Preference reasoning and aggregation over combinatorial domains in uncertain and multi-agent scenarios

PhD student: Cristina Cornelio *Supervisor: Francesca Rossi*

Doctoral School in Mathematical Sciences
Computer Science area, XXVIII course
University of Padua, Italy

February 2016

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Sede Amministrativa: Università degli Studi di Padova
Dipartimento di Matematica

SCUOLA DI DOTTORATO DI RICERCA IN: SCIENZE MATEMATICHE
INDIRIZZO: INFORMATICA
CICLO: XXVIII

PREFERENCE REASONING AND AGGREGATION OVER COMBINATORIAL DOMAINS IN UNCERTAIN AND MULTI-AGENT SCENARIOS

Direttore della Scuola: Ch.mo Prof. Pierpaolo Soravia

Coordinatore d'indirizzo: Ch.mo Prof. Francesca Rossi

Supervisore: Ch.mo Prof. Francesca Rossi

Dottorando: Cristina Cornelio

Contents

Riassunto

Le preferenze, intese come opinioni di utenti su un insieme di oggetti, sono ampiamente presenti nelle nostre vite e recentemente sono diventate molto studiate in Intelligenza Artificiale. In molti contesti della nostra vita, non consideriamo gli oggetti come entità atomiche, ma consideriamo un insieme di caratteristiche/attributi che le caratterizzano e che interagiscono tra di loro. Siamo dunque particolarmente interessati in questi tipi di scenario: preferenze condizionali su domini combinatori e multi-attributo.

L'abilità di rappresentare le preferenze in maniera compatta è essenziale, in particolare nel contesto di modellazione e ragionamento con preferenze multi-attributo, in presente un esplosione combinatoria di informazione che causa un alto costo computazionale. A questo scopo sono state sviluppate in letteratura una serie di linguaggi di rappresentazione compatta. In questa tesi ci focalizziamo inizialmente su un modello grafico di rappresentazione delle preferenze condizionali: le conditional preference networks (*CP-nets*). Analizziamo quindi i vantaggi e gli svantaggi delle CP-net concentrandoci su scenari incerti e multi-agente.

Gli scenari reali sono spesso dinamici e incerti: un utente può cambiare le sue idee nel tempo e le sue preferenze potrebbero essere affette da errori o rumore. In questa tesi proponiamo una nuova strutture chiamata *PCP-net* (probabilistic conditional preference networks), generalizzazione delle CP-net, capace di rispondere ai cambiamenti attraverso aggiornamenti e di supportare informazione probabilistica. Le PCP-net possono essere usate anche per rappresentare i contesti multi-agente dove ogni agente è rappresentato da una CP-net e le probabilità vengono usate per riconciliare i conflitti tra gli utenti.

In questa tesi analizziamo anche un altro linguaggio di rappresentazione compatta, simile alle CP-net: i *vincoli soft*. I vincoli soft sono meno restrittivi rispetto alle CP-net, ma le complessità computazionali rimangono le stesse per i task principali. Per questo motivo, ripensiamo lo scenario

multi-agente usando un profilo di agenti che esprimono le loro preferenze attraverso vincoli soft, invece che tramite una collezione di CP-net poi aggregate in una PCP-net.

La letteratura riguardante le CP-net presenta anche molte altre generalizzazioni, poichè le CP-net sono per certi versi restrittive e limitate nell'espressività. Ad esempio, le CP-net sono state estese con vincoli, con inconsistenza locale e incompletezza (GCP-net), con funzioni di utilità (UCP-net), etc. Sono dunque stati sviluppati molti formalismi differenti per descrivere le preferenze condizionali e ognuno di essi ha una sintassi e una semantica ah hoc e algoritmi specifici. In questa tesi specifichiamo un nuovo framework con lo scopo di unificare tutti questi modelli. La forza della nostra formulazione è la diretta espressione del modello in logica standard al primo ordine, come una teoria Datalog vincolata. Questa formulazione è ricca abbastanza da esprimere le CP-nets e tutte le sue estensioni.

Concludiamo la tesi, studiando un applicazione delle preferenze in un scenario reale. Analizziamo come migliorare gli algoritmi di scambi di reni aumentando il numero di trapianti e di durata di vita aspettata, fornendo alcuni risultati incoraggianti. Quindi forniamo anche alcune idee preliminari riguardo a come incorporare le preferenze nelle procedure di matching attualmente utilizzate.

Abstract

Preferences, intended as user opinions over items, are conspicuously present in our lives and recently became widely studied in Artificial Intelligence. In many contexts of our life, we do not consider items only as entire entities, but we consider a set of features/attributes that characterize them and that interact to each other. Therefore, we are particularly interested in this kind of scenarios representing conditional preferences over combinatorial and multi attribute domains.

The ability of representing preferences in a compact way is essential, especially in the context of multi-attribute preference modelling and reasoning since it causes combinatorial explosion of information and a high computational cost. For this purpose, a number of compact representation languages have been developed in the literature. We initially focus on a recently developed framework for representing conditional preferences over a graphical structure: conditional preference networks (*CP-nets*). We analyse the advantages and the drawbacks of CP-nets focusing on uncertain and multi-agent scenarios.

Real life scenarios are often dynamic and uncertain: a user can change her mind over time and her preferences could be affected by errors or noise. We propose the new *PCP-net* structure (probabilistic conditional preference networks), as a generalisation of the CP-nets framework, able to respond to change through updates and to support probabilistic informations. PCP-nets can be used also to represent a multi-agent scenario where each agent is represented by a CP-net and probabilities are used to reconcile conflicts between users.

We analyse another compact representation language, similar to CP-nets, namely soft constraints. Soft constraints are less restrictive, with respect to CP-nets, but the computational complexity remains almost the same for the main tasks. For this reason, we rethink the multi-agent context by using a profile of agents expressing their preferences via soft constraints, instead of via a collection of CP-nets aggregated into a PCP-net.

The CP-nets literature presents also many other generalisations, since CP-nets are restrictive and limited in expressiveness. For example, CP-nets have been extended with constraints, with local inconsistency and incompleteness (GCP-nets), with utility functions (UCP-nets), etc. Therefore, several different frameworks were developed to describe conditional preferences. Each of these formalisms have ad hoc syntax, semantics and specialised algorithms. In this thesis, we specify a new framework with an unification purpose of all these models. The strength of our formulation is the direct expression of the model in the standard first order logic, as constrained Datalog theories. This formulation is rich enough to express CP-nets and all its extensions.

We conclude this work, studying an application of preferences in a real-life scenario. We analyse how to improve kidney exchanging algorithms increasing the number of transplantations and the expected life duration, providing some encouraging results. Then we provide also some preliminary ideas about how to incorporate preferences in the matching procedure currently used.

Introduction & Background

1. Introduction

1.1 Context

Preferences play an important role in artificial intelligence, in automated decision making and multi-agent systems [17, 21, 49, 51, 91]. The ability to express preferences in a faithful way, which can be handled efficiently, is essential in many reasoning tasks. In settings such as e-commerce, on demand video, and other scenarios where supply outstrips an individuals ability to view all the available choices, we require an efficient formalism to model and reason with complex, interdependent preferences. We may also use these preferences to make decisions about joint plans, actions, or items in multi-agent environments. Agents express their preferences over a set of candidate decisions and these preferences are aggregated into one decision which satisfies as many agents as possible.

Often multi-attribute preference modeling and reasoning causes a combinatorial explosion leading to high computational cost [39, 41, 52]. The set of candidates is often described as a product of multiple features. Consider for example a car: usually it is not seen as a single item, but as a combination of features, such as its engine, its shape, its color, and its cost. Each of these features has some possible values, and a car is the combination of such feature values. This is often because each decision has a combinatorial structure, that is, can be seen as the combination of certain features, where each feature has a set of possible values. This occurs in several AI applications, such as combinatorial auctions, web recommender systems, and configuration systems [79]. When the set of decisions is small, a user may just present an order over them to express their preferences. However, when the set of objects is very large, this is infeasible. Usually the values for each feature are not many, but, because of the combinatorial structure, the set of decisions can still be very large. Consider an example where we need to choose a menu for a meal, composed of a first course, a main course, a dessert, and a wine. Assuming 5 choices for each of these, we should choose

among $5^4 = 625$ different meals. In this thesis we assume that the decision set has a combinatorial structure.

A number of *compact representation languages* have been developed in the literature to tackle the computational challenges arising from these problems. The main ones are conditional preference structures (CP-nets) [11], soft constraints [10, 78, 91], GAI-nets [53] and graphical utility models [7]. In this work we firstly assume individual agents express their preferences as CP-nets, a qualitative preference modeling framework that allows for conditional preference statements, and then as soft constraints, a quantitative preference model that allows undirected conditional preferences.

A structure such as a CP-net or a soft constraint problem automatically induces a partial or a total order over the set of decisions. For such formalisms, the literature tells us how computationally expensive it is to find the most preferred decision, or to test if a decision is the most preferred, or also to test whether one decision is more preferred to another one [52, 78]. For example, for both types of structures, it is NP-hard to find the most preferred decision, although it becomes linear in the number of features when some topological restrictions hold [52, 78].

There is some experimental evidence suggesting qualitative preferences are more accurate than quantitative preferences elicited from individuals in uncertain information settings, [92]. CP-nets are a qualitative preference modeling framework that allow for conditional preference statements [11]. They are compact, arguably quite natural, intuitive in many circumstances, and widely used in many applications in computer science such as recommendation engines [49]. There exist many generalizations of CP-net, such as *General CP-nets* [52] that define CP-nets that can be incomplete or locally inconsistent. *CP-theories* [111] are a specialized formalism, with its own *ad hoc* syntax and semantics, in which preferences may be conditioned on *indifference* to certain features. *Comparative preference theories* [112] permit preferences to be defined on a set of features simultaneously. Along another direction, [12] and [42] introduce the idea of adding (hard and soft) constraints to CP-nets.

Often, we must make a joint decision and we need to compromise our preferences with those of other people. To come back to the meal example above, we may think of a set of friends who want to organize a dinner and need to agree on a common menu. This is a kind of scenario we consider in this thesis: we consider a set of agents, each expressing their preferences via soft constraints or via CP-nets over a common set of decisions with a combinatorial structure. The goal is to aggregate such preferences and to select a joint decision. While compact preference formalisms mentioned above are suitable for expressing the preferences of a single agent, they may lead to undesirable results if used for multi-agent preference aggregation. For example, in soft constraints, the negative opinion of an agent over a specific value of a feature would impact negatively on any decision containing such a value, even if all other agents like it. An alternative could be to search for a decision belonging to the Pareto frontier of the multi-agent preferences, as it is done in [54] for preferences expressed via General Additive Independence (GAI) utility functions. Another option, that we propose in this thesis, is to aggregate the agents' preferences via voting rules.

1.2 Problem Statement and Research Goals

1.2.1 Probabilistic and Dynamic scenarios

Real life scenarios are often dynamic. A user can change her mind over time or the system under consideration can change its laws: preferences may change over time. Thus, we need a structure that can respond to change through updates, without the need to completely rebuild the structure. Additionally, we often meet situations characterized by some form of uncertainty. We may be unsure about our preference ordering over certain items, or there could be noise in our preference structure due to lack of precision in elicitation or sensor collection (e.g., measurement error from remote sensors). In order to model this, we need a structure for handling uncertainty and change, for example via probabilistic information. The need for encoding uncertain, qualitative information has seen some work in the recommendation engine area [46, 85] and is a motivating example.

Consider a household of two people and their Netflix account. The recommendation engine only observes what movies are actually watched, at what time they are watched, and their final rating. There are two people in the house and let us say that one prefers drama movies to action movies while the other one has the opposite preference. When making a recommendation about what type of movie to watch, the engine may consider several facts. Comedies may always be watched in the evening, so we can put a deterministic, causal link between time of day and type of movie. However, we cannot observe which user is sitting in front of the television at a given time. There is strong evidence from the behavioral social sciences showing that adding uncertainty to preference frameworks may be a way to reconcile transitivity when eliciting input from users [88], among other nice properties [73]. Using this idea, we propose to add a probabilistic dependency between our belief about who is in front of the television and what we should recommend. We may want to update the probability associated with this belief based on the browsing or other real-time observable habits of the user. Users could change their preferences over time or there could be different users with different preferences. To do this we need an updatable and changeable structure that allows us to encode uncertainty. This is the main proposal of this thesis, that we call PCP-net. While keeping the preferential dependency structure of a CP-net, in a PCP-net a preference ordering over a variable domain is replaced by a probability distribution over all possible preference orderings over that domain.

Once we have this new dynamic and probabilistic structure (PCP-net), we want to perform two main tasks: optimality and dominance. Optimality corresponds to understand how to compute the most probable alternative or the most probable user representation. Dominance corresponds to understanding, given two alternatives, which one has the higher probability of being preferred.

1.2.2 Multi-agent scenarios

Firstly we consider multi-agent scenarios where individual agents express their preferences as CP-nets, then we consider another model to represent an profile of agents: soft constraints. Agents, in this scenario, may use these preferences frameworks to make decisions about joint plans, actions, or items in multi-agent environments: agents express their preferences over a set of alternative decisions, and such preferences are aggregated into a collective decision. Often in these settings

we need to handle partial preferences of agents [90] or reconcile preferences that directly conflict. Voting has been used extensively to resolve these conflicts [68, 72, 74, 75, 114].

PCP-net aggregation

We investigate the use of PCP-nets (Probabilistic CP-nets) as a new structure to represent the preferences of a group of agents, since a PCP-net defines a probability distribution over a collection of CP-nets: all those CP-nets that can be obtained from the PCP-net by choosing one variable ordering out of the whole probability distribution over them.

Given a PCP-net, we focus on three collective reasoning tasks that agents may want to perform: finding the candidate which represents best the group of agents, finding the outcome that is most preferred by the agents, or decide whether an outcome is collectively preferred to another (a dominance query).

Previous efforts to tackle these collective reasoning problems over CP-nets have primarily focused on the question of optimality [68] or dominance [96] and have not dealt with uncertainty. Generally, a voting method is defined in order to determine the most preferred alternative. These methods are usually sequential (multi-issue voting) and work at the level of the individual variables [65, 67, 68, 114, 116]. In contrast, we propose to aggregate the collection of CP-nets into a single structure, a PCP-net, on which we directly perform collective reasoning tasks, including but not limited to testing optimality. This provides a formalism to perform other preference reasoning tasks such as answering dominance queries. Moreover, we can store or communicate between agents a single, compact structure instead of a possibly large collection of CP-nets.

We analyse different methods for aggregating a collection of CP-nets into a single PCP-net and compare them theoretically and experimentally.

We also study how our methods to compute the main reasoning tasks (optimality and dominance) perform on our resulting PCP-net in comparison to the given collection of CP-nets, both from an accuracy point of view and also from a computational complexity point of view, also in comparison with state of the art results and methods.

Soft-constraints aggregation

With respect to CP-nets, soft constraints allow one to avoid imposing many restrictions on the agents' preferences, since they are not directional, and thus information can flow from one variable of a constraint to another one without a predefined ordering between them, in a multi-agent setting where preferences are aggregated. This allows one to not tie the variable ordering used by the sequential procedure (multi-issue voting) to the topology of the constraint graph of each agent. Moreover, soft constraints can model also strict requirements, which are often necessary in multi-issue settings where one needs to rule out some combinations of feature values. Not only are CP-nets different from soft constraints syntactically, but also they induce different ordering over the solutions. In fact, CP-nets never induce orderings with ties which is typical in soft constraint problems but usually present many incomparable outcomes. Moreover, while it is computationally difficult to compare two solutions in CP-nets, in soft constraint problems it is easy. In terms of reasoning complexity, computing the optimal solution in an acyclic CP-net is easy, while it is computationally difficult for general CP-nets; it is in general difficult to find an optimal solution

in a soft constraint problem, while it is polynomial if the constraint graph of the problem has no cycles or a bounded treewidth. We are interested to undergo a study similar to the one performed in [65] for CP-nets in scenarios where the agents express their preferences via soft constraints.

1.2.3 Logic scenarios

CP-nets represent one end of the expressiveness/tractability spectrum. While useful in practice, CP-nets are limited in expressiveness: a rule may specify a preference for exactly one value over another (in the same feature domain). The literature presents many extension and variants (General CP-nets, CP-theories, Comparative preference theories, hard and soft constrained CP-nets, etc.) each one with ad hoc syntax, semantics and algorithms. We want to specify a new framework that unifies all the models above and that does not need different syntax, semantics and algorithms. We analyze the idea that conditional preferences can be directly expressed in standard first order logic, as constrained Datalog theories [19, 59, 105].

We propose a semantics, described in first-order logic, rich enough to express CP-nets and each of its extensions discussed above, including algorithms for consistency, dominance and optimality. We plan to analyze the properties and computational complexity of these main tasks.

Introducing constraints in first order logic is natural, thus we do not have to introduce them in an *ad hoc* fashion as in [12, 42, 84]. For example in [84] authors use an ad hoc semantic for flipping sequences: *consistent flipping worsening sequence*, flipping sequence that allow worsening flips only between consistent outcomes.

We want also to exploit the advantage of the introduction of conditional preference theories as Datalog programs [20]: thus we make Datalog's rich semantic, algorithmic and implementation framework available in service of conditional preferences.

1.3 Summary of Contributions

1.3.1 Probabilistic and Dynamic framework

We propose a new framework: *PCP-net* (Probabilistic CP-nets). This is a generalisation of CP-nets which allows for uncertainty and online modification of the dependency structure and preferences of CP-nets. PCP-nets provide a way to express probabilities over dependency links and over preference orderings in conditional preference statements. Given a PCP-net we study the computational complexity of the main reasoning tasks concerning preferences: outcome optimization and dominance queries.

Concerning optimality in a PCP-net there are two interpretations: the first one considers that a PCP-net defines a probability distribution over a set of CP-nets; the second one considers the set of outcomes. Thus we show how to find the most probable induced CP-net, that is, the CP-net in the distribution that has the highest probability to appear in reality. We define the notion of optimal outcome in two natural ways: the most probable optimal outcome and the optimal outcome of the most probable CP-net induced by the PCP-net. If the dependency structure of the PCP-net has bounded in-degree, both kinds of optimal outcomes can be found in time polynomial in the size of the PCP-net.

We then study dominance which is in general computationally a difficult task. For PCP-nets whose dependency graph is a polytree, for which dominance is still difficult, we give a polynomial time approximation in the form of a lower and an upper bound to the correct probability value for dominance. We show, experimentally, that often the range between the lower and upper bound is small, and in particular that the lower bound is very close to the correct value.

1.3.2 Multi-agent context

We consider scenarios where several agents must aggregate their preferences over a large set of candidates with a combinatorial structure and we assume agents to compactly express their preferences over the candidates via firstly CP-nets and then soft constraints.

PCP-nets in a multi-agent context

While keeping the preferential dependency structure of a CP-net, in a PCP-net a preference ordering over a variable domain is replaced by a probability distribution over all possible preference orderings over that domain. Thus, a PCP-net defines a probability distribution over a collection of CP-nets: all those CP-nets that can be obtained from the PCP-net by choosing one variable ordering out of the whole probability distribution over them.

We then adapt, to the multi-agent context, the two collective reasoning tasks that agents may want to perform, optimality and dominance, applying the existing notions and algorithms developed for PCP-nets. Given a set of agents each expressing their preferences as a CP-net, we may want to find the outcome that is most preferred by the agents (optimality over outcomes) or to find the candidate which is most preferred by the agents (the most probable induced CP-net), or decide whether one candidate is collectively preferred to another(dominance query). We propose to aggregate the collection of CP-nets into a single structure, a PCP-net, on which we directly perform collective reasoning tasks. This provides a formalism to perform other preference reasoning tasks such as answering dominance queries, in addition to finding the most preferred outcome. Moreover, we can store or communicate between agents a single, compact structure instead of a possibly large collection of CP-nets.

We consider two methods for aggregating a collection of CP-nets into a single PCP-net: by extracting the probability distribution directly from the CP-nets (proportional method, *PR*) or by minimizing a notion of error between the aggregated structure and the original set of CP-nets (least-squares method, *LS*). We then analyze the methods experimentally, showing that when aggregating the agents' CP-nets, the proportion method yields a PCP-net which more accurately captures the preferences of the agents than the least-square method.

When the PCP-net is obtained by aggregating a collection of CP-nets via the proportion method, we show experimentally that our approximation of dominance closely models dominance computed on the collection of given CP-nets. The same result holds for a deterministic form of dominance we call MP-dominance.

Aggregation of Soft Constraint Profiles

To aggregate the preferences of the agents, represented via soft constraints, we consider a sequential procedure that asks the agents to vote on one variable at a time. At each step, all agents express their

preferences over the domain of a variable; based on such preferences, a voting rule is used to select one value for that variable. When all variables have been considered, the selected values constitute the returned variable assignment, that is, the elected candidate. We study several properties of this procedure (such as Condorcet consistency, anonymity, neutrality, monotonicity, consistency, efficiency, participation, independence of irrelevant alternatives, non dictatorship, and strategy-proofness), by relating them to corresponding properties of the adopted voting rules used for each variable. Moreover, we perform an experimental study that shows that the proposed sequential procedure yields a considerable saving in time with respect to a non-sequential approach, while the winners satisfy the agents just as well, independently of the variables ordering. We provide also results on real preferences obtained from the PrefLib database and describing the TripAdvisor users preferences over features describing hotels.

1.3.3 Logic framework

We observe that constrained conditional preferences systems can be expressed as particular kinds of constrained Datalog programs which we call (constrained) *LCP theories*. These programs can be run using tabling in several Prolog systems, such as XSB Prolog. We show that existing conditional preference frameworks (CP-nets [11], CP-theories [111], GCP [52], comparative preferences [112] are all particular cases of *flat* LCP theories. We also introduce a new, powerful form of conditional preferences, *recursive conditional preferences*, that can be formulated as LCP theories (i.e. within Datalog). We establish complexity results for flat and recursive LCP for the canonical conditional preference algorithms (consistency, dominance, optimality), extending prior results. We provide new polynomial results on special forms of conditional preference theories, namely acyclic and tree-structured theories, extended to Flat LCP. We introduce a new complexity measure for conditional preferences, *data-complexity*, leveraging data-complexity notions from Datalog. We implement a compiler for our system into XSB Prolog. The compiler recognizes special cases of LCP theories (e.g. acyclic) and generates efficient (poynomial-time) code; otherwise the general tabling-based mechanism is used to answer basic queries. We analyze several new notion of dominance for CP-nets that are more efficient from a computational point of view, taking advantage of the Datalog formulation of the theory. We analyze the properties of these new semantics and we provide also a comparison between them and the classical formulation of dominance.

1.3.4 Application to a real-life problem: The kidney exchange protocol

We study a particular application of preferences in a real-life scenario: the kidney exchange protocol. Kidney exchange is widely used in many countries, and the success of this procedure depends on how many patients (with the corresponding donor) partecipate to the program. In Italy, and in particular in our context of Padua, only very few patients join the kidney exchange program. We study how to encourage the donor/patient pairs to participate and we analyze how to improve the existing algorithms to increase the number of transplantations and the expected life span. We provide a procedure that improves the number of transplantations in the dataset of Padua, and we provide encouraging results. We also the provide some preliminary ideas about how to incorporate preferences in the matching procedures currently used, to minimize transplantation failures: the

number of rejected proposed matchings.

1.4 Thesis statement

The thesis analyzes preferences over combinatorial domains, providing two generalizations of conditional preference frameworks. The first one, located also in a multi-agent scenario, follows a probabilistic point of view allowing uncertain and dynamic reasoning. The second stream instead, with a unification purpose of the main models for representing preferences over combinatorial domains, consists into a logic generalization that expands the notion of preferences through the new concept of recursive preferences. These generalizations are useful to have a comprehensive and complete formulation of conditional preferences over combinatorial domains, preserving the state-of-the-art computational complexities.

1.5 Publications

The scientific findings contained in this thesis have been presented at international conferences and workshops. The most relevant ones are:

- Conferences:
  - ⋆ **"Dynamic and Probabilistic CP-nets"**, *C. Cornelio*, Proceedings of the Doctoral Program of International Conference on Principles and Practice of Constraint Programming 2013, CP-13.
  - ⋆ **"Updates and Uncertainty in CP-net"**, *C. Cornelio, U. Grandi, J. Goldsmith, N. Mattei, F. Rossi and K.B. Venable*, Proceedings of AUAI 2013: Advances in Artificial Intelligence, 26th Australasian Joint Conference, 2013.
  - ⋆ **"Reasoning with PCP-nets in a Multi-Agent Context"**, *C. Cornelio, U. Grandi, J. Goldsmith, N. Mattei, F. Rossi and K.B. Venable*, Proceedings of the International Conference on Autonomous Agents & Multiagent Systems 2015, AAMAS-15.
  - ⋆ **"Models for Conditional Preferences as extensions of CP-nets"**, *C. Cornelio*, Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (extended abstract), Doctoral Consortium, IJCAI-15.
- Workshops:
  - ⋆ **"Dynamic Probabilistic CP-nets"**, *C. Cornelio, U. Grandi, J. Goldsmith, N. Mattei, F. Rossi and K.B. Venable*, Proceedings of the 7th Multidisciplinary Workshop on Advances in Preference Handling of the International Joint Conference on Artificial Intelligence 2013, MPREF-13.
  - ⋆ **"Voting with CP-nets using a Probabilistic Preference Structure"**, *C. Cornelio, U. Grandi, J. Goldsmith, N. Mattei, F. Rossi and K.B. Venable*, Proceedings of the 5th International Workshop on Computational Social Choice, ComSoC-14.
  - ⋆ **"Logical conditional preference theories"**, *C. Cornelio, A. Loreggia, and V. Saraswat*, Proceedings of the MPREF workshop of the International Joint Conference on Artificial Intelligence 2015, MPREF-15.
- Submitted:

- ⋆ **"Multi-agent soft constraint aggregation via sequential voting: theoretical and experimental results"**, *C. Cornelio, G. Dalla Pozza, M.S. Pini, F. Rossi, K.B.Venable*.
- ⋆ **"Reasoning with PCP-nets"**, *C. Cornelio, U. Grandi, J. Goldsmith, N. Mattei, F. Rossi and K.B. Venable*.

1.6 Structure of the thesis

The thesis is structured in three main parts: the first part is about background notions and the state of the art of the context analysed in this thesis; the second part describes the theoretical contributions of the thesis and the use of preference formalisms in an application scenario; the third part is the concluding section with a summary and some future directions.

In details the thesis is organized in the following chapters:

- Chapter 1: *Introduction.* In this chapter we introduce an overview of the thesis focusing on the context in which we allocate our work, the problems that have been addressed, and the solutions/original contributions provided.
- Chapter 2: *Background.* In this chapter we deal the background notions needed in the thesis regarding preference representations: Bayesian Networks, CP-nets, variants and extensions of CP-nets, soft constraints and voting theory.
- Chapter 3: *PCP-net.* In this chapter we introduce PCP-nets and we analyse the algorithmic properties of the associated main tasks. We also provide a dynamic interpretation of PCP-nets that allows changes and modifications of the preferences during the time. We also provide an alternative formulation of preferences in a probabilistic context, using a different notion of probability: the Dempster-Shafer theory of probability. Moreover, we propose an algorithm for learning a PCP-net in the preliminary case of separable dependency structure.
- Chapter 4: *Aggregation of preferences.* In this chapter we analyse two different kinds of aggregation of preferences in a multi-agents context: the first one aggregating CP-nets into a PCP-net and the second one aggregating the preferences of a group of agents expressed via (fuzzy) soft constraints. In both the two sections we provide theoretical and experimental analysis. In Section 4.1.6 we provide a method to build a polytree dependency graph for a PCP-net aggregated from a O-legal profile of CP-nets.
- Chapter 5: *A logic framework for conditional preferences.* In this chapter we introduce the new unifying and logic framework for conditional preferences: LCP-theories. We analyse the algorithmic properties and computational complexity of the associated main tasks. We provide also the definition and analysis of three alternative semantics for the dominance task that are particularly useful in this context.
- Chapter 6: *Preferences in a real-life scenario.* In this chapter we study the kidney exchange protocols and we propose a way to improve the currently existing exchange algorithms. We provide also some preliminary ideas of introducing patients preferences in the computation of the matching to obtain a further improvement.
- Chapter 7: *Summary and Conclusions.* In this chapter we present a summary of the main contributions of the thesis and we provide a concluding discussion about the obtained results.

- Chapter 8: *Future directions.* We conclude the thesis introducing some future work directions presenting also some preliminary considerations.

2. Background and State of the art

Preferences play a key role in artificial intelligence therefore it is essential to provide frameworks to represent them in an effective way. In the following sections we introduce the background notions, frameworks and methodologies used in the preferences context.

There exists many interpretation of what a "preference" is, but all these definitions can be classified in two main classes. The first point of view is "*quantitative*": this interpretation is based on utility functions that represent how much the users like or dislike an item or an alternative. The second point of view instead is "*qualitative*": this interpretation is based on the comparison between two or more alternatives; the user provide her preference between two items but not specifying how much more she likes the preferred one.

Preferences can be expressed over several kinds of items: candidates of an election, films, music, etc. In many contexts of our life in which we have to make a decision over a set of alternatives, we do not express our preferences over the alternatives considered as entire entities but we consider the set of features that characterize them. In these kind of contexts we are talking about alternatives with *multi-attribute domains*. When the number of features describing the items increase, for humans, it becomes difficult to compare outcomes, since it is difficult to evaluate alternatives over multiple dimensions. Also for machines is difficult to handle multi-attribute domains since it requires to save in memory an exponential amount of information. For these reasons, the relations between the different features have an important role and are widely studied in the literature. In the *quantitative* context the preferences are represented by utility functions defined as a combination of sub-functions each one corresponding to a different feature. In the *qualitative* scenario preferences are divided focusing on subset of features with particular properties. Thus become obvious that the features (properties that characterize an alternative) can be either interconnected to each other or independent to each other, and this idea permits to simplify the computational complexity of the

frameworks. A popular way to represent the interaction between the features corresponds to the use of graphical models, taking advantage on the independence between the features. A significant number of such a models have been developed, starting from Bayesian networks that are a compact way to represent a probability distribution over a set of events defined in a combinatorial domain. Afterwards other graphical models have been defined to represent qualitative and quantitative preferences. The main graphical model concerning qualitative preferences are *CP-nets*, defined in 2004 by Boutilier et al. [11].

This framework (CP-nets) represents in a compact way the conditional preferences of a user over a set of features. For example the statement "Given a Ferrari I prefer the red color to the yellow; but I prefer the opposite for a Lamborghini." means that the feature *color* depends on the feature *model* of the car. In this chapter firstly we introduce CP-nets, analyzing the strengths and the drawbacks of this formulation.

In the literature many generalizations and variants of CP-nets have been developed. In the following sections we analyse these main generalizations:

- TCP-nets [15]: an extension of CP-nets that permits to represent also the notion of *relative importance* and *conditional relative importance* between the features;
- CP-trees [112]: a formulation of preferences more restrictive than a CP-net but useful in practice since in allows polynomial computation for the main reasoning tasks;
- CI-net [14]: a framework that depends on the notion of *conditional importance statement*;
- UCP-nets [13]: an extension of CP-nets in which the preferences are expressed in a quantitative way, exploiting a utility function defined on the single features;
- mCP-nets [90]: an extension of CP-net formalism to model and handle the preferences of multiple agents;
- GCP-nets [41, 52]: a general form of conditional and qualitative preferences that allow local inconsistency and incompleteness;
- CP-nets with constraints [12, 84]: a model to reason with CP-net and hard constraints jointly;
- CP-theories [111]: a formulation that allows a more general form of cp-statements, with respect to CP-nets;
- Comparative preference theories [112]: a framework that allows to express preferences on several features simultaneously.

Soft-constraints [78] are another way, analysed in this chapter, to represent preferences exploiting graphical structures, but in a quantitative way. Then we introduce the *voting theory* environment: a voting system is a method by which a set of voters choose one among a set of options/candidates. Voters opinions can be expressed by a submission of a vote or by their preference ordering over the set of candidates (or part of it). Voting theory is widely used in the preferences analysis context.

Chapter structure

The chapter is organized as follows.

- In Section 2.1 we introduce the Bayesian networks framework, providing the state-of-the-art procedures for the main tasks and their related computational complexity.
- In Section 2.2 we present the CP-nets model. We provide the formal definition and we also

introduce the notions of optimal outcome and outcome comparison, providing algorithms and associated computational complexities.

- In Section 2.3 we study the variants and generalizations of the CP-nets framework: TCP-nets, CP-trees, CI-net, UCP-nets, mCP-nets, GCP-nets, CP-nets with constraints, CP-theories and Comparative preference theories.
- In Section 2.4 we present a preferences representation model alternative to CP-nets, "quantitative" oriented: Soft constraints. We analyse also optimality in this scenario.
- In Section 2.5 we introduce the voting theory environment. We provide the definition of the main voting rules and we define the axiomatic properties that are usually considered.

2.1 Bayesian Networks

The Bayesian networks formalism was invented to allow an efficient representation of uncertain knowledge and to reason rigorously with it ([29], [31] and [93]). This approach is now one of main research topics on uncertain reasoning and expert systems.

The concept of a Bayesian network encompasses the importance of independence and conditional independence relationships in simplifying probabilistic representations of the world. In this section we introduce a systematic way to represent such relationships explicitly in the form of Bayesian networks. We then show how probabilistic inference, although computationally exponential in the worst case, can be done efficiently in many practical situations.

2.1.1 Independence and Conditional Independence

Before explaining what a Bayesian network is, we have to introduce the notions of independence and conditional independence.

Definition 2.1.1 Let X and Y be two discrete random variables. X is *independent* of Y if

$$\mathbb{P}(X,Y) = \mathbb{P}(X)\mathbb{P}(Y).$$

We have also that, if X is *independent* of Y, the following equalities hold:

$$\mathbb{P}(X|Y) = \mathbb{P}(X), \qquad \mathbb{P}(Y|X) = \mathbb{P}(Y).$$

The definition of independence can be extended to sets of variables, as follows.

Definition 2.1.2 The set of variables $X_1,\cdots,X_l$ is *independent* of the set of variables $Y_1,\cdots,Y_m$ if

$$\mathbb{P}(X_1,\cdots,X_l,Y_1,\cdots,Y_m) = \mathbb{P}(X_1,\cdots,X_l)\mathbb{P}(Y_1,\cdots,Y_m).$$

Definition 2.1.3 Let X, Y and Z be three discrete random variables. X is *conditionally independent* of Y given Z if

$$\mathbb{P}(X|Y,Z) = \mathbb{P}(X|Z).$$

We have also that if X is *conditionally independent* of Y given Z then:

$$\mathbb{P}(X,Y|Z) = \mathbb{P}(X|Z)\mathbb{P}(Y|Z), \qquad \mathbb{P}(X|Y,Z) = \mathbb{P}(X|Z).$$

The definition of conditional independence, as the definition of independence, can be extended to sets of variables.

Definition 2.1.4 The set of variables $X_1,\cdots,X_l$ is *conditionally independent* of the set of variables $Y_1,\cdots,Y_m$, given the set of variables $Z_1,\cdots,Z_n$, if

$$\mathbb{P}(X_1,\cdots,X_l|Y_1,\cdots,Y_m,Z_1,\cdots,Z_n) = \mathbb{P}(X_1,\cdots,X_l|Z_1,\cdots,Z_n).$$

These notion combined with Bayes's rule are the basis of the Bayesian networks.

Definition 2.1.5 — Bayes's rule. Given two discrete random variables X and Y, Bayes rule is as follows:

$$\mathbb{P}(X|Y) = \mathbb{P}(Y|X)\frac{\mathbb{P}(X)}{\mathbb{P}(Y)}.$$

2.1.2 Representation of Bayesian networks

The full joint probability distribution of a set of variables can answer all the questions about the domain, however, such a domain, can become very large as the number of variables grows. Independence and conditional independence among variables can greatly reduce the number of probabilities that need to be specified in order to define the full joint distribution. *Bayesian networks (BN)* are data structures defined to represent the dependencies among variables. A BN can represent also any full joint probability distribution and in many cases can do so very concisely.

Definition 2.1.6 A *Bayesian network* is a directed graph in which:

- each node corresponds to a random variable (discrete or continuous);
- a set of directed edges connects pairs of nodes (if there is an edge from node X to node Y, X is said to be a *parent* of Y);
- the graph has no directed cycles and hence is a directed acyclic graph (DAG);
- each node X_i has a family of conditional probability distribution $\mathbb{P}(X_i|Parents(X_i))$ that quantifies the effect of the parents on the node.

Definition 2.1.7 If the nodes are discrete variables, each of them X_i has a *conditional probability table (CPT)*, in which each row contains a family of conditional probability distributions, each one associated with a possible combination of values for the parent nodes $\mathbb{P}(X_i|Parents(X_i))$.

Observation 2.1 Consider a BN in which the variables have domains of cardinality equal to d. In general, a table for a variable with k parents contains d^{k+1} independently specifiable probabilities. A node with no parents has only one row representing the prior probabilities of each possible value of that variable.

A Bayesian network can often be far more compact than the full joint distribution, since the full joint distribution needs d^n values while BN needs nd^{k+1} values (that could be less then d^n if

$\max\{k\}$ is small). ■

Observation 2.2 Considering Boolean variables, once you know that the probability of a *true* value is p, the probability of false must be $1-p$, so we often omit the second number. In general, a table for a Boolean variable with k Boolean parents contains 2^k independently specifiable probabilities. ■

We now show a Bayesian network.

■ **Example 2.1** Consider a Bayesian network with four variables A, B, C and D with domains $\mathscr{D}_A = \{a_1, a_2, a_3\}$, $\mathscr{D}_B = \{b_1, b_2\}$, $\mathscr{D}_C = \{c_1, c_2, c_3\}$ and $\mathscr{D}_D = \{d_1, d_2\}$. Assume there are two edges: from B to C and from B to D (as depicted in Figure 2.1).

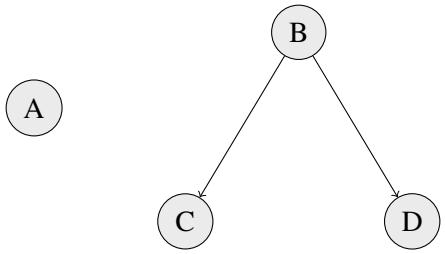


Figure 2.1: Bayesian network (Example 2.1)

The variables have the following CPTs:

- Variable A:

a_1	a_2	a_3
0.4	0.5	0.1

- Variable B:

b_1	b_2
0.7	0.3

- Variable C:

B values	c_1	c_2	c_3
b_1	0.2	0.3	0.5
b_2	0.6	0.2	0.2

- Variable D:

B values	d_1	d_2
b_1	0.4	0.6
b_2	0.8	0.2

We note that there are 9 degrees of freedom in the CPTs, but if we take all the joint probabilities we would have 35 values. ■

We will see, in the following section, that the combination of the topology of the network and the conditional distributions of the variables suffices to specify the full joint distribution for all the variables.

2.1.3 The joint probability

The previous section described what a network is, but not what it means.

One way to define what the network means is to define the way in which it represents a specific joint distribution over all the variables.

The parameters associated with each node correspond to conditional probabilities $\mathbb{P}(X_i|Pa(X_i))$. A generic entry in the joint distribution is the probability of a conjunction of particular assignments to each variable, such as $\mathbb{P}(X_1 = x_1 \cdots X_n = x_n)$. The value of this entry is given by the formula:

$$\mathbb{P}(X_1 = x_1, \cdots, X_n = x_n) = \prod_{i=1}^{n} \mathbb{P}(X_i = x_i | Pa(X_i) = Pa(x_i)). \tag{2.1}$$

where $Pa(x_i)$ is the assignment of the parents of X_i according to the assignment $(X_1 = x_1 \cdots X_n = x_n)$. Thus, each entry in the joint distribution is represented by the product of the appropriate elements of the conditional probability tables (CPTs) in the Bayesian network.

Equation 2.1 is given by the applying of the *chain rule* to the joint probability $\mathbb{P}(X_1 = x_1 \cdots X_n = x_n)$.

Definition 2.1.8 (Chain rule) The *chain rule* is a repeated application of the definition of conditional probability to the joint probability of a set of variables. Consider an indexed set of variables $A_1, ..., A_n$. We can apply, to the joint probability, the definition of conditional probability to obtain:

$$\mathbb{P}(A_n, \cdots, A_1) = \mathbb{P}(A_n | A_1, \cdots, A_{n-1})$$

Repeating this process with each final term creates the product:

$$\mathbb{P}(A_n, \cdots, A_1) = \prod_{i=1}^{n} \mathbb{P}(A_i | \bigcap_{j=1}^{i-1} (A_j) \ )$$

that is the chain rule.

We can obtain equation:

$$\mathbb{P}(X_1, \cdots, X_n) = \prod_{i=1}^{n} \mathbb{P}(X_i | Pa(X_i))$$

by applying the chain rule to the joint probability of $X_1, \cdots, X_n$ where the X_i are ordered in such a way that the variables representing the causes precede the variables representing the consequences. We then remove, in the factors $\mathbb{P}(A_i | \bigcap_{j=1}^{i-1}(A_j))$ of the product, the variable$A_{\bar{j}}$ that are conditional independent of A_i given the other A_j.

Observation 2.3 Each variable is conditionally independent of its non-descendants, given its parents. For example, in Example 2.1, C is independent of D given the value of B.

Another important independence property is that a node is conditionally independent of all other nodes in the network, given its parents, children, and children's parents (this subset of nodes is called *Markov blanket*). ■

2.1.4 Inference

We now introduce *inference* in a Bayesian networks, given information E, called *evidence*.

Inference tasks

The main tasks of inference in Bayesian network are:

- *Belief updating (Bel)*:
  This task look for the probability v^*, define as follows.
  - ◇ *Single marginal*, given a variable X:
    $v^* = \mathbb{P}(X = x|E = e)$.
  - ◇ *Subjoint*, given a subset of variables S:
    $v^* = \mathbb{P}(S = s|E = e)$.
- *MPE*:
  Finding *most probable explanation (MPE)* is finding the most probable assignment for all the variables, according to the evidence:

$$(x_1^*, \cdots, x_n^*) = \arg\max_{(x_1,\cdots,x_n)} \mathbb{P}(X_1 = x_1, \cdots, X_n = x_n|E = e).$$

- *MAP*:
  Finding the *maximum a-posteriori hypothesis (MAP)* is used when we are interested on a subset of variables. We suppose to have the set of n variables `Var` and we are interested on a subset of m variables $A_1, \cdots, A_m$ of `Var`. The set $\texttt{Var} \setminus \{A_1, \cdots, A_m\}$ is equal to $\{B_1, \cdots, B_l\}$. So, MAP finds the most probable assignment for $A_1, \cdots, A_m$ summing over all possible values of the set $\{B_1, \cdots, B_l\}$, according to the evidence.

$$(a_1^*, \cdots, a_m^*) = \arg\max_{(a_1,\cdots,a_m)} \sum_{\forall(b_1,\cdots,b_l)} \mathbb{P}(A_1 = a_1, \cdots, A_m = a_m, B_1 = b_1, \cdots B_l = b_l|E = e).$$

Inference algorithms

The inference method that we will use in this thesis is *Variable Elimination*, also called *Bucket Elimination* [95] [29] [31].

The idea, like that of direct application of factoring, is simple. First, establish an elimination ordering for the variables not in the result. Then, for each variable in the elimination ordering in turn:

1. remove and gather all distributions indexed by the parameter;
2. combine them, marginalizing over the parameter;
3. and place the result in the distribution set.

We consider the following *Dechter's algorithms* [31]: algorithms obtained by combining the procedure of bucket elimination with the query of Belief updating, MPE and MAP.

- Dechter's algorithm for *Bel*:

Algorithm elim-bel

Input: A belief network $BN = \{P_1, \cdots, P_n\}$; an ordering of the variables $d = X_1, \cdots, X_n$; evidence e.
Output: The belief in $X_1 = x_1$.
1. Initalize: Generate an ordered partition of the conditional probability matrices, $bucket_1, \cdots, bucket_n$ where $bucket_i$ contains all matrices whose highest variable is X_i. Put each observed variable in its bucket. Let $S_1, \cdots, S_j$ be the subset of variables in the processed bucket on which matrices (new or old) are defined.
2. Backward: For $p \leftarrow n$ downto 1, do
for all the matrices $\lambda_1, \cdots, \lambda_j$ in $bucket_p$, do
$\diamond$ **if** $X_p = x_p$ appears in $bucket_p$ (bucket with observed variable), assign $X_p = x_p$ to each λ_i and then put each resulting function in appropriate bucket.
$\diamond$ **else** $U_p \leftarrow \bigcup_{i=1}^{j}(S_i \setminus \{X_p\})$. Generate $\lambda_p = \sum_{X_p} \prod_{i=1}^{j} \lambda_i$ and add λ_p to the bucket of largest-index variable in U_p.
3. Return: $Bel(x_1) = \alpha \mathbb{P}(x_1) \cdot \prod_i \lambda_i(x_1)$ (where the λ_i are in $bucket_1$, α is a normalizing constant).

- Dechter's algorithm for *MPE*:

Algorithm elim-mpe

Input: A belief network $BN = \{P_1, \cdots, P_n\}$; an ordering of the variables $d = X_1, \cdots, X_n$; evidence e.
Output: The most probable assignment.
1. Initalize: Generate an ordered partition of the conditional probability matrices, $bucket_1, \cdots, bucket_n$ where $bucket_i$ contains all matrices whose highest variable is X_i. Put each observed variable in its bucket. Let $S_1, \cdots, S_j$ be the subset of variables in the processed bucket on which matrices (new or old) are defined.
2. Backward: For $p \leftarrow n$ downto 1, do
for all the matrices $\lambda_1, \cdots, \lambda_j$ in $bucket_p$, do
$\diamond$ **if** $X_p = x_p$ appears in $bucket_p$ (bucket with observed variable), assign $X_p = x_p$ to each λ_i and then put each in appropriate bucket.
$\diamond$ **else** $U_p \leftarrow \bigcup_{i=1}^{j}(S_i \setminus \{X_p\})$. Generate $\lambda_p = \max_{X_p} \prod_{i=1}^{j} \lambda_i$ and $x_p^o = \arg\max_{X_p} \lambda_p$. Add λ_p to the bucket of largest-index variable in U_p.
3. Return: Assign values in the ordering d using the recorded functions x^o in each bucket.

- Dechter's algorithm for *MAP*:

Algorithm elim-map

Input: A belief network $BN = \{P_1, \cdots, P_n\}$; a subset of variables $A = \{A_1, \cdots, A_k\}$; an ordering of the variables d in which the A's are first in the ordering; evidence e.
Output: The most probable assignment $A = a$.
1. Initalize: Generate an ordered partition of the conditional probability matrices, $bucket_1, \cdots, bucket_n$ where $bucket_i$ contains all matrices whose highest variable is X_i.
2. Backward: For $p \leftarrow n$ downto 1, do
for all the matrices $\lambda_1, \cdots, \lambda_j$ in $bucket_p$, do
$\diamond$ **if** $X_p = x_p$ appears in $bucket_p$ (bucket with observed variable), assign $X_p = x_p$ to each λ_i and then put each in appropriate bucket.
$\diamond$ **else** $U_p \leftarrow \bigcup_{i=1}^{j}(S_i \setminus \{X_p\})$. If X_p not in A and if $bucket_p$ contains new functions, then $\lambda_p = \sum_{X_p} \prod_{i=1}^{j} \lambda_i$; else, $X_p \in A$ and $\lambda_p = \max_{X_p} \prod_{i=1}^{j} \lambda_i$ and $a^o = \arg\max_{X_p} \lambda_p$. Add λ_p to the bucket of largest-index variable in U_p.
3. Return: Assign values, in the ordering $d = A_1, \cdots, A_k$ using the information a^o recorded in each bucket.

We now introduce some definitions that we will need in what follows.

Definition 2.1.9 The *moral graph* G^M of a directed graph G is formed by connecting nodes (in G) that have a common child, and then making all edges in the graph undirected.

Definition 2.1.10 The *width* $w_o(X)$ of a *variable* X in graph G along the ordering o is the number of nodes preceding X in the ordering and connected to X (earlier neighbors).

The *width* w_o of a *graph* is the maximum width $w_o(X)$ among all nodes.

Definition 2.1.11 The *induced graph* G' along the ordering o is obtained by recursively connecting earlier neighbors of each node, from last to the first in the ordering.

Definition 2.1.12 Given an ordering o for the variables, the width of the induced graph G' is called the *induced width* of the graph G and it is denoted as w_o^*.

Theorem 2.1.1 Complexity of VE-inference is $O(n\exp(w_o^* + 1))$ in time and $O(n\exp(w_o^*))$ in space, when w_o^* is the induced-width of the moral graph along the ordering o for the variables of the graph G.

The hard problem of this computation is identifying an order in which to marginalize the variables not in the query. Once that has been determined, it is simple (polynomial time) to VE-inference. Finding the order o^* such that:

$$o^* = \operatorname*{argmin}_o w_o^*,$$

is NP-hard [31]. There are some greedy algorithms to compute it, as *min-fill* ([32]) that works well.

Inference is also NP-hard in general case ([29] and [31]).

2.2 CP-nets

CP-nets [11] are a graphical model for compactly representing conditional and qualitative preference relations. CP-nets are sets of *ceteris paribus* preference statements. For instance, the statement *"I prefer red wine to white wine if meat is served."* asserts that, given two meals that differ *only* in the kind of wine served *and* both containing meat, the meal with red wine is preferable to the meal with white wine.

2.2.1 Preferences and orderings

Before introducing the CP-net formalism, we introduce the notion of ordering and preference.

Orderings

Let us consider the concepts of *total order*, *preorder* and *partial order*.

Definition 2.2.1 Suppose that P is a set and that $\succeq$ is a binary relation on P.

Then $\succeq$ is a *total order* if it is:

- total;
- antisymmetric;
- transitive.

Thus, $\forall a, b$ and $c \in P$, we have that:

- $a \succeq b$ or $b \succeq a$ (totality);
- if $a \succeq b$ and $b \succeq a$ then $a = b$ (antisymmetry);
- if $a \succeq b$ and $b \succeq c$ then $a \succeq c$ (transitivity).

Definition 2.2.2 Suppose that P is a set and that $\succeq$ is a binary relation on P.

Then $\succeq$ is a *preorder* if it is:

- reflexive;
- transitive.

Thus, $\forall a, b$ and $c \in P$, we have that:

- $a \succeq a$ (reflexivity);
- if $a \succeq b$ and $b \succeq c$ then $a \succeq c$ (transitivity).

Definition 2.2.3 Suppose that P is a set and that $\succeq$ is a binary relation on P.

Then $\succeq$ is a *total preorder* if it is:

- total;
- reflexive;
- transitive.

Definition 2.2.4 Suppose that P is a set and that $\succeq$ is a binary relation on P.

Then $\succeq$ is a *partial order* if it is:

- reflexive;
- antisymmetric;
- transitive.

Thus, $\forall a, b$ and $c \in P$, we have that:

- $a \succeq a$ (reflexivity);
- if $a \succeq b$ and $b \succeq a$ then $a = b$ (antisymmetry);
- if $a \succeq b$ and $b \succeq c$ then $a \succeq c$ (transitivity).

All partial orders are preorders, but preorders are more general.

Definition 2.2.5 Suppose that P is a set and that $\succeq$ is a binary relation on P.

Then $\succeq$ is a *strict weak order* if it is:

- irreflexive;
- antisymmetric;
- transitive;
- transitive of incomparability.

Thus, $\forall a, b$ and $c \in P$, we have that:

- $a \not\succ a$ (irreflexive);
- if $a \succeq b$ and $b \succeq a$ then $a = b$ (antisymmetry);
- if $a \succeq b$ and $b \succeq c$ then $a \succeq c$ (transitivity);
- if $a \bowtie b$ and $b \bowtie c$ then $a \bowtie c$ (transitivity of incomparability).

Qualitative Preferences

We introduce in what follows the notion of qualitative preferences.

Definition 2.2.6 A *preference ranking* is a total preorder ($\succeq$) over a set of values: $v_1 \succeq v_2$ means that outcome v_1 is *equally or more preferred* to the decision maker than v_2. We use $v_1 \succ v_2$ to denote the fact that outcome v_1 is *strictly more preferred* by the decision maker than v_2.

We will use the terms preference ordering and relation interchangeably with ranking.

We assume a set of features (or variables) $\mathtt{Var} = \{X_1, \cdots, X_n\}$ over which the decision maker has preferences. Each variable X_i is associated with a domain $\mathscr{D}(X_i) = \{x_{i_1}, \cdots, x_{i_{n_i}}\}$ of values it can take.

Definition 2.2.7 An *assignment* x of values to a set $X \subseteq \mathtt{Var}$ of variables is a function that maps each variable in X to an element of its domain. If $X = \mathtt{Var}$, x is a *complete assignment*, otherwise x is called a *partial assignment*.

Given a problem defined over n variables with domains $\mathscr{D}(X_1), \cdots, \mathscr{D}(X_n)$, there are $|\mathscr{D}(X_1)| \cdot |\mathscr{D}(X_2)| \cdot \cdots \cdot |\mathscr{D}(X_n)|$ assignments (exponential number).

We denote the set of all assignments to $X \subseteq \mathtt{Var}$ by $Asst(X)$. If x and y are assignments to disjoint sets X and Y, respectively ($X \cap Y = \emptyset$), we denote the combination of x and y by xy. If $X \cup Y = \mathtt{Var}$, we call xy a *completion* of assignment x. We denote by $Comp(x)$ the set of completions of x. Complete assignments correspond directly to the outcomes over which a user have to make a decision. For any outcome o, we denote by $o \restriction_X$ the value $x \in \mathscr{D}(X)$ assigned to variable X by that outcome.

Definition 2.2.8 A set of variables X is *preferentially independent* of its complement $Y = \mathtt{Var} \setminus X$ iff , $\forall x_1, x_2 \in Asst(X)$ and $\forall y_1, y_2 \in Asst(Y)$, we have:

$$x_1 y_1 \succeq x_2 y_1 \text{ iff } x_1 y_2 \succeq x_2 y_2.$$

If the relation above holds, we say x_1 is *preferred* to x_2 *ceteris paribus*.

We define conditional preferential independence analogously:

Definition 2.2.9 Let X, Y, and Z be nonempty sets that partition $\mathtt{Var}$. X is *conditionally preferentially independent* of Y given an assignment z to Z iff, $\forall x_1, x_2 \in Asst(X)$ and $\forall y_1, y_2 \in Asst(Y)$, we have:

$$x_1 y_1 z \succeq x_2 y_1 z \text{ iff } x_1 y_2 z \succeq x_2 y_2 z.$$

In other words, X is preferentially independent of Y when Z is assigned z. If X is conditionally preferentially independent of Y $\forall z \in Asst(Z)$, then X is conditionally preferentially independent of Y given the set of variables Z.

2.2.2 Representation of CP-nets

The model of CP-nets [11] is similar to a Bayesian network on the surface; however, the nature of the relation between nodes within a network is generally quite weak. CP-nets representation is used to capture statements of qualitative conditional preferential independence.

Definition 2.2.10 A *CP-net (conditional preference networks)* over variables (features) $\mathtt{Var} = \{X_1, \cdots, X_n\}$ (with finite domains $\mathscr{D}(X_1), \cdots, \mathscr{D}(X_n)$), is a direct graph G (*dependency graph*) over $X_1, \cdots, X_n$ in which:

- each node corresponds to a variable $X_i \in \mathtt{Var}$;
- a set of direct edges connects pairs of nodes (if there is an edge from node X_i to node X_j, X_i is said to be a *parent* of X_j, $X_i \in Pa(X_j)$);
- each node X_i has a *conditional preference table* (*CP-table*) that associates a total order $\succ_u^i$ with each instantiation $u \in Asst(U)$ of X_i's parents $Pa(X_i) = U$.

An *acyclic CP-net* is one in which the dependency graph is acyclic.

Given $Pa(X_i)$ we have that X_i is conditionally preferentially independent of $\mathtt{Var} \setminus (Pa(X_i) \cup \{X_i\})$.

Observation 2.4 Suppose we have CP-nets in which the features have domains of cardinality d. So, in general, a CP-table for a variable with k parents will contain d^k orderings. A feature with no parents has only one row representing the ordering over its domain. ■

We now show an example of CP-net.

■ **Example 2.2** Consider a CP-net whose features are A, B, C, and D, with binary domains $\mathscr{D}(F) = \{f, \bar{f}\}$ if F is the name of the feature (as shown in Figure 2.2).

The preference statements are as follows:

- Feature A:

ordering for A
$a \succ \bar{a}$

- Feature B:

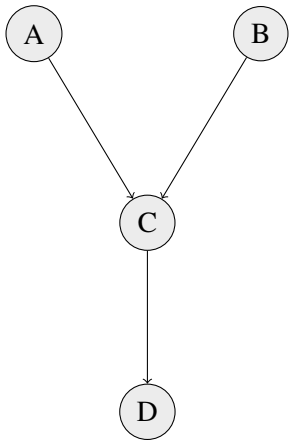


Figure 2.2: CP-net (Example 2.2)

ordering for B
$b \succ \bar{b}$

- Feature C:

A & B	ordering for C
$a\ b$	$c \succ \bar{c}$
$\bar{a}\ b$	$\bar{c} \succ c$
$\bar{a}\ \bar{b}$	$c \succ \bar{c}$
$a\ \bar{b}$	$\bar{c} \succ c$

- Feature D:

C values	ordering for D
c	$d \succ \bar{d}$
$\bar{c}$	$\bar{d} \succ d$

Here, statement $a \succ \overline{a}$ represents the unconditional preference for $A = a$ over $A = \overline{a}$, while statement $c : d \succ \overline{d}$ states that $D = d$ is preferred to D=$\overline{d}$, given that $C = c$. ∎

The semantics of CP-nets depends on the notion of *worsening flip*.

Definition 2.2.11 A *worsening flip* is a change in the value of a variable to a less preferred value according to the cp-statement for that variable.

For example, in the CP-net in the Example 2.2, passing from the complete assignment $abcd$ to $ab\bar{c}d$ is a worsening flip since c is better than $\overline{c}$ given a and b.

Definition 2.2.12 One outcome α is *better* than another outcome β (written $\alpha \succ \beta$) iff there is a chain of worsening flips from α to β.

This definition induces a preorder over the outcomes, which is a partial order if the CP-net is acyclic.

The two main reasoning task concerning CP-nets are *outcome optimization* and *outcome comparison*. We analyse in detail this two tasks in the two following sections.

2.2.3 Optimal outcomes of CP-nets

Given a set of outcomes, an outcome o is optimal if and only if does not exists any other outcome o' such that $o' \succ o$.

Definition 2.2.13 Given a CP-net inducing a preorder $\succ$ over the set of outcomes, an *optimal outcome* is an outcome that is maximal for $\succ$ in the set of outcomes.

There could exists more then one optimal outcome for the same CP-net, but if the CP-net has an acyclic dependency graph there exists a unique optimal outcome.

Goldsmith et al. [52] prove that finding an optimal outcome of a general CP-net is PSPACE-complete. However, in acyclic CP-nets, there is only one optimal outcome and this can be found in linear time by sweeping through the CP-net [11], assigning the most preferred values in the preference tables.

More rigorously:

Definition 2.2.14 The sweep forward procedure, in acyclic CP-nets, to achieve the optimal value is as follows:

- *Step* 0: We choose for the CP-net's independent features (feature that do not have parents in the dependency graph) the value that it is ranked first in the ordering of the domain's values.
- Step i: For each feature that it is not yet assigned and that has the value's assignments of all the parents, we choose the value that it is most preferred (so ranked first) in the ordering of the domain's values (in the rows corresponding to that particular assignments for the parents's values).

■ **Example 2.3** We will show the sweep forward procedure applied to the CP-net (in Figure 2.2) of Example 2.2. First we have to choose the value for A and B because are independent features. Thus we have $A = a$ and $B = b$.

Then we can process C, assign $C = c$, and finally $D = d$. ■

In the general case the optimal outcome coincides with the solutions of a constraint problem obtained replacing each cp-statement with a constraint. For example, the following cp-statement (derived from the statements $ab : c \succ \bar{c}$ and $\bar{a}\bar{b} : c \succ \bar{c}$ in Example 2.2) $(a \wedge b) \vee (\overline{a} \wedge \overline{b}) : c \succ \overline{c}$ would be replaced by the constraint $(a \wedge b) \vee (\overline{a} \wedge \overline{b}) \Rightarrow c$.

2.2.4 Outcome comparison

Two outcomes o and o' can stand in one of three possible relations with respect to a CP-net $\mathscr{C}$:

- o dominates o': $\mathscr{C} \models o \succ o'$
- o' dominates o: $\mathscr{C} \models o \succ o'$
- o and o' are incomparable: $\mathscr{C} \models o \not\succ o' \wedge \mathscr{C} \models o' \not\succ o$ written also as $\mathscr{C} \models o \bowtie o'$

There are two different ways in which we can compare two outcomes using a CP-net: *dominance queries* and *ordering queries*

Dominance queries

A dominance query represents preferential comparison between outcomes.

Definition 2.2.15 Given a CP-net $\mathscr{C}$ and a pair of outcomes o and o', a *dominance query* returns *True* if $\mathscr{C} \models o \succ o'$, *False* otherwise.

$\mathscr{C} \models o \succ o'$ means that there exists a worsening path (chain of worsening flips) from o' to o.

Proposition 2.2.1 If there is an improving flipping sequence for CP-net $\mathscr{C}$ from o to o' then $\mathscr{C} \models o \succ o'$ and if $\mathscr{C}$ is an acyclic CP-net and there is no improving flipping from o to o' then $\mathscr{C} \not\models o' \succ o$.

Boutilier et al. [11] prove that evaluating a dominance query for binary-valued tree structured CP-nets has complexity $O(n^2)$, where n is the number of features. This complexity result is improved by Bigot et al. [9] that prove that dominance for tree-shaped CP-nets takes $O(n)$ time. Boutilier et al. [11] prove that dominance is polynomial also for polytrees but NP-complete for acyclic CP-nets that are directed path singly connected, and NP-complete for CP-nets with a bounded number of paths between two nodes. Goldsmith et al. [52] prove that dominance for a general structure is PSPACE-complete using the fact that finding a flipping sequence between two comparable outcome can be also seen as a planning problem.

Ordering queries

An ordering query represents preferential comparison between outcomes in a weaker form with respect to dominance.

Definition 2.2.16 Given a CP-net $\mathscr{C}$ and a pair of outcomes o and o', a *ordering query* returns *True* if $\mathscr{C} \not\models o' \succ o$, *False* otherwise. If $\mathscr{C} \not\models o' \succ o$, this means that there exists an preference ordering consistent with $\mathscr{C}$ that satisfies $o \succ o'$.

Ordering queries with respect to acyclic CP-nets can be answered in time linear in the number of variables [11], with a computational complexity of $O(n)$ where n is the number of features.

2.2.5 Random generation of CP-nets

The problem of an automatic generation of CP-nets is very important because it is fundamental to perform fair experiments that require simulated datasets.

Randomly generating CP-nets in a uniform way is a difficult problem [2]. A CP-net can model a subset of partial orders, and the set of all the possible partial orders could be exponentially large since we are considering combinatorial domains. Also counting the number of partial orders induced by a CP-net is a well-known difficult problem [2].

Allen et al. [2] propose a method to generate CP-nets of a given number of nodes, uniformly at random. The method consists into an heuristic that follow the idea of counting (therefore also ranking) and generating CP-nets given their ranks, considering the binary case, and with exponential complexity.

2.3 Variants and Extensions of CP-nets

CP-nets are deeply studied and widely used, thus many variants and generalizations have been developed. The main variants are described in the following sections.

2.3.1 TCP-nets

TCP-nets (Tradeoffs-enhanced CP-nets) [15] are an extension of CP-nets that permit the representation of the notion of *relative importance* and *conditional relative importance*. A variable X is relatively more important with respect to a variable Y ($X \rhd Y$) if it is more important that the value of X be high than that the value of Y be high. For instance "The length of the journey is more important to me than the choice of airline". A variable X is relatively more important with respect to a variable Y given an assignment of a variable Z ($Z = z_0 : X \rhd Y, Z = z_1 : Y \rhd X$) if, given that particular assignment of $Z = z_0$, it is more important that the value of X be high than that the value of Y be high. For instance "The length of the journey is more important to me than the choice of airline provided that I am lecturing the following day. Otherwise the choice of airline is more important". This formulation introduces a dependency graph in which we have one node for each variable and three type of edges. The first type represents preferential dependencies, the second type represents relative importance relations and the third one, undirected, represents conditional importance relations. Each node is associated with a conditional preference table (CPT) as in CP-nets and the conditional importance edges to a *CI-table (conditional importance table)* that express the conditional importance statements.

Formally the definition of a TCP-net is:

Definition 2.3.1 A TCP-net is a tuple $(\texttt{Var}, cp, i, ci, cpt, cit)$ such that:

- $\texttt{Var}$ is a set of n nodes ($\texttt{Var} = \{X_1, \cdots, X_n\}$);
- cp is a set of directed edges: *conditional preference edges*;
- i is a set of directed edges: *importance edges*;
- ci is a set of undirected edges: *conditional importance edges*;
- cpt are the conditional preference tables (defined as in CP-nets);
- cit are the conditional importance tables associated with the ci edges.

■ **Example 2.4** Given five features A, B, C and D a possible TCP-net is described in Figure 2.3. In this TCP-net we have three cp-edges: (A,C) and (A,D); one i-edge: (A,B) and one ci-edge: (C,D). The relative importance of C and D depends on the assignments of the variables A and B, unless when A and B are assigned $\bar{a}\bar{b}$, when it is not specified the relative importance between C and D.

The resulting induced preference order is described by the following statements, given the variables R, R_1 and $R_2 \in Dom(B)$, S, S_1 and $S_2 \in Dom(C)$ and T, T_1 and $T_2 \in Dom(D)$:

- $\{aR_1ST \succ \bar{a}R_2ST\}$;
- $\{abcT_1 \succ ac\bar{b}T_2\}$ and $\{abSd \succ acS\bar{d}\}$;
- $\{a\bar{b}cT \succ a\bar{b}\bar{c}T\}$ and $\{a\bar{b}S_1\bar{d} \succ a\bar{b}S_2d\}$;
- $\{\bar{a}b\bar{c}T \succ \bar{a}bcT\}$ and $\{\bar{a}bS_1\bar{d} \succ \bar{a}bS_2d\}$;
- $\{\bar{a}\bar{b}\bar{c}T \succ \bar{a}\bar{b}cT\}$ and $\{\bar{a}\bar{b}S\bar{d} \succ \bar{a}\bar{b}Sd\}$.

■

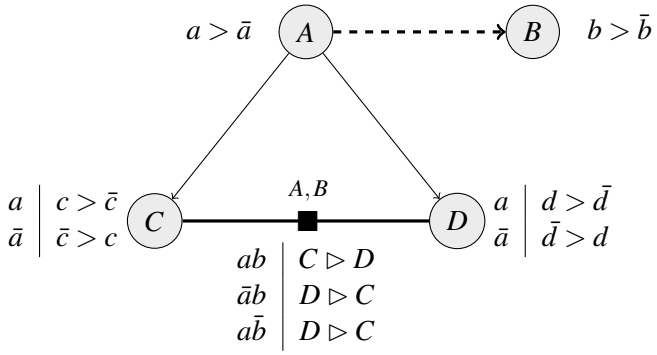


Figure 2.3: A TCP-net (Example 2.4)

2.3.2 CP-trees

A CP-tree [112] is a particular form to represent users preferences. A CP-tree is made by *CP-nodes*.

Definition 2.3.2 Given a set of variables $\texttt{Var} = \{X_1, \cdots, X_n\}$ a CP-node r is a tuple

$$\langle A_r, a_r, Y_r, \geq_r \rangle$$

where $A_r \subseteq \texttt{Var}$ A is a set of variables, a_r is an assignment to the variables in A_r, $Y_r \subseteq \texttt{Var} \setminus A_r$ is a non-empty set of variables and $\geq_r$ is a weak order on the assignments of the variables in Y_r.

Definition 2.3.3 A CP-tree corresponds to a directed tree in which each node is a CP-node and the ancestors of a node r (the nodes on the path from root to the parent) are the set A_r. Given an edge $r \to r'$, this edge is associated with an assignment y to variables Y_r, and this assignment is different from all the assignment to Y_r associated with any other edges from node r. Then $A_{r'} = A_r \cup Y_r$ and and $a_{r'}$ is a_r extended with the assignment $Y_r = y$. $A_{root} = \emptyset$.

■ **Example 2.5** In Figure 2.4 we show an example of a CP-tree over five variables $X_1, \cdots, X_5$ with binary domains $Dom(F) = \{f, \bar{f}\}$

The resulting induced preference order is described by the following statements, given the variables R_1 and $R_2 \in Dom(B)$, S_1 and $S_2 \in Dom(C)$ and T_1 and $T_2 \in Dom(D)$:

- $\{x_1 R_1 S_1 T_1 \succ \bar{x}_1 R_2 S_2 T_2\}$;
- $\{x_1 \bar{x}_2 x_3 T_1 \succ x_1 x_2 \bar{x}_3 T_2\}$;
- $\{\bar{x}_1 R_1 x_3 T_1 \succ \bar{x}_1 R_2 \bar{x}_3 T_2\}$;
- $\{x_1 \bar{x}_2 x_3 x_4 \succ x_1 \bar{x}_2 x_3 \bar{x}_4\}$;
- $\{x_1 x_2 \bar{x}_3 \bar{x}_4 \succ x_1 x_2 \bar{x}_3 x_4\}$.

■

Optimality and Dominance

CP-trees define a weak order and thus a notion of dominance:

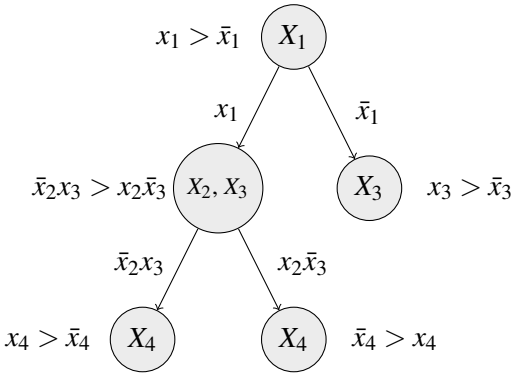


Figure 2.4: A CP-tree (Example 2.5)

Definition 2.3.4 Given a CP-tree $\mathscr{C}$, the relation $\succeq_{\mathscr{C}}$ is defined as: given two outcomes o and o', then $o \succeq_{\mathscr{C}} o'$ if and only if

- $o \restriction_{Y_r} \geq_r o' \restriction_{Y_r}$;
- o and o' are represented by two nodes of the tree l and l';
- r is the deepest node that is in common between the two paths: from root to l and from the root to l'. r represents the set of variables Y_r;
- o and o' both extend a_r, that is defined by the labels of the path from the root to the node r

Both dominance and optimality can be computed in polynomial time in the number of nodes of the tree [112].

■ **Example 2.6** 2.5 The optimal outcome of Example 2.5 is $x_1\bar{x}_2x_3x_4$. Given two outcomes $o = \bar{x}_1\bar{x}_2x_3x_4$ and $o' = \bar{x}_1x_2\bar{x}_3\bar{x}_4$, the answer to the dominance query $o \succ o'$? is *True* because o and o' have the same path: $X_1 \rightarrow_{\bar{x}_1} X_3$ and $x_3 > \bar{x}_3$. ■

2.3.3 CI-net

CI-nets (Conditional Importance networks) [14] represent ordinal preferences over the set of alternatives. The definition of CI-nets depends on the notion of *conditional importance statement*:

Definition 2.3.5 A *conditional important statement (CI-statement)* on a set of variables `Var` is a quadruple $\gamma = (S^+, S^-, S_1, S_2)$ (where $S*$ are subset of the set of variables `Var` that are pairwise disjoint), written as:

$$S^+, S^- : S_1 \rhd S_2$$

and S^+ are called *positive preconditions*, S^- are called *negative preconditions* and S_1, S_2 are called *compared sets*.

CI-net consider a set `Var` of binary variables, with domain in $\{0,1\}$. Each variable correspond to an item and if the variable is assigned to 1 this means that the user obtained that item, otherwise is equal to 0.

Informally a CI-statement can be read as: " Given all the items in S^+ (all the variables in S^+ are assigned to 1) and any items in S^- (all the variables in S^- are assigned to 0), I prefer obtaining all the items in S_1 (to assign to 1 to all the variables in S_1) to obtaining all those in S_2 (to assign to 1 to all the variables in S_1)".

Definition 2.3.6 A *CI-net* on a set of variables $\mathtt{Var}$ is a set of conditional importance statements.

■ **Example 2.7** Given four variables A, B, C and D, with binary domains $Dom(F) = \{f, \bar{f}\}\ \forall F \in \{A,B,C,D\}$, a possible CI-net is defined by the following set of CI-statements:

$$\{a : d \rhd bc, a\bar{d} : b \rhd c, d : b \rhd c\}$$

■

A CI-net induces a *preference relation* $\succ$: a strict partial order (irreflexive, asymmetric and transitive binary relation) over the set $2^{\mathtt{Var}}$ where $\mathtt{Var}$ is the set of variables. CI-nets can express all strict monotonic preference relations.

The semantics of a CI-net is based on the notion of *worsening flip*.

Definition 2.3.7 Given a CI-net C on a set of variables $\mathtt{Var}$ and given two subsets V_1 and $V_2 \subseteq \mathtt{Var}$ there is a worsening flip from V_1 to V_2 if $V_2 \subsetneq V_1$ and there is a CI-statement $S^+, S^- : S_1 \rhd S_2$ such that if $\bar{S} = \mathtt{Var} \setminus (S^+ \cup S^- \cup S_1 \cup S_2)$ then:

- $S_1 \cup S^+ \subseteq V_1$ and $S_2 \cup S^+ \subseteq V_2$
- $V_1 \cap S^- = V_2 \cap S^- = V_1 \cap S_2 = V_2 \cap S_1 = \emptyset$
- $\bar{S} \cap V_1 = \bar{S} \cap V_2$

Worsening flips define a preference graph on $2^{\mathtt{Var}}$.

Satisfiability

Given a preference relation $\succ$ over $2^{\mathtt{Var}}$, it satisfies a CI-statement $S^+, S^- : S_1 \rhd S_2$ if for every $S' \subseteq \mathtt{Var} \setminus (S^+ \cup S^- \cup S_1 \cup S_2)$, we have $S' \cup S^+ \cup S_1 \succ S' \cup S^+ \cup S_2$. A preference relation $\succ$ over $2^{\mathtt{Var}}$ satisfies a CI-net if $\succ$ satisfies each CI-statements in the CI-net and $\succ$ is monotonic.

Definition 2.3.8 A CI-net C is called *satisfiable* if there exists a preference relation satisfying C.

Proposition 2.3.1 Any CI-net with an acyclic dependency graph is satisfiable.

Testing satisfiability for CI-nets is PSPACE-complete.

Dominance

Definition 2.3.9 Given a CI-net C and its preference relation $\succ$, a dominance query corresponds to aswer to the question:

$$V_1 \succ_{\mathtt{Var}} V_2? \quad \text{with } V_1, V_2 \subseteq \mathtt{Var}.$$

Dominance in satisfiable CI-nets is PSPACE-complete, but if we restrict to satisfiable SCI-nets (CI-nets precondition-free and singleton-comparing CI-statements) dominance is in P.

2.3.4 UCP-nets

UCP-nets (Utility Conditional Preference networks) [13] are an extension of CP-nets in which the preferences are expressed in a quantitative formulation, and not only qualitative as in CP-nets. The idea of this extension is to combine generalized additive models (GAI-networks) and qualitative preferences described with CP-nets, using the notion of dependency graph to represent the relation between the variables.

The preferences of the user over each variables are expressed in a table that contains the user utility values of the values in the variable domain. A CP-statement of the form $x : y > \bar{y}$ is represented in UCP-nets by $f_Y(y|x) = u_1$, $f_Y(\bar{y}|x) = u_2$ and $u_1 > u_2$ where f_Y is the utility function of the variable Y. Thus each feature X is associated with a function $f_X(v,u)$ where v is a value in the domain of X and u is an assignment of the X parents nodes $Pa(X)$, that corresponds to the utility function of the feature X. The utility function $u(o)$ of a complete outcome o defined over the variables $X_1, \cdots, X_n$ corresponds to:

$$u(o) = \sum_{i=1}^{n} f_{X_i}(o \restriction_{X_i}, o \restriction_{Pa(X_i)})$$

Definition 2.3.10 Given a utility function $u(X_1, \cdots, X_n)$ defined over a set of variables $X_1, \cdots, X_n$, a *UCP-network* is a directed acyclic graph (DAG) G over $X_1, \cdots, X_n$ and a set of factors $f_{X_i}(X_i, Pa(X_i))$, such that

$$u(X_1, \cdots, X_n) = \sum_{i=1}^{n} f_{X_i}(X_i, Pa(X_i))$$

and G is a valid CP-net for the the preference relation $\succeq$ induced by $u(X_1, \cdots, X_n)$. This means that $\succeq$ can be written as a valid set of CP-statement for each feature X_i.

■ **Example 2.8** In Figure 2.5 we show an example of a UCP-net defined on 4 features $X_1, \cdots, X_4$. Considering the outcome $x_1 x_2 \bar{x}_3 \bar{x}_4$ we can compute the following utility function:

$$u(x_1 x_2 \bar{x}_3 \bar{x}_4) = f_{X_1}(x_1) + f_{X_2}(x_2) + f_{X_3}(\bar{x}_3, x_1, x_2) + f_{X_4}(\bar{x}_4, \bar{x}_3) = 5 + 6 + 0.1 + 0.3 = 11.4$$

■

Also for UCP-nets we can consider the two main tasks: outcome optimization and dominance queries.

Optimality

Definition 2.3.11 Given a UCP-net, an *outcome optimization query corresponds to find the outcome that has the maximum utility given some partial assignment x, i.e. solve this problem:*

$$\arg\max\{u(v) : v \in Comp(x)\}$$

where $Comp(x)$ correspond to the set of completion of the partial assignment x.

In a general UCP-net the computational complexity of this task is a hard problem, but if we

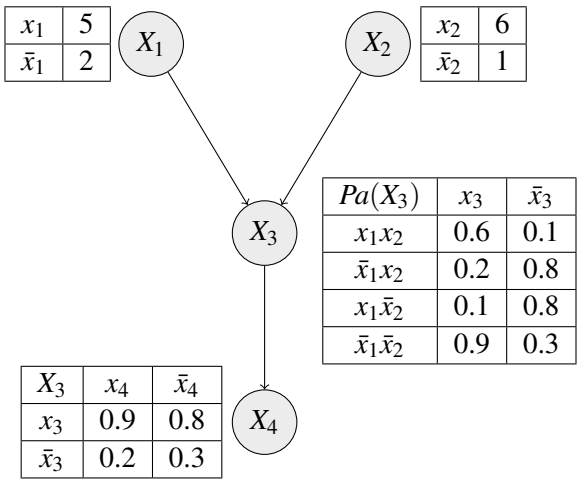


Figure 2.5: A UCP-net (Example 2.8)

use *tradeoff weights* (for details see [13]) and we can use the sweep forward procedure, and the complexity become polynomial in the number of variables.

Dominance

Definition 2.3.12 Given a UCP-net, an *dominance query corresponds to understand which outcome has the higher utility , i.e. solve this problem:*

$$u(v_1) > u(v_2)?$$

Computing dominance is easy for UCP-nets, since it correspond to sum n elements.

2.3.5 mCP-nets

mCP-nets (multi-agent Conditional Preference networks) [90] are an extension of CP-nets formalism to model and handle the preferences of multiple agents. A mCP-net is a set of m partial CP-nets (CP-nets with partial or missing information) which can share some features. A mCP-net dependency graph is obtained by combining the graphs of the partial CP-nets having one occurrence of each shared feature. A feature of an agent a can be

- *shared* between the agents: these features are ranked in the CP-net of a and also in at least one other of the partial CP nets;
- *visible* for the agent a: these features can be used as precondition in the cp-statements, but not ranked in a;
- *private* if is ranked only in the CP-net of a and is not visible to the other agents CP-nets.

■ **Example 2.9** Given two agents that define their preferences over three features A, B and C, in Figure 2.6 we show a possible mCP-net of the two agents. The features A_1, B_1,B_2, C_1, C_2 are private, the feature D is shared between the two agents and the feature A_2 is visible for agent 1 and

used as precondition in the cp-statements of the feature D.

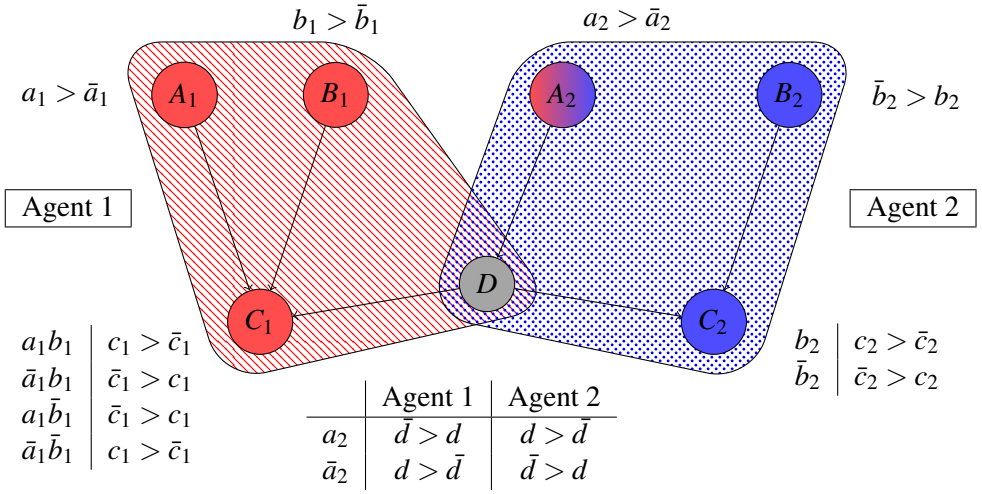


Figure 2.6: A mCP-net (Example 2.9)

■

Optimality and Dominance

Optimality and dominance tasks are seen as voting procedures over the CP-nets that composing the mCP-net, by querying each CP-net in turn and collecting together the results.

Using different voting rules we can obtain different semantics. Have been considered five main voting rules: Pareto, majority, max, lex and rank. An optimal outcome can be derived using each of them but in exponential time. Also the dominance task is hard: the computational complexity is exponential in the input description.

2.3.6 GCP-nets

GCP-nets ([52] and [41]) allow a general form of conditional and qualitative preferences to be modeled compactly.

Definition 2.3.13 A **Generalized CP-net** (**GCP-net**) C over a set of variables with binary domains `Var` is a set of conditional preference rules. A **conditional preference rule** is an expression $p : l > \bar{l}$, where l is a literal of some atom $X \in$ `Var` and p is a propositional formula over `Var` that does not involve variable X. A GCP-net corresponds to a directed graph (dependency graph) where each node is associated with a feature and the edges are pairs (Y,X) where Y appears in p in some rule $p : x > \bar{x}$ or $p : \bar{x} > x$. Each node X is associated with a *CP-table* which expresses the user preference over the values of X. Each row of the CP-table corresponds to a conditional preference rule.

The CP-tables of a GCP-net can be *incomplete* (i.e. for some values of some variables' parents, the preferred value of X may not be specified) and/or *locally inconsistent* (i.e. for some values of some variables' parents, the table may both contain the information $x > \bar{x}$ and $\bar{x} > x$). CP-nets [11] are a special case of GCP-nets in which the preferences are locally consistent and locally complete.

Consistency and Optimality

Given any GCP-net the problems of consistency checking and finding optimal outcomes are PSPACE-complete [52]. Moreover, there could be several different maximal elements. The optimal outcomes for acyclic nets are unique and can be found in polynomial time in N. The procedure used to this purpose is usually called a *sweep forward* and takes N steps [11].

Dominance

The problem of dominance testing (i.e. determining if one outcome is preferred to another) is PSPACE-complete for GCP-nets. It is polynomial if the GCP-net is tree structured or poly-tree structured [39, 52].

2.3.7 CP-nets with constraints

Preferences and constraints are strongly related. We often have the simultaneous presence of preferences such as "I prefer pizza to pasta" and hard constraints such as "I can't eat peanuts because I'm allergic".

With the purpose of handle this union of qualitative preferences and hard constraints, there are two main approaches ([12] and [84]). Both of them provide algorithmic results to the problem.

Boutilier et al. method

In this work Boutilier et al. method [12] consider the case of acyclic CP-nets. They defined a new notion of optimal outcome: the optimal outcome is a feasible outcome (i.e., consistent with the hard constraints) that is not dominated by any other feasible outcome.

Thus, given a set of feasible outcomes S and a CP-net $\mathscr{C}$ an outcome $o \in S$ is said to be *Pareto optimal* with respect to the preference order induced by the CP-net $\mathscr{C}$ if and only if there is no $o' \in S$ such that $o' \succ o$. In this context we are looking for a Pareto optimal outcome in the set S, that corresponds to the set of feasible outcomes. But determining the set of Pareto-optimal feasible outcomes for general sets of constraints is NP-hard.

The idea of this work is to use a branch-and-bound algorithm to determine the set of feasible outcomes using the structure of user preferences. If we stop the algorithm in any time, it always provide a solutions set of outcomes that contain only Pareto-optimal solutions.

Prestwich et al. method

In this work Prestwich et al. [84] define a new structure that consists of the union of a CP-net and a set of hard constraints.

Definition 2.3.14 A *constrained CP-net* is a CP-net plus a set of constraints defined on subsets of variables of the CP-net. Formally is a pair $\langle \mathscr{C}, V \rangle$ where $\mathscr{C}$ is a CP-net and V is a set of hard constraints.

We have also a different notion of worsening flips, in which now are allowed only feasible outcomes:

Definition 2.3.15 Given a constrained CP-net $\langle \mathscr{C}, V \rangle$, and two outcomes o and o', we have that $o \succ o'$ iff there is a chain of flips from o to o' such that each flip is a worsening flip for $\mathscr{C}$ and each outcome in the chain is feasible for V.

This is the main difference with the method described in [12]: in [12] they allow also worsening chains with unfeasible outcomes.

Considering only binary variables we can consider the constraints as Boolean clauses (in normal form) and we have the following translation of the CP-statements $u : a \succ \bar{a}$:

$$(u \wedge \bigwedge_{\varphi \in V \wedge \bar{a} in \varphi} \varphi \restriction_{a=True}) \rightarrow a$$

where $\varphi \restriction_{a=True}$ is the clause where we have deleted $\bar{a}$.

An outcome is optimal for $\langle \mathscr{C}, V \rangle$ iff satisfies the Boolean satisfaction problem of the set of all these kind of constraints and the translation of the constraints in V. The same results apply also in the non-binary case.

2.3.8 CP-theories

CP-theories (CP-th) are introduced in [111] as a logic of conditional preference which generalizes CP-nets.

Definition 2.3.16 Given a set of variables $\mathtt{Var} = \{X_1, \cdots, X_N\}$ with domains $Dom(X_i)$, $i = 1, \ldots, n$, the language $L_{\mathtt{Var}}$ is defined by all the statements of the form: $u : x_i \succ x_i'[W]$ where u is an assignment of a set of variables $U \subseteq \mathtt{Var} \setminus \{X_i\}$, $x_i \neq x_i' \in Dom(X_i)$ and W is a set of variables such that $W \subseteq (\mathtt{Var} \setminus U \setminus \{X_i\})$.

Definition 2.3.17 Given a language $L_{\mathtt{Var}}$ as defined above, a *conditional preference theory (CP-theory)* Γ on $\mathtt{Var}$ is a subset of $L_{\mathtt{Var}}$. Γ generates a set of preferences that corresponds to the set $\Gamma^* = \bigcup_{\varphi \in \Gamma} \varphi^*$ where, given $\varphi = u : x_i \succ x_i'[W]$, φ^* is defined as $\varphi^* = \{(tuxw, tux'w') : t \in \mathtt{Var} \setminus (\{X_i \cup U \cup W\}),\ w, w' \in W\}$.

A CP-net is a particular case of a CP-theory where $W = \emptyset$ for all $\varphi \in \Gamma$.

Two graphs are associated to a CP-theory: $H(\Gamma) = \{(X_j, X_i) | \exists \varphi \in \Gamma \text{ s.t. } \varphi = u : x_i \succ x_i'[W] \text{ and } X_j \in U\}$ and $G(\Gamma) = H \cup \{(X_i, X_j) | \exists \varphi \in \Gamma \text{ s.t. } \varphi = u : x_i \succ x_i'[W] \text{ and } X_j \in W\}$.

■ **Example 2.10** A CP-th defined on three features $C = cake$, $D = drink$ and $W = weather$ described by the sentences: " I?d prefer to eat a carrot cake rather then a chocolate cake irrespective of the choices of the drink." and "If it is a cold day I prefer to drink a tea, otherwise a lemonade" and "I prefer cold days" can be seen in Figure 2.7.

The CP-th in Figure 2.7 induce the order described by the following sets:

- $\{\ carrot\ d_1\ w > chocolate\ d_2\ w \mid d_1, d_2 \in Dom(D),\ w \in Dom(W)\}$
- $\{\ c\ tea\ cold > c\ lemonade\ cold \mid c \in Dom(C)\}$
- $\{\ c\ lemonade\ warm > c\ tea\ warm \mid c \in Dom(C)\}$

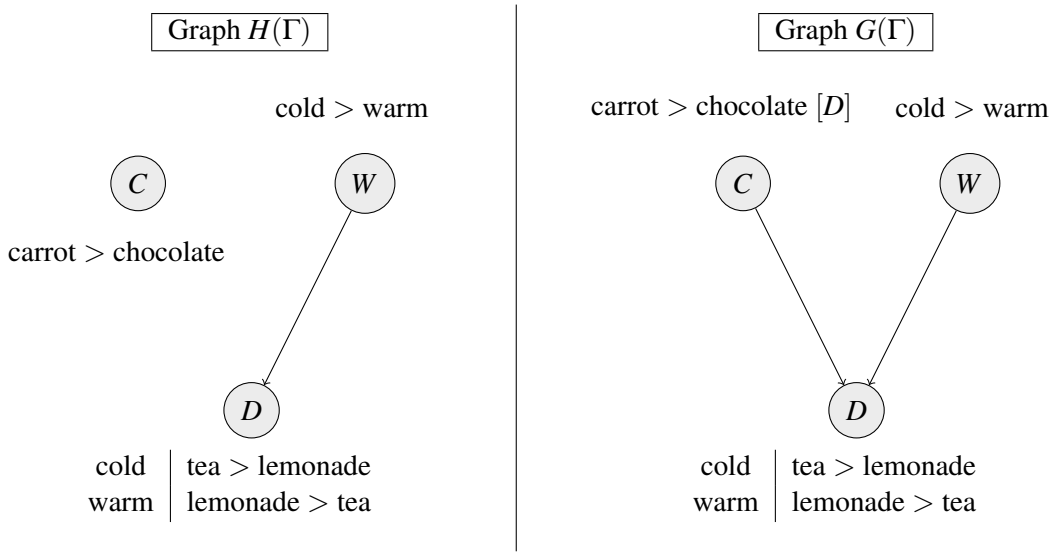


Figure 2.7: A CP-theory Γ: the two graphs $G(\Gamma)$ and $H(\Gamma)$ of Example2.10

- $\{\ c\ d\ cold > c\ d\ warm \mid d \in Dom(D),\ c \in Dom(C)\}$

■

The semantics of CP-theories depends on the notion of a *worsening swap*, which is a change in the assignment of a set of variables to an assignment which is less preferred by a rule $\varphi \in \Gamma$. We say that one outcome o is better than another outcome o' ($o \succ o'$) if and only if there is a chain of worsening swaps (a *worsening swapping sequence*) from o to o'. Thus are well defined the notions of optimality and dominance. Outcome optimization can be computed in polynomial time for tree structured CP-theories but is PSPACE-complete for general CP-theories. Dominance queries instead are PSPACE-complete also for acyclic structures.

Definition 2.3.18 A CP-theory Γ is *locally consistent* if and only if for all $X_i \in \mathtt{Var}$ and $u \in Pa(X_i)$ in the graph $H(\Gamma)$, $\succ_u^{X_i}$ is irreflexive.

Local consistency can be determined in time proportional to $|\Gamma|^2 N$. Given a CP-theory Γ, if the graph $G(\Gamma)$ is acyclic, Γ is consistent if and only if Γ is locally consistent, thus global consistency has the same complexity as local consistency given an acyclic graph $G(\Gamma)$.

2.3.9 Comparative preference theories

Comparative preference theories [112] are an extension of CP-theories.

Definition 2.3.19 The comparative preference language $\mathscr{CL}_{\mathtt{Var}}$ is defined by all statements of the form: $p > q||T$ where P, Q and T are subsets of Var and p and q are assignments respectively

of the variables in P and in Q.

Definition 2.3.20 Given a language $\mathscr{CL}_{\mathtt{Var}}$ as defined above, a *comparative preference theory* Λ on $\mathtt{Var}$ is a subset of $\mathscr{CL}_{\mathtt{Var}}$. Λ generates a set of preferences that corresponds to the set $\Lambda^* = \bigcup_{\varphi\in\Lambda} \varphi^*$ where if $\varphi = p > q||T$, φ^* is defined as a pair (α,β) of outcomes such that α extends p and β extends q and α and β agree on T: $\alpha \restriction_T = \beta \restriction_T$.

2.4 Soft Constraints

A soft constraint [78] involves a set of variables and associates a value from a (totally or partially ordered) set to each instantiation of its variables. Such a value is taken from a preference structure $\langle A,+,\times,0,1\rangle$, where A is the set of preference values, $+$ induces an ordering over A (where $a \leq b$ iff $a+b=b$), $\times$ is used to combine preference values, and 0 and 1 are respectively the worst and best element.

Definition 2.4.1 A Soft Constraint Satisfaction Problem (SCSP) is a tuple $\langle \mathtt{Var}, D, C, A\rangle$ where $\mathtt{Var}$ is a set of variables, D is the domain of the variables, and C is a set of soft constraints (each one involving a subset of $\mathtt{Var}$) associating values from A.

An instance of the SCSP framework is obtained by choosing a specific preference structure. For instance, choosing $S_{FCSP} = \langle [0,1], max, min, 0, 1\rangle$ means that preference values are in [0,1] and we want to maximize the minimum preference value. This is the setting of fuzzy CSPs (FCSPs) [78, 97]. Figure 2.8 shows the constraint graph of an FCSP where $\mathtt{Var} = \{X,Y,Z\}$, $D = \{f,\bar{f}\}\forall Fin\mathtt{Var}$ and $C = \{c_X, c_Y, c_Z, c_{XY}, c_{YZ}\}$. Each node models a variable and each arc models a binary constraint, while unary constraints define variables' domains. For example, c_Y associates preference value 0.4 to $Y = y$ and 0.7 to $Y = \bar{y}$. Default constraints such as c_X and c_Z, where all variable assignments get value 1.

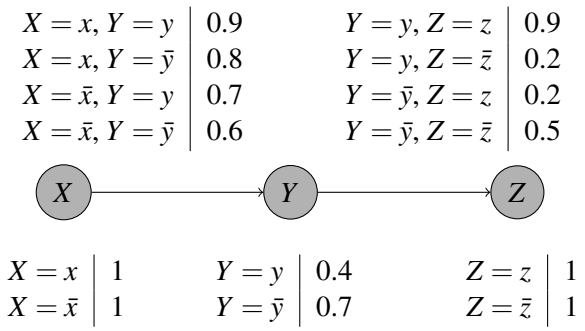


Figure 2.8: An FCSP.

Given an assignment o to all the variables of an SCSP, the preference value of o is obtained by combining (via the operator $\times$ of the preference structure) the preference values associated by

each constraint to the sub-tuples of o. For example, in FCSPs, the preference value of a complete assignment is the minimum preference value given by the constraints. Considering the example in Figure 2.8, the assignment xyz has preference value equal to $\min\{1, 0.9, 0.4, 0.9, 1\} = 0.4$

2.4.1 Optimality

Solving an SCSP means finding the top element in the ordering induced by the constraints over the set of all the solutions, i.e., over the set of all complete variable assignments. In the case of FCSPs, such an ordering is a total order with ties. In the example above, the induced ordering has $(X = x, Y = \bar{y}, Z = \bar{z})$ and $(X = \bar{x}, Y = \bar{y}, Z = \bar{z})$ at the top with a preference value 0.5, $(X = x, Y = y, Z = z)$ and $(X = \bar{x}, Y = y, Z = z)$ just below with 0.4, and all others tied at the bottom with preference value 0.2. An optimal solution, say o, of an SCSP is then a complete assignment with an undominated preference value (thus $(X = x, Y = \bar{y}, Z = \bar{z})$ or $(X = \bar{x}, Y = \bar{y}, Z = \bar{z})$ in this example). Given a variable X, we write $s_{\downarrow x}$ to denote the value of X in o. Finding an optimal solution is in general an NP-hard problem, unless certain restrictions are imposed, such as a tree-shaped constraint graph [33]. The FCSP shown in Figure 2.8 has a tree-shape constraint graph.

Constraint propagation [8] may improve the computational efficiency of the search for an optimal solution.

We will consider a specific form of constraint propagation called directional arc consistency (DAC). Given a variable ordering o, a FCSP is DAC if all its constraints are so. Assuming all constraints are binary, that is, involve two variables, a fuzzy constraint over variables X and Y such that X precedes Y in the ordering o, is DAC iff, for each a in the domain of X,

$$f_X(x) = max_{v \in Dom(Y)}(min(f_X(x), f_{XY}(x,v), f_Y(v)))$$

, where f_X, f_Y, and f_{XY} are the preference functions of c_X, c_Y and c_{XY}. This definition can be generalized to any instance of the SCSP approach by replacing max and min with the two operations $+$ and $\times$ of the preference structure.

If a FCSP is not DAC, we can make it so in polynomial time with an algorithm that follows backward the ordering o. In our running example, if we choose the variable ordering $\langle X, Y, Z \rangle$, achieving DAC means first enforcing the property over the constraint on Y and Z and then over the constraint on X and Y. The first step modifies the preference value of $Y = \bar{y}$ to 0.5, while the second step sets the preference values of both $X = x$ and $X = \bar{x}$ to 0.5. We note that, by achieving DAC w.r.t. ordering o, we obtain a total order with ties over the values of the first variable in o, where each value is associated to the preference value of the best solution having such a variable instantiated to such a value. Given an SCSP P and one of its variables V, we will denote as $top(V, P)$ the set of values of V that are assigned the highest preference value in such an ordering. In our running example, achieving DAC brings both values of X in a tie and in $top(X, P)$, with preference value 0.5, that is the preference value of an optimal solution.

DAC is enough to find the preference value of an optimal solution when the constraint graph of the problem (i.e., the graph where the nodes represent the variables and the arcs represent the constraints) has no cycles (and thus it has a tree shape) [78]. In fact, if we have DAC in a tree

problem, such an optimum preference value is the best preference value in the domain of the root variable. In this situation, to find an optimal solution, it is then enough to perform a backtrack-free search which instantiates variables in the order o (thus, from the root of the tree to its leaves).

In our running example, if we choose the variable ordering $\langle X,Y,Z\rangle$, achieving DAC means first enforcing the property over the constraint on Y and Z and then over the constraint on X and Y. The first phase modifies the preference value of $Y=\bar{y}$ to

$$\max(\min(f_Y(\bar{y}),f_{Y,Z}(\bar{y},z),f_Z(z)),\min(f_Y(\bar{y}),f_{Y,Z}(\bar{y},\bar{z}),f_Z(\bar{z})))$$

$$=\max(\min(0.7,0.2,0.1),\min(0.7,0.5,1))=\max(0.2,0.5)=0.5\ .$$

Similarly, the second phase sets the preference values of both $X=x$ and $X=\bar{x}$ to 0.5. We note that, by achieving DAC w.r.t. ordering o, we obtain a total order with ties over the values of the first variable in o, where each value is associated to the preference value of the best solution having such a variable instantiated to such a value. Given an SCSP P and one of its variables V, we will denote as $top(V,P)$ the set of values of V that are assigned the highest preference value in such an ordering. In our running example, achieving DAC brings both values of X in a tie and in $top(X,P)$, with preference value 0.5, that is the preference value of an optimal solution.

2.5 Voting theory

A voting rule allows a set of voters to choose one among a set of candidates. Voters need to submit their vote, that is, their preference ordering over the set of candidates (or part of it), and the voting rule aggregates such votes to yield a result, usually called the winner. In the classical setting [6], given a set of n candidates C, a *profile* P is a collection of total orderings over the set of candidates, one for each voter C_i $i\in\{1,\ldots,n\}$. Given a profile, a *voting rule* also known as a social choice function, maps it onto a single winning candidate (if necessary, ties are broken appropriately).

2.5.1 Voting rules

The formal definition of a voting rule is:

Definition 2.5.1 Let P a profile of m voters over a set of alternatives X, a *voting rule* $r:P\rightarrow X$ is a function that maps each profile P into an alternative $r(P)\in X$.

Some examples of widely used voting rules are (we assume a tie-breaking mechanism to assure a single winner):

- *Plurality*, where each voter states whom the most preferred candidate is, and the candidate who is preferred by the largest number of voters wins;
- *Majority*: like plurality, but with only two candidates; (with just 2 candidates, it's equivalent to Majority voting).
- *Borda*, where, given m candidates, each voter gives a ranking of all candidates, the i^{th} ranked candidate scores $m-i$, and the candidate with the greatest sum of scores wins;
- *Approval*, where each voter approves between 1 and $m-1$ candidates of m total candidates, and the candidate with most votes of approval wins;

- *Copeland*, where the winner is the candidate who wins the most pairwise competitions against all the other candidates.

2.5.2 Voting rules on combinatorial domains

Voting rules can be defined also on multi-issue domains.

Definition 2.5.2 Let $\texttt{Var} = X_1, \cdot, X_n$ denote a set of n features (variables), with domains $Dom(X_i)$. *Combinatorial voting* refers to the voting setting where the set of alternatives corresponds to $C = Dom(X_1) \times \cdots \times Dom(X_n)$. C is called a multi-issue domain or combinatorial domain.

Considering the general setting we encounter a problem: suppose that all the variables X_i are binary-valued, then we obtain a set of alternative of cardinality 2^n. As soon as the number of alternative is large, the voters are likely to be unhappy about expressing a their preferences and moreover, the result of voting could be completely insignificant. For instance, with 5 voters and 6 binary issues, it is probable that all the 5 voters vote for a different alternatives (since there are 64 alternatives). In this case the winner, under the plurality rule, might be disliked by all but one voter. This phenomenon is a type of multiple-election paradox [26].

Sequential Voting

One natural approach to combinatorial voting is sequential voting. Given $O = X_1, X_2 \cdots X_n$ an ordering over the issues, sequential voting selects the winner in n steps. In each step i, the voters vote on the i-th issue in O following their preferences on that particular variable. The agents votes over the i-th feature are collected by applying a local voting rule, and the winning value is then announced to all the voters. More rigorously:

Definition 2.5.3 Given a vector of local rules $r_1, \cdots, r_n$, where each r_i is a voting rule on $Dom(X_i)$, the sequential composition of $r_1, \cdots, r_n$ following O, denoted by $Seq_O(r_1, \cdots, r_n)$ [65], is defined as follows: $Seq_O(r_1, \cdots, r_n)[P] = (c_1, \cdots, c_n) \in C$, so that for any $c_i = r_i[P \restriction_{Xi:c_1 \cdots c_{i-1}}]$.

This approach can be used to avoid multiple-election paradoxes [62].

2.5.3 Properties of the voting rules

Research on voting theory has mainly been concerned with the definition of properties of voting systems that are desirable. The main are the following ones [6, 104]:

- *Condorcet-consistency*: A voting rule is Condorcet-consistent if, when a candidate who beats every other in pairwise elections (namely, a Condorcet winner) exists, that candidate is always elected; The Condorcet winner is unique and may not exist.
- *Anonymity*: A voting rule is anonymous when the results of an election are the same even if a permutation on the voters' preferences is applied. This means that the result does not depend on the names of the voters, or in which order they are considered, but it depends only on their set.
- *Neutrality*: A voting rule is neutral when, for any permutation on the set of candidates, the permuted results of an election are identical to the results of the same election where the permutation on the set of candidates occurs before applying the voting rule. This is the

analogous of anonymity for the candidates: the result does not depend on the names of the candidates.

- *Monotonicity*: A voting rule is monotonic if, when a candidate wins, and a voter improves his vote in favor of this candidate, then the same candidate still wins. Formally: given an issue x and a set of voters E, a voting rule r is monotonic if, for any profile $P = \langle V_1, ..., V_{|E|} \rangle$ over x and for any $P' = \langle V'_1, ..., V'_{|E|} \rangle$ over x such that each V'_i is obtained from V_i by raising only $r(P)$, then $r(P') = r(P)$.
- *Strong Monotonicity*: A voting rule is strongly monotonic if, given a profile P and a subset of the candidates Y, and any profile P' obtained by P by increasing the preference values of the candidates in Y, without changing their relative position, the winner in P' is either the winner in P or is in Y.

  Formally: given an issue x, a voting rule r is strongly monotonic if, for any profile P over x, any $Y(x) \subseteq D(x)$, and any P' obtained from P only by raising the candidates in $Y(x)$ while keeping their relative positions unchanged, we have $r(P') \in r(P) \cup Y(x)$.
- *Consistency*: A voting rule is consistent if, when considering preferences of two disjoint sets of voters - who decide over the same candidates and have identical final results - the result obtained by the profile of the joint set of voters is the same as the ones obtained by the disjoint set of voters.
- *Participation*: A voting rule is participative if, given any profile, and given a new vote over a set of candidates by a new voter, the result obtained from the new profile is equally or more preferred by the new voter. This means that a voter has an incentive to participate in the voting process, since by doing so he can get a better result.
- *Efficiency*: A voting rule is efficient if, given the winner in an election, there's no candidate who is preferred to it by all voters. This means that, when all voters agree that a certain candidate should win, he will be declared the winner.
- *Independence of Irrelevant Alternatives (IIA)*: A voting rule is IIA if, whenever a candidate y looses to some winner x, and the relative rankings of x and y do not change in the profile, then y cannot win (independently of any possible change w.r.t. other irrelevant alternatives).
- *Non-dictatorship*: A voting rule is non-dictatorial if there is no voter such that the winner is always a top-ranked alternative of this voter (the dictator).
- *Strategy-proof*: A voting rule is strategy-proof if it is not manipulable, that is, no voter can get better off by lying on its preference ordering.

Table 2.1 summarizes the properties of the voting rules viewed so far [6]. All such rules are anonymous and neutral. Moreover, only Copeland is Condorcet consistent, and all but Copeland are consistent and participative. All the above rules are non-dictatorial and manipulable in general, while only Approval is IIA. All these rules satisfy efficiency and monotonicity but not strong monotonicity.

Rule	*C.C.*	*Anon.*	*Neutr.*	*Mon.*	*S.M.*	*Cons.*	*Eff.*	*Part.*	*IIA*	*Non-dict.*	*St-pr.*
Plurality	*No*	*Yes*	*Yes*	*Yes*	*No*	*Yes*	*Yes*	*Yes*	*No*	*Yes*	*No*
Borda	*No*	*Yes*	*Yes*	*Yes*	*No*	*Yes*	*Yes*	*Yes*	*No*	*Yes*	*No*
Approval	*No*	*Yes*	*Yes*	*Yes*	*No*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*No*
Copeland	*Yes*	*Yes*	*Yes*	*Yes*	*No*	*No*	*Yes*	*No*	*No*	*Yes*	*No*

Table 2.1: Properties of the voting rules.

II Original contributions

3. PCP-nets

We often meet situations characterized by uncertainty and probabilistic information. A user could be unsure about her preferences or we could have error and noise in the learning/elicitation procedures or during the data collection. Another scenario in which we have to handle uncertainty is the multi-agent scenario: a group of agents have to make a decision but the preferences of the different users could be in conflict.

With the purpose to model this kind of uncertain preferences we define a new formalism introducing probabilities in the CP-nets framework. While keeping the preferential dependency structure of a CP-net, in a PCP-net a preference ordering over a feature domain is replaced by a probability distribution over all possible preference orderings over that domain. We define thus a generalization of traditional CP-nets with probabilities on individual cp-statements, making the assumption that these probabilities are independent. In this way we can use algorithms and techniques from Bayesian networks to efficiently compute outputs for restricted dependency structures.

A PCP-net defines a probability distribution over a collection of CP-nets: all those CP-nets that can be obtained from the PCP-net by choosing one feature ordering out of the whole probability distribution over them.

Given a PCP-net, we focus on the two main reasoning tasks: optimality and dominance. We define different notions of optimality. Concerning outcome optimization, one can define the optimal feature assignment in two natural ways. The first definition of optimal outcome corresponds to the most probable optimal outcome, the second to the optimal outcome of the most probable CP-net induced by the PCP-net. If the PCP-net has a bounded dependency structure, both top outcomes can be found in polynomial time in the size of the PCP-net. Concerning instead user optimization, we can define the notion of *most probable induced CP-net*, that is the most probable description of a

"standard" user. Also this task can be computed in polynomial time in the size of the PCP-net, if the PCP-net has a bounded dependency structure. We then study dominance. Given two outcomes, it is in general computationally difficult to compute the probability that one dominates the other one, since this is the sum of all probabilities of the CP-nets, induced by the PCP-net, where dominance holds. For PCP-nets whose dependency graph is a polytree, for which dominance is still difficult, we give a polynomial time approximation in the form of a lower and an upper bound to the correct probability value for dominance. We show, experimentally, that often the range between the lower and upper bound is small, and in particular that the lower bound is very close to the correct value.

We analyse also how to learn a PCP-net from a set of dominance examples, in a preliminary case of separable structure. We assume also that the set of example is consistent and thus our algorithm returns always at least one solution.

Many generalization of PCP-nets are still possible. We analyse two of them: a dynamic generalization and a generalization using a different notion of probability. In the dynamic context we give attention to the possible dynamic modifications to the structure of a PCP-net and how to implements their effects in an efficient way. We also analyse the effects of these dynamic modification on computing the most probable optimal outcome and the most probable induced CP-net. In the second scenario instead we focus on the Dempster-Shafer theory of probability as a generalization of the classical definition of uncertainty, allowing the representation of ignorance and using a rule to combine the effect of different sources of evidence in a more general context. We provide a description of this new formalism to describe probabilistic conditional preferences.

Related work

There has been a lot of work in the study and modeling of probabilistic preferences in a variety of areas such as philosophy, logic, and economics. Probabilistic analysis in social choice has often focused on the *impartial culture* model: a model in which all preference orders (for a set of candidates) are equally likely. The plausibility of this particular model has been called into question by behavioral social choice theorists [89]. Thus were developed frameworks that reflect a more realistic probabilistic scenarios: probabilistic models of preferences or parameterized families of distributions over rankings.

A different stream of research has been developed around a logic description for probabilistic preferences: the incorporation of preferences into information systems, such as databases, has recently seen a surge in interest, with a special interest on modeling uncertainty in these domains. In a work of [71] they combine probabilistic reasoning with logical representations. The authors provide probabilistic ontologies used to rank answers to a user?s queries, since the preference model may be in conflict with the preferences induced by the probabilistic model. Other works instead apply the concept of probabilistic preferences into the recommender systems context: [106] propose a framework, namely Preference Network (PN), a probabilistic model that systematically combines both content-based filtering and collaborative filtering into a single conditional Markov random field; [58] presents a preference model using mutual information in a statistical framework, combing information of joint features and alleviates problems arising from sparse data. These approaches are located in the specific area of recommender systems, thus they can not be applied

to a more general scenario. A next step can be seen in [70], where they focus on the point of view of the intersection of databases and knowledge representation, where preferences are usually represented as strict partial orders over the set of tuples in a database or the consequences of a knowledge base. They introduce the probabilistic preference logic networks (PPLNs), combining such preferences with probabilistic uncertainty.

A similar approach to PCP-net can be seen in the work of de Amo et al. [3] in which they propose *CPrefMiner* for learning a Bayesian Preference Network (BPN) from a given set of user preferences between two alternatives. In a scenario of preference elicitation, they focus on preferences (expressed by a BPN) that can change according to the context in which they are located.

Chapter structure and related publications

The work presented in this chapter has appeared in the proceedings of the following conferences and international workshops.

- ⋆ **"Dynamic and Probabilistic CP-nets"**, *C. Cornelio*, Proceedings of the Doctoral Program of International Conference on Principles and Practice of Constraint Programming 2013, CP-13.
- ⋆ **"Updates and Uncertainty in CP-net"**, *C. Cornelio, U. Grandi, J. Goldsmith, N. Mattei, F. Rossi and K.B. Venable*, Proceedings of AUAI 2013: Advances in Artificial Intelligence, 26th Australasian Joint Conference, 2013.
- ⋆ **"Dynamic Probabilistic CP-nets"**, *C. Cornelio, U. Grandi, J. Goldsmith, N. Mattei, F. Rossi and K.B. Venable*, Proceedings of the 7th Multidisciplinary Workshop on Advances in Preference Handling of the International Joint Conference on Artificial Intelligence 2013, MPREF-13.

The chapter is organized as follows.

- In Section 3.1 we define the new PCP-nets framework, a generalization of the CP-nets model allowing uncertainty. We formally define the model and provide some examples.
- In Section 3.2 we introduce the set of CP-nets that are induced by a PCP-net. A PCP-net defines a probability distribution over this set.
- In Section 3.3 we provide a formula to compute the probability of existence of an edge in the PCP-nets dependency graph. We provide two formulation of the formula in the two cases of binary domains features and of general features.
- In Section 3.4 we study the optimality task in PCP-nets. We consider different notion of optimality: about induced CP-nets and about outcomes. We provide algorithms to compute these different tasks.
- In Section 3.5 we analyse the dominance task. We provide an exact procedure in the case of separable PCP-nets, and an approximate procedure for the general scenario. We also provide an experimental analysis to understand the accuracy of the approximation.
- In Section 3.6 we investigate the problem of exact learning of separable PCP-nets, given a set of consistent dominance examples.
- In Section 3.7 we consider the case of PCP-nets with dynamic modification of the structure

or of the PCP-tables.

- In Section 3.8 we introduce a generalization of PCP-nets considering an alternative probability theory: the Dempster-Shafer theory of probability.
- In Section 3.9 we provide a summary of the chapter and a discussion of the results.

3.1 Probabilistic CP-nets (PCP-nets)

In the following section we will consider features with general domains, but in some parts, for instance in the examples, we restrict for simplicity to binary domains and bounded (by a constant) induced width k (see Definition 2.1.12) of the dependency graph. Where not specified the extension to non-binary domains is a trivial exercise.

Definition 3.1.1 A **PCP-net** (Probabilistic CP-net) [28] [9] is a directed graph where each node represents a variable (often called feature) $\texttt{Var} = \{X_1, \ldots, X_n\}$ each with binary domains $Dom(X_1), \ldots, Dom(X_n)$. For each feature X_i, there is a set of *parent* features $Pa(X_i)$ that can affect the preferences over the values of X_i. This defines a *dependency graph* in which each node X_i has edges from all features in $Pa(X_i)$. Given this structural information, for each feature X_i, instead of giving a preference ordering over the domain of X_i (as in the CP-nets), we give a probability distribution over the set of all preference orderings (total orderings over $Dom(X_i)$) for each complete assignment on $Pa(X_i)$.

Definition 3.1.2 Given a feature X in a PCP-net, its *PCP-table* is a table associating each combination of the values of the parent features of X with a probability distribution over the set of total orderings over the domain of X.

Note that we are assuming that the probabilities of different orderings for different parents instantiations are independent to each other, considering each rule as a random variable. This may look as a strong simplification but it make sense considering the typical kind of data that we are managing: a huge amount of information from a lot of individuals, each one gives only a few preferences thus generally providing sparse data. Let think about recommender systems: usually each user give information on a small set of features or items and any other information on the others. For this reason, when we aggregate all of the user preferences we obtain that it clearly makes sense to maintain these rules independent from each other, since each rule is obtained from a different user.

We discuss in Section 8.1 as future direction, how to extended this representations of the probability distributions over the statements, to a richer one, breaking the independence assumption.

■ **Example 3.1** This example shows the difference between a CP-net $\mathscr{C}$ shown in Figure 3.1 and a PCP-net $\mathscr{Q}$ shown in Figure 3.2 with two features, X_1 and X_2, with domains $Dom(X_1) = \{x_1, \bar{x}_1\}$ and $Dom(X_2) = \{x_2, \bar{x}_2, \underline{x}_2\}$.

■

Note that PCP-nets are a strict generalization of CP-nets. When a PCP-net is restricted to probability distributions in $\{0,1\}$ we recover the definition of CP-nets [11].

$\bar{x}_1 > x_1$

x_1	$x_2 > \bar{x}_2 > \underline{x}_2$
$\bar{x}_1$	$\underline{x}_2 > \bar{x}_2 > x_2$

Figure 3.1: A CP-net $\mathscr{C}$ (Example 3.1)

$x_1 > \bar{x}_1$	0.2
$\bar{x}_1 > x_1$	0.8

x_1	$x_2 > \bar{x}_2 > \underline{x}_2$	0.1
	$\underline{x}_2 > \bar{x}_2 > x_2$	0.3
	$\bar{x}_2 > \underline{x}_2 > x_2$	0.6
$\bar{x}_1$	$x_2 > \bar{x}_2 > \underline{x}_2$	0.7
	$\bar{x}_2 > x_2 > \underline{x}_2$	0.3

Figure 3.2: A PCP-net $\mathscr{Q}$ (Example 3.1)

3.2 The set of induced CP-nets

We note that a PCP-net is a compact representation of a probability distribution over a set of CP-nets (for example, representing the preferences of a certain population in a multi-agent context). In fact, given a PCP-net, we can generate many different CP-nets, one for each choice that can be performed on the PCP-net.

Definition 3.2.1 Given a PCP-net $\mathscr{Q}$, a *CP-net induced by* $\mathscr{Q}$ has the same features, with the same domains, as $\mathscr{Q}$. The CP-table of each feature X_i is generated choosing specific rows of the PCP-table of X_i: an ordering is selected for each combination of the values of the parent features $Pa(X_i)$.

It is important to notice that in the generation process of the induced CP-tables we can obtain also redundant information, if we select rows that are equal for each parents assignment. Considering two variables X_1 and X_2 such that $\{X_1\} = Pa(X_2)$, if we we choose the same ordering o_{X_2} of the domain values of the variable X_2 for each assignment of the parent node X_1, we obtain that X_2 is independent from X_1 and thus we can contract the CP-table using o_{X_2} and we can remove the edge between X_1 and X_2. In this way we are reducing induced CP-nets in their *normal form*, removing redundant information.

Definition 3.2.2 A CP-net is in *normal form* if for each node X does not exists any node $Y \in Pa(X)$ such that the rows of the CP-table of X, in correspondence to the parents assignments $u_i y_1, u_i y_2, \cdots, u_i y_l$, have the same ordering for the values of the variable X, for each u_i in

the set $\{u_1,\cdots,u_m\} = Dom(Y_1)\times\cdots\times Dom(Y_k)$ where $Dom(Y) = \{y_1,\cdots,y_l\}$ and $Pa(X) = \{Y,Y_1,\cdots,Y_k\}$.

It is important to notice that a CP-net induces the same preorder over the set outcomes that is induced by its normal form.

■ **Example 3.2** This example show the diffirence between a CP-net not in normal form and its normal form. In Figure 3.4 is shown the normal form of the CP-net in Figure 3.3

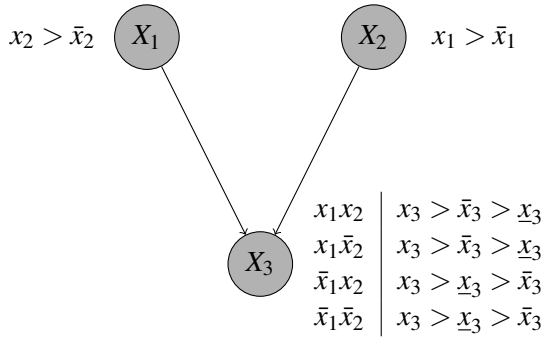


Figure 3.3: A CP-net not in normal form (Example 3.2)

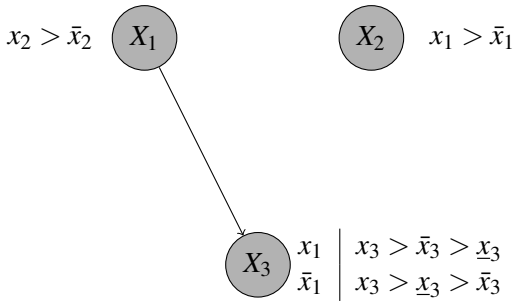


Figure 3.4: The normal form of CP-net in Figure 3.3 (Example 3.2)

■

Considering the normal form of the induced CP-nets, we observe that the edges are a subset of the edges in the PCP-net. Thus, CP-nets induced by the same PCP-net may have different dependency graphs, if considered in their normal form.

Definition 3.2.3 Given a PCP-net $\mathscr{Q}$, the *set of the induced CP-nets by* $\mathscr{Q}$ is the set that contains all the induced CP-net by $\mathscr{Q}$.

Each induced CP-net has an associated probability, obtained from the PCP-net by taking the product of the probabilities of the orderings chosen in the CP-net.

Notice that it is correct to do so because the choice of the ordering of a feature is independent to the choice of the ordering of the other features.

Definition 3.2.4 Given a PCP-net $\mathscr{Q}$ and an induced CP-net C, we can define C by its CPTs (Conditional Preference Tables). Let $\bar{q}$ be the vector containing the probabilities of all rows of the P-CPTs of $\mathscr{Q}$, and $\mathscr{X}_C$ a function that associates to each row i the value 1 if the row i defines C (it appears in the CPTs of C) 0 otherwise. We define the probability of C, $fp_C(\bar{q})$, to be the product of the probabilities of the q_i with $\mathscr{X}_C(i) = 1$

$$\mathbb{P}(C) = fp_C(\bar{q}) = \prod_{i:\mathscr{X}_C(i)=1} q_i$$

Hence, a PCP-net induces a probability distribution over the set of all induced CP-nets.

■ **Example 3.3** Consider the PCP-net in Figure 3.5 with two features, X_1 and X_2, with domains $Dom(X_i) = \{x_i, \bar{x}_i\}$. Figure 3.6 describes a CP-net that has been induced from the PCP-net of Figure 3.5. The probability associated to this CP-net is defined by the following formula: $fp_C(\bar{q}) = [q_2 \cdot q_4 \cdot q_5]$, given $\mathscr{X}_C = (0,1,0,1,1)$.

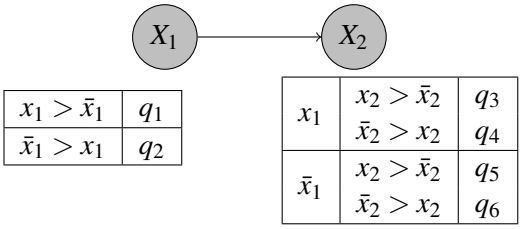


Figure 3.5: A PCP-net.

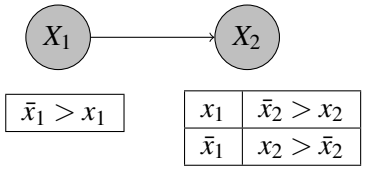


Figure 3.6: A CP-net induced by the PCP-net in Figure 3.5.

■

3.3 Probability of the edges

In this section we consider all the CP-nets in normal form. A PCP-net induces a profile of CP-nets, each one with a possibly different dependency graph. Each edge of the CP-net dependency graph can appear or not in a particular induced CP-net. Thus each dependency is associated with a probability of existing.

Definition 3.3.1 Given a PCP-net Q and an edge e in the dependency graph of Q, we define the probability of the existence of the edge e as the sum of the probability of the induced CP-nets in normal form that have e in their dependency graph:

$$\mathbb{P}(e) = \sum_{i \in I \text{ s.t. } e \in C_i} \mathbb{P}(C_i)$$

where I is the index set of induced CP-nets C_i in normal form.

This definition implies that the probability of non-existence of an edge e, thus the probability of having an induced CP-net without the edge e, is equal to $1 - \mathbb{P}(e)$.

This probability can can be derived using the PCP-table of the child node. We consider an edge e, from a feature X to a feature Y. The set of Y's parent nodes are the features in the set $Pa(Y)$ with cardinality $k = |Pa(Y)|$. Let's take

$$\mathscr{E} = \bigotimes_{F \in (Pa(Y) \setminus X)} Dom(F)$$

where $Dom(F) = \{f, \bar{f}\}$ is the domain for a feature F.

Theorem 3.3.1 Given a PCP-net Q with features with binary domains, the probability of an edge $e = (X,Y)$ can be computed as follows:

$$\mathbb{P}(e) = 1 - \sum_{\sigma \in \Upsilon} \left(\prod_{v \in \mathscr{E}} \mathbb{P}[\sigma(v)|vx] \; \mathbb{P}[\sigma(v)|v\bar{x}] \right)$$

where $\Upsilon = \{\sigma : \mathscr{E} \to \{y > \bar{y}, \bar{y} > y\}\}$ is the set of all possible functions σ that associate to each element $v \in \mathscr{E}$ an element of the set $\{y > \bar{y}, \bar{y} > y\}$ (the set of all the possible total orders of the domain of Y).

Proof. We equivalently prove that the probability of the edge $e = (X,Y)$ not existing is:

$$\sum_{\sigma \in \Upsilon} \left(\prod_{v \in \mathscr{E}} \mathbb{P}[\sigma(v)|vx] \; \mathbb{P}[\sigma(v)|v\bar{x}] \right) .$$

To prove the formula we have to reason about the particular configurations of the PCP-table of Y that induce CP-nets that not have the edge e.

A general PCP-table for a feature Y corresponds to:

v_1x	$y > \bar{y}$ $\bar{y} > y$	p_{1x} $1 - p_{1x}$
$v_1\bar{x}$	$y > \bar{y}$ $\bar{y} > y$	$p_{1\bar{x}}$ $1 - p_{1\bar{x}}$
$\dots$	$\dots$	$\dots$
v_ix	$y > \bar{y}$ $\bar{y} > y$	p_{ix} $1 - p_{ix}$
$v_i\bar{x}$	$y > \bar{y}$ $\bar{y} > y$	$p_{i\bar{x}}$ $1 - p_{i\bar{x}}$
$\dots$	$\dots$	$\dots$
$v_{2^{k-1}}x$	$y > \bar{y}$ $\bar{y} > y$	$p_{2^{k-1}x}$ $1 - p_{2^{k-1}x}$
$v_{2^{k-1}}\bar{x}$	$y > \bar{y}$ $\bar{y} > y$	$p_{2^{k-1}\bar{x}}$ $1 - p_{2^{k-1}\bar{x}}$

where $v_1, \cdots, v_{2^{k-1}}$ are the assignments of the parent nodes in $Pa(Y) \setminus X$.

We define an *i-configuration* (with $i \in \{1, \cdots, 2^{k-1}\}$) as a particular combination of the rows in which, given a particular assignment v_i for the parents of Y different from X, the ordering for the values of the feature Y is the same for both x and $\bar{x}$. This corresponds to the definition of independence of Y to X given v_i: the ordering over the domain of Y is independent to the assignment of X given the assignment v_i for the features in $Pa(Y) \setminus X$, if the two rows of the CP-tables corresponding to v_ix and $v_i\bar{x}$ have the same ordering over the domain of Y.

We have two possible i-configurations for each v_i:

v_ix	$y > \bar{y}$	p_{ix}
$v_i\bar{x}$	$y > \bar{y}$	$p_{i\bar{x}}$

or

v_ix	$\bar{y} > y$	$1 - p_{ix}$
$v_i\bar{x}$	$\bar{y} > y$	$1 - p_{i\bar{x}}$

The first i-configuration has probability $p_{ix} \cdot p_{i\bar{x}}$ and the second one $(1 - p_{ix}) \cdot (1 - p_{i\bar{x}})$.

A *general configuration* correspond to a combination of *i-configurations*, each one defined on a particular assignment of the features in $Pa(Y) \setminus X$, and its probability corresponds to the product of the probability of the single i-configurations.

A feature Y is independent to another feature X if it is independent for each combination of assignment of $Pa(Y) \setminus X$. Thus a *general configuration* corresponds to a combination of *i-configurations* for all the assignment of $Pa(Y) \setminus X$. It is well defined a correspondence between the set of all the possible combinations of *i-configurations* and the set of function $\sigma : \mathscr{E} \to \{y > \bar{y}, \bar{y} > y\}$ that maps each element $v \in \mathscr{E}$ to a total order over the domain of Y.

We partition the set of induced CP-nets that not have the edge e in equivalence classes: each class correspond to the set of induced CP-nets corresponding to a *general configuration*, as described above. Thus in each class we are fixing the rows of the CP-table of Y and we let free to change the other CP-tables. Each class contains the same number of CP-nets: each CP-net has fixed the the rows of the CP-table of Y and the number of CP-nets in the equivalence class corresponds

to the number of all the possible values combination of the remaining CP-tables (the number of combination is the same, independently to the assignment of the Y CP-table).

The probability of a *general configuration* (the product of the probability of the chosen rows) correspond to the probability of the corresponding equivalence class. This because inside of a class we are considering all the possible combination for the free rows, that sum up to one.

The probability of a *general configuration* is defined by the formula:

$$\prod_{v \in \mathscr{E}} \mathbb{P}[\sigma(v)|vx]\mathbb{P}[\sigma(v)|v\bar{x}]$$

where the function σ associates to each v an element of the set $\{y > \bar{y}, \bar{y} > y\}$:

$$\sigma : \mathscr{E} \to \Theta(Y) \ .$$

Now we have to consider all the possible *general configuration* and to sum their probabilities. In this way we are summing the probability of the equivalence classes and so we obtain the probability of the whole partitioned set: the set of induced CP-net that not have the edge e in their dependency graph. We simply sum over all the possible functions σ, obtaining the initial formula:

$$\sum_{\sigma \in \Upsilon} \left(\prod_{v \in \mathscr{E}} \mathbb{P}[\sigma(v)|vx] \ \mathbb{P}[\sigma(v)|v\bar{x}] \right) \ .$$

■

The previous theorem can be generalized to PCP-net with features with non-binary domains as follow:

Theorem 3.3.2 The probability of the edge $e = (X, Y)$ can be computed as follows:

$$\mathbb{P}(e) = 1 - \sum_{\sigma \in \Upsilon} \left[\prod_{v \in \mathscr{E}} \left(\prod_{x_j \in Dom(X)} \mathbb{P}[\sigma(v)|vx_j] \right) \right]$$

where $\Theta(Y)$ is the set of all the possible total orders of the domain of Y and $\Upsilon = \{\sigma : \mathscr{E} \to \Theta(Y)\}$ is the set of all possible functions σ that associate to each element$v \in \mathscr{E}$ a total order over the domain of Y (an element of the set $\Theta(Y)$).

The proof follows step by step the proof of the binary case.

3.4 Optimality

Given a PCP-net we study mainly two optimality tasks: finding the most probable induced CP-net and finding the most probable optimal outcome. These two reasoning tasks have slightly different semantics and may be of use to different groups in the preference reasoning community. The most probable induced CP-net is analogous, in our Netflix example from earlier, to the CP-net that most likely represent a generic viewer in the household. The most probable optimal outcome would be what a recommendation engine should suggest to maximize the probability of a correct

recommendation. One can be seen as an aggregated model, that still retains usefulness for prediction and sampling, while the other is an aggregated outcome, that maximizes the probability of being correct.

3.4.1 The Most Probable Induced CP-net

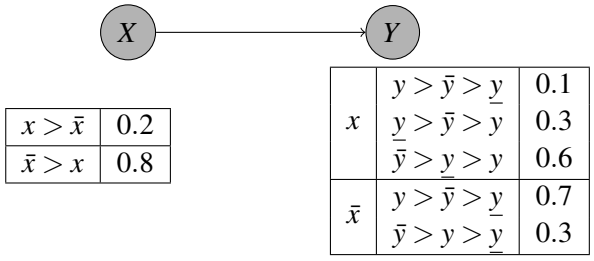
We reduce the problem of finding the most probable induced CP-net to that of finding an assignment with maximal joint probability of an appropriately defined BN.

Given a PCP-net $\mathscr{C}$, we define the BN called the *general network*, or *G-net($\mathscr{C}$)*, associated with $\mathscr{C}$, as follows. We create a variable for each independent feature A (without parents in the dependency graph) of the PCP-net, with domain equal to the set of all possible total orderings over the domain of A. The probability distribution over the orderings is given by the PCP-table of A. For each dependent feature B of the PCP-net, we add as many variables to the G-net as the possible assignments of the parents. Each of these variables, say X, will have the same domain, the set of total orderings over the domain of B.

Definition 3.4.1 Given a PCP-net $\mathscr{C}$, its *G-net* is a separable Bayesian network N such that:

- for each *independent node* $X^{\mathscr{C}}$ in $\mathscr{C}$ N has a node X^N; its domain $Dom(X^N)$ consists into all the orderings over the values in $Dom(X^{\mathscr{C}})$; the probability distribution over $Dom(X^N)$ correspond to the probability distribution over the orderings in the PCPTs of node $X^{\mathscr{C}}$ in $\mathscr{C}$.
- for each node *dependent node* $Y^{\mathscr{C}}$ in $\mathscr{C}$ N has nodes $Y_1^N, \cdots, Y_m^N$, where $m = |Dom(Z_1^{\mathscr{C}}) \times \cdots \times Dom(Z_l^{\mathscr{C}})|$ and $\{Z_1^{\mathscr{C}}, \cdots, Z_l^{\mathscr{C}}\} = Pa(Y^{\mathscr{C}})$; all the nodes $Y_1^N, \cdots, Y_m^N$ have the same domain $Dom(Y_1^N) = \cdots = Dom(Y_m^N)$ and consists into all the orderings over the values in $Dom(Y^{\mathscr{C}})$; the probability distribution over $Dom(Y_i^N)$ corresponds to the probability distribution over the orderings in the PCPTs of node $X^{\mathscr{C}}$ in $\mathscr{C}$ corresponding to the i-th assignment of the parent nodes $Pa(Y^{\mathscr{C}})$.

Observation 3.1 All the variables in the G-net are independent, this follows from the fact that we consider all the cp-statements as independent. The definition of the G-net could seem trivial in the context in which the the cp-statements are independent, but it make sense considering the typical kind of data that we are managing: a huge amount of information from a lot of individuals, each one giving only a few preferences, thus generally providing sparse data. Let think about recommender systems: usually each user gives informations on a small set of features or items and any other information on the others. For this reason, when we aggregate user preferences, we obtain that it clearly makes sense to maintain these rules independent from each other, since each rule is obtained from a different user. We plan to extend this scenario to a more rich framework. In this generalization the G-net assumes a relevant importance, since it expresses the relation between the cp-statements. The structure of the G-et now admits edges between the nodes, and these edges represents the connections between the cp-statements. For example we can observe that there are strong relations between the probability distributions of a variable considering the assignment of its parents, or the probability distributions associated to different variables. ■



$x > \bar{x}$		0.2
$\bar{x} > x$		0.8

x	$y > \bar{y} > \underline{y}$	0.1
	$\underline{y} > \bar{y} > y$	0.3
	$\bar{y} > \underline{y} > y$	0.6
$\bar{x}$	$y > \bar{y} > \underline{y}$	0.7
	$\bar{y} > y > \underline{y}$	0.3

Figure 3.7: A PCP-net $\mathscr{Q}$

■ **Example 3.4** Considering the PCP-net with two features X and Y with domains $Dom(X) = \{x, \bar{x}\}$ and $Dom(Y) = \{y, \bar{y}, \underline{y}\}$ in Figure 3.7, the corresponding G-net is shown in Figure 3.8. The variables have the following domains: $Dom(X) = \{x > \bar{x}, \bar{x} > x\}$, $Dom(Y_x) = Dom(Y_{\bar{x}}) = \{y > \bar{y} > \underline{y}, \underline{y} > \bar{y} > y, \bar{y} > \underline{y} > y, \bar{y} > y > \underline{y}\}$.

$x > \bar{x}$	0.2
$\bar{x} > x$	0.8

$y > \bar{y} > \underline{y}$	0.1
$\underline{y} > \bar{y} > y$	0.3
$\bar{y} > \underline{y} > y$	0.6
$\bar{y} > y > \underline{y}$	0

$y > \bar{y} > \underline{y}$	0.7
$\underline{y} > \bar{y} > y$	0
$\bar{y} > \underline{y} > y$	0
$\bar{y} > y > \underline{y}$	0.3

Figure 3.8: The G-net associated with $\mathscr{C}$.

■

We define a correspondence between each probability distribution in the PCP-net to a random variable (a node) in the G-net. In this way, a complete assignment of the G-net corresponds to a induced CP-net, since we are fixing a particular ordering for each variable in the dependency graph for all the assignment of the parents nodes.

Observation 3.2 Given a PCP-net $\mathscr{C}$ and the corresponding G-net N, there is a one-to-one correspondence between the assignments of N and the induced CP-nets of $\mathscr{C}$. ■

Observation 3.3 Given a PCP-net $\mathscr{C}$, the probability of realizing one of its induced CP-nets, say $\mathscr{C}_i$, is the joint probability of the corresponding assignment in the G-net for $\mathscr{C}$. We take exactly the product of the chosen rows in the PCP-tables. ■

Proposition 3.4.1 The probabilities over the induced CP-nets of a certain PCP-net form a probability distribution.

Proof. The probability defined in Observation 3.3 is computed as a product of non-negative factors, thus it is non-negative. Moreover, the sum of the probabilities of all the CP-nets in the set of the

induced CP-nets is equal to 1, because there's a 1-1 correspondence between the assignments of the G-net with positive (not zero) probability and the induced CP-nets, and the sum of the probabilities of all the assignments of a BN is equal to 1. ■

Observation 3.4 Given a PCP-net $\mathscr{C}$ and its induced CP-nets, the most probable of such induced CP-nets can be computed by computing the assignment with maximal joint probability of the G-net for $\mathscr{C}$. ■

Because the G-net has a separable dependency graph (there are no edges and all the node are independent), to compute the assignment with maximal joint probability takes $O(N)$ time, where N is the number of the node of the G-net.Given k the maximal number of parents nodes for each node and d the maximal cardinality of the domain of the nodes of the PCP-net, then $N = n \cdot d^k$ (n is the number of nodes in the PCP-net). We obtain the computational complexity $O(nd^k)$ that is polynomial in the size of the description of the PCP-net.

Once we have computed the most probable induced CP-net, we can compute its optimal outcome using a classical procedure of outcome optimization for CP-nets. In this way we find the *optimal outcome of the most probable induced CP-net*. This notion of outcome optimization is an alternative to the *most probable optimal outcome* that we define in the following section.

3.4.2 The Most Probable Optimal Outcome

The most probable optimal outcome is the outcome that occurs with the greatest probability as the optimal one in the set of induced CP-nets. The probability that an outcome is optimal corresponds to the sum of the probabilities of the CP-nets that have that outcome as optimal.

To reason about this task, first observe that the most probable optimal outcome may not be the optimal outcome of the most probable CP-net.

■ **Example 3.5** Let us consider a PCP-net with only one feature X with domain $D_X = \{x_1, x_2, x_3\}$ and the following PCP-table:

- $x_1 > x_2 > x_3$ with probability 0.3
- $x_1 > x_3 > x_2$ with probability 0.3
- $x_3 > x_2 > x_1$ with probability 0.4

The most probable CP-net is the one corresponding to the third ordering and it has the optimal outcome x_3. The other CP-nets have x_1 as optimal, so $\mathbb{P}(x_1) = 0.6$ and $\mathbb{P}(x_3) = 0.4$. The most probable optimal outcome is therefore x_1 but the optimal outcome of the most probable CP-net is x_3. ■

To find the most probable optimal outcome, we cannot use the most probable induced CP-net (by the G-net procedure described above) and then find its optimal outcome; we must make use of another BN. In order to find the most probable optimal outcome, we make use of another Bayesian network we call the *Opt-net*.

In general, given a PCP-net $\mathscr{C}$ the *optimals' network (Opt-net)* for $\mathscr{C}$ is a BN with the same dependencies graph as $\mathscr{C}$. Thus, the Opt-net has a variable for each of the PCP-net's features. The domains of the variables in the Opt-net are the values of the corresponding features that are

ranked first in at least one ordering with non-zero probability. The conditional probability tables of the Opt-net are obtained from the corresponding PCP-tables as follows: for each assignment of the parent variables, we consider the corresponding probability distribution over the values of the dependent variable defined in the PCP-table. The probability of a value for the dependent variable is the sum of the probabilities of the orderings that have that particular value as most preferred according to that distribution.

Definition 3.4.2 Given a PCP-net $\mathscr{C}$, its *G-net* is a separable Bayesian network N such that:

- for each node $X^{\mathscr{C}}$ in $\mathscr{C}$ N has a node X^N;
- the domain $Dom(X^N) \subseteq Dom(X^{\mathscr{C}})$ and consists into the union
- for each edge $(X^{\mathscr{C}}, Y^{\mathscr{C}})$ in $\mathscr{C}$ N has a an edge (X^N, Y^N);
- the probability distribution over $Dom(X^N)$ are defined as:

$$\mathbb{P}(w|u_i) = \sum_{j=1 \text{ and } o_j[1]=w}^{m} p_j^i \quad \forall w \in Dom(X^N)$$

where:

- $Pa(X^N) = \{Z_1^N, \cdots, Z_k^N\}$;
- $l = |Dom(Z_1^{\mathscr{C}}) \times \cdots \times Dom(Z_l^{\mathscr{C}})|$;
- $u_1, \cdots, u_l$ are all the possible assignment of $Pa(X^N)$;
- m is the number of possible total ordering of the values in $Dom(X^N)$: $o_1, \cdots, o_m$;
- if $o = v_1 > v_2 > \cdots > v_d$ then $o[1] = v_1$ is the first element in the ordering o;
- p_j^i is the probability in $\mathscr{C}$ of the ordering o_j given the assignment of the parents nodes u_i.

■ **Example 3.6** Consider the PCP-net $\mathscr{C}$ with three features A, B and C with domains $D_A = \{a_1, a_2\}$, $D_B = \{b_1, b_2\}$ and $D_C = \{c_1, c_2, c_3\}$ shown in Figure 3.9a.

The Opt-net has the same dependency graph as $\mathscr{C}$, with three variables A, B and C with domains: $D_A = \{a_1, a_2\}$, $D_B = \{b_1, b_2\}$ and $D_C = \{c_1, c_2\}$, and two edges AC and BC. The domain of variable C in the Opt-net does not contain value c_3 because it never appears as most preferred in any ordering. Therefore, the Opt-net has a table for entry a_1b_2 where c_1 appears with probability 0.2 and c_2 appears with probability 0.8. The Opt-net is shown in Figure 3.9b.

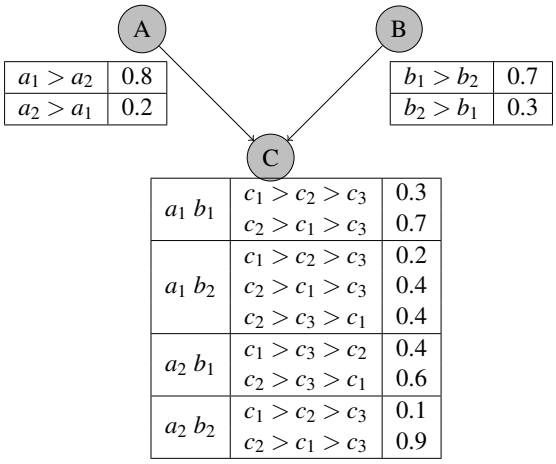
■

Observation 3.5 Given a PCP-net $\mathscr{C}$ and the Opt-net, there is a one-to-one correspondence between the assignments (with non-zero probability) of the Opt-net and the outcomes that are optimal in at least one induced CP-net of $\mathscr{C}$. ■

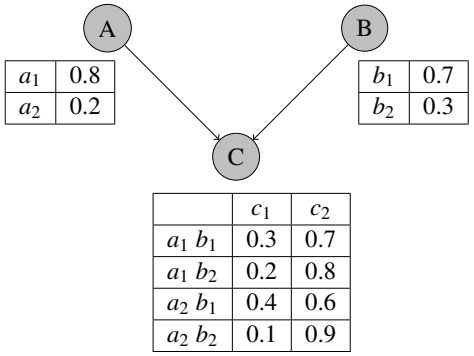
From here we consider only the assignments of the Opt-net with positive probability.

Proposition 3.4.2 Given a PCP-net $\mathscr{C}$, the probability that an outcome is optimal is the joint probability of the corresponding assignment in the Opt-net. If no such corresponding assignment exists, then the probability of being optimal is 0.

Proof. By construction, the set of assignments of the Opt-net is a subset of those of $\mathscr{C}$. By the



(a) A PCP-net $\mathscr{C}$.



(b) The Opt-net associated with $\mathscr{C}$.

Figure 3.9: A PCP-net and its associated Opt-net

definition of the Opt-net, if an assignment of $\mathscr{C}$ is not an assignment of the Opt-net then it cannot be optimal in any induced CP-net.

Let us now focus on the assignments of $\mathscr{C}$ that have a corresponding assignment in the Opt-net. Let $x = (x_1, x_2, ..., x_n)$ be one of these assignments. We denote by $P_{opt}(x)$ the joint probability of x, $\mathbb{P}(X_1 = x_1, ..., X_n = x_n)$ in the Opt-net. We recall that the probability that x is optimal in the PCP-net is the sum of the probabilities of the induced CP-nets that have assignment x as optimal. We call this probability $P_{cp}(x)$. We must prove that $P_{opt}(x) = P_{cp}(x)$.

Let us consider A_x, the set of induced CP-nets that have x as their optimal assignment. We have $P_{cp}(x) = \sum_{\mathscr{C} \in A_x} \mathbb{P}(\mathscr{C})$.

When we compute the optimal value for a CP-net, we sweep forward, starting from the independent features, assigning features their most preferred value. This means that only one subset of the rows of the CP-tables is considered when computing the optimal outcome. We can thus split a CP-net $\mathscr{C}$ into two parts, one affecting the choice of the optimal outcome (denoted with $\mathscr{C}_*$) and one not involved in it (denoted with $\mathscr{C}_{-*}$). If we consider the probability that that CP-net is induced by the PCP-net, we see that these two parts are independent. Thus we have $P_{cp}(x) = \sum_{\mathscr{C} \in A_x} \mathbb{P}(\mathscr{C}) = \sum_{\mathscr{C} \in A_x} \mathbb{P}(\mathscr{C}_*)\mathbb{P}(\mathscr{C}_{-*})$.

Regarding $\mathscr{C}_*$, observe that the optimal outcome x can be produced in many different ways, as there can be many different orderings that produce the same result. For example, the orderings $a_1 > a_2 > a_3$ and $a_1 > a_3 > a_4$ produce both the optimal value a_1 for variable X_1. So we can do a disjoint partition of the set A_x into k subsets $A_{x_1}, ..., A_{x_k}$ for some k.

Two CP-nets $\mathscr{C}$ and D that belong to the same A_{x_i} are equal in the part that actively affects the choice of the optimal value and different in the other parts: $\mathscr{C}_* = D_*$ and $\mathscr{C}_{-*} \neq D_{-*}$.

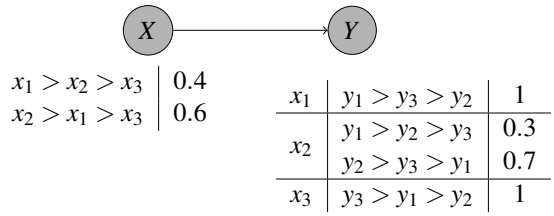
Let $\mathscr{C}^i_*$ be the part that is equal for all the members of A_{x_i}. The probability becomes: $P_{cp}(x) = \sum_{i=1}^{k} \mathbb{P}(\mathscr{C}^i_*) \sum_{\mathscr{C} \in A_{x_i}} \mathbb{P}(\mathscr{C}_{-*})$. We note that $\sum_{\mathscr{C} \in A_{x_i}} \mathbb{P}(\mathscr{C}_{-*}) = 1\ \forall i = 1, ..., k$, since we are summing the probability of all possible cases regarding $\mathscr{C}_{-*}$. Thus, the probability becomes $P_{cp}(x) = \sum_{i=1}^{k} \mathbb{P}(\mathscr{C}^i_*)$.

However, we have $\mathbb{P}(X_1 = x_1, ..., X_n = x_n) = \sum_{i=1}^{k} \mathbb{P}(\mathscr{C}_*^i)$ and, thus, $P_{cp}(x) = P_{opt}(x)$. since we built the rows of the probability tables for the variables $X_1, ..., X_n$ by summing the probability of the orderings that have the same head. This is the same as summing the probabilities over the subset A_{x_i}. ■

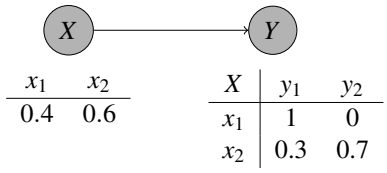
To find the most probable optimal outcome for a PCP-net $\mathscr{C}$, it is sufficient to compute the assignment with the maximal joint probability on the Opt-net.

If we have evidence on the variables, for instance an assignment for a subset of features, we do not have to re-build the entire structure of the Opt net. We need only to re-build the connected components of the features affected (all the nodes that can be reached by a direct path that starts from a node with evidence), applying evidence.

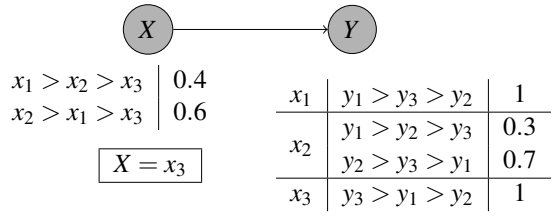
■ **Example 3.7** We consider the PCP-net over two features X and Y with domains $Dom(X) = \{x_1, x_2, x_3\}$ and $Dom(Y) = \{y_1, y_2, y_3\}$ and the associated Opt-net over two features X and Y with domains $Dom(X) = \{x_1, x_2\}$ and $Dom(Y) = \{y_1, y_2\}$ described in Figure 3.10. Considering the given evidence of $X = x_3$, we obtain the PCP-net with domains $Dom(X) = \{x_1, x_2, x_3\}$ and $Dom(Y) = \{y_1, y_2, y_3\}$ and the associated Opt-net over two features X and Y with domains $Dom(X) = \{x_1, x_2, x_3\}$ and $Dom(Y) = \{y_1, y_2, y_3\}$ described in Figure 3.10.



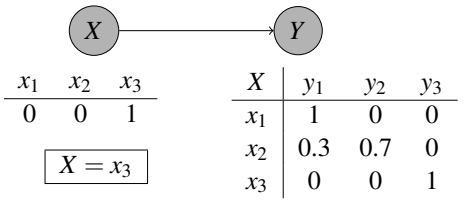
(a) PCP-net.



(b) Opt-net.



(c) PCP-net with evidence $X = x_3$.



(d) Opt-net with evidence $X = x_3$.

Figure 3.10: A PCP-net and its associated Opt-net, with and without evidence $X = x_3$.

■

Observation 3.6 We focus on the most probable optimal outcome and the optimal outcome of the most probable induced CP-net since both of them can be computed in polynomial time. Other notions of optimality, concerning outcome optimization, could be defined, such as the most preferred outcome. The most preferred outcome can be computed giving a score to each outcome, based on the position of its ranking in an induced CP-net, and then taking the outcome with the highest score among the profile of induced CP-nets. This procedure is similar to the Borda voting rule and requires the explicit computation of the whole preference order over the set of possible outcomes, for each agent. Since the number of possible outcomes is exponential in the number of features (in the case of n binary variables, the number of possible outcomes is 2^n), and the cardinality of the induced profile of CP-nets is exponential in the PCP-net description, then also this procedure becomes exponential. Thus we focus on the most probable optimal outcome and the optimal outcome of the most probable induced CP-net, notions of outcome optimization that can be computed in a tractable amount of time. ■

3.5 Dominance

We investigate the problem of dominance queries for binary-valued acyclic PCP-nets. Given a binary-valued and acyclic PCP-net $\mathscr{Q}$ and two outcomes o and o', the goal is to compute the probability of:

$$\mathbb{P}(o > o')$$

First we consider the rigorous definition of dominance for PCP-nets, recalling the definition for CP-nets.

Definition 3.5.1 — Dominance for CP-nets. Given a CP-net C and a pair of outcomes o and o', a *dominance query* asks whether $C \models o > o'$. If this relation holds, o is preferred o', and we say that o dominates o' with respect to C.

Definition 3.5.2 — Dominance for PCP-nets. Given a PCP-net $\mathscr{Q}$ and a pair of outcomes o and o', a *dominance query* asks for the probability that $\mathscr{Q} \models o > o'$. That, we compute:

$$\mathbb{P}_{\mathscr{Q}}(o > o') = \sum_{\substack{C \text{ induced by } \mathscr{Q} \\ C \models o > o'}} \mathbb{P}(C)$$

If we consider a CP-net C as a particular PCP-net with all probabilities in the PCP-table belonging to the set $\{0,1\}$ then the result of the dominance query is a probability that belongs to the set $\{0,1\}$ too, thus 0 correspond to $C \not\models o > o'$ and 1 corresponds to $C \models o > o'$

Computational complexity of dominance queries

Bigot et al. [9] prove that dominance for PCP-nets is #P-hard, even if the structure is acyclic, the longest path has length 3 and each node has at most one outgoing edge and at most 4 parents. They also give an algorithm to obtain dominance for a binary-valued tree structured PCP-net that takes $O(2^{2k^2}n)$ where n is the number of features and k is the number of variables which have a different value in o and o'.

A CP-net is a particular PCP-net with all probabilities in the PCP-table set to 0 or 1. Thus a CP-net can be seen as a special case of PCP-net.

Dominance testing for binary-valued CP-nets is:

- $O(n)$ for tree-structured CP-nets [9, 11]
- polynomial for polytree-structured CP-nets [11]
- NP-complete for directed-path singly connected acyclic CP-nets [11]
- NP-complete for CP-nets with a bounded number of pathsbetween two nodes [11]
- PSPACE-complete for general CP-nets [52]

To compute the probability of dominance for separable PCP-nets (that have no edges in their dependency graph) is polynomial, and there is an exact formula to compute it. Because to compute the exact value of the dominance for a more structured PCP-net is hard, we compute an interval in which the exact value belongs. Thus we compute a lower bound and an upper bound in polynomial time.

3.5.1 Separable PCP-nets

We analyze now how to compute the dominance in the case of separable PCP-nets. We provide results for a binary-domains setting. The generalization to non-binary domains is an easy extension.

Given two outcomes o and o' we generate two sets $Diff^+(o,o')$ and $Diff^-(o,o')$ such that:

- $Diff^+ = \{i \in \{1,\cdots,n\} | x_i \in o \text{ and } \bar{x}_i \in o'\}$
- $Diff^- = \{i \in \{1,\cdots,n\} | \bar{x}_i \in o \text{ and } x_i \in o'\}$

The probability of the dominance can be computed analytically by the following formula:

$$\mathbb{P}(o \succ o') = \left(\prod_{i \in Diff^+(o,o')} p_i \right) \cdot \left(\prod_{j \in Diff^-(o,o')} (1-p_j) \right)$$

with the convention that for a feature X_i with domain $\mathscr{D}_i = \{x_i, \bar{x}_i\}$ we have $p_i = \mathbb{P}(x_i \succ \bar{x}_i)$ in the PCP-net (and thus $(1-p_i) = \mathbb{P}(\bar{x}_i \succ x_i)$).

Observation 3.7 The formula is a product of at most n factors. The computation of each factor is $O(1)$, hence the formula takes $O(n)$ computation time. ∎

Proposition 3.5.1 Given a PCP-net $\mathscr{Q}$ over n feature $X_1,\cdots,X_n$ with binary domains $\mathscr{D}_i = \{x_i, \bar{x}_i\} \forall i \in \{1,\cdots,n\}$ and given two outcome o and o' then

$$\mathbb{P}(o \succ o') = \left(\prod_{i \in Diff^+(o,o')} p_i \right) \cdot \left(\prod_{j \in Diff^-(o,o')} (1-p_j) \right)$$

with the convention that for a feature X_i with domain $\mathscr{D}_i = \{x_i, \bar{x}_i\}$ we have $p_i = \mathbb{P}(x_i \succ \bar{x}_i)$ in the PCP-net (and thus $(1-p_i) = \mathbb{P}(\bar{x}_i \succ x_i)$).

Proof. We call $\mathscr{I}(\mathscr{Q})$ the set of CP-nets induced by $\mathscr{Q}$, and $\mathscr{O}(C)$ the partial order defined by a

CP-net C. We know that

$$\mathbb{P}(o \succ o') = \sum_{\substack{C \in \mathscr{I}(\mathscr{Q}) \text{ s.t.} \\ (o \succ o') \in \mathscr{O}(C)}} \mathbb{P}(C)$$

Now we show that:

1. if $i \in Diff^+(o,o')$ then the induced CP-nets that have $\bar{x}_i \succ x_i$ don't have $o \succ o'$ in their partial order among the features ($o \not\succ o'$).
2. if $i \in Diff^-(o,o')$ then the induced CP-nets that have $x_i \succ \bar{x}_i$ don't have $o \succ o'$ in their partial order among the features ($o \not\succ o'$).

We prove only the first assertion because the second is symmetric.

We suppose that $i \in Diff^+(o,o')$. We must have a chain of worsening flip from o to o', in such a way that $o \succ o'$.

We consider $o = (yx_iz)$ and $o' = (y'\bar{x}_iz')$ where y and y' are the assignments of the variables $x_1, \cdots, x_{i-1}$ and z and z' are the assignments of the variables $x_{i+1}, \cdots, x_n$ respectively of o and o'.

Because of the features are all indipendent, to have a chain of worsening flip from o to o' we must have a chain of worsening flip from y to y', from z to z' and for x_i to $\bar{x}_i$. But, in some induced CP-nets that have $\bar{x}_i \succ x_i$, the first two may be possible, the third one is an improvement. So there not exist a chain of worsening flip from o to o' in the induced CP-nets that have $\bar{x}_i \succ x_i$.

Thus we have to consider only the induced CP-net that for all the features $i \in Diff^+(o,o')$ have $x_i \succ \bar{x}_i$, and for all the features $i \in Diff^-(o,o')$ have $\bar{x}_i \succ x_i$.

$$\mathbb{P}(o \succ o') = \sum_{\substack{C \in \mathscr{I}(\mathscr{Q}) \text{ s.t.} \\ (x_i \succ \bar{x}_i) \forall i \in Diff^+(o,o') \\ (\bar{x}_i \succ x_i) \forall i \in Diff^-(o,o')}} \mathbb{P}(C)$$

We now observe that all of these induced CP-nets have $(o \succ o') \in \mathscr{O}(C)$. That is because the variables $x_j \notin (Diff^+(o,o') \cup Diff^-(o,o'))$ have se same values in o and o', and so we can build always a chain of worsening flip for the variables $x_j \in (Diff^+(o,o') \cup Diff^-(o,o'))$ by definition, and the others are equal.

So we can compute the probability $\mathbb{P}(o \succ o')$:

$$\begin{aligned}
\mathbb{P}(o \succ o') &= \sum_{\substack{C \in \mathscr{I}(\mathscr{Q}) \text{ s.t.} \\ (x_i \succ \bar{x}_i) \\ \forall i \in Diff^+(o,o') \\ (\bar{x}_i \succ x_i) \\ \forall i \in Diff^-(o,o')}} \mathbb{P}(C) \\
&= \sum_{\substack{C \in \mathscr{I}(\mathscr{Q}) \text{ s.t.} \\ (x_i \succ \bar{x}_i) \\ \forall i \in Diff^+(o,o') \\ (\bar{x}_i \succ x_i) \\ \forall i \in Diff^-(o,o')}} \left(\prod_{j \text{ s.t. } x_j \succ \bar{x}_j} p_j \prod_{j \text{ s.t. } \bar{x}_j \succ x_j} (1-p_j) \right) \\
&= \prod_{i \in Diff^+(o,o')} p_i \prod_{i \in Diff^-(o,o')} (1-p_i) \sum_{\substack{C \in \mathscr{I}(\mathscr{Q}) \text{ s.t.} \\ (x_i \succ \bar{x}_i) \\ \forall i \in Diff^+(o,o') \\ (\bar{x}_i \succ x_i) \\ \forall i \in Diff^-(o,o')}} \left(\prod_{\substack{j \text{ s.t. } x_j \succ \bar{x}_j \\ j \notin Diff^+(o,o')}} p_j \prod_{\substack{j \text{ s.t. } \bar{x}_j \succ x_j \\ j \notin Diff^-(o,o')}} (1-p_j) \right)
\end{aligned} \tag{3.1}$$

But we know that:

$$\sum_{\substack{C \in \mathscr{I}(\mathscr{Q}) \text{ s.t.} \\ (x_i \succ \bar{x}_i) \\ \forall i \in Diff^+(o,o') \\ (\bar{x}_i \succ x_i) \\ \forall i \in Diff^-(o,o')}} \left(\prod_{\substack{j \text{ s.t. } x_j \succ \bar{x}_j \\ j \notin Diff^+(o,o')}} p_j \prod_{\substack{j \text{ s.t. } \bar{x}_j \succ x_j \\ j \notin Diff^-(o,o')}} (1-p_j) \right) = 1$$

Then we have that:

$$\mathbb{P}(o \succ o') = \left(\prod_{i \in Diff^+(o,o')} p_i \right) \left(\prod_{i \in Diff^-(o,o')} (1-p_i) \right)$$

■

■ **Example 3.8** Now we show an example of how to compute the dominance for separable PCP-nets. Let's take a separable PCP-net $\mathscr{Q}$ with five nodes: $X_1, \cdots, X_5$. The PCP-table are described in Figure 3.11.

We consider two different outcomes: $o = x_1x_2\bar{x}_3x_4x_5$ and $o' = x_1\bar{x}_2x_3x_4\bar{x}_5$. We obtain that $\mathbb{P}(o > o') = \mathbb{P}(x_2 > \bar{x}_2) \cdot \mathbb{P}(\bar{x}_3 > x_3) \cdot \mathbb{P}(x_5 > \bar{x}_5) = 0.4 \cdot 0.9 \cdot 0.7 = 0.252$ given $Diff^+(o,o') = \{2,5\}$ and $Diff^-(o,o') = \{3\}$. ■

3.5.2 A lower bound for the dominance value

We consider only binary-valued polytree PCP-nets, that are directed-path singly connected in which each node has at most one parent in common with another node. This category includes directed trees.

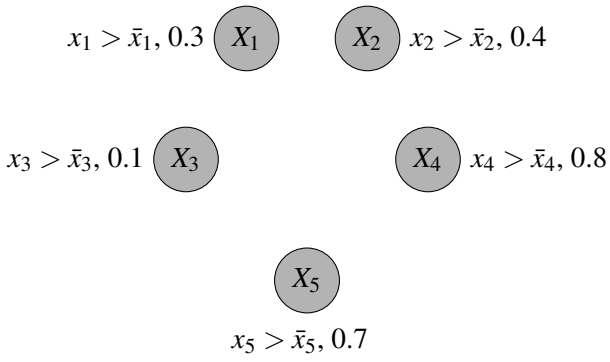


Figure 3.11: PCP-net $\mathscr{Q}$ of Example 3.8

We suppose also to have an order O for the nodes $X_1, \cdots, X_n$ such that each node X_i has only parents with a lower index: $Pa(X_i) \subseteq \{X_1, \cdots, X_{i-1}\}$.

Notation: Given an outcome $o = o_1 o_2 \cdots o_n$ we use the notation $o \restriction_{X_i}$ to denote o_i and $o \restriction_S = o \restriction_{S_1,\cdots,S_m}$, where $S = \{S_1, \cdots, S_m\} \subseteq \{X_1, \cdots, X_n\}$ and if $S_i \in S$ also $Pa(S_i) \in S$, to denote $o \restriction_{s_1} \cdots o \restriction_{s_m}$. Given a CP-net C and two outcomes o and o', when we use the notation $o \restriction_S > o \restriction_S$ we mean that the comparison is considered in the sub-CP-net C' that have the nodes in S and the arcs in C that connect them, with the corresponding CP-tables.

Definition 3.5.3 Given a PCP-net $\mathscr{Q}$ with n variables $X_1, \cdots, X_n$ and given two outcomes o and o', we define the set $I_{o,o'}$ (just I i what follow to simplify notation) of the indices of nodes that have different value on o and o'. The dominance lower bound for $o > o'$ in $\mathscr{Q}$ is:

$$\mathbb{P}^L_{\mathscr{Q}}(o > o') = \prod_{i \in I}(1 - p_i)$$

where $p_i \; \forall i \in I$ is defined as:

$$p_i = \begin{cases} \mathbb{P}(o' \restriction_{X_i} > o \restriction_{X_i} \forall u \in U_i^{o,o'}) & \text{if } Pa(X_i) \neq \emptyset \\ \mathbb{P}(o' \restriction_{X_i} > o \restriction_{X_i}) & \text{if } Pa(X_i) = \emptyset \end{cases}$$

where u is an assignment of the parents of X_i that belongs to $U_i^{o,o'}$ that is the set of all u assignments of $Pa(X_i)$ such that: given $Y \in Pa(X_i)$, if $o \restriction_Y = o' \restriction_Y$ then $u \restriction_Y = o \restriction_Y = o' \restriction_Y$ otherwise, if $o \restriction_Y \neq o' \restriction_Y$ then $u \restriction_Y \in \{y, \bar{y}\}$.

We observe that the formulation of the p_i coefficients is equivalent to:

$$p_i = \begin{cases} \prod_{u \in U_i^{o,o'}} \mathbb{P}(o' \restriction_{X_i} > o \restriction_{X_i} | u) & \text{if } Pa(X_i) \neq \emptyset \\ \mathbb{P}(o' \restriction_{X_i} > o \restriction_{X_i}) & \text{if } Pa(X_i) = \emptyset \end{cases}$$

Theorem 3.5.2 Given a PCP-net $\mathscr{Q}$, the formula defined in Definition 3.5.3:

$$\mathbb{P}^{L}_{\mathscr{Q}}(o > o') = \prod_{i \in I}(1 - p_i)$$

returns a lower bound for the probability of the dominance as defined in Definition 3.5.2:

$$\mathbb{P}_{\mathscr{Q}}(o > o') = \sum_{\substack{C \text{ induced by } \mathscr{Q} \\ C \models o > o'}} \mathbb{P}(C)$$

To prove this theorem we need the following propositions.

Proposition 3.5.3 Given an acyclic CP-net C with n nodes and two outcomes $o = o_1 o_2 \cdots o_n$ and $o' = o'_1 o'_2 \cdots o'_n$ such that $o > o'$ and given i an index in $\{1, \cdots, n\}$ such that $o_i \neq o'_i$ and $o_j = o'_j \forall j < i$ then

$$o \restriction_{X_1 \cdots X_j} > o' \restriction_{X_1 \cdots X_j} \qquad \forall j \geq i$$

Proof. Because of $o > o'$ there exists a worsening path from o to o': $P = o, o_1, \cdots, o'$. We prove that $P \restriction_{X_1,\cdots,X_j} = o \restriction_{X_1,\cdots,X_j}, o_1 \restriction_{X_1,\cdots,X_j}, \cdots, o' \restriction_{X_1,\cdots,X_j}$ is a worsening path from $o \restriction_{X_1,\cdots,X_j}$ to $o' \restriction_{X_1,\cdots,X_j}$. We have to prove that each step is a worsening flip. Considering a flip of the value of the variable X_l. It is independent from the values of the children or from the values of the nodes that are independent to X_l. All the nodes $X_{j+1}, \cdots, X_n$ are either children or nodes independent to X_l, thus, removing that variables it is irrelevant for the variables $X_1, \cdots, X_j$. Thus if a step is a worsening flip in the path P, it is also a worsening flip in the path $P \restriction_{X_1,\cdots,X_j}$. ■

We observe that the same proposition holds also for '$<$' simply renaming the outcomes.

Proposition 3.5.4 Given an acyclic CP-net C with n nodes and two outcomes o and o' such that exists index $i \in \{1, \cdots, n\}$ such that $o \restriction_{X_1 \cdots X_i} \bowtie o' \restriction_{X_1 \cdots X_i}$ then given each index j such that $i \leq j \leq n$ then $o \restriction_{X_1 \cdots X_j} \bowtie o' \restriction_{X_1 \cdots X_j}$

Proof. We prove the proposition by contradiction. Suppose that $o \restriction_{X_1 \cdots X_j} \not\bowtie o' \restriction_{X_1 \cdots X_j}$, then either $o \restriction_{X_1 \cdots X_j} > o' \restriction_{X_1 \cdots X_j}$ or $o \restriction_{X_1 \cdots X_j} < o' \restriction_{X_1 \cdots X_j}$. We suppose to have $o \restriction_{X_1 \cdots X_j} > o' \restriction_{X_1 \cdots X_j}$ (the other case is analogous). Since $o \restriction_{X_1 \cdots X_j} > o' \restriction_{X_1 \cdots X_j}$ there exists a worsening path P from $o \restriction_{X_1 \cdots X_j}$ to $o' \restriction_{X_1 \cdots X_j}$ and $P = o \restriction_{X_1 \cdots X_j}, o_1 \restriction_{X_1 \cdots X_j}, \cdots, o' \restriction_{X_1 \cdots X_j}$. For Proposition 3.5.3 we have that $\forall k \in \{1, \cdots, j\}$ $o \restriction_{X_1,\cdots,X_k} \geq o \restriction_{X_1,\cdots,X_k}$ because the path $P \restriction_{X_1,\cdots,X_k} = o \restriction_{X_1,\cdots,X_k}, o_1 \restriction_{X_1,\cdots,X_k}, \cdots, o' \restriction_{X_1,\cdots,X_k}$ is a worsening path from $o \restriction_{X_1,\cdots,X_k}$ to $o' \restriction_{X_1,\cdots,X_k}$ in the sub-CP-net of the nodes $X_1, \cdots, X_k$. In particular for $k = i$, thus $o \restriction_{X_1 \cdots X_i} > o' \restriction_{X_1 \cdots X_i}$ that is a contradiction. ■

Corollary 3.5.5 Given an acyclic CP-net C with n nodes and two outcomes o and o' such that exists index $i \in \{1, \cdots, n\}$ such that $o \restriction_{X_1 \cdots X_i} \bowtie o' \restriction_{X_1 \cdots X_i}$ then $o \bowtie o'$.

Proposition 3.5.6 Given an acyclic CP-net C with n nodes and two outcomes o and o' then $o \bowtie o'$ if and only if exists index $i \in \{1, \cdots, n\}$ such that

$$o \restriction_{X_1 \cdots X_j} \not\bowtie o' \restriction_{X_1 \cdots X_j} \quad \forall j < i \quad \text{and} \quad o \restriction_{X_1 \cdots X_l} \bowtie o' \restriction_{X_1 \cdots X_l} \quad \forall l \geq i$$

Proof. $\Leftarrow$ It is obvious for Proposition 3.5.4.

$\Rightarrow$ We suppose to have $o \bowtie o'$ and we prove that exists index i such that

$$o \restriction_{X_1 \cdots X_j} \not\bowtie o' \restriction_{X_1 \cdots X_j} \quad \forall j < i \quad \text{and} \quad o \restriction_{X_1 \cdots X_l} \bowtie o' \restriction_{X_1 \cdots X_l} \quad \forall l \geq i$$

by induction on the number of variables.

Base case: The base case for the induction consider a number of variables equals to two. That because in the case with only one variable there not exist two incomparable outcomes.

We suppose to have two variables A and B. In the case in which there is an edge between them, there are not incomparable outcomes. Thus we consider only the cases in which there is no edge between the two variables. We have four cases for the CP-tables:

- $a > \bar{a}$ and $b > \bar{b}$ that implies that $a\bar{b} \bowtie \bar{a}b$
- $a > \bar{a}$ and $\bar{b} > b$ that implies that $\bar{a}\bar{b} \bowtie ab$
- $\bar{a} > a$ and $b > \bar{b}$ that implies that $\bar{a}\bar{b} \bowtie ab$
- $\bar{a} > a$ and $\bar{b} > b$ that implies that $a\bar{b} \bowtie \bar{a}b$

In each of them we obtain $i = 1$.

Induction step: We suppose true for n and we prove for $n+1$.

We consider a CP-net with $n+1$ variables in which $o \bowtie o'$. We consider $o \restriction_{X_1,\cdots,X_n}$ and $o' \restriction_{X_1,\cdots,X_n}$. We can have three cases:

1. $o \restriction_{X_1,\cdots,X_n} > o' \restriction_{X_1,\cdots,X_n}$
2. $o \restriction_{X_1,\cdots,X_n} < o' \restriction_{X_1,\cdots,X_n}$
3. $o \restriction_{X_1,\cdots,X_n} \bowtie o' \restriction_{X_1,\cdots,X_n}$

In the case 3 we can apply the induction hypothesis. The cases 1 and 2 are symmetric so we can prove only the case 1. We observe that $o \restriction_{X_1,\cdots,X_n} \neq o' \restriction_{X_1,\cdots,X_n}$, because, in that case, the two outcome o and o' are a flipping couple, that is never incomparable.

So we consider the case in which $o \bowtie o'$ and $o \restriction_{X_1,\cdots,X_n} > o' \restriction_{X_1,\cdots,X_n}$. We have to prove that $\forall j < n$ $o \restriction_{X_1,\cdots,X_j} > o' \restriction_{X_1,\cdots,X_j}$. We prove it by contradiction.

- We suppose that $\exists j < n$ such that $o \restriction_{X_1,\cdots,X_j} \bowtie o' \restriction_{X_1,\cdots,X_j}$. We apply Proposition 3.5.4 and we obtain that also $o \restriction_{X_1,\cdots,X_n} \bowtie o' \restriction_{X_1,\cdots,X_n}$ that is a contradiction.
- We suppose that $\exists j < n$ such that $o \restriction_{X_1,\cdots,X_j} < o' \restriction_{X_1,\cdots,X_j}$. Thus there is a worsening path P from $o \restriction_{X_1,\cdots,X_n}$ to $o' \restriction_{X_1,\cdots,X_n}$. Considering P restricted to the variables $X_1,\cdots,X_j$ we obtain $\bar{P}$ that is a worsening path from $o \restriction_{X_1,\cdots,X_j}$ to $o' \restriction_{X_1,\cdots,X_j}$ that is a contradiction.

■

Proposition 3.5.7 Given an acyclic CP-net C with n nodes and suppose that C is formed by two connected components C_1 and C_2 and given two different outcomes o and o' such that

$$(o \restriction_{C_1} > o' \restriction_{C_1} \quad \text{and} \quad o \restriction_{C_2} < o' \restriction_{C_2}) \quad \text{or} \quad (o \restriction_{C_1} < o' \restriction_{C_1} \quad \text{and} \quad o \restriction_{C_2} > o' \restriction_{C_2})$$

then $o \bowtie o'$.

Proof. We prove only the case in which $o \restriction_{C_1} > o' \restriction_{C_1}$ and $o \restriction_{C_2} < o' \restriction_{C_2}$, because the other one is symmetric.

We suppose by contradiction that $o \not\bowtie o'$. Then we can have either $o > o'$ or $o < o'$. We suppose to have $o > o'$ (the other case is analogous). Since $o > o'$ there exists a worsening path P from o to o', $P = o, o_1, o_2, \cdots, o'$. Each step o_i of the path correspond to a worsening flip of a variable X_j that belongs either to C_1 or to C_2.Thus the two sub-path P_1 and P_2 such that:

$$P_1 = P \restriction_{C_1} = o \restriction_{C_1}, o_1 \restriction_{C_1}, o_2 \restriction_{C_1}, \cdots, o' \restriction_{C_1} \quad \text{and} \quad P_2 = P \restriction_{C_2} = o \restriction_{C_2}, o_1 \restriction_{C_2}, o_2 \restriction_{C_2}, \cdots, o' \restriction_{C_2}$$

are path (with repetition) such that P_1 is a worsening path from $o \restriction_{C_1}$ to $o' \restriction_{C_1}$ in the CP-net C_1 and P_2 is a worsening path from $o \restriction_{C_2}$ to $o' \restriction_{C_2}$ in the CP-net C_2. We obtain that $o \restriction_{C_1} > o' \restriction_{C_1}$ and $o \restriction_{C_2} > o' \restriction_{C_2}$ that is a contradiction. ■

We can generalize Proposition 3.5.7 in the following proposition.

Proposition 3.5.8 Given an acyclic CP-net C with n nodes and suppose that C is formed by m connected components $C_1, C_2, \cdots, C_m$ and given two different outcomes o and o' such that exists two indexes $i \neq j \in \{1, \cdots, n\}$ such that

$$(o \restriction_{C_i} > o' \restriction_{C_i} \quad \text{and} \quad o \restriction_{C_j} < o' \restriction_{C_j}) \quad \text{or} \quad (o \restriction_{C_i} < o' \restriction_{C_i} \quad \text{and} \quad o \restriction_{C_j} > o' \restriction_{C_j})$$

then $o \bowtie o'$.

Definition 3.5.4 Given two outcome o and o' such that $o \bowtie o'$ in a CP-net C then we say that the index i is the *discriminator* for the incomparability (or that o and o' are incomparable for index i) if i is the minimum index such that

$$o \restriction_{X_1 \cdots X_j} \bowtie o' \restriction_{X_1 \cdots X_j} \forall j \geq i \text{ and } o \restriction_{X_1 \cdots X_j} \not\bowtie o' \restriction_{X_1 \cdots X_j} \forall j < i$$

Now we can prove the Theorem 3.5.2.

Proof. (Theorem 3.5.2) Given o and o' such that $o = (x_1 x_2 \cdots x_n)$ and $o' = y_1 y_2 \cdots y_n$ the formula can be written as:

$$(1 - p_{i_0})(1 - p_{i_1}) \cdots (1 - p_{i_m})$$

where i_j are the ordered index that belongs to the set I of the index if the variables that change value from o to o'. Computing some products we obtain the following equivalent formulation:

$$[(1-p_{i_0})] - [(1-p_{i_0})(p_{i_1})] - [(1-p_{i_0})(1-p_{i_1})(p_{i_2})] - \cdots - [(1-p_{i_0})(1-p_{i_1}) \cdots (1-p_{i_{m-1}})(p_{i_m})]$$

Considering the first term: $1 - p_{i_0}$ If the variable X_{i_0} is independent then $p_{i_0} = \mathbb{P}(o' \restriction_{X_{i_0}} > o \restriction_{X_{i_0}})$ then $(1 - p_{i_0}) = \mathbb{P}(o \restriction_{X_{i_0}} > o' \restriction_{X_{i_0}})$. Otherwise the set $U_{i_0}^{o,o'}$ has only an element u. Thus $p_{i_0} = \mathbb{P}(o' \restriction_{X_{i_0}} > o \restriction_{X_{i_0}} | u)$ then $(1 - p_{i_0}) = \mathbb{P}(o \restriction_{X_{i_0}} > o' \restriction_{X_{i_0}} | u)$ So, we call q_{i_0} the probability $\mathbb{P}(o \restriction_{X_{i_0}} > o' \restriction_{X_{i_0}})$ (or $\mathbb{P}(o \restriction_{X_{i_0}} > o' \restriction_{X_{i_0}} | u)$). Then the formula is equivalent to:

$$[q_{i_0}] - [(q_{i_0})(p_{i_1})] - [(q_{i_0})(1-p_{i_1})(p_{i_2})] - \cdots - [(q_{i_0})(1-p_{i_1}) \cdots (1-p_{i_{m-1}})(p_{i_m})]$$

We note that each term of the sum contains q_{i_0}. In terms of induced CP-net by $\mathscr{Q}$, the probability

q_{i_0} is the probability of the induced CP-nets that have $x_{i_0} > y_{i_0}$ (or $x_{i_0} > y_{i_0}|u$). Thus we are restrict the set of induced CP-nets that have that particular row in their CP-tables. That because the set of these CP-net contains all the CP-nets that allow $o > o'$ and not contain no one CP-net that allows $o' > o$. Thus we call this set S_0 and it contains only CP-nets in which $o > o'$ or $o \bowtie o'$. The idea is to remove from S_0 all the induced CP-net in which $o \bowtie o'$. To each step of the sum we remove a subset of S_0. Analyzing a generic j-term of the sum:

$$[(q_{i_0})(1-p_{i_1})\cdots(1-p_{i_{j-1}})(p_{i_j})]$$

we observe that we are removing from S_0 a subset $\bar{S}_j$:the set of the induced CP-nets defined by $[(q_{i_0})(1-p_{i_1})\cdots(1-p_{i_{j-1}})(p_{i_j})]$. We consider also the set S_j of the induced CP-nets defined by $[(q_{i_0})(1-p_{i_1})\cdots(1-p_{i_{j-1}})(1-p_{i_j})]$ and we observe that S_j and $\bar{S}_j$ are a partition of the set S_{j-1}. That implies that we do not remove the same CP-net two or more time, because we are moving in partitions. In particular when we remove the probability of the j-term of the sum, we remove the probability of the set $\bar{S}_j$. The final probability that we obtain is the probability of the set S_m (as we can see from the first formulation of the formula).

We prove that each set $\bar{S}_j$ contains all the induced CP-net that have $o \bowtie o'$ for index i_j (see Definition 3.5.4), and thus set S_j contains only CP-nets such that:

$$(x_1x_2\cdots x_{i_j}z > y_1y_2\cdots y_{i_j}z \ \ \forall z) \quad \text{and} \quad (x_1x_2\cdots x_{i_j}z_1 \underset{\bowtie}{>} y_1y_2\cdots y_{i_j}z_2 \ \ \forall z_1 \neq z_2)$$

where z, z_1, z_2 are a completion for a complete outcome over all the variables $X_1, \cdots, X_n$.

Thus we prove that, given an induced CP-net C from $\mathscr{D}$ and two outcome o and o', if for C they are incomparable for index i_j then $C \in \bar{S}_j$.

We suppose that the two outcome o and o' are incomparable for index i_j for the induced CP-net C and we have to prove that $C \in \bar{S}_j$.

We prove this by contradiction. If $C \notin \bar{S}_j$ then either (1) $C \in \bar{S}_k$ with $k < j$ or (2) $C \in S_j$.

(1) $C \in \bar{S}_k$ with $k < j$

This implies that o and o' are incomparable for $i_k < i_j$ and it is a contradiction because i_j is the minimum index for the incomparability.

(2) $C \in S_j$

Because of o and o' are incomparable for index i_j we have that $o \restriction_{X_1,\cdots,X_{i_j-1}} \not\bowtie o' \restriction_{X_1,\cdots,X_{i_j-1}}$. We suppose that $o \restriction_{X_1,\cdots,X_{i_j-1}} > o' \restriction_{X_1,\cdots,X_{i_j-1}}$ (the case with '$<$' is symmetric). Then we prove that also $o \restriction_{X_1,\cdots,X_{i_j}} > o' \restriction_{X_1,\cdots,X_{i_j}}$, and it is a contradiction.

If X_{i_j} is an independent node it is easy to show that $o \restriction_{X_1,\cdots,X_{i_j}} > o' \restriction_{X_1,\cdots,X_{i_j}}$. We take a worsening path P from $o \restriction_{X_1,\cdots,X_{i_j-1}}$ to $o' \restriction_{X_1,\cdots,X_{i_j-1}}$,

$$P = o \restriction_{X_1,\cdots,X_{i_j-1}} \to O_1 \to O_2 \to \cdots \to o' \restriction_{X_1,\cdots,X_{i_j-1}}$$

The path

$$\bar{P} = o \restriction_{X_1,\cdots,X_{i_j-1}} o_{i_j} \to O_1o_{i_j} \to O_2o_{i_j} \to \cdots \to o' \restriction_{X_1,\cdots,X_{i_j-1}} o_{i_j} \to o' \restriction_{X_1,\cdots,X_{i_j-1}} o'_{i_j}$$

is a worsening path from $o \restriction_{X_1,\cdots,X_{i_j}} (= o \restriction_{X_1,\cdots,X_{i_j-1}} o_{i_j})$ to $o' \restriction_{X_1,\cdots,X_{i_j}} (= o' \restriction_{X_1,\cdots,X_{i_j-1}} o'_{i_j})$.

Now we consider the case in which X_{i_j} is not an independent node. Because of C is singly connected, the sub-CP-net $\tilde{C}$ of the nodes $X_1,\cdots,X_{i_{j-1}}$ is formed by $|Pa(X_{i_j})|$ connected components, each one containing exactly one parent of X_{i_j}. Thus there aren't conflicts with the needs of the ancients. So, we have two cases:

- X_{i_j} has no shared parents.

  In this case, for each $u \in U_{i_j}^{o,o'}$ we can have a worsening path P from $o|X_1,\cdots,X_{i_j-1}$ to $o'|X_1,\cdots,X_{i_j-1}$ that contains u because the parents belongs to difference connected components and we can permutate the changing of the single connected component . Than suppose to have

  $$P = o \restriction_{X_1,\cdots,X_{i_j-1}} \to o_1 \to o_2 \to \cdots \to o_l \to \cdots \to o' \restriction_{X_1,\cdots,X_{i_j-1}}$$

  where o_l contains u ($o_l \restriction_{Pa(X_{i_j})} = u$). Because of, if a CP-net belongs to S_j means that exists al least one $u \in U_{i_j}^{o,o'}$ such that in the CP-table of X_{i_j} there is the row $u : x_{i_j} > y_{i_j}$, then a worsening path from $o \restriction_{X_1,\cdots,X_{i_j}}$ to $o' \restriction_{X_1,\cdots,X_{i_j}}$ is

  $$\tilde{P} = o \restriction_{X_1,\cdots,X_{i_j-1}} x_{i_j} \to o_1 x_{i_j} \to o_2 x_{i_j} \to \cdots \to o_l x_{i_j} \to o_l y_{i_j} \to \cdots \to o' \restriction_{X_1,\cdots,X_{i_j-1}} y_{i_j}$$

- X_{i_j} has shared parents.
  Also in this case, there exists a path P from $o|X_1,\cdots,X_{i_j-1}$ to $o'|X_1,\cdots,X_{i_j-1}$ that contains u, for each $u \in U_{i_j}^{o,o'}$, that we build a path $\tilde{P}$ as in the previous point, but we have to consider the interaction of the other nodes that have shared parents with X_{i_j}. Suppose to share a parent X_p with another node X_c there are two cases:
  - X_c needs the same value for X_p as X_{i_j}. In this case we can change the value of X_{i_j} and X_c in an arbitrary order
  - X_c needs a different value for X_p as X_{i_j}. In this case in the path the value of X_p change, for example, from a value x_p to $\bar{x}_p$. If X_c needs x_p then first we change its value then we change X_p and finally X_{i_j}. If X_c needs $\bar{x}_p$ then first we change the value of X_{i_j} then we change X_p and finally X_c. It is possible because X_{i_j} and X_c have only a shared parent. If the two nodes have more than one shared parent maybe they need two incomparable assignments of the shared parent and in that case not exists a path between them.

Thus $C \in \bar{S}_j$. ■

■ **Example 3.9** Now we show an example of computing the lower bound using the formula defined in Definition 3.5.3.

We consider a PCP-net $\mathscr{Q}$ as defined in Figure 3.12 with five nodes $X_1,\cdots,X_5$ and four edges: X_1X_3, X_1X_4, X_2X_4 and X_4X_5.

We consider two different outcomes: $o = x_1x_2\bar{x}_3x_4x_5$ and $o' = x_1\bar{x}_2x_3x_4\bar{x}_5$.

Thus the lower bound is:

$$\mathbb{P}^L_{\mathscr{Q}}(o > o') = [1 - \mathbb{P}(\bar{x}_2 > x_2)] \cdot [1 - \mathbb{P}(x_3 > \bar{x}_3|x_1)] \cdot [1 - \mathbb{P}(\bar{x}_5 > x_5|x_4)]$$

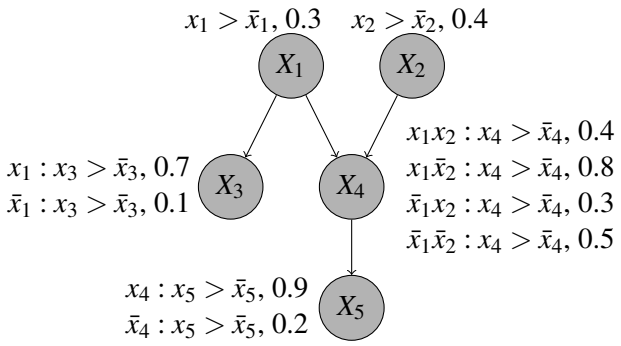


Figure 3.12: PCP-net $\mathscr{2}$ of Example 3.9

$$= \mathbb{P}(x_2 > \bar{x}_2) \cdot \mathbb{P}(\bar{x}_3 > x_3 | x_1) \cdot \mathbb{P}(x_5 > \bar{x}_5 | x_4) = 0.4 \cdot 0.3 \cdot 0.9 = 0.108$$

■

Computational complexity

The formula defined in Definition 3.5.3 is a product of at most n factors. The computation time of each factor is, in the worst case, equal to $O(2^k)$ where k is the maximal number of parents. Thus the computation complexity of the formula takes $O(n2^k)$ time. If we consider the case of trees (in which all nodes have at most one parent) we obtain $O(n)$.

Restriction to separable PCP-nets

We observe that if we apply the formula for the lower bound to separable PCP-nets, the probability resulting is the same as the exact value of the dominance probability.

Theorem 3.5.9 Given a PCP-net $\mathscr{2}$ with a separable dependency graph and two outcomes o and o', then:

$$\mathbb{P}_{\mathscr{2}}(o > o') = \mathbb{P}^L_{\mathscr{2}}(o > o')$$

Proof. We observe that each formulation is a product of factor and each factor corresponds to a variable that has a different value in o and o'. And each variable that has a different value in o and o' has a unique factor in each formulation. We prove that the factors that correspond of the same variable coincide in the two formulations.

We suppose to have a PCP-net that have the following PCP-tables: $X_i : x_i > \bar{x}_i, q_i$ where q_i is the probability $\mathbb{P}(x_i > \bar{x}_i)$

We consider two cases: the first one is the case of a variable X_i such that $o \restriction_{X_i} = x_i$ and $o' \restriction_{X_i} = \bar{x}_i$.

- in the formulation of dominance for separable PCP-net, we have a factor that correspond to q_i because the variable X_i belongs to the set $Diff^+(o, o')$

- in the lower bound formulation we have a factor that correspond to:

$$(1-\mathbb{P}(o' \restriction_{X_i} > o \restriction_{X_i})) = (1-\mathbb{P}(\bar{x}_i > x_i)) = \mathbb{P}(x_i > \bar{x}_i) = q_i$$

The second case is the case of a variable X_i such that $o \restriction_{X_i} = \bar{x}_i$ and $o' \restriction_{X_i} = x_i$.

- in the formulation of dominance for separable PCP-net, we have a factor that correspond to $(1-q_i)$ because the variable X_i belongs to the set $Diff^-(o,o')$
- in the lower bound formulation we have a factor that correspond to:

$$(1-\mathbb{P}(o' \restriction_{X_i} > o \restriction_{X_i})) = (1-\mathbb{P}(x_i > \bar{x}_i)) = 1-q_i$$

Thus the two formulations coincide. ■

■ **Example 3.10** We show now an example of computing the lower bound and the exact value of the dominance using the formula for a separable PCP-nets in Figure 3.13.

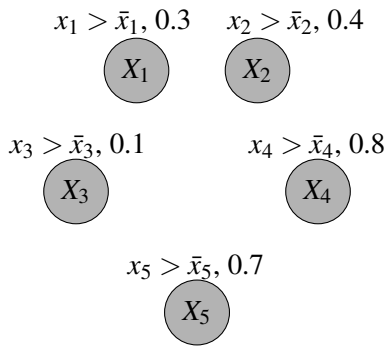


Figure 3.13: PCP-net $\mathscr{Q}$ of Example 3.10

We consider two different outcomes: $o = x_1x_2\bar{x}_3x_4x_5$ and $o' = x_1\bar{x}_2x_3x_4\bar{x}_5$. The sets $Diff^+(o,o')$ and $Diff^-(o,o')$ are the following:

$$Diff^+(o,o') = \{2,5\} \qquad Diff^-(o,o') = \{3\}$$

Thus we have:

$$\mathbb{P}(o > o') = \mathbb{P}(x_2 > \bar{x}_2) \cdot \mathbb{P}(\bar{x}_3 > x_3) \cdot \mathbb{P}(x_5 > \bar{x}_5) = 0.4 \cdot 0.9 \cdot 0.7 = 0.252$$

The lower bound is computed as follows:

$$\mathbb{P}^L(o > o') = [\mathbb{P}(\bar{x}_2 > x_2)] \cdot [\mathbb{P}(x_3 > \bar{x}_3)] \cdot [\mathbb{P}(\bar{x}_5 > x_5)]$$

$$= \mathbb{P}(x_2 > \bar{x}_2) \cdot \mathbb{P}(\bar{x}_3 > x_3) \cdot \mathbb{P}(x_5 > \bar{x}_5) = 0.4 \cdot 0.9 \cdot 0.7 = 0.252$$

The two values coincide. ∎

3.5.3 An upper bound for the dominance value

We recall the notion of *ancestor nodes* in graph theory: given a node X in a graph G, the set $Anc(X)$ is the set of nodes in G such that exists a direct path from each node in $Anc(X)$ to X.

Definition 3.5.5 Given a PCP-net $\mathscr{2}$ with n variables X_i and given two outcomes o and o', we define the set $Diff$ to be the indices of nodes that have different value in o and o' and that have fixed ancestor (that have the same value in o and o' for all the ancestor nodes) in o and o'. The upper bound for $\mathbb{P}(o > o')$ in $\mathscr{2}$ is:

$$\mathbb{P}^U_{\mathscr{2}}(o > o') = \min_{j \in Diff} \mathbb{P}(o \restriction_{X_j} > o' \restriction_{X_j} | u)$$

where u is the fixed assignment of the parents.

Theorem 3.5.10 Given a PCP-net $\mathscr{2}$ and two outcomes o and o', $\mathbb{P}^U_{\mathscr{2}}(o > o')$ is an upper bound for $\mathbb{P}_{\mathscr{2}}(o > o')$.

Proof. It is equivalent to prove that all CP-nets induced by $\mathscr{2}$ that support $o > o'$ have the row $u : o \restriction_{X_j} > o' \restriction_{X_j}$ for all the variables $X_j \in (Diff \cap FA)$. We prove this sentence by contradiction: we consider an induced CP-net C that contains the row $u : o' \restriction_X > o \restriction_X$ for a $X \in (Diff \cap FA)$ and we prove that $C \models (o' > o) \vee (o \bowtie o')$.We have two cases: *(1)* X is an independent node. Then the flip $o \restriction_X \to o' \restriction_X$ can't be a worsening flip. That implies that all the worsening paths starting from o do not contain a flip for variable X, so o' can't be reached. *(2)* X is a dependent node. Thus, the set $Anc(X) = \{X_{i_1}, \cdots, X_{i_k}\}$ with $k \geq 1$ and all of them have fixed value on o and o'. We call u the assignment of $Pa(X)$ in o and o'. Thus, each worsening path from o has the value u for $Pa(X)$ in each step of the path, because all the ancestors of X are fixed. This implies that no worsening path from o contains other assignments for the parents of X. Thus, no worsening path starting from o contains a flip for variable X, so o' can't be reached. Thus $C \not\models (o > o')$ and so $C \models (o' > o) \vee (o \bowtie o')$. ∎

∎ **Example 3.11** Now we show an example of computing the upper bound.

We consider a PCP-net $\mathscr{2}$ as defined in Example 3.9 (see Figure 3.12) with five nodes $X_1, \cdots, X_5$ and four edges: X_1X_3, X_1X_4, X_2X_4 and X_4X_5.

We consider two different outcomes: $o = x_1x_2\bar{x}_3x_4x_5$ and $o' = x_1\bar{x}_2x_3x_4\bar{x}_5$.

The set $Diff$ is equal to $Diff = \{2,3\}$ because they have different values on o and o' and also fixed ancestors. The variable X_5 has different values on o and o' but its ancestor X_2 has not a fixed value in o and o'.

Thus the upper bound is:

$$\mathbb{P}^U_{\mathscr{2}}(o > o') = \min\{\mathbb{P}(x_2 > \bar{x}_2), \mathbb{P}(\bar{x}_3 > x_3 | x_1)\} = \min\{0.4, 0.3\} = 0.3$$

∎

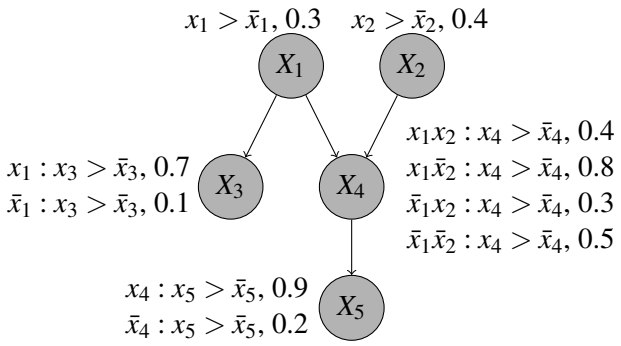


Figure 3.14: PCP-net $\mathscr{Q}$ of Example 3.11

Computational complexity

The formula defined in Definition 3.5.5 is a minimum between at most n factors. The computation time of each factor is $O(1)$, then computation complexity of the formula is $O(n)$ time.

3.5.4 The interval

Computing the lower bound and the upper bound allows to have a interval in which the exact value of the dominance probability belongs.

We test experimentally the size of the interval varying the number of nodes and, fixing the number of nodes, varying the maximal number of parents that a node can have.

For all our experiments we randomly generate CP-nets, and PCP-nets. Generating PCP-nets i.i.d. is non-trivial [2] and therefore we use an approximation method for random generation of CP-nets and probabilities. We generate polytree structured CP-nets and PCP-nets, considering a fixed ordering $X_1, \cdots, X_n$ for the variables. We also take as input the maximum in-degree for each feature, k. For each CP-net in the profile we first generate its acyclic dependency graph. For each feature X_i, we randomly choose its in-degree d, $0 \leq d \leq \min\{k, i-1\}$. Next, for each node X_i we add parents, one by one randomly, using rejection sampling on $X_1, \cdots, X_{i-1}$ (to ensure acyclicity), rejecting those that add cycles in the underlying *undirected* graph, until we reach the in-degree d or exhaust the set of possible parents. When the graph is built, we fill in the CP tables choosing randomly one element of the domain (since the domain is binary). For a PCP-net, we generate the dependency graph and CP tables as for a CP-net, and then we randomly assign probabilities to CPT rows.

Figure 3.15 shows the size of the interval as a function of the number of features, n, and of k. In this experiment we vary $n \in [0, 30]$ and fix the maximum k to $n-1$, $n/2$ and $n/4$. We compute the mean of the dominance interval over 100 PCP-nets for each value of n and for each PCP-net we take the mean over 100 randomly generated outcome pairs.

We observe that the size of the interval grows with the number of parents. Thus if we consider

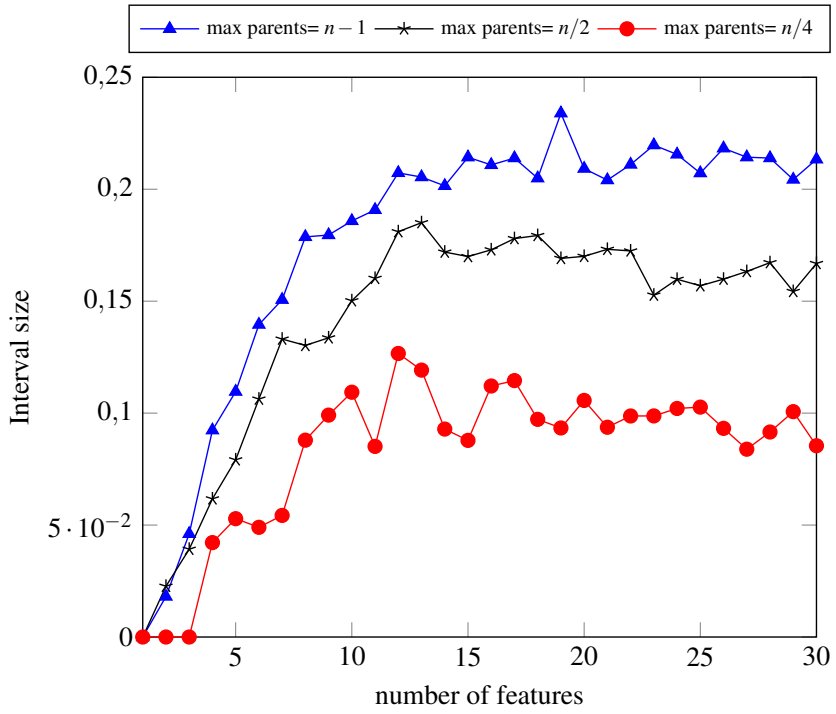


Figure 3.15: Interval size vs. number of features.

a bound over the maximal number of parents, we can reduce the size of the interval.

We observe that the maximal interval size is 0.23 and the mean interval size is 0.2 with $n-1$ maximal parents (the worst case). If we have a number of parents boundend by a constant, the size of the interval is much smaller. Thus the results shows that the interval is small in such cases: with $n/4$ maximal parents we have an interval size less then 10% of the whole interval $[0,1]$. This results suggest that in a real-world datasets the error will be very small since in real-world data the number of maximal parent usually is bounded by a small constant (because people usually take decisions dependently to a small set of attributes).

3.5.5 Lower bound as approximation of the dominance exact value

Then we test where is located the exact value of the dominance probability value in the interval and the following experiment show that the exact value is very close to the lower bound. We conclude that the lower bound is a good estimation for the exact value of the dominance.

Table 3.1 shows the distance between our lower bound and the true value of the dominance probability as we vary the number of features. In this experiment we vary the number of features $n \in [0,7]$ and we fix the maximum k to $n-1$, $n/2$ and $n/4$. We compute the mean over 20 PCP-nets for each value of n and for each PCP-net we take the mean over 25 outcome pairs (total mean over 500 cases).

We limit our experimental setting considering only a maximum of 7 nodes because the computational complexity for computing the exact value of the dominance is exponential: in the worst case we have to test a dominance query for each different induced CP-net, and the number of induced CP-net are exponential in the PCP-net description. Considering the a PCP-net with 7 nodes and

the maximum number of parents less then $n-1$, usually the computation on 20 PCP-nets with 25 compared pairs takes about one month.

n :	1	2	3	4	5	6	7
$Pa : n-1$	0.0	0.0	0.002	0.002	0.002	0.008	0.004
$Pa : n/2$	0.0	0.0	0.008	0.006	0.0	0.004	0.004
$Pa : n/4$	0.0	0.0	0.0	0.0	0.0034	0.002	0.006

Table 3.1: Distance between our lower bound and the true probability of dominance.

We observe that the true value is very close to the lower bound and the maximal distance is 0.006.

We observe that also the percentage of the distance from the dominance exact value to lower bound respect to the whole interval is very small. The percentage grows augmenting the number of the nodes and of the parents, but in our experiments the maximal distance is about the 1.2% of the interval.

In our experiments, the distance between the lower bound to the true dominance probability is generally very small. It seems to grow as a function of the number of nodes and parents. Thus, the lower bound can be considered a good approximation of the true dominance probability with a maximal error equal to the size of the interval.

3.5.6 Dominance as a decision problem

In some settings it could be enough to get a yes/no answer from a dominance query, rather than the exact probability value.

For example, if a PCP-net encodes the preferences of a community, we can ask if the users prefer one outcome or another. For this reason we introduce others concept of dominance seen as decision problem:

Definition 3.5.6 — DDP for PCP-nets. Given a PCP-net $\mathcal{Q}$ and a pair of outcomes o and o', the *Dominance Decision Problem* over o and o' for $\mathcal{Q}$ correspond to ask $True$ or $False$ to the question if the PCP-net prefer o or o'.

- if $\mathbb{P}(o > o') \geq 0.5$ then $DDP_{\mathcal{Q}}(o,o') = True$
- if $\mathbb{P}(o > o') < 0.5$ then $DDP_{\mathcal{Q}}(o,o') = False$

The threshold is fixed to 0.5 because we are considering majority as voting rule when we ask if the users prefer one outcome or another, since we consider only two options $True$ and $False$.

Definition 3.5.7 — MP-Dominance for PCP-nets. Given a PCP-net $\mathcal{Q}$ and a pair of outcomes o and o', a *Most Probable dominance query (MP-dominance)* is to ask a *dominance query* to the most probable induced CP-net.

We performed some experiments to compare the two methods that approximate the dominance decision problem:

- the lower bound;
- MP-dominance.

The two following experiment analyze the number of times in which the MP-dominance correspond to the value of DDP in a scale of $[0,1]$ where 1 correspond to a perfect match.

Figure 3.16 shows the frequency of right answer of the most probable induced CP-net (in $[0,1]$), varying the number of nodes. In this experiment we vary the number of nodes $n \in [0,5]$ and we fix the parents $\leq n/2$. We compute the mean over 50 PCP-nets for each value of n and for each PCP-net we take the mean over 20 outcomes pairs.

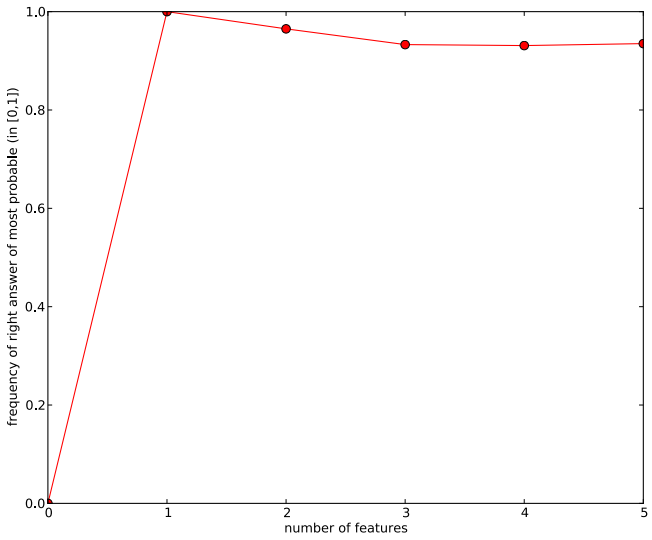


Figure 3.16: Frequency of right answer of the most probable induced CP-net, varying the number of nodes

The percentage of correct answer is very high: up to 90%.

Figure 3.17 shows the frequency of right answer of the most probable induced CP-net (in $[0,1]$), varying the maximal number of parents. In this experiment we fix the number of nodes to $n = 5$ and we vary the maximal number of parents $m \in [0,4]$. We compute the mean over 30 PCP-net for each value of n and for each PCP-net we take the mean over 20 outcomes pairs.

The percentage of correct answer results very high: up to 90%.

Concluding, these two experiments show that MP-dominance is an accurate method. We want to compare now MP-dominance with the lower bound based decision dominance. In the following two experiments we provide this comparison varying first the number of nodes and than the maximal number of parents.

Figure 3.18 shows the frequency of right answer of the Mp-dominance and the lower bound based decision dominance (in $[0,1]$), varying the number of nodes. In this experiment we vary the number of nodes $n \in [0,5]$ and we fix the parents $\leq n/2$. We compute the mean over 50 PCP-net for each value of n and for each PCP-net we take the mean over 20 outcomes couple.

The Lower bound based dominance is more correct than MP-dominance.

Figure 3.19 shows the frequency of right answer of the Mp-dominance and the Lower bound

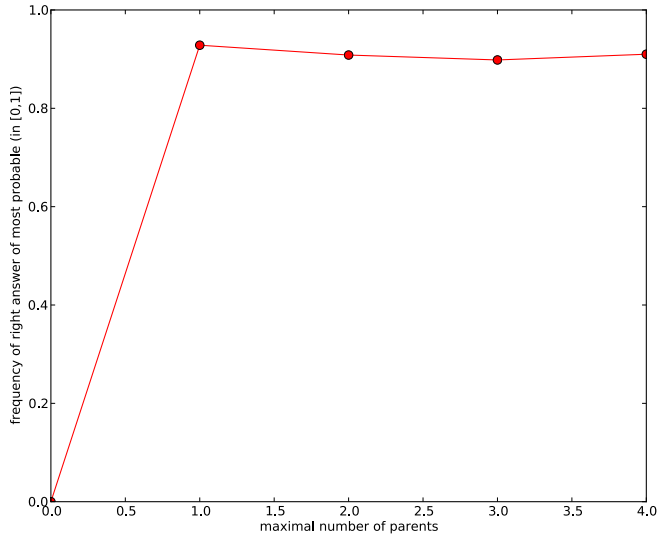


Figure 3.17: Frequency of right answer of the most probable induced CP-net, varying the maximal number of parents

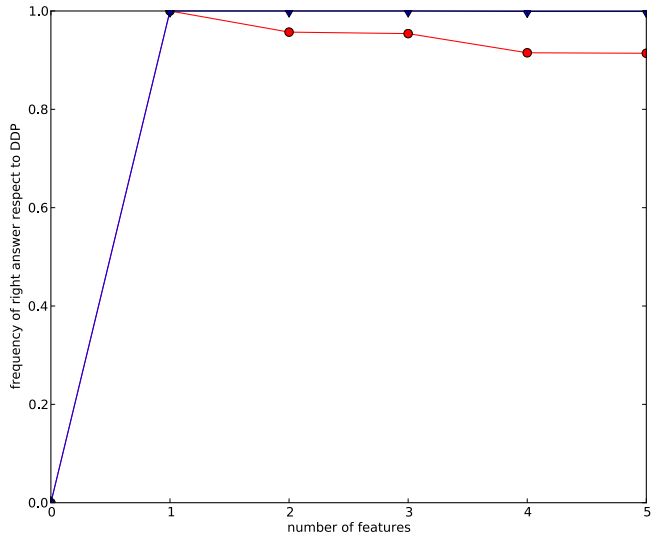


Figure 3.18: Frequency of right answer of the Mp-dominance and the Lower bound based dominance, varying the number of nodes

based dominance (in $[0,1]$), varying the maximal number of parents. In this experiment we fix the number of nodes to $n = 5$ and we vary the maximal number of parents $m \in [0,4]$. We compute the mean over 30 PCP-net for each value of n and for each PCP-net we take the mean over 20 outcomes couple.

In conclusion the more accurate method for approximate the DDP is the lower bound based

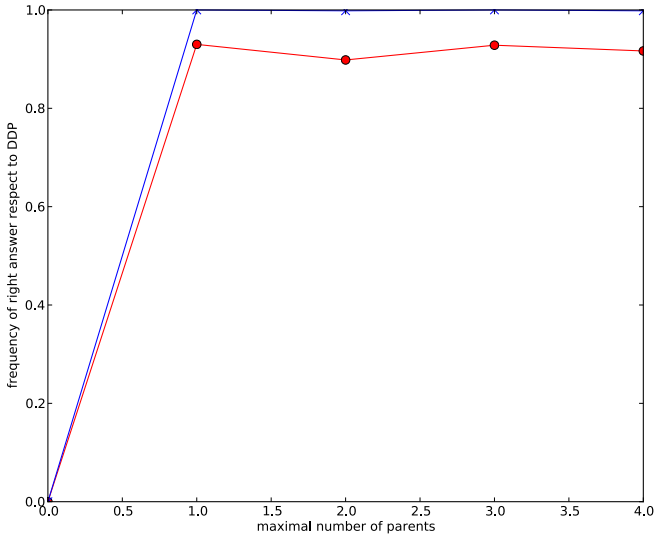


Figure 3.19: Frequency of right answer of the Mp-dominance and the Lower bound based dominance, varying the maximal number of parents

decision dominance.

3.6 Learning of separable PCP-nets

We investigate the problem of exact learning of PCP-nets in the initial case of separable structure. We consider this case since we have a polynomial method to compute optimality tasks and the exact value of dominance. Moreover separable preferences are a very important class of preference relations on multi-attribute domains, as we can see in the economics or social choice literature (see for example [57]). Focusing first on separable preferences is nontrivial and it is necessary as first step towards learning more complex preferences. Furthermore in literature is well-known that a simple structure generalize concepts in a better way than more structured framework despite the fact that more complex structures may fit the examples better.

There is an active stream of research on preference learning, both for passive and active learning.

Passive learning considers systems that have a set of preferences between alternatives (dominance pairs) given by a user. The goal is to be able to reason about her preferences so as to predict choices she will make on new pairs of alternatives.

This approach is completely different from the *active learning* of preferences (also called *preference elicitation*) in which the system interacts with the user, making queries and asking to her to express a preference on the value of some attributes, until she has found her target object or left the system (see [18, 61, 110]).

The literature research initially focused mainly on preferences on simple, non-combinatorial domains and multi-issues domains are considered mainly associated with an active learning of utility functions [40, 53, 108]. Regarding instead qualitative preferences we must cite the works on

learning lexicographic preferences on multi-attribute domains ([38, 98, 117]).

Active learning on combinatorial domains has been studied in the work of Koriche and Zanuttini in [60], Dimopoulos et al. in [37], Eckhardt and Vojtáš in [43, 44], Liu et al. in [69], and the most recent work of Guerin et al. in [56] that guarantee to produce a CP-net in polynomial time given a constant bound on the number of parents.

Passive learning has been analyzed on multi-issues domains by Chevaleyre in [22] and by Lang and Mengin in [66]. This last work considers our same context: the special case of separable CP-nets. They proved that finding a separable CP-net from a consistent set of examples can be solved in time polynomial, since the VC-dimension (a measure of the the expressive power of a learning algorithm) of separable CP-nets is polynomial in the input dimension and thus separable CP-nets are PAC-learnable. We will compare our procedure with this particular work in the end of this section.

In this section we focus on passive learning: given a set of consistent dominance examples, we exact learn a PCP-net consistent with the set of examples. The goal is to identify a probabilistic distribution over the possible preference orderings generated by a binary-valued PCP-net. The learning is performed using a set of consistent examples turned to a system of non-linear equations, that, once solved, returns the solution/solutions: a PCP-net or a set of PCP-net that are consistent with the set of examples.

Suppose to have n independent features $X_1, \cdots, X_n$ with binary domains (the non-binary case is a trivial extension): $\mathscr{D}_i = \{x_i, \bar{x}_i\}$, we want to learn a PCP-net $\mathscr{Q}$ as described in Figure 3.20.

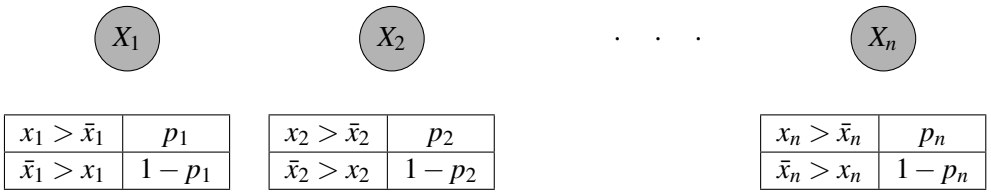


Figure 3.20: PCP-net $\mathscr{Q}$

Since we consider a bias on the structure of the PCP-net (i.e. without edges), our unknown are only the n parameters $p_1, \cdots, p_n$ that represent the probability distribution over the possible ordering among the variables values.

The set of examples

We learn the parameters from a set of examples of the form:

$$e = (o, o', p)$$

where o and o' are two outcome and p is the probability (or normalized frequency) of the example.

If we consider only positive examples, p is the following probability: $\mathbb{P}(o \succ o') = p$. But we can consider also negative or neutral examples. Thus we have three possible types of examples:

- Positive examples: $e = (o, o', p)^+$ where $p = \mathbb{P}(o \succ o')$
- Incomparable examples: $e = (o, o', p)^{\bowtie}$ where $p = \mathbb{P}(o \bowtie o') = 1 - (\mathbb{P}(o \succ o') + \mathbb{P}(o' \succ o))$
- Negative examples: $e = (o, o', p)^-$ where $p = \mathbb{P}(o \not\succ o') = \mathbb{P}(o' \succ o) + \mathbb{P}(o \bowtie o')$

We immediately observe that a negative example could been turned into a positive one. If we have $e = (o, o', p)^-$ where $p = \mathbb{P}(o' \succ o) + \mathbb{P}(o \bowtie o')$ we can consider

$$p' = 1 - p = 1 - (\mathbb{P}(o' \succ o) + \mathbb{P}(o \bowtie o')).$$

But we know that:

$$1 = \mathbb{P}(o \succ o') + \mathbb{P}(o' \succ o) + \mathbb{P}(o \bowtie o')$$

thus

$$p' = 1 - p = \mathbb{P}(o' \succ o),$$

and we can consider the example $e' = (o', o, p')^+$

So, in our model, we can have two choices of set of examples:

- a set of only positive examples;
- a set of positive and incomparable examples.

Observation 3.8 We will consider only the first case (set of only positive examples), since the second is an easy generalization:

$$p = \mathbb{P}(o \bowtie o') = 1 - \mathbb{P}(o > o') - \mathbb{P}(o' > o)$$

We have a formula to compute the probability of a dominance query over two outcomes in function of the parameter of the PCP-net description. To have a formula is the only requirement that the following method needs. ■

The second assumption that we make is that the set of examples is consistent. This means that given a set of examples, there exists at least one PCP-net that entails them, and thus there is always at least one solution of the problem.

3.6.1 The method

The method goal is to find a consistent PCP-net with the set of input examples (or a set of consistent PCP-nets). This means that if we compute dominance testing over a PCP-net in the solution set it returns exactly the probability associated with the example.

The method consists into generate an equation from each example in the set of examples, and then to solve the system of generated equations.

The system of equations

Given a positive example $e = (o, o', p)$ we generate two sets $Diff^+(o, o')$ and $Diff^-(o, o')$ such that:

- $Diff^{+}(e) = \{i \in \{1,\cdots,n\} | x_i \in o \text{ and } \bar{x}_i \in o'\}$
- $Diff^{-}(e) = \{i \in \{1,\cdots,n\} | \bar{x}_i \in o \text{ and } x_i \in o'\}$

Thus the equation, over the p_i variables, associated with the example $e = (o, o', p)$ is $eq_e(p_1, \cdots, p_n)$:

$$\left(\prod_{i \in Diff^{+}(e)} p_i\right) \cdot \left(\prod_{j \in Diff^{-}(e)} (1 - p_j)\right) = p$$

These equations are non-linear and over at most n variables. We have a number of equations equal to the cardinality of the set of examples. We have also n additional constraints about the feasible domains for the p_i (p_i are probabilities):

$$p_i \in [0,1] \quad \forall i \in \{1,\cdots,n\}$$

Since the set of examples is consistent, there is always at least one solution, but we could also have an infinity number of solutions (in this case we can use a tie breaking rule such as the solution with minimum entropy etc.).

Observation 3.9 — Consistency. The PCP-net $\mathscr{Q}$ obtained solving the system of non-linear equations built associating to each example $e = (o, o', p)$ the equation

$$\left(\prod_{i \in Diff^{+}(e)} p_i\right) \cdot \left(\prod_{j \in Diff^{-}(e)} (1 - p_j)\right) = p$$

is consistent with the set of examples. That means that computing the probability $\mathbb{P}(o \succ o')$ over $\mathscr{Q}$ it returns p.

The equation fro the dominance in a separable PCP-net corresponds to the formula of the method. ■

■ **Example 3.12** We provide an example of how the method runs. Given a set of examples T over three variables X_1, X_2 and X_3 with binary domains $\mathscr{D}_1 = \{x_1, \bar{x}_1\}$, $\mathscr{D}_2 = \{x_2, \bar{x}_2\}$ and $\mathscr{D}_3 = \{x_3, \bar{x}_3\}$, we have the separable PCP-net $\mathscr{Q}$ (Figure 3.21), in which we have to learn the three parameters: p_1, p_2 and p_3.

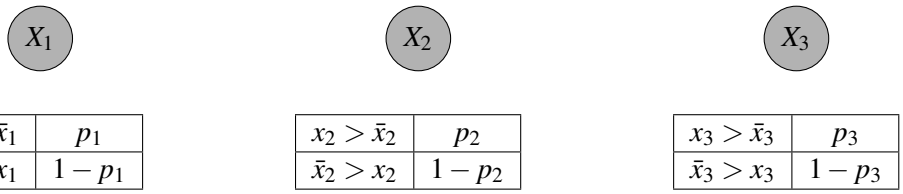


Figure 3.21: PCP-net $\mathscr{Q}$

We also have the following set of examples T with cardinality 3:

- $e_1 = (x_1x_2x_3, \bar{x}_1x_2\bar{x}_3, 0.6)$;
- $e_2 = (x_1\bar{x}_2x_3, x_1x_2x_3, 0.3)$;
- $e_3 = (x_1x_2\bar{x}_3, \bar{x}_1x_2x_3, 0.2)$.

Thus we have:

- $Diff(e_1)^+ = \{1,3\}$, $Diff(e_1)^- = \emptyset$;
- $Diff(e_2)^+ = \emptyset$, $Diff(e_2)^- = \{2\}$;
- $Diff(e_3)^+ = \{1\}$, $Diff(e_3)^- = \{3\}$;

and we generate the following equations:

- $eq_{e_1}(p_1,p_2,p_3) : p_1p_3 = 0.6$;
- $eq_{e_2}(p_1,p_2,p_3) : 1 - p_2 = 0.3$;
- $eq_{e_3}(p_1,p_2,p_3) : p_1(1-p_3) = 0.2$;

Solving them we find the solution:

$$p_1 = 0.8 \quad p_2 = 0.7 \quad p_3 = 0.75$$

■

Observation 3.10 This approach can be applied to the deterministic case of CP-nets learning, considering the probability of the examples equal to 1: examples of the form: $e = (o, o', 1)$.

We observe that our approach, restricted to CP-nets, results equivalent to the method proposed by Chevaleyre et al. in[22] and studied in [66]. To see this equivalence it is sufficient to apply the relation between logic operators and numerical operator: *AND* is the numerical product, *OR* is the numerical sum.

Lang and Mengin in [66] proved that finding a separable CP-net from a consistent set of examples can be solved in time polynomial, since the VC-dimension (a measure of the the expressive power of a learning algorithm) of separable CP-nets is polynomial in the input dimension and thus separable CP-nets are PAC-learnable. This result is consistent with our result since also our procedure is computable in a time polynomial in the number of features. ■

3.7 Dynamic Probabilistic CP-nets

We now turn our attention to dynamic modifications to the structure of a PCP-net. These changes can be implemented in an efficient way and their effects on computing the most probable optimal outcome and the most probable induced CP-net are minimal, in terms of complexity. One can think of modifying the structure of the PCP-net as entering evidence in a BN framework. By adding or removing an arc or setting an ordering for a variable we can fix parts of the probability distribution and compute the outcomes of the resulting structure.

3.7.1 Main dynamic modifications

There are many possible dynamic modifications of the structure of a PCP-net, but the ones that we will consider are the following:

- adding a dependency;

- deleting a dependency;
- adding a feature;
- deleting a feature.

Any other modification can be represented as a combination of the ones listed above.

To add a dependency $e = (X,Y)$ we just add the edge to the dependency graph and update the respective probability table (of the Y node), adding the new informations.

The delete a dependency $e = (X,Y)$ we focus on the subspace of the events in which the two nodes X and Y are independent to each other. We recompute the probabilities in the PCP-table of the node Y conditioning to the fact that X and Y, this means that, given an assignment u_i of the parents nodes $Pa(Y) \setminus X$, the probability of a total order o_j on the domain $Dom(Y)$ given u_i corresponds to the probability of the case in which we have o_j for each assignment for the variable X given u_i. Given the PCP-table described in Table 3.2 for the node Y focusing on a particular assignment u_i of the parents nodes $Pa(Y) \setminus X$ and where $o_1, \cdots, o_m$ are the possible total orders on the domain $Dom(Y)$ of Y, $\{x_1, \cdots, x_l\} = Dom(X)$, we recompute the probability of an order o_j as:

$$\mathbb{P}(o_j|u_i, \text{independence from } X) = \frac{1}{K}\prod_{k=1}^{l}\mathbb{P}(o_j|x_k) = \frac{1}{K}\prod_{k=1}^{l} p_{i,k}^{o_j}$$

where K is the proper normalizing constant.

$Pa(Y)$	ordering	$\mathbb{P}$
$u_i x_1$	o_1	$p_{i,1}^{o_1}$
	$\cdots$	$\cdots$
	o_m	$p_{i,1}^{o_m}$
$u_i \cdots$	o_1	$\cdots$
	$\cdots$	$\cdots$
	o_m	$\cdots$
$u_i x_l$	o_1	$p_{i,l}^{o_1}$
	$\cdots$	$\cdots$
	o_m	$p_{i,l}^{o_m}$

Table 3.2: A PCP-net table

■ **Example 3.13** Let's consider the PCP-net in Figure 3.22 with binary domains features X_1, X_2 and X_3. Removing the edge from X_2 to X_3 we obtain the PCP-net in Figure 3.23

■

To add a feature we just add a new node with the respective probability table. To remove a feature we first remove each of its outgoing edges, following the procedure described above, and then we simply remove the node and the corresponding PCP-table.

Additionally, due to the independence assumptions, we can modify probabilities over ordering and features at a local level, with no need to recompute the entire structure when new information is added.

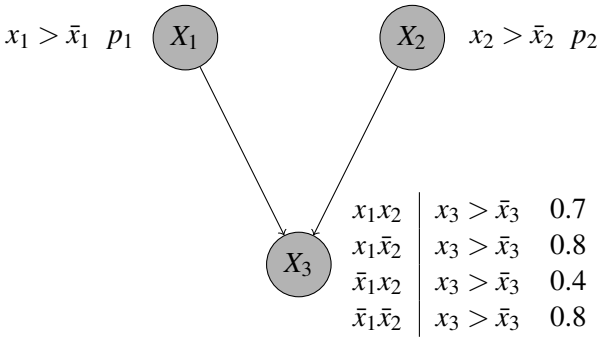


Figure 3.22: A PCP-net (Example 3.13)

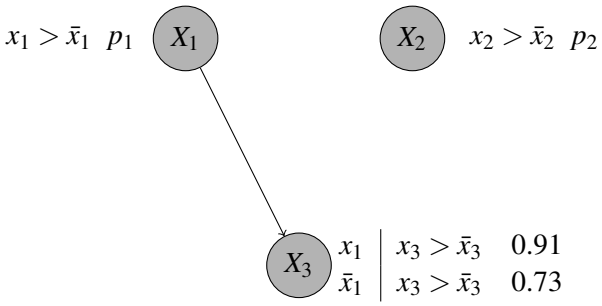


Figure 3.23: A PCP-net (Example 3.13)

3.7.2 Effects of the modifications on the G-net

When we modify a PCP-net $\mathscr{C}$ we also need to modify the associated G-net and this can have an effect on computing the most probable induced CP-net. To add or delete a dependency or feature, independent or dependent, we need to recompute the probabilities for the connected component of the children of the feature affected, which can be done quickly if this component is small, and recompute the maximal joint probability for the G-net. This same procedure works for updating a probability table with either evidence or changing the distribution. All these steps take constant time in the size of the connected component of the change.

3.7.3 Effects of the modifications on the Opt-net

When we modify a PCP-net $\mathscr{C}$, the changes affect the associated Opt-net. Consider the dependency of feature B on feature A. When we add or delete this dependency, or when we change its probability, we only need to recompute the probability table of B in the Opt-net. When computing the most probable optimal outcome, we note that, in the worst case, we must recompute the whole maximal joint probability of the Opt-net. The same can be said when we delete a feature, as this amounts

to the deletion of a set of dependencies, or when we modify the probability distribution over the orderings on B for a specific assignment to all of its parents. When we add a feature A to $\mathscr{C}$, we must add the corresponding node in the Opt-net and generate the corresponding probability table. This new node is independent. Thus, revising the current most probable optimal outcome is easy: the new optimal is the current one extended with the optimal value of the new feature.

3.8 A PCP-net generalization by using the Dempster Shafer theory

In this section we provide a generalization of PCP-nets. We extend CP-net, and its generalization CP-theories, using a different theory of probability to describe the uncertainty over the events/alternatives. We will use the *Dempster-Shafer* theory of probability [99]. This theory has many advantage: it allows the representation of ignorance and provide a rule to combine the effect of different sources of evidence.

3.8.1 Dempster-Shafer theory of probability

The Dempster-Shafer theory [99] (known also as *evidence theory* or *theory of belief functions*) is a mathematical theory of evidence and plausible reasoning. We will define in what follows the basic definition and notions of this theory.

Let Θ be a finite set with mutually exclusive elements. A source triple over Θ corresponds to $(\Omega, \mathbb{P}, \Gamma)$ where Ω is the finite abstraction set, $\mathbb{P}$ is a probability distribution over Ω such that $\forall \omega \in \Omega : \mathbb{P}(\omega) \neq 0$ and Γ is a function such that $\Gamma : \Omega \to 2^{\Theta} \setminus \emptyset$. Given a statement $\varphi \in \Theta$, we use the notation $[\varphi]$ to indicate the set of all the models that satisfy the statement φ.

A mass function m is defined over the source triple:

$$m(\varphi) = \sum_{\omega \in \Omega : \Gamma(\omega) = \varphi} \mathbb{P}(\omega) \ .$$

The mass function has the following properties: $m(\emptyset) = 0$ and $\sum_{\omega \subseteq \Omega} m(\omega) = 1$. The quantity $m(\omega)$ is a measure of the belief that is assigned to exactly the set ω and not to any proper subset of ω. A measure of the total belief in ω is associated to the belief function *Bel* defined as

$$Bel(\varphi) = \sum_{\omega \in \Omega : \Gamma(\omega) \subseteq \varphi} \mathbb{P}(\omega) \ .$$

If we suppose to have that the information associated to the source triple $(\Omega, \mathbb{P}, \Gamma)$ is a combination of the information of a finite set of source triples: $(\Omega_i, \mathbb{P}_i, \Gamma_i)$, we can combine these informations using the *Dempster's rule of combination*. The *Dempster's rule of combination* combine these rules using independence assumptions between the mass functions defined on the source triples $(\Omega_i, \mathbb{P}_i, \Gamma_i)$. This combination can be seen as a mapping from the set of source triples $(\Omega_i, \mathbb{P}_i, \Gamma_i)$ to the source triple $(\Omega, \mathbb{P}, \Gamma)$, defined as follow. Suppose to have:

- m source triples $(\Omega_i, \mathbb{P}_i, \Gamma_i)$;
- the set $\Omega' = \bigotimes_{i=1}^{m} \Omega_i$. If $\omega \in \Omega'$ that $\omega = (\omega_1, \cdots, \omega_m)$ where $\omega_i \in \Omega_i$;
- the function $\Gamma' : \Omega' \to 2^{\Theta}$ such that $\Gamma'(\omega) = \bigcap_{i=1}^{m} \Gamma_i(\omega_i)$;

- the probability function $\mathbb{P}'$ such that $\mathbb{P}'(\omega) = \prod_{i=1}^{m} \mathbb{P}_i(\omega_i)$ (independence assumptions)

then we can define the source triple $(\Omega, \mathbb{P}, \Gamma)$ as:

- $\Omega = \{\omega \in \Omega' | \Gamma'(\omega) \neq \emptyset\}$;
- $\Gamma = \Gamma' \restriction_{\Omega}$ the restriction of Γ' to Ω;
- $\mathbb{P}(\omega) = \frac{\mathbb{P}'(\omega)}{\mathbb{P}'(\Omega)}$ where $\frac{1}{\mathbb{P}'(\Omega)}$ is the measure of the conflict between evidences;

Thus he have defined a mass function m and a belief function Bel on Θ as follow:

$$m(\varphi) = \sum_{\omega \in \Omega : \Gamma(\omega) = \varphi} \left(\prod_{i=1}^{m} \frac{\mathbb{P}_i(\omega_i)}{K_\Omega} \right) = \sum_{\substack{\omega \in \Omega : \\ \bigcap_{i=1}^{m} \Gamma_i'(\omega_i) = \varphi}} \left(\prod_{i=1}^{m} \frac{\mathbb{P}_i(\omega_i)}{K_\Omega} \right),$$

$$Bel(\varphi) = \sum_{\omega \in \Omega : \Gamma(\omega) \subseteq \varphi} \left(\prod_{i=1}^{m} \frac{\mathbb{P}_i(\omega_i)}{K_\Omega} \right) = \sum_{\substack{\omega \in \Omega : \\ \bigcap_{i=1}^{m} \Gamma_i'(\omega_i) \subseteq \varphi}} \left(\prod_{i=1}^{m} \frac{\mathbb{P}_i(\omega_i)}{K_\Omega} \right),$$

where $K_\Omega = \frac{1}{\mathbb{P}'(\Omega)}$.

3.8.2 Our model

Given a set of n features $V = \{X_1, \cdots, X_n\}$ with binary domains $D(X_i) = \{x_i, \bar{x}_i\}$ we call $\underline{V}$ the set of all complete assignments of the variables in V (outcomes): $\underline{V} = \bigotimes_{i=1}^{n} D(X_i)$ where given $o \in \underline{V}$ then $o \restriction_{X_i}$ corresponds to the assignment of the variable X_i in o. We define the language Λ as the set of all the rule/statements over V such that:

$$\Lambda = \{u : x_i > \bar{x}_i[W]\} \cup \{u : \bar{x}_i > x_i[W]\}$$

where $i \in \{1, \cdots, n\}$, u is an assignment of a subset $S \subseteq V \setminus \{X_i\}$ with cardinality in $\{0, \cdots, n-1\}$ (partial outcome $u \in \underline{S} = \bigotimes_{X_i \in S} D(X_i)$) and W is a subset of $V \setminus \{X_i\} \setminus S$ with cardinality in $\{0, \cdots, n-1-|S|\}$.

Λ is the general language, but making some restriction to Λ we can recover for example CP-theories and CP-nets that are subset of Λ. We will consider the natural assumption that for each variable X_i the set U is a subset of $\{X_1, \cdots, X_{i-1}\}$ and the set W is a subset of $\{X_{i+1}, \cdots, X_n\}$.

Following the Dempster-Shafer approach we define Θ as the set of all total orderings over $\underline{V}$. This set is finite and with mutually exclusive elements. In the following two section we propose two different but equivalent ways to define the set of source triples $(\Omega_i, \mathbb{P}_i, \Gamma_i)$: the first one uses binary sets Ω_i and the second one uses general sets Ω_i.

Formulation 1: binary sets

For each statement $\varphi_i \in \Lambda$ we define a binary set Ω_i such that:

- $\Omega_i = \{\omega_i, \neg\omega_i\}$
- $\Gamma_i(\omega_i) = [\varphi_i]$
- $\Gamma_i(\neg\omega_i) = \Theta$
- $\mathbb{P}_i(\omega_i) = p_i$ and $\mathbb{P}_i(\neg\omega_i) = 1 - p_i$

■ **Example 3.14** Given a language Λ defined on two variables $V = \{X_1, X_2\}$ with the restriction

that for each variable X_i then the set U must be a subset of $\{X_1, \cdots, X_{i-1}\}$, we define Λ as follows:

$$\Lambda = \{\varphi_1, \varphi_2, \cdots, \varphi_1 2\}$$

where:

$\varphi_1 = (x_1 > \bar{x}_1)$ $\varphi_4 = (\bar{x}_1 > x_1[X_2])$ $\varphi_7 = (x_2 > \bar{x}_2[X_1])$ $\varphi_{10} = (x_1 : \bar{x}_2 > x_2)$
$\varphi_2 = (\bar{x}_1 > x_1)$ $\varphi_5 = (x_2 > \bar{x}_2)$ $\varphi_8 = (\bar{x}_2 > x_2[X_1])$ $\varphi_{11} = (\bar{x}_1 : x_2 > \bar{x}_2)$
$\varphi_3 = (x_1 > \bar{x}_1[X_2])$ $\varphi_6 = (\bar{x}_2 > x_2)$ $\varphi_9 = (x_1 : x_2 > \bar{x}_2)$ $\varphi_{12} = (\bar{x}_1 : \bar{x}_2 > x_2)$

Let Θ the set of all the total orderings over $\underline{V}$, then we have 12 source triples $(\Omega_i, \Gamma_i, \mathbb{P}_i)$ defined on Λ, one for each statement $\varphi_i \in \Lambda$. For example Ω_2 is defined as:

- $\Omega_2 = \{\omega_2, \neg\omega_2\}$
- $\Gamma_2(\omega_2) = [\bar{x}_1 > x_1]$
- $\Gamma_2(\neg\omega_2) = \Theta$
- $\mathbb{P}_2(\omega_2) = p_2$ and $\mathbb{P}_2(\neg\omega_2) = 1 - p_2$

The mass function of $\varphi_2 = (\bar{x}_1 > x_1)$ is:

$$m(\varphi_2) = \frac{p_2}{K_\Omega}$$

The belief function of $\varphi_2 = (\bar{x}_1 > x_1)$ is:

$$Bel(\varphi_2) = \frac{p_2 + p_4}{K_\Omega}$$

■

It is important to notice that in this formulation the constant $\frac{1}{\mathbb{P}'(\Omega)}$, the measure of the conflict between evidences, is high. This because there are many elements of Ω that have $\Gamma'(\omega) = \emptyset$. For example the element (ω_1, ω_2) in the Example 3.14 has:

$$\Gamma'(\omega_1, \omega_2) = [\varphi_1] \cap [\varphi_2] = [x_1 > \bar{x}_1] \cap [\bar{x}_1 > x_1] = \emptyset\ .$$

Formulation 2: general sets

For each feature $X \in V$ and u assignment of the parents of X, we have a set of statements $\{\varphi_{i_1}, \cdots, \varphi_{i_k}, \varphi_{i_{k+1}}\} \subseteq \Lambda$ such that $\forall j \in \{1, \cdots, k\}$ $\varphi_{i_j} = (u : x > \bar{x}[W])$ and $\varphi_{i_{k+1}} = \Theta$, we define a set Ω_i such that:

- $\Omega_i = \{\omega_{i_1}, \cdots, \omega_{i_k}\}$
- $\Gamma_i(\omega_{i_j}) = \{\varphi_{i_j}\}$
- $\mathbb{P}_i(\omega_{i_j}) = p_{i_j}$ and $\sum_{j=1}^{k} p_{i_j} = 1$

■ **Example 3.15** In the same scenario of Example 3.14, we have only 4 source triples, defined as:
$\Omega_1 = \{\omega_{1_1}, \omega_{1_2}, \omega_{1_3}, \omega_{1_4}, \omega_{1_5}\}$

- $\Gamma_1(\omega_{1_1}) = [\varphi_1], \Gamma_1(\omega_{1_2}) = [\varphi_2], \Gamma_1(\omega_{1_3}) = [\varphi_3], \Gamma_1(\omega_{1_4}) = [\varphi_4], \Gamma_1(\omega_{1_5}) = \Theta$;
- $\mathbb{P}_1(\omega_{1_j}) = p_{1_j}$ and $\sum_{j=1}^{5} p_{1_j} = 1$

$\Omega_2 = \{\omega_{2_1}, \omega_{2_2}, \omega_{2_3}, \omega_{2_4}, \omega_{2_5}\}$

- $\Gamma_2(\omega_{2_1}) = [\varphi_5], \Gamma_2(\omega_{2_2}) = [\varphi_6], \Gamma_2(\omega_{2_3}) = [\varphi_7], \Gamma_2(\omega_{2_4}) = [\varphi_8], \Gamma_2(\omega_{2_5}) = \Theta$;

- $\mathbb{P}_2(\omega_{2_j}) = p_{2_j}$ and $\sum_{j=1}^{5} p_{2_j} = 1$

$\Omega_3 = \{\omega_{3_1}, \omega_{3_2}, \omega_{3_3}\}$

- $\Gamma_3(\omega_{3_1}) = [\varphi_9]$, $\Gamma_3(\omega_{3_2}) = [\varphi_{10}]$, $\Gamma_2(\omega_{3_3}) = \Theta$,
- $\mathbb{P}_3(\omega_{3_j}) = p_{3_j}$ and $\sum_{j=1}^{3} p_{3_j} = 1$

$\Omega_4 = \{\omega_{4_1}, \omega_{4_2}, \omega_{4_3}\}$

- $\Gamma_4(\omega_{4_1}) = [\varphi_{11}]$, $\Gamma_4(\omega_{4_2}) = [\varphi_{12}]$, $\Gamma_4(\omega_{4_3}) = \Theta$,
- $\mathbb{P}_4(\omega_{4_j}) = p_{4_j}$ and $\sum_{j=1}^{3} p_{4_j} = 1$

■

It is important to notice that in this formulation the constant $\frac{1}{\mathbb{P}'(\Omega)}$, the measure of the conflict between evidences, is lower that in the first formulation. But there are elements of Ω that have $\Gamma'(\omega) = \emptyset$. For example the element (ω_1, ω_2) in the Example 3.14 has:

$$\Gamma'(\omega_{2_1}, \omega_{3_2}) = [\varphi_5] \cap [\varphi_{10}] = [x_2 > \bar{x}_2] \cap [x_1 : \bar{x}_2 > x_2] = \emptyset \, .$$

Proposition 3.8.1 *Formulation 1* and *Formulation 2* are equivalent.

Proof. We can obtain *Formulation 2* from *Formulation 1*: given k source triples $(\Omega'_i, \Gamma'_i, \mathbb{P}'_i)$ from the *Formulation 1*, we take (in the variables order) a set of the source triples $(\Omega'_i, \Gamma'_i, \mathbb{P}'_i)$, where each one corresponds to a statements in the the set $\{\forall j \in \{1, \cdots, k\} \ \varphi_{i_j} = (u : x > \bar{x}[W])\}$ for a particular u and X. Then we combine, using the Dempster-Shafer rule, this set of source triples to obtain a source triple in *Formulation 2*. If we iterate this process to all the sets $\{\forall j \in \{1, \cdots, k\} \ \varphi_{i_j} = (u : x > \bar{x}[W])\}$ for each assignments u we obtain the second formulation. ■

Observation 3.11 The PCP-nets are a particular case of this Dempster-Shafer generalization model, considering the following language (with $W = \emptyset$):

$$\Lambda_P = \{u : x_i > \bar{x}_i\} \cup \{u : \bar{x}_i > x_i\}$$

■

Using a Bayesian Network to describe the relation between the sets Ω_i

When we define the source triple using the Dempster-Shafer theory we make an independence assumptions in the relation between the sets Ω_i. We want now to relax this condition to obtain a more general and sometimes natural formulation.

Given the source triples $(\Omega_i, \mathbb{P}_i, \Gamma_i)$, we defined the probability function $\mathbb{P}$ as:

$$\mathbb{P}(\omega) = k \cdot \prod_{i=1}^{m} \mathbb{P}_i(\omega_i)$$

where $k = \frac{1}{\mathbb{P}'(\Omega)}$. Now we define a new probability function $\mathbb{P}$ as:

$$\mathbb{P}(\omega) = k \cdot \mathbb{P}_{BN}(\omega)$$

where $k = \frac{1}{\mathbb{P}'(\Omega)}$ and $\mathbb{P}_{BN}(\omega)$ is the joint probability of the Bayesian Network over ω.

We replaced the simple product of the probability implied by the independence assumption with a more complex probability distribution over the spaces Ω_i.

We can make two main assumptions on the structure of the Bayesian:

- *Independence assumption.* This assumption is the most common, and it is used in the classical Dempster-Shafer theory. It consists to consider the source triple independent to each other and thus to learn a Bayesian-net with a separable graph (without edges).
- *Non-Independence assumption.* It consists to consider the source triple dependents to each other and thus to learn a Bayesian-net with a general acyclic graph. The acyclicity is a common assumption. For a cyclic dependency graph we can use instead a Markov model to represents the function $\mathbb{P}_{BN}(\omega)$.

Dominance

Once we have this probabilistic structure, we may want to make a comparisons between outcomes. To compute the dominance between two outcomes is hard also in this case, but for poly-tree and tree structures we can use an efficient procedure to find a good approximation of the value. We can apply the method described in [95] to the source triples, to compute the an approximation of the dominance value using Monte-Carlo. We can use this method to compute also an approximation of the dominance value for a PCP-net. The basic algorithm [95] is to generate randomly an induces CP-net, following the probabilities in the G-net, and test the dominance. This is a good approximation and can be done in polynomial time if the PCP-net is poly-tree or tree structured, because dominance in these cases is polynomial for CP-nets.

3.8.3 A language equivalently expressive as Λ

In this section we propose a new language that is equivalently expressive as the language Λ.

The language Λ is defined as $\Lambda = \{u : x_i > \bar{x}_i[W]\} \cup \{u : \bar{x}_i > x_i[W]\}$. The formulation of Λ follows the definition of CP-theory statements. We study what happen if we follow instead the definition of CP-net statements and we add a new kind of statements that corresponds to the generalization from CP-net to CP-theories. Following this idea we define a new language $\tilde{\Lambda}$:

$$\tilde{\Lambda} = \{u : x_i > \bar{x}_i\} \cup \{u : \bar{x}_i > x_i\} \cup \{u : X_i > X_j\}$$

where $i \neq j$ and the statements $u : X_i > X_j$ means that the variable X_i is more important then the variable X_j given the assignments u of the parents variables.

The obtained language $\tilde{\Lambda}$ is equivalently expressive to the language Λ, but it has a richer semantics.

Proposition 3.8.2 The two languages Λ and $\tilde{\Lambda}$, defined above, are equivalent.

Proof. First we prove that all the statement of Λ can be obtained from $\tilde{\Lambda}$.

- The statements $u : x_i > \bar{x}_i[W]$ in Λ can be expressed as the following conjunction of the statements of $\tilde{\Lambda}$:

$$(u : x_i > \bar{x}_i) \wedge \left[\bigwedge_{Y \in W} (u : X_i > Y) \right] .$$

Now we prove that all the statement of Λ can be obtained from $\tilde{\Lambda}$

- The statements $u : x_i > \bar{x}_i$ in $\tilde{\Lambda}$ are coincident with the statements in Λ with $W = \emptyset$
- The statements $u : X_i > X_j$ in $\tilde{\Lambda}$ can be defined with a combination of the statements of Λ:

$$\forall w(uw : x_i > \bar{x}_i[X_j]) \ \vee \ (uw : \bar{x}_i > x_i[X_j]) \ .$$

where w is an assignments of the variables in $V \setminus (\{X_i, X_j\} \cup U)$

■

We can use equivalently the language $\tilde{\Lambda}$ or the language Λ in the two different formulation described in Section 3.8.2.

3.9 Summary and Discussion

In this chapter we analyzed a way to introduce probability in the CP-nets framework with the purpose of managing situations characterized by some form of uncertainty. A PCP-net could represent a situations in which we are unsure about our preference ordering over certain items, or there could be noise in our preference structure due to lack of precision in elicitation or sensor collection (e.g., measurement error from remote sensors). In order to model this, we need a structure for handling uncertainty and changes, for example via probabilistic information. Thus we generalized the CP-nets framework introducing preferences at the statements level. This introduction reflects a probability distribution also over the structure of the dependency graph. We then analyzed the set of CP-nets that are induced by a PCP-net, the set of all the deterministic realizations generated from a PCP-net. A PCP-net defines a probability distribution over this set.

Once defined this new structure, we studied how to reason with this new framework in terms of optimality and dominance, the two main reasoning tasks regarding conditional preferences.

First we studied the optimality task in PCP-nets. We defined and compared different notions of optimality: the most probable induced CP-net, the most probable optimal outcome and the optimal outcome of the most probable induced CP-net. The first notion concerns optimality about induced CP-nets and the last two about outcomes. We provide algorithms to compute these different tasks, that use a transposition two properly defined BNs: the Opt-net and the G-net.

The G-net defines the probability distribution over the set of induced CP-nets. For this reason the G-net is defined as a separable BN, since we are assuming that the probabilities of different orderings for different parents instantiations are independent to each other, considering each rule as a random variable. This may look as a strong simplification but it makes sense considering the typical kind of data that we are managing: a huge amount of information from a lot of individuals, each one giving only a few preferences thus generally providing sparse data. Lets think about recommender systems: usually each user gives information on a small set of features or items and any other information on the others. For this reason, when we aggregate user preferences, we obtain that it clearly makes sense to maintain these rules independent from each other, since each rule is obtained from a different user.

Obviously this independence assumption could be considered as a limit of our model, indeed

we propose a way to follow in Section 8.1 as future direction, for extending this representations of probability distributions over the statements, to a richer one, considering more structured G-nets.

Concearning outcome optimization, we focused on the most probable optimal outcome (computed using the Opt-net) and the optimal outcome of the most probable induced CP-net since both of them can be computed in polynomial time. Other notions of optimality could be defined, but, since the number of possible outcomes is exponential in the number of features, also the procedure to compute them is exponential. Thus we focused on the most probable optimal outcome and the optimal outcome of the most probable induced CP-net, notions of outcome optimization that can be computed in a tractable amount of time.

We then studied the dominance task, analyzing its computational cost. We showed that dominance for an acyclic PCP-net is a "hard" problem. Thus we studied the cases in which the dominance values are polynomially computable. In the case in which a PCP-net has a separable dependency graph we provided an exact procedure that has a polynomial computational complexity. The separable case is a very important class of preference relations on multi-attribute domains, as we can see in the economics or social choice literature (see for example [57]), and it is a first step towards analyzing dominance on more complex preferences.

We are also able to perform exact dominance for a general scenario, but, since this task is computationally hard we defined also an algorithm that compute an approximate value for the dominance. This procedure is computable in polynomial time and we proved experimentally that the result is really close to the exact value.

PCP-nets can be seen as a way to bring together BNs and CP-nets, thus allowing to model preference and probability information in one unified structure. We are interested in the use of a unique structure for example to represent a population of user in a multi agent context: to manage a unique structure instead of an entire population of users, is an advantage in several tasks, with respect to the standard voting procedures. For instance with a PCP-net we are able to compute a dominance query, task that is not allowed using the main standard voting procedures. Given a PCP-net representing a set of users we can compute the characteristics of the standard user, the user (described by a CP-net) that best represent the population preferences. Moreover, with a PCP-net it is not required to take in memory the profile of users, and it is not required, for the computation of a task, to query the whole profile.

We then investigated the problem of learning of PCP-nets, considering the case in which the PCP-nets have a separable dependency graph. We focus on passive learning: given a consistent set of dominance examples, we exact learn a PCP-net that is consistent with the set of examples. Focusing first on separable preferences is nontrivial and it is necessary as first step towards learning more complex preferences. Moreover in literature is well-known that a simple structure generalize concepts in a better way than more structured framework despite the fact that more complex structures may fit the examples better.

We compared our approach with the state of the art procedures about preference learning and we observed that restricting to the deterministic case, we obtain exactly the procedure proposed by Chevaleyre et al. in [22] also studied in [66]. Lang and Mengin in [66] proved that finding a separable CP-net from a consistent set of examples can be solved in polynomial time, since the

VC-dimension (a measure of the the expressive power of a learning algorithm) of separable CP-nets is polynomial in the input dimension and thus separable CP-nets are PAC-learnable. This result is consistent with our result since both methods are computable in a time that is polynomial in the number of features.

Many generalization of PCP-nets are still possible. We analysed two of them: a dynamic generalization and a generalization using a different notion of probability.

We considered the case of PCP-nets with dynamic modification of the structure or of the PCP-tables. We defined and showed how to reason with this new dynamic PCP-nets framework that can be updated without recomputing their entire structure, analysing the effects of these dynamic modification on computing the most probable optimal outcome and the most probable induced CP-net.

Finally we introduced a generalization of PCP-nets considering an alternative probability theory: the Dempster-Shafer theory of probability, allowing the representation of ignorance and using a rule to combine the effect of different sources of evidence in a more general context. We provided a description of this new formalism to describe probabilistic conditional preferences.

In the following chapter we will specifically study the interpretation of PCP-nets as multi-agent scenario, using preferences to reconcile possible conflicts between the users opinions.

4. Preference aggregation

Probabilistic conditional preference networks (PCP-nets) provide a compact representation of a probability distribution on a collection of CP-nets. In the first section of this chapter we view a PCP-net as the result of aggregating a collection of CP-nets into a single structure. We use the resulting PCP-net to perform collective reasoning tasks, e.g., determining the most preferred alternative, when a group of agents expresses their preferences via CP-nets or determining the most probable "standard user", the user that best represents the population of users, that could also not belong to the set of users in input. We propose two PCP-net based methods to perform CP-net aggregation: by extracting the probability distribution directly from the CP-nets (*proportion method*) or by minimizing the error between the aggregated structure and the set of CP-nets (*least square method*). We combine these two methods with the two ways of extracting an optimal outcome from a PCP-net, obtaining four approaches which can be seen as voting rules that take as input a profile of individual CP-nets and output a winning candidate. We show that the output of the sequential voting method with majority coincides with one of the four methods we define. This is not surprising, since this method has been defined to obtain the outcome which best satisfies the collection of agents. However, we can obtain this outcome by generating a PCP-net representing the collection of given CP-nets. This allows us to do more than just finding a collectively optimal outcome, since the PCP-net can be used also to respond to dominance queries or to find the next best outcome in a linearisation of the induced preference ordering. We analyze the four methods according to several axiomatic properties usually considered to be desirable in voting rules. We also perform an experimental comparison of the four methods, by taking their output and asking the individual CP-nets for a dominance query between pairs of such outcomes, with the aim of determining which candidate is the one that most satisfies the individual agents' preferences. The experimental results show that the proportional method strictly dominate the least-squares method.

Moreover, the optimal outcome of the most probable induced CP-net is empirically always better than the most probable optimal outcome.

With respect to CP-nets, soft constraints are less restrictive since they are not directional, and thus require no predefined ordering between the variables. This allows one to not tie the variable ordering used by the sequential procedure to the topology of the constraint graph of each agent. CP-nets and soft constraints profiles are really different to each other. The orderings induced by CP-net are different from the ones induced by soft constraints, since CP-nets do not support ties. Ties are typical in the soft constraints formulation. Another important difference is that outcome comparison, which is computationally difficult in CP-nets, easy in soft constrains, but computing the optimal solution in an a CP-net has similar computational complexity to that in a soft constraint problem.

We now conduct a study to undergo a study similar to the one performed in [65] for CP-nets in scenarios where the agents express their preferences via soft constraints. It shows that a sequential single-feature voting protocol can find a winner in polynomial time, and has several other desirable properties, when the CP-nets satisfy certain conditions on their dependencies. In [65], the CP-nets must be acyclic, and their dependency graphs must all be compatible with a given graph ordered according to the feature ordering in the voting procedure.

In the second section of this chapter we study a sequential preference aggregation procedure based on voting rules for settings where several agents express their preferences over a common set of variable assignments via soft constraints. We analyse this approach by providing both theoretical and experimental results. In particular, we show several theoretical properties of the approach also compared to the non-sequential one. We evaluate the performance of our approach experimentally by considering several widely used voting rules, i.e., Plurality, Borda, Copeland, and Approval, and we show that in these cases the sequential approach is convenient in terms of computation time, while satisfying the agents preferences just as much as the non-sequential approach. Also, the quality of the returned solution is not affected by the ordering of the features. We conclude by providing results on real preferences obtained from the PrefLib database.

Chapter structure and related publications

The work presented in this chapter has appeared in the proceedings of the following conferences and international workshops.

- ⋆ **"Voting with CP-nets using a Probabilistic Preference Structure"**, *C. Cornelio, U. Grandi, J. Goldsmith, N. Mattei, F. Rossi and K.B. Venable*, Proceedings of the 5th International Workshop on Computational Social Choice, ComSoC-14.
- ⋆ **"Reasoning with PCP-nets in a Multi-Agent Context"**, *C. Cornelio, U. Grandi, J. Goldsmith, N. Mattei, F. Rossi and K.B. Venable*, Proceedings of the International Conference on Autonomous Agents & Multiagent Systems 2015, AAMAS-15.

The chapter is organized as follows.

- In Section 4.1 we consider the multi-agent scenario in which agents represent their preferences by CP-nets. We introduce two different aggregation methods: *PR* and *LS*. We compare them using four voting rules, two defined on *PR* and two on *LS*. We provide both theoretical

and experimental analysis to perform this comparison. We also introduce a procedure to aggregate a profile of O-legal CP-nets into a poly-tree structured PCP-net, since, on this particular structure, the dominance task is polynomially computable.

- In Section 4.2 we consider the multi-agent scenario in which agents represent their preferences by soft constraints. We present a sequential voting procedure to aggregate preferences and we apply it to the main used voting rule: Plurality, Approval, Borda and Copeland. We compare these obtained sequential procedures to the non-sequential ones. We also provide an experimental analysis using both randomly generated profiles and real-world profiles.
- In Section 4.3 we provide a summary of the chapter and a discussion of the results.

4.1 Aggregation of CP-nets into a PCP-net

The aim of this section is the aggregation of the preferences of a set of users, represented using CP-nets, into the unique structure: a PCP-net. We are interested in this purpose because to manage a unique structure instead of an entire population of users, is an advantage in several tasks, with respect to the standard voting procedures. For instance with a PCP-net we are able to compute a dominance query, task that is not allowed using the main standard voting procedures. Given a PCP-net representing a set of users we can compute the characteristics of the *standard user*, the user (described by a CP-net) that best represent the population preferences. To compute this tasks we calculate the most probable induced CP-net and the user in output could also not belong to the initial profile since it represents common preferences. Moreover, with a PCP-net it is not required to take in memory the profile of users, that could be very large (for example in the case of elections we manage profiles with a very large number of voters). Also considering the computational time required for a tasks, such as computing the most probable optimal outcome, we note that using the standard voting procedure this computational time depends on how many users are in the profile. Using instead a unique structure we are avoiding this additional cost. Thus we have a computational complexity that is significantly lower since we are not querying to each agent the same task but only to one agent (the PCP-net) representing the profile distribution of preferences. Furthermore if we have evidence on some variables, or if we have to recompute a task under different conditions, we do not have to consider again the whole profile but only the aggregated PCP-net.

We consider collections of CP-nets [11], also called *profiles* in voting theory terminology. A profile of CP-nets is a set of CP-nets on the same set of variables: $P = (C_1, \cdots, C_m)$. We focus on O-legal profiles [65] of acyclic CP-nets where each variable has a binary domain.

Definition 4.1.1 (O-legality) A profile of m CP-nets over n variables is said to be O-legal if all the dependency graphs of the CP-nets share the same topological ordering of the variables.

Note that this assumption does not mean that all the CP-nets in the profile have the same dependency graph, but all the dependency graphs can be linearized with the same linear order (there exists at least one linear order but there could exist also more then one). This assumption is widely used in computational social choice and preference reasoning, and it is often sufficient to obtain tractable results.

Given a multi-attribute domain, the fundamental idea, developed in the research area of voting

theory, of decompose the instance into smaller, local sub-instances is based on the notion of O-legality: [63] defined sequentially decomposable voting rules: voting rules defined over all profiles that are compatible with a given order O. In this way the main reasoning tasks became tractable also in multi-issues contexts; thus O-legal profiles are fundamental and largely adopted in literature. In our scenario we require this condition for two main reasons: we need O-legality to compare our methods to sequential voting rule procedures; and we need O-legality to obtain an acyclic dependency graph of the aggregated PCP-net. We consider only acyclic-structured PCP-net since acyclic graphs have particular properties and tractable computational complexity for the main reasoning tasks.

Moreover, in many real-life domains it is reasonable to assume O-legality, for instance in the cases in which preferential dependencies between variables coincide for all agents, or considering on line elicitation of preferences: usually the users fill a fixed order sequence of questions each one regarding a single variable, and this order is equal for all the users.

We start from an O-legal profile of m CP-nets over variables $X_1, \cdots, X_n$ each with a binary domain. We define the *relative frequency* of a CP-net C_i, written as $freq_i$ as the percentage of times C_i appears in P. In the following, a profile will be written as:

$$P = ((C_1, freq_1), \cdots, (C_k, freq_k)) \quad \text{, with } \sum_{i=1}^{k} freq_i = 1 \ .$$

The use of relative frequencies is relevant in domains with a high number of voters variables and a low number of variables as in some cases (e.g. political parties) many agents may express the same preferences. We use the same formulation in the case where CP-net is unique, we can "count" all these as 1's and we merely introduce the counts in this way to ease discussion.

Given a profile P of CP-nets, there may be no PCP-net which induces exactly the same distribution over the CP-nets in P. However, the function that maps a PCP-net to its set of induced CP-nets is injective. Thus, we can make the following statements.

Observation 4.1 Given a empirical probability distribution over a set of CP-nets (that what we call a profile P

$$P = ((C_1, freq_1), \cdots, (C_k, freq_k)) \quad \text{, with } \sum_{i=1}^{k} freq_i = 1$$

where *freq* define the empirical distribution), there may exist no PCP-net inducing it, even if they have the same dependency graph.

To show this we consider a profile of CP-nets over two binary variables X_1 and X_2:

- $(C_1, 0.5)$: $(x_1 > \bar{x}_1), (x_1 : x_2 > \bar{x}_2)$ and $(\bar{x}_1 : \bar{x}_2 > x_2)$
- $(C_2, 0.4)$: $(x_1 > \bar{x}_1), (x_1 : \bar{x}_2 > x_2)$ and $(\bar{x}_1 : x_2 > \bar{x}_2)$
- $(C_3, 0.1)$: $(x_1 > \bar{x}_1)$ and $(x_2 > \bar{x}_2)$.

The PCP-net representing such a profile must satisfy the following system of equations, where $\bar{q} = (q_1^1, q_1^2, q_2^2)$, q_1^1 is the probability of $x_1 > \bar{x}_1$, q_1^2 is the probability of $x_1 : x_2 > \bar{x}_2$, and q_2^2 is the probability of $\bar{x}_1 : x_2 > \bar{x}_2$:

$$\begin{cases} fp_{C_1}(\bar{q}) = & q_1^1 q_1^2 (1-q_2^2) \\ fp_{C_2}(\bar{q}) = & q_1^1 (1-q_1^2) q_2^2 \\ fp_{C_3}(\bar{q}) = & q_1^1 q_1^2 q_2^2 \end{cases} \Rightarrow \begin{cases} fp_{C_1}(\bar{q}) = & 0.5 \\ fp_{C_2}(\bar{q}) = & 0.4 \\ fp_{C_3}(\bar{q}) = & 0.1 \end{cases}$$

This system has no solution for $\bar{q} \in [0,1]^3$. ■

Observation 4.2 Given a probability distribution over a set of CP-nets, we can compute a PCP-net to fit this distribution, if it exists.

To prove that f, the function mapping a PCP-net to its set of induced CP-nets, is injective we have to show that two different PCP-nets cannot generate the same set of induced CP-nets with the same probability distribution.

This is the same as proving that, if the system generated by a probability distribution over a set of CP-nets has a solution, it is unique.

We consider a PCP-net and then, to compute the probabilities, we consider the G-net associated. We suppose that the G-net has n variable $X_1, X_2, ..., X_n$ with domains $D_1 = \{a_j^1$: for $j = 1$ to $k_1\}$, ... , $D_n = \{a_j^n$: for $j = 1$ to $k_n\}$ (the values a_j^i can be an ordering, 0 or 1). We define p_j^i as the probability $\mathbb{P}(X_i = a_j^i)$.

Our system of equations (by definition) contains the numerical values of all the joint probabilities of every assignment to the n variables.

We suppose now that there are two distinct solution ${p_j^i}^*$ and ${p_j^i}^\diamond$ ($\forall i = 1, ..., n$ and $\forall j = 1, ..., k_i$).

We know that p_j^i is equal to:

$$\sum_{t_1=1}^{k_1} .. \sum_{t_{i-1}=1}^{k_{i-1}} \sum_{t_{i+1}=1}^{k_{i+1}} .. \sum_{t_n=1}^{k_n} \mathbb{P}(X_1 = a_{t_1}^1, .., X_{i-1} = a_{t_{i-1}}^{i-1}, X_i = a_j^i, X_{i+1} = a_{t_{i+1}}^{i+1}, .., X_n = a_{t_n}^n)$$

But all the numerical values of the joint probabilities are known (for all assignments of the n variables), so all the probabilities:

$$\mathbb{P}(X_1 = a_{t_1}^1, .., X_{i-1} = a_{t_{i-1}}^{i-1}, X_i = a_j^i, X_{i+1} = a_{t_{i+1}}^{i+1}, .., X_n = a_{t_n}^n)$$

are known as well.

Let us consider the value of p_j^i, which we call it b_j^i. Now since both ${p_j^i}^*$ and ${p_j^i}^\diamond$ are equal to b_j^i, then they coincide.

This allows us to conclude that there is a unique solution. ■

Observation 4.1 and 4.2 give us interesting clues about the practicality of eliciting and aggregating CP-nets. It indicates that we, in some instances, can derive a PCP-net from a profile of CP-nets. We may be able to use PCP-nets as a compact encoding of distributions over CP-nets as a reasonable model underlying recommendation systems for large databases of highly configurable products [46, 85].

Note that assuming that the probabilities of different orderings for different parents instantiations

are independent to each other (considering each rule as a random variable), may look as a strong simplification and approximation of a preference distribution over a population. Usually we expect that population preferences are highly correlated (discussed in Section 8.1 as future direction), but our approach focus on the typical kind of data that we are managing nowadays: a huge amount of information from a lot of individuals, each one gives only a few preferences thus generally providing sparse data. Lets think about recommender systems: usually each user give information on a small set of features or items and any other information on the others. For this reason, when we aggregate all of the user preferences we obtain that it clearly makes sense to maintain these rules independent from each other, since each rule is obtained from a different user.

In general, the system is over-constrained and will rarely admit a solution. Therefore, we need to define aggregation methods that work even when there is no PCP-net that exactly recovers the input profile of CP-nets.

4.1.1 Aggregation methods

We now define two methods to represent a profile of CP-nets using a PCP-net. As we are not guaranteed to find a PCP-net representing the exact distribution of induced CP-nets in the profile we must resort to methods approximating this input distribution. The first method we propose generates a PCP-net by taking the union of the dependency graphs of the given CP-nets and determining the probabilities in the PCP-tables from the frequencies of the CP-nets in the profile.

Definition 4.1.2 Given a profile of CP-nets $P = (C_i, freq_i)$, the *Proportion (PR)* aggregation method defines a PCP-net whose dependency graph is the union of the graphs of the CP-nets in the profile. Given a variable X and an assignment u to its parents, the probabilities in the PCP-tables are defined as follows:

$$\mathbb{P}(x > \bar{x}|u) = \sum_{C_i : x > \bar{x}|u} freq_i$$

(and $\mathbb{P}(\bar{x} > x|u) = 1 - \mathbb{P}(x > \bar{x}|u)$)), i.e., the probability of the ordering $x > \bar{x}$ for variable X, given assignment u of $Pa(X)$, is the sum of probabilities of the CP-nets that have that particular ordering over the domain of X, given u.

Observation 4.3 Note that the PCP-tables, built following Definition 4.1.2, are well defined. For each CP-net C_i we are able to say if $C_i : x > \bar{x}|u$ for each variable X and set of parents nodes U. Each parent of X in C_i is also parent of X in the aggregated PCP-net. This is true because the dependency graph of the PCP-net is the union of the graphs of the CP-nets in the profile, thus the dependency graph of C_i is a sub-graph of the dependency graph of the PCP-net. Thus U, the set of parent of X in the PCP-net are a super-set of the set of parents of X in C_i. Lets call the set of parents of X in C_i as the set V. Thus we obtain that $C_i \models x > \bar{x}|u$ if and only if $C_i \models x > \bar{x}|u \restriction_V$. ∎

The second method minimizes the mean squared error between the probability distribution induced by the PCP-net over the CP-nets given in the input and the relative frequency observed in the input.

Definition 4.1.3 Let $P = (C_i, freq_i)$ be a profile of CP-nets. The *Least Square (LS)* aggregating method defines a PCP-net whose underlying graph is the union of the graphs of the CP-nets and the probabilities q^i_j in the PCP-tables that solve the following problem:

$$\underset{\bar{q}\in[0,1]^r}{\operatorname{argmin}} \sum_{i=1}^{k} [fp_{C_i}(\bar{q}) - freq_i]^2$$

where $\bar{q}$ is the vector of q^i_j ordered lexicographically with i as the first variable, q^i_j is the probability variable of the j-th row of the PCP-table of the variable X_i in the PCP-net, r is the number of PCP-table-rows in the whole PCP-net, C_i are the k CP-nets observed in the profile P and $fp_{C_i}(\bar{q})$ is the function of probability of the CP-net C_i introduced in Definition 3.2.4.

Observation 4.4 The LP formulation of LS is the following:

$$\underset{\bar{q}}{\operatorname{argmin}} \sum_{i=1}^{k} [fp_{C_i}(\bar{q}) - freq_i]^2$$
$$q_i \geq 0 \ \forall i$$
$$q_i \leq 1 \ \forall i$$

The *LS* method objecting function is a sum of a linear number of components (one for each CP-net in the initial profile), however, to evaluate fp_{C_i} may require exponential time as the PCP-net resulting from a generic profile may have an exponential number of cp-statements. We can ensure that the union graph of an O-legal profile has bounded width – making *LS* a polynomial method – by assuming the following O-boundedness condition:

Definition 4.1.4 A profile satisfy the O-boundedness condition if for each feature X_j there are sets $PP(X_j) \subseteq \{X_1, \ldots, X_n\}$ of possible parents such that

- $|PP(X_j)| < K$ for all j and a given constant K
- for all individuals i, $Pa_i(X_j) \subseteq PP(X_j)$.

■

Computing the PCP-net using method *PR* may also require exponential time, as for *LS*, because the PCP-net resulting from a generic profile may have an exponential number of cp-statements. However, *PR* also becomes a polynomial method if we have the O-boundedness condition.

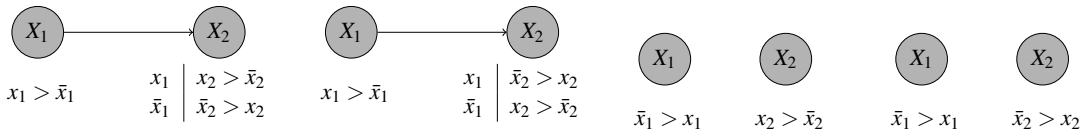
As *LS* formalized a notion of "distance to the best" we thought it was a fitting complement to *PR*'s "item by item majority best" method. We wanted two slightly different methods to compare and contrast.

4.1.2 Computing Optimality in a multi-agent context

Let $\mathscr{P}$ be the set of all CP-net profiles P of m voters over a set of alternatives X, a CP-voting rule $r : \mathscr{P} \to X$ is a function that maps each profile P into an alternative $r(P) \in X$.[1]

We define four voting rules by combining the two aggregation methods *PR* and *LS* presented

[1] In what follows we assume that CP-voting rules use lexicographic tie-breaking to return a unique winner.



(a) CP-net C_1. (b) CP-net C_2. (c) CP-net C_3. (d) CP-net C_4.

Figure 4.1: A CP-net profile P of the proof of Proposition 4.1.1

in Definition 4.1.2 and 4.1.3 with the two possible ways of extracting an optimal outcome from a PCP-net:

- PR_O: *PR* and most probable optimal outcome;
- PR_I: *PR* and optimal outcome of most probable induced CP-net;
- LS_O: *LS* and most probable optimal outcome;
- LS_I: *LS* and optimal outcome of most probable induced CP-net.

Computing the optimal outcome for PR_O and PR_I is polynomial if the graph of the resulting PCP-net has bounded width. This property can be obtained via the O-boundedness condition.

The four methods may produce different outcomes:

Proposition 4.1.1 There exists a profile of CP-nets P such that

$$\{PR_O(P), PR_I(P)\} \cap \{LS_O(P), LS_I(P)\} = \emptyset$$

and there exists P such that

$$PR_O(P) \neq PR_I(P)$$

and such that

$$(\text{or } LS_O(P) \neq LS_I(P)).$$

Proof.

- First we prove that there exists a profile of CP-nets P such that $\{PR_O(P), PR_I(P)\} \cap \{LS_O(P), LS_I(P)\} = \emptyset$. Let us take the following profile P in Figure 4.1 of four CP-nets over two variables X_1 and X_2:
  - **C_1** with probability 0.095. C_1 has the edge from X_1 to X_2 and CP-tables: $x_1 > \bar{x}_1$ and $x_1 : x_2 > \bar{x}_2, \bar{x}_1 : \bar{x}_2 > x_2$.
  - **C_2** with probability 0.505. C_2 has the edge from X_1 to X_2 and CP-tables: $x_1 > \bar{x}_1$ and , $x_1 : \bar{x}_2 > x_2, \bar{x}_1 : x_2 > \bar{x}_2$.
  - **C_3** with probability 0.005. C_3 does not have the edge from X_1 to X_2 and has CP-tables: $\bar{x}_1 > x_1$ and $x_2 > \bar{x}_2$.
  - **C_4** with probability 0.395. C_4 does not have the edge from X_1 to X_2 and has CP-tables: $\bar{x}_1 > x_1$ and $\bar{x}_2 > x_2$.

  PR gives us the PCP-net $(x_1 > \bar{x}_1, 0.6)$ and $(x_1 : x_2 > \bar{x}_2, 0.51)$, $(\bar{x}_1 : x_2 > \bar{x}_2, 0.1)$, while LS outputs $(x_1 > \bar{x}_1, 0.59)$ and $(x_1 : x_2 > \bar{x}_2, 0.29)$, $(\bar{x}_1 : x_2 > \bar{x}_2, 0)$. Thus we obtain that $PR_O(P) = \bar{x}_1\bar{x}_2$, $PR_I(P) = x_1x_2$, $LS_O(P) = x_1\bar{x}_2$ and $LS_I(P) = x_1\bar{x}_2$.

- The fact that there exists P such that $PR_O(P) \neq PR_I(P)$ (or $LS_O(P) \neq LS_I(P)$) is a immediate consequence of the fact that the most probable optimal outcome and the outcome of the most probable CP-net can be different. For example, consider a PCP-net $\mathscr{2}$ on two variables X_1 and X_2 with the PCP-tables $(x_1 > \bar{x}_1, 0.6)$, $(x_1 : x_2 > \bar{x}_2, 0.6)$ and $(\bar{x}_1 : x_2 > \bar{x}_2, 0)$. P is the profile generated by the set of induced CP-net by the PCP-net $\mathscr{2}$. The most probable induced CP-net has $x_1 > \bar{x}_1$, $x_1 : x_2 > \bar{x}_2$ and $\bar{x}_1 : \bar{x}_2 > x_2$, thus x_1x_2 is the optimal outcome. However, the most probable optimal outcome of the PCP-net is $\bar{x}_1\bar{x}_2$. Therefore, PR_O (respectively, LS_O) will differ from PR_I (resp. LS_I) on a profile P that produce this PCP-net.

■

It is interesting to observe that PR_I returns the same result as the *sequential voting rule with majority* [65] (see Section 2.5.2), that consists of applying the majority rule "locally" on each feature in the order given by O.

Theorem 4.1.2 Given any profile of CP-nets, PR_I produces the same result as sequential voting with majority.

Proof. Consider a variable X_i with domain $\{x_i, \bar{x}_i\}$, and an assignment u for the parents of X_i. With sequential voting we choose the value of the domain that corresponds to the first value of the ordering that maximizes the following:

$$\max_{j \in \{1,\cdots,m\}} \{[\sum_{C_j : x_i > \bar{x}_i | u} \mathbb{P}(C_j)], [1 - \sum_{C_j : x_i > \bar{x}_i | u} \mathbb{P}(C_j)]\}.$$

With PR_I, we create a PCP-net that has, for the row in the PCP-table of X_i corresponding to assignment u for its parents, the probability $\sum_{C_j : x_i > \bar{x}_i | u} \mathbb{P}(C_j)$ for $x_i > \bar{x}_i$ and $1 - \sum_{C_j : x_i > \bar{x}_i | u} \mathbb{P}(C_j)$ for $\bar{x}_i > x_i$. To compute the CP-tables of the most probable induced CP-net, we choose the orderings with maximal probability: for each variable X_i, given u, we choose the ordering that maximizes the following probability:

$$\max_{j \in \{1,\cdots,m\}} \{[\sum_{C_j : x_i > \bar{x}_i | u} \mathbb{P}(C_j)], [1 - \sum_{C_j : x_i > \bar{x}_i | u} \mathbb{P}(C_j)]\}.$$

To compute the optimal outcome of the most probable induced CP-net, we choose the greater literal in the ordering that appears in the CP-table, given u. This is for a generic variable X_i and assignment u, thus is true for all the variables and assignment of their parents. ■

Other notions of optimal outcome may be defined. For example, the outcome that maximizes the average number of outcomes worse than it in the induced CP-nets. But this kind of optimality is computationally hard and thus we focus on optimal outcomes that can be computed in polynomial time.

4.1.3 Axiomatic Properties

A first way to compare the four voting rules is by checking if they satisfy a set of desirable axiomatic properties [80]. We analyse a selection of properties corresponding to the classical

Property	PR_O	PR_I	LS_O	LS_I
Anonymity	$\surd$	$\surd$	$\surd$	$\surd$
Neutrality	$\surd$	$\surd$	$\surd$	$\surd$
Homogeneity	$\surd$	$\surd$	$\surd$	$\surd$
Opt-Monotonicity	$\surd$	$\surd$		
Consistency		$\surd$ [65]		
Participation	$\surd$	$\surd$		
Consensus	$\surd$*	$\surd$		

Table 4.1: Axiomatic Properties of PR_O, PR_I, LS_O and LS_I. *= true only over a single feature

axiomatic properties, usually considered in literature ([80], [65], etc.).

We recall the definitions of these properties (see Section 2.5.3), adapting them to the context of CP-nets profiles.

- **Anonymity** holds when the result is not sensitive to any permutation of the voters (that is, given a permutation σ on voters and the set of alternatives X, $r(\sigma(P),X) = r(P,X)$).
- **Neutrality** holds if, for any profile P and any permutation σ on alternatives X, then $r(P,X) = r(P,\sigma(X))$.
- **Homogeneity** holds when, for any profile P and any $s \in \mathbb{N}$, we have $r(P) = r(sP)$, where sP is a profile in which there are s copies of each voter in P.
- **Opt-Monotonicity** holds if, given two profiles $P = (C_1, \cdots, C_m)$ and $P' = (C'_1, \cdots, C'_m)$ where C'_i is obtained by C_i by changing the CP-tables so that $r(P)$ is the optimal outcome for C'_i, we have $r(P) = r(P')$
- **Consistency** holds if, given two disjoint profiles P_1 and P_2 such that $r(P_1) = r(P_2)$, we have $r(P_1 \cup P_2) = r(P_1) = r(P_2)$.
- **Participation** holds if, for any profile P and any CP-net C, we have $r(P \cup \{C\}) >_C r(P)$.
- **Consensus** holds if, for any profile $P = (C_1, \cdots, C_m)$, there is no alternative o such that $o >_{C_i} r(P)$, for all $i \in \{1, \cdots, m\}$.

Anonymity and neutrality both hold for all four voting rules. We know PR_I satisfies a stronger version of monotonicity and consistency (hence homogeneity) as it coincides with the sequential voting on O-legal profiles studied by Lang and Xia [65].

Our results in this section are summarized in the table below.

Theorem 4.1.3 *PR_O and and PR_I satisfy homogeneity.*

Proof. Consider a profile $P = (C_1, f_1), \cdots, (C_k, f_k)$ from which we can get the normalised frequencies $(C_1, \frac{f_1}{m}), \cdots, (C_k, \frac{f_k}{m})$. Considering N times each CP-net, we obtain the following distribution over Nm CP-nets $P' = (C_1, Nf_1), \cdots, (C_k, Nf_k)$ with the following normalised frequencies $(C_1, \frac{Nf_1}{Nm}), \cdots, (C_k, \frac{Nf_k}{Nm})$. The probability of a generic CP-net C_i in P' is $\frac{Nf_i}{Nm} = \frac{f_i}{m}$ which is the same as the probability generated by P. ■

Theorem 4.1.4 *LS_O and LS_I satisfy homogeneity.*

Proof. In the proof of Theorem 4.1.3 we add N copies of each CP-net to the original profile resulting in the same probability distribution over the CP-nets. This fact is true for any collection of CP-nets, therefore, we generate the same set of equations to minimize, and thus the same solution (PCP-net). ■

Theorem 4.1.5 *PR_O and PR_I satisfy opt-monotonicity.*

Proof. Let us consider two profiles $P = (C_1, \cdots, C_m)$ and $P' = (C'_1, \cdots, C'_m)$ where C'_i is obtained by C_i by changing the CP-tables so that $r(P)$ is the optimal outcome for C'_i. Let $\mathscr{Q}$ and $\mathscr{Q}'$ represent, respectively, the PCP-nets obtained using the PR aggregation method on P and P'. Let S be the set of rows of the PCP-tables in $\mathscr{Q}$ relevant to $r(P)$. By definition of PR, the only difference between $\mathscr{Q}$ and $\mathscr{Q}'$ is in the probabilities of the rows in S. More specifically, the changes required to obtain C'_i from C_i are such that the probability of the orderings favoring the values assigned in $r(P)$ will be higher in $\mathscr{Q}'$, while those favoring the values opposite to those in $r(P)$ will be lower. Thus, we have that $r(P) = r(P')$.

The result for PR_I can be directly obtained from Theorem 4.1.2, that is, by using the equivalence of PR_I with sequential majority. Let us denote with o the optimal outcome of C_i. By replacing C_i with C'_i we increase by one the number of votes for the values that are in $r(P)$ and not in o, we decrease by one those votes for values that are in o and not in $r(P)$ and we leave the vote count the same for the other values (that is, the common ones) unchanged. Thus, since $r(P)$ was the winner for sequential majority given P it will still be the winner given P'. Given Theorem 4.1.2 we can conclude that the same holds for PR_I. ■

Theorem 4.1.6 *PR_I and PR_O satisfy participation.*

Proof. We first consider the case $r = PR_I$. Consider a profile $P = (C_1, \cdots, C_m)$ and an additional CP-net C. We have to prove that $r(P \cup \{C\}) >_C r(P)$. PR gives us a PCP-net $\mathscr{Q}$ for P and a PCP-net $\mathscr{Q}'$ for $P \cup \{C\}$. Since P and $P \cup \{C\}$ are O-legal, let X be the first variable according to O such that $r(P \cup \{C\})|_X \neq r(P)|_X$ and let $u = r(P)|_{PA(X)}$. This means that in the most probable induced CP-net of $\mathscr{Q}$ there is a preference statement $u : r(P)|_X > r(P \cup \{C\})|_X$ and in $\mathscr{Q}'$ there is $u : r(P \cup \{C\})|_X > r(P)|_X$. Thus the probability $\mathbb{P}[u : r(P)|_X > r(P \cup \{C\})|_X] \geq 0.5$ in $\mathscr{Q}$, but ≤ 0.5 in $\mathscr{Q}'$. This means that C has the row $u : r(P \cup \{C\})|_X > r(P)|_X$ in X's CP-table. Thus the outcome $r(P \cup \{C\}) \geq_C r(P)$.

A similar reasoning applies to the case $r = PR_O$. In the PCP-tables of $\mathscr{Q}'$, the probabilities of the rows corresponding to the CP-tables of C increase with respect to $\mathscr{Q}$. Thus, for each feature X and each assignment u of its parents, the probability to obtain the first ranked value in the u rows of C CP-tables, increases. This means that it improves the result for C.

■

Theorem 4.1.7 *PR_I satisfies consensus and PR_O satisfies consensus over a single feature.*

Proof. We first consider the case $r = PR_I$. Consider profile $P = (C_1, \cdots, C_m)$ and assume there is an alternative o s.t. $o >_{C_i} r(P) \forall i \in \{1, \cdots, m\}$. Since P is O-legal, let X be the first variable according to O such that $o|_X \neq r(P)|_X$. Let u be the assignment to X's parents in o and $r(P)$ ($o|_{PA(X)} = r(P)_{PA(X)}$ since X is the first variable according to O in which they differ). Since $o >_{C_i} r(P) \forall i \in \{1, \cdots, m\}$, it must be that $u : o|_X > r(P)|_X \forall C_i$. Let us now consider the PCP-net $\mathcal{Q}$ obtained from P by PR. It is easy to see that in the PCP-table of X the probability of $u : o|_X > r(P)|_X$ will be strictly higher than the probability of $u : o|_X < r(P)|_X$, which implies that the most probable induced CP-net must have the row $u_i : o|_X > r(P)|_X$. The optimal outcome of the most probable induced CP-net must have the assignment $o|_X$ for the variable X because is ranked first in the table in the u row. But $r(P)|X \neq o|X$ and we have a contradiction.

A similar reasoning applies to the case $r = PR_O$, but in a weaker version. We will prove that, for any profile P, there is no alternative o such that o differs from $r(P)$ on only a single variable and $o >_{C_i} r(P) \forall C_i$. Assume that there were an alternative o such that $o >_{C_i} r(P)$, $\forall C_i$ and o differs from $r(P)$ only on the variable X. The probability of $u : o|_X > r(P)|_X$ in the PCP-table of X is equal to 1 because all the CP-nets in the profile prefer o to $r(P)$. Thus the probability of $r(P)$ is equal to 0, and we have a contradiction. ■

In conclusion, the aggregation method PR, generating the voting rules PR_O and PR_I, satisfies a good number of desirable axiomatic properties. Obtaining results for the LS method is rather hard, given that it is based on numerical optimization. In the next section we will compare the two methods experimentally.

4.1.4 Experimental Evaluation of Optimality

In this section we describe our experiments to evaluate the quality of the outcomes returned by the four voting rules. We compare the results of these four voting rules with a baseline named $PLUR$, which outputs the result of the plurality voting rule applied to the initial profile of CP-nets. $PLUR$ takes the optimal outcome of each CP-net and returns the outcome which is optimal for the largest number of CP-nets. We then compare these five voting rules using two different scoring functions, each of which is computed using dominance queries on the input profile of CP-nets.

Let F and G be two CP-voting rules, and let T be a set of O-legal CP-profiles. Given a profile P, we first define three metrics: $d_P^{>}(F,G) = |\{C \in P : F(P) >_C G(P)\}|$, $d_P^{<}(F,G) = |\{C \in P : G(P) >_C F(P)\}|$ and $d_P^{\bowtie}(F,G) = |\{C \in P : F(P) \bowtie_C G(P)\}|$. Note that $d_P^{>} + d_P^{<} + d_P^{\bowtie} = |P|$. We then define the function $Dom_P(F,G)$ as:

$$Dom_P(F,G) = \begin{cases} 1 & \text{if } d_P^{>}(F,G) > \max\{d_P^{<}(F,G), d_P^{\bowtie}(F,G)\} \\ -1 & \text{if } d_P^{<}(F,G) > \max\{d_P^{>}(F,G), d_P^{\bowtie}(F,G)\} \\ 0 & \text{otherwise.} \end{cases}$$

We use the following notion of pairwise score:

$$PairScore_T(F,G) = \frac{\sum_{P\in T} Dom_P(F,G)}{|T|}$$

Note that this score belongs to the interval $[-1,1]$. Our second scoring function is inspired by Copeland scoring:

$$CopelandScore_T(F) = \sum_{G\in V\setminus\{F\}} PairScore_T(F,G)$$

where $V = \{PR_O, PR_I, LS_O, LS_I, PLUR\}$. Observe that this score belongs to the interval $[-4,4]$.

For all our experiments we randomly generate profiles of CP-nets, and PCP-nets. Generating PCP-nets i.i.d. is non-trivial [2] and therefore we use an approximation method for random generation of CP-nets and probabilities.

To generate O-legal profiles of CP-nets, we consider a fixed ordering $X_1, \cdots, X_n$ of features. We also take as input the maximum in-degree for each feature, k. For each CP-net in the profile we first generate its acyclic dependency graph. For each feature X_i, we randomly choose its in-degree d, $0 \leq d \leq \min\{k, i-1\}$. Next, we randomly (using a uniform distribution) choose d parents from the features $\{X_1, \cdots, X_{i-1}\}$. When the graph is built, we fill in the CP tables choosing uniformly one element of the domain (since the domain is binary). For a PCP-net, we generate the dependency graph and CP tables as for a CP-net, and then we randomly assign probabilities to CPT rows. In all our experiments ties are broken lexicographically.

We show results for some specific parameter values, similar results were observed also for other values of n and k.

In both the experiments in Figure 4.2 and Figure 4.3, we compute the mean *CopelandScore* over 100 O-legal profiles of CP-nets and we consider two parameters: the number of features and the number of individuals in the profile. In the experiment in Figure 4.2 we investigate the quality of all the voting rules, varying the number of features and fixing the number of individuals in the profile. In Figure 4.3 we analyze the quality of the voting rules, varying the number of individuals in the profile and fixing the number of features.

In the first set of experiments (Figure 4.2) the profiles have 20 individual CP-nets and the number of features varies from 1 to 10, and each has at most 2 parents. We limit our experimental setting considering only a maximum of 10 nodes each one with at most 2 parents, since the computational complexity for computing dominance is exponential: for each pair of rule we have to test a dominance query for each different CP-net.

According to the *CopelandScore* score, the best voting rule is PR_I, and we observe that *PR* is consistently better than *LS* using either the *O* or *I* method. PR_I is consistently better than PR_O and LS_I is also better than LS_O. Hence, the most probable optimal outcome (*O*) is worse than the optimal outcome of the most probable induced CP-net (*I*) in both the *PR* and the *LS* method. The variance in Figure 4.2 could be explained by the fact that, with more features, the number of pairs of outcomes that are incomparable increases. There is also a noticeable difference of behavior between even and odd numbers of features (and even and odd numbers of individuals in Figure 4.3). Our conjecture is that an odd number of features (resp. individuals) leads to more decisiveness in

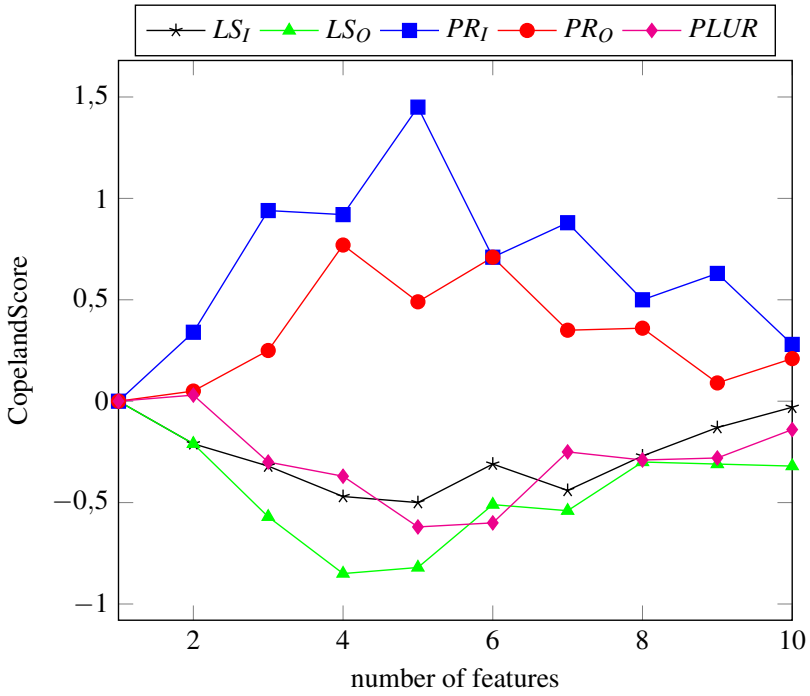


Figure 4.2: *CopelandScore* vs. the number of features.

the plurality voting rules.

In the second set of experiments (Figure 4.3) the profiles have $n = 3$ and at most $k = 1$ parent per feature, and the number of individual CP-nets varies in $[1,30]$. We observed that the number of CP-nets in the profile does not significantly influence the *CopelandScore* of the voting rule.

4.1.5 Computing Dominance in a multi-agent context using PR

In this section we analyse how *PR* performs on dominance testing. We show experimentally that the aggregation method *PR* is also efficient in terms of dominance. To show this we compare the result of dominance test on random pairs of outcomes in a CP-net profile with the result of dominance tests on the aggregated PCP-net corresponding to the CP-nets profile. Given two outcomes o and o', the dominance value $Dom(o,o')$ defined on an initial profile of CP-nets corresponds to the relative frequency of CP-nets that entail $o > o'$.

In the following experiment we generate randomly profiles of CP-nets and PCP-nets as we did in the previous section. The experiments, in Table 4.2, are performed on 50 iteration on profiles with 50 individual CP-nets and with a number of features that varies from 1 to 6, and each has at most 2 parents. For each profile we take 50 pairs of outcomes, and we consider the mean over these of the absolute value of the difference between the two dominance values. Also here we limit our experimental setting because of the computational costs of dominance.

In our experiments, in Table 4.2, the two values of dominance had maximum difference 0.047 when varying the number of features and the number of CP-nets in the profile. Thus the approximation induced by a PCP-net generated with the *PR* method, given a profile of CP-nets, is accurate both for optimality and for dominance.

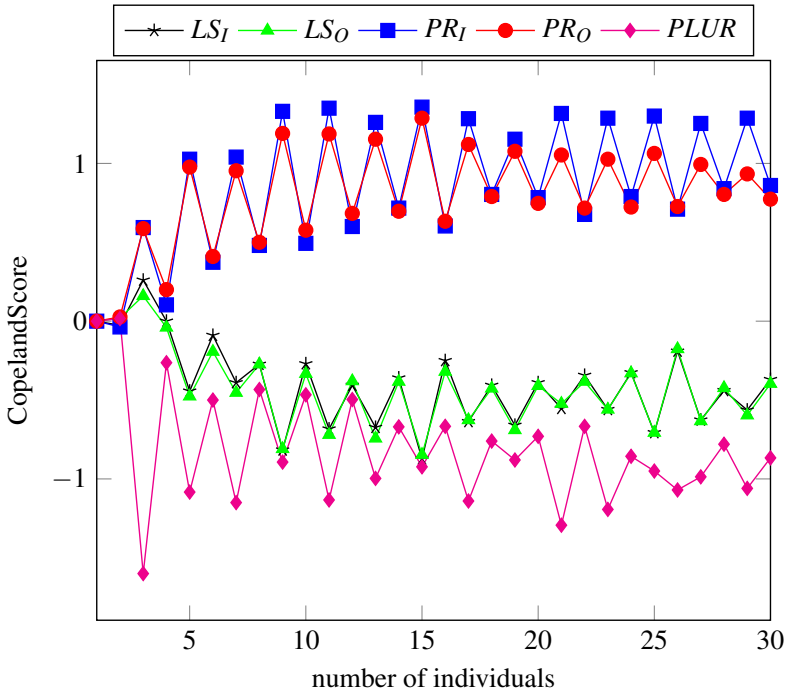


Figure 4.3: *CopelandScore* vs. the number of CP-nets.

n features	1	2	3	4	5	6
difference	0.0	0.023	0.028	0.037	0.045	0.047

Table 4.2: Difference between the value of dominance in the initial profile of CP-nets and in the PCP-net aggregated with PR.

Dominance as a Decision Problem

In some settings it could be enough to get a yes/no answer from a dominance query, rather than the exact probability value.

Given a profile of CP-nets and two outcomes, we compare MP-dominance queries on the PCP-net aggregated with *PR* with the answers to the true/false dominance queries on both (1) the initial profile of CP-nets and (2) the profile of CP-nets induced by the PCP-net. A profile of CP-nets returns *True* if the frequency of CP-nets that entail dominance is greater then the frequency of those that do not, and *False* otherwise. Our experiments show that the percentage of times these different methods give the same yes/no result is above 90% for both of the comparisons. This supports the fact that aggregating CP-nets with the *PR* method to get a PCP-net is also reasonable for dominance seen as a decision-making problem.

4.1.6 From a profile of O-legal CP-nets to a polytree-structured PCP-net

In section 4.1, when we consider a profile of agents described by CP-nets, we consider O-legal profile with bounded width by assuming the O-boundedness condition. If the aggregated PCP-net has a poly-tree structure for the dependency graph then we have many desirable properties, such as a good approximation of the dominance value computable in polynomial time. Thus we study how to aggregate an O-legal profile of CP-net (with the O-boundedness condition)into a PCP-net with a

poly-tree dependency graph.

We provide the following procedure:

- *step 0:* we aggregate the profile of CP-net in a PCP-net $\mathscr{Q}$ using the *PR* aggregating method, in the usual way;
- *step 1:* we find the minimum spanning polytree pT of the $\mathscr{Q}$ dependency graph;
- *step 2:* we build a new PCP-net $\mathscr{R}$ with pT as dependency graph.

We explain in details how to perform *step 1* and *step 2* in the following two sections. *step 0*, the *PR* aggregating method, is presented in Section 4.1 and it generates a PCP-net by taking the union of the dependency graphs of the given CP-nets and determining the probabilities in the PCP-tables from the frequency of the CP-nets in the profile.

In what follows we consider binary variables to simplify the notation; the non-binary case is an easy generalization.

Step 1: minimum spanning polytree

In this section we analyze the problem of find the minimum spanning polytree in a directed acyclic graph(DAG).

We recall the definition of polytree:

Definition 4.1.5 A polytree is a directed acyclic graph whose underlying undirected graph is a tree.

We note that find the minimum spanning polytree in a directed acyclic graph correspond to find the minimum spanning tree in another graph obtained from the DAG in input but undirected.

We can consider two different notions of minimality for the procedure, one that considers the problem without weights on the edges and one that considers weights that corresponds to the probability pot existence of the edges. Thus we can consider two different problems:

1. remove the minimal number of edges from the DAG to obtain the tree;
2. remove a set of edges that have a minimal weight from the DAG to obtain the tree.

For both of the notions of minimality we can use one of the classical algorithms for finding the minimum spanning tree in a DAG [27]:

- Kruskal algorithm with computational complexity equal to $O(m \log n)$;
- Prim algorithm with computational complexity equal to$O(n \log n + m \log n)$;
- Boruvka algorithm with computational complexity equal to$O(m \log n)$;

where m is the number of edges in the DAG and n is the number of nodes.

Once obtained the minimum undirected spanning tree we insert the directions on the edges corresponding to the initial directions in the original DAG.

Step 2: build the new PCP-net

The new PCP-net $\mathscr{R}$ has the pT dependency graph, where pT is the minimum spanning polytree computed on the initial PCP-net $\mathscr{Q}$ (step 1). In this section we explain how to build the PCP-tables of $\mathscr{R}$ from the PCP-tables of $\mathscr{Q}$.

Contraction procedure

For each node we make a transformation from its PCP-table in $\mathscr{Q}$ to its PCP-table in $\mathscr{R}$. We

have a number of step equals to the number of nodes, where at each step we are marginalize the probability of the entries of the PCP-table, using the assumption of a uniform distribution of each parent value.

We consider a node X with $k+l$ parents $Pa(X) = \{Pa_1Pa_2, \cdots, Pa_l, Y_1, \cdots, Y_k\}$ and we have to remove the parents $Y_1, \cdots, Y_k$.

X has the the following PCP-table:

$Pa(X)$	X	$\mathbb{P}$
$u_1y_1y_2\cdots y_k$	$x > \bar{x}$	$p_{1,1}$
$u_1\bar{y}_1y_2\cdots y_k$	$x > \bar{x}$	$p_{1,2}$
$\cdots$	$\cdots$	$\cdots$
$u_1\bar{y}_1\bar{y}_2\cdots\bar{y}_k$	$x > \bar{x}$	$p_{1,2^k}$
$\cdots$	$\cdots$	$\cdots$
$u_iy_1y_2\cdots y_k$	$x > \bar{x}$	$p_{i,1}$
$u_i\bar{y}_1y_2\cdots y_k$	$x > \bar{x}$	$p_{i,2}$
$\cdots$	$\cdots$	$\cdots$
$u_i\bar{y}_1\bar{y}_2\cdots\bar{y}_k$	$x > \bar{x}$	$p_{i,2^k}$
$\cdots$	$\cdots$	$\cdots$
$u_{2^k}y_1y_2\cdots y_k$	$x > \bar{x}$	$p_{2^k,1}$
$u_{2^k}\bar{y}_1y_2\cdots y_k$	$x > \bar{x}$	$p_{2^k,2}$
$\cdots$	$\cdots$	$\cdots$
$u_{2^k}\bar{y}_1\bar{y}_2\cdots\bar{y}_k$	$x > \bar{x}$	$p_{2^k,2^k}$

where u_i with $i \in \{1, \cdots, 2^l\}$ are all the possible assignments of the parents $Pa_1Pa_2, \cdots, Pa_l$ and the orderings $\bar{x} > x$ have implicitly the probability $1 - \mathbb{P}(x > \bar{x})$.

The new PCP-table for X is:

$Pa(X)$	X	$\mathbb{P}$
u_1	$x > \bar{x}$	$[\sum_{j=1}^{k} p_{1,j}]/2^k$
$\cdots$	$\cdots$	$\cdots$
u_i	$x > \bar{x}$	$[\sum_{j=1}^{k} p_{i,j}]/2^k$
$\cdots$	$\cdots$	$\cdots$
u_{2^l}	$x > \bar{x}$	$[\sum_{j=1}^{k} p_{2^l,j}]/2^k$

where the orderings $\bar{x} > x$ have implicitly the probability $1 - \mathbb{P}(x > \bar{x})$.

Thus for an assignment u_i of the parents $Pa_1Pa_2, \cdots, Pa_l$ the probability of the ordering $x > \bar{x}$ in the new PCP-table is:

$$\mathbb{P}(x > \bar{x}|u_i) = [\sum_{j=1}^{k} \mathbb{P}(x > \bar{x}|u_iv_j)]/2^k$$

where v_j with $j \in \{1, \cdots, 2^k\}$ are all the possible assignments of the parents $Y_1, Y_2, \cdots, Y_k$.

The total computational complexity is $O(n2^{|Pa(X)|})$. But the number of parents is bounded and we obtain $O(n2^k)$ that is $O(n)$.

Theorem 4.1.8 The contraction procedure minimize the mean quadratic error between the initial and final probability of outcomes.

Proof. We suppose to have an initial PCP-net $\mathscr{Q}$ and a final PCP-net $\mathscr{R}$ with a polytree dependency graph. The probability of an outcome o is computed using the bayesian network associated Opt-net. We build Opt-net($\mathscr{Q}$) and Opt-net($\mathscr{R}$). Opt-net($\mathscr{R}$) is the Bayesian network that minimize the quadratic error between the probability distribution over outcomes from Opt-net($\mathscr{Q}$).

The Opt-net is bayesian network associated to a given the PCP-net. Given the following PCP-table of a node X of the PCP-net:

$Pa(X)$	X	$\mathbb{P}$
u_1	$x > \bar{x}$	p_1
u_1	$\bar{x} > x$	$1-p_1$
$\dots$	$\dots$	$\dots$
u_i	$x > \bar{x}$	p_i
u_i	$\bar{x} > x$	$1-p_i$
$\dots$	$\dots$	$\dots$
u_k	$x > \bar{x}$	p_k
u_k	$\bar{x} > x$	$1-p_k$

the table for node X in the associated Opt-net is:

$Pa(X)$	X	$\mathbb{P}$
u_1	x	p_1
u_1	$\bar{x}$	$1-p_1$
$\dots$	$\dots$	$\dots$
u_i	x	p_i
u_i	$\bar{x}$	$1-p_i$
$\dots$	$\dots$	$\dots$
u_k	x	p_k
u_k	$\bar{x}$	$1-p_k$

How remove dependencies in a Bayesian network minimizing the quadratic error between the initial and final probability of outcomes is a well known problem [23] and it is solved as described in the contraction procedure but applied to the bayesian tables (using the analogies between PCP-tables and Opt-net tables). Opt-net($\mathscr{R}$) is the Bayesian network that minimize the mean quadratic error between the probability distribution over outcomes from Opt-net($\mathscr{Q}$). This implies that PCP-net $\mathscr{R}$ is the PCP-net that minimize the quadratic error between the probability distribution over outcomes from the PCP-net $\mathscr{Q}$. ■

4.2 Aggregation of soft constraint problems

Instead of considering profiles of agents in which preferences are represented via CP-nets, as described in the previous section, we can explore other formulations. For example in this section

we consider a soft constraints formulation of profiles.

In this section we present a sequential preference aggregation procedure based on voting rules for settings where several agents express their preferences over a common set of variable assignments via soft constraints. A similar approach has been considered when agents express their preferences via CP-nets in [65]. It shows that a sequential single-feature voting protocol can find a winner in polynomial time, and has several other desirable properties, only under certain conditions. In [65], the CP-nets must be acyclic, and their dependency graphs must all be *compatible* with a given graph ordered according to the feature ordering in the voting procedure. In [114] both such restrictions are relaxed via a procedure that exploits a graph defined on complete assignments. Other more specific approaches to sequential composition of voting rules with CP-nets that have been proposed are based on maximum likelihood estimators [115] and the maximin criterion [87]. Variants of CP-nets, such as LP trees or lexicographic trees have been considered in [26, 64].

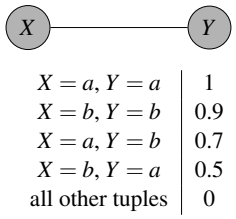
In this section we perform a theoretical study similar to [65] for soft constraints. Moreover, we provide also an experimental study of our sequential voting procedure for some widely used voting rules (i.e., Plurality, Borda, Copeland, and Approval).

The theoretical results that we will show for soft constraints are similar to those obtained in [65] for CP-nets. With respect to CP-nets, soft constraints allow one to avoid imposing many restrictions on the agents' preferences. In fact, contrarily to CP-nets, constraints are not directional, and thus information can flow from one variable of a constraint to another one without a predefined ordering between them. This allows one to not tie the variable ordering used by the sequential procedure to the topology of the constraint graph of each agent. Moreover, soft constraints can model also strict requirements, which are often necessary in multi-issue settings where one needs to rule out some combinations of feature values. Notice that the tractability assumption over the constraint graphs (that they should be tree-shaped) is similar to the assumptions that CP-nets are acyclic. However, contrarily to what is needed for CP-nets, with soft constraints one does not need to impose that the constraint graphs are compatible among them and with a graph structure based on the variable ordering.

The sequential procedure with soft constraints of this section has also been studied in terms of its resistance to bribery in [82, 83]. The bribery problem is defined by an external agent (the briber) who wants to influence the result of the voting process by convincing some voters to change their vote, in order to get a collective result which is more preferred to him; there is usually a limited budget to be spent by the briber to convince the voters [45]. Bribery issues in a sequential procedure have been considered also in [72, 74, 75, 77]. However, in these approaches agents express their preferences via CP-nets. In [72, 77] agents may also interact and influence each other.

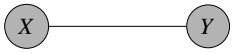
4.2.1 The sequential procedure

Assume a set of agents, each one expressing its preferences over a common set of objects via an SCSP (see the background Section 2.4) whose variable assignments correspond to the objects. Since the objects are common to all agents, this means that all the SCSPs have the same set of variables and the same variable domains but they may have different constraints, as well as different preferences over the variable domains. This is the notion of *soft profile*, which is formally defined



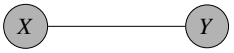
$X=a, Y=a$	1
$X=b, Y=b$	0.9
$X=a, Y=b$	0.7
$X=b, Y=a$	0.5
all other tuples	0

(a) P_1 and P_2.



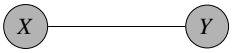
$X=a, Y=b$	1
$X=c, Y=c$	1
$X=b, Y=a$	0.9
all other tuples	0

(b) P_3.



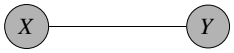
$X=a, Y=b$	1
$X=d, Y=d$	1
$X=b, Y=a$	0.9
all other tuples	0

(c) P_4.



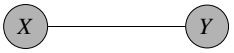
$X=a, Y=b$	1
$X=e, Y=e$	1
$X=b, Y=a$	0.9
all other tuples	0

(d) P_5.



$X=a, Y=b$	0.9
$X=f, Y=f$	1
$X=b, Y=a$	1
all other tuples	0

(e) P_6.



$X=a, Y=g$	0.9
$X=g, Y=b$	1
$X=b, Y=a$	1
all other tuples	0

(f) CP-net P_7.

Figure 4.4: A fuzzy profile.

as a triple $(\texttt{Var}, D, P)$ where $\texttt{Var}$ is a set of variables (also called issues), D is a sequence of $|\texttt{Var}|$ ordered finite domains, and P a sequence of m SCSPs over variables in $\texttt{Var}$ with domains in D. [2]

A *fuzzy profile* is a soft profile with fuzzy soft constraints. An example of a fuzzy profile where $\texttt{Var} = \{X, Y\}$, $Dom(X) = Dom(Y) = \{a, b, c, d, e, f, g\}$), and P is a sequence of seven FCSPs (Fuzzy Soft Constraints Profiles), is shown in Figure 4.4.

We consider fuzzy profiles where each voter has a tree-shaped set of fuzzy constraints.

The idea is to sequentially vote on each variable via a voting rule, possibly using a different voting rule for each variable. Given a soft profile $(\texttt{Var}, D, P)$, assume $|\texttt{Var}| = n$, and consider an ordering of such variables $O = \langle v_1, \ldots, v_n \rangle$, and a corresponding sequence of voting rules $R = \langle r_1, \ldots, r_n \rangle$ (that will be "local"). The sequential procedure is a sequence of n steps, where at each *STEP i* we process the variable X_i performing the following three points in order:

1. All agents are first asked for their preferences over the domain of variable X_i, yielding profile p_i over such a domain. To do this, the agents will achieve DAC on their SCSP, considering the ordering O. Ties are broken lexicographically if needed. For example, if we use Plurality we only need to break ties at the top level, while if we use Approval we don't need to break ties, since we give to the rule the set of optimal solutions. Using Borda we need to break all the ties in the whole agents' preference ordering.
2. Then, the voting rule r_i is applied to profile p_i, returning a winning assignment for variable X_i, say d_i. If there are ties, the first one following the given lexicographical order will be taken.
3. Finally, the constraint $X_i = d_i$ is added to the preferences of each agent and DAC is applied

[2] Notice that a soft profile consists of a collection of SCSPs over the same set of variables, while a profile (as in the classical social choice setting) is a collection of total orderings over a set of candidates.

to propagate its effect considering the reverse order of O.

After all n steps have been executed, the winning assignments are collected in the tuple $\langle d_1,\ldots,d_n\rangle$, i.e., the winner of the election. This is denoted by $Seq_{O,R}(\texttt{Var},D,P)$.

In the soft profile above, assume the variable ordering is $\langle X,Y\rangle$ and $r_i =$ Approval for all $i = 1,...,n$. In step 1, agents achieve DAC. This changes the preferences of the agents over X. For example, in P_1 and P_2, $X = a$ maintains preference value 1, $X = b$ gets preference value 0.9, and all other domain values get preference value 0, while in P_3, $X = a$ and $X = c$ maintain preference value 1, $X = b$ gets preference value 0.9, while all others get preference value 0. Then, Approval is applied over the domain of X where the sets of approved values are: $\{a\}$ for the first two voters and respectively $\{c,a\}$, $\{d,a\}$, $\{e,a\}$, $\{f,b\}$, and $\{g,b\}$ for the others. Thus, $X = a$ is chosen and the constraint $X = a$ added to all SCSPs, and its effect is propagated via DAC on the domain of Y. In step 2, DAC does not modify any preference value (since Y is the last variable) and the sets of approved values for Y are all equal and contain only b. Thus the elected solution with the sequential procedure is $s = (X = a,Y = b)$, which has preference value 0.7 for P_1 and P_2, 1 for P_3, P_4, and P_5, and 0.9 for P_6 and P_7.

An alternative to this sequential procedure would be to generate the preference orderings for each voter from their FCSPs, and then to aggregate them in one step via Approval. In our example, $(X = a,Y = b)$ gets 3 votes, $(X = b,Y = a)$ gets 2 votes, $(X = a,Y = a),(X = f,Y = f)$, $(X = d,Y = d)$, $(X = c,Y = c)$, $(X = e,Y = e)$, and $(X = g,Y = g)$ each gets 1 vote, while all other solutions get no vote. Thus the winner (breaking ties lexicographically) is $(X = a,Y = b)$.

The variable ordering which is used in the sequential procedure, is assumed to be given in this section. In practice, the variable ordering is chosen by the chair of the preference aggregation process, for example by using priority arguments (one may vote first on the most important features), or by aggregating the agents's preferences on such orderings (if the number of features is small, this voting process is feasible).

4.2.2 Properties of sequential vs. local voting rules

We will now prove several properties of this sequential preference aggregation procedure.

We will start by providing a general result which shows that if the sequential voting procedure satisfies a given property, so do all the local voting rules. We then show that that the opposite holds for anonymity, consistency, efficiency, (strong) monotonicity, and non-dictatorship.

The results are summarized in Table 4.3. In particular, we consider a sequential voting procedure where at each step we apply the local voting rule r_i to variable X_i, that is, $\text{Seq}_{\langle X_1,\ldots,X_n\rangle,\langle r_1,\ldots,r_n\rangle}$. The second column describes results regarding whether a property satisfied by all r_i is also satisfied by $\text{Seq}_{\langle X_1,\ldots,X_n\rangle,\langle r_1,\ldots,r_n\rangle}$, while the third column does the opposite. Notice that one of the results of efficiency holds only in the restricted case occurring when all the ordering induced by the SCSPs have a single top element.

From properties of the sequential procedure to properties of the local voting rules

In what follows, we prove that if one of the local rules does not satisfy a property, neither does the sequential rule. This is obviously equivalent to proving that if the sequential rule satisfies a

	Local. → Seq.	Seq. → Local
Condorcet Consist.	No	Yes
Anonimity	Yes	Yes
Neutrality	No	Yes
Consistency	Yes	Yes
Participation	No	Yes
Efficiency	Yes (unique top)	Yes
Monotonicity	Yes	Yes
Strong Monot.	Yes	Yes
IIA	No	Yes
Non-dictatorship	Yes	Yes
Strategy-proofness	No	Yes

Table 4.3: Property preservation.

given property so do all the local rules. To do so, we introduce a mechanism that lifts a profile on a single variable to a profiles defined on profile of SCSPs. We are in the context of a set of variables `Var` with domain D, and a set of m voters, given a profile p_i over a variable v_i. In this context we define a soft profile $Ext(p_i) = (\texttt{Var}, D, P)$ over `Var`, such that for any sequential procedure $\text{Seq}_{O,R}$ $\text{Seq}_{O,R}(Ext(p_i)) \downarrow v_i = r_i(p_i)$ (r_i is the i-th component of R), and, in each SCSP in P, the preference value of any solution coincides with the preference value of its projection on v_i. To achieve this, P consists of SCSPs with only unary constraints: the constraint over v_i respects the ordering in p_i, all other unary constraints are the same for all voters and associate preference value 1 to exactly one value per variable and 0 to all other values. Also, all voting rules are assumed to be unanimous. Intuitively, $Ext(p_i)$ extends the orderings over variable v_i given by p_i to a soft profile over all variables where the only significant preferences are those on variable v_i.

Theorem 4.2.1 If the sequential procedure satisfies property h, so do all the local rules.

Proof. As mentioned above, we will prove this in the opposite direction. Let us assume that one of the voting rules, say r_i does not satisfy h. Let us consider the set of soft profiles which can defined as extensions of profiles of variable v_i as described above. The correspondence between a profile p_i over v_i and soft profile $Ext(p_i)$ is such that the behavior of the sequential rule over extended profile $Ext(p_i)$ coincides exactly with the behavior of r_i on p_i. This in particular means that any set of profiles on v_i which constitute an example of how r_i violates h can be lifted to a set of soft profiles on which the sequential rule violates h. ■

From properties of the local voting rules to properties of the sequential procedure

Order independence. We may wonder how much the choice of the ordering O influences the result of our procedure. Of course, in general, different orderings will lead to different results. However, some orderings will produce the same result. Intuitively, these are orderings that differ just for the relative position of variables that are independent on each other, according to all agents.

More precisely, given an ordering O over `Var`, this can be used to direct all edges in the constraint graph of each agent, by setting the direction from v_1 to v_2 if v_1 is earlier than v_2 in O.

Let us call $G_i(O)$ this directed graph relative to agent i. If two variables v and v' are not connected in the transitive closure of this graph, then they are said to be independent.

Observation 4.5 Consider any two orderings O_1 and O_2 over $\mathtt{Var}$, that differ for the relative position of two variables v_1 and v_2, and assume these two variables are independent in $G_i(O)$, for all i. Then $\text{Seq}_{O_1,R}(\mathtt{Var},D,P) = \text{Seq}_{O_2,R}(\mathtt{Var},D,P)$.

To show this statement, we just need to prove that, if two variables v_i and v_j are independent for an agent, its preference ordering over v_i, given that v_j has already been instantiated, is the same as the one when v_i has still its whole domain. This is certainly so, since the fact that they are independent means that there is no constraint, nor path of constraints, involving them and thus assigning a value to v_j does not produce any change in the preferences assigned to values of v_i through propagation. ■

Given a soft profile, we may wonder if there exists a *best* variable ordering to use for the sequential voting procedure, where by *best* we mean that the resulting variable assignment reflects the preferences of the agents as much as possible. In Section 4.2.3 we show experimentally that while the variable ordering may lead to different winners, the voters are, on average, equally satisfied with any of such winners.

Condorcet consistency. It is natural to ask ourselves if the result returned by the sequential voting procedure has some relation with what is considered to be most preferred by the agents. In this respect, it is natural to consider the notion of Condorcet winner, which is classical in voting theory.

As defined above, a Condorcet winner (CW) is a candidate which is preferred to any other candidate by a majority of agents. Given a totally ordered profile, as in classical voting theory, there can be zero or one Condorcet winner. In our context, however, since we may have ties in the preference orderings of the agents, there could be more than one Condorcet winner, since several variable instantiations could be considered optimal for a majority of agents. We are thus considering weak Condorcet consistency, but for simplicity we refer to it here as Condorcet consistency. For any voting rule, it is desirable that it is Condorcet-consistent, that is, that it returns a Condorcet winner if there is one.

First, we define the notion of sequential Condorcet winner (SCW).

Definition 4.2.1 — SCW. Given a soft profile $(\mathtt{Var},D,P)$ with m agents and n variables, and an ordering O over $\mathtt{Var}$, $\langle d_1,\ldots,d_n\rangle$ is a SCW iff, for all $j = 1,\ldots,n$, $|\{i|d_j \in top(v_j,P_i|_{v_1=d_1,\ldots,v_{j-1}=d_{j-1}})\}| > m/2$. Where given an SCSP Q, we will denote as $Q|_{v_1=d_1,\cdots,v_h=d_h}$ the problem obtained from Q by fixing variables $v_1,\cdots,v_h$ to the corresponding values, and P_i denote the fuzzy constraint problem of agent i.

In words, a sequential Condorcet winner is the combination of local Condorcet winners.

If all the local rules are Condorcet consistent, the sequential voting procedure returns a SCW by definition, if it exists. However, to conclude that Seq is Condorcet consistent, we need to prove that SCW = CW.

Theorem 4.2.2 Given a soft profile $(\texttt{Var}, D, P)$ and an ordering O over $\texttt{Var}$, if d is a CW for $(\texttt{Var}, D, P)$, then it is a SCW for $(\texttt{Var}, D, P)$.

Proof. If $d = (d_1, \ldots, d_n)$ is a CW, then a majority of voters prefer it to all other candidates. Thus, at each step i of the sequential voting procedure, after DAC is enforced, each value in the domain of v_i will be associated with the highest preference associated to a solution containing it. This implies that (at least) the same majority will prefer d_i to all other values in the domain of v_i, given the values already chosen for the previous variables. ■

From Theorems 4.2.1 and 4.2.2 we know that if the sequential voting procedure is Condorcet consistent, all local voting rules are so. Unfortunately, the opposite does not hold in general, even if all voting rules are Condorcet consistent.

Theorem 4.2.3 If all local voting rules are Condorcet consistent, the sequential voting procedure may be not Condorcet consistent.

Proof. Consider a fuzzy profile $(\texttt{Var}, D, P)$ where: $\texttt{Var} = \{X, Y\}$, $Dom(X) = Dom(Y) = \{a, b\}$ and there are 5 agents. The fuzzy SCSPs of all agents have a single constraint over $\{X, Y\}$. For two agents we have: $(X = a, Y = b) = 0.9$, $(X = b, Y = b) = 0.8$, $(X = a, Y = a) = 0.7$, $(X = b, Y = a) = 0.6$; for one agent: $(X = a, Y = a) = 0.9$, $(X = a, Y = b) = 0.8$, $(X = b, Y = a) = 0.7$, $(X = b, Y = b) = 0.6$; for the other two agents: $(X = b, Y = a) = 0.9$, $(X = b, Y = b) = 0.8$, $(X = a, Y = a) = 0.7$, $(X = a, Y = b) = 0.6$. When each agent solves the problem and projects on variable X, for the first two agents we have $pref_X(a) = 0.9$ and $pref_X(b) = 0.8$; for the third agent $pref_X(a) = 0.9$ and $pref_X(b) = 0.7$; and for the last two agents $pref_X(a) = 0.7$ and $pref_X(b) = 0.9$. Thus, 3 over 5 agents agree that $X = a$ is optimal. Since the voting rule r_X is Condorcet-consistent, this value will be chosen for X. Given $X = a$, the preferences of the agents for Y are: for the first two agents $pref_Y(a) = 0.7$ and $pref_Y(b) = 0.9$; for the third agent $pref_Y(a) = 0.9$ and $pref_Y(b) = 0.7$; for the last two agents $pref_Y(a) = 0.7$ and $pref_Y(b) = 0.6$. Thus $Y = a$ will be chosen, since r_Y is Condorcet consistent, and $(X = a, Y = a)$ will be the SCW. However, $(X = a, Y = a)$ is not a CW, since the majority of the agents prefers $(X = b, Y = b)$. This means that there could be results of the sequential voting procedure that are not CWs. ■

To make sure that the procedure is Condorcet consistent, we need to impose some restrictions on the profile, similarly to what is done in [65]. One such restriction is, for example, imposing that each voter associates with each candidate (complete assignment) a different preference value. This ensures, that when considering the preference values associated to a single variable, after DAC, the values corresponding to the SCW will have a preference value strictly higher than all other values.

Anonymity. It is important to make sure that a preference aggregation system does not depend on the names or the order of the agents. This corresponds to saying that the rule is anonymous.

In our setting, a permutation of voter set corresponds, basically, to a permutation of the soft constraint problems. It is easy to see that anonymity at the local level implies global anonymity.

Proposition 4.2.4 If all the local rules are anonymous, so is the resulting sequential procedure.

Neutrality. Neutrality is a property that requires a rule to be insensitive to permutations of the candidates. This means that the result does not depend on the names of the candidates, but only on their position in the preference orderings. It is thus very important that a voting system is neutral.

We note that the candidates of the local voting rules are the values in the variable domains, while the candidates of the sequential voting rule are the complete assignments to all variables.

Theorem 4.2.5 Neutrality of the local rules does not imply neutrality of the sequential procedure.

Proof. While a permutation of the values in the domains always corresponds to a permutation of the variable assignments, not all of the permutations of variable assignments can be obtained via permutations of domain values. For example, if we have assignments: $s_1 = (X = a, Y = a), s_2 = (X = a, Y = b), s_3 = (X = b, Y = a)$, and $s_4 = (X = b, Y = b)$, the permutation that swaps s_1 and s_2, leaving s_3 and s_4 fixed, cannot be modelled by a permutation of domain values. In fact, a permutation that swaps s_1 and s_2 implies, at a local level swapping values a and b in the domain of variable Y. Now consider a soft profile of variables X and Y where all voters have the same SCSP assigning preference value 1 to $X = b$ and 0 to $X = a$, and preference value 1 to $Y = b$ and 0.5 to $Y = a$. The winner of the sequential procedure before swapping s_1 and s_2 is s_3. However if we run the sequential procedure after swapping s_1 and s_2, ans thus $Y = a$ with $Y = b$, assuming the local rules for X and Y are neutral, will return s_4. This implies that the sequential procedure is not neutral since s_3 was fixed in the permutation defined over complete outcomes. ■

Consistency. As defined above, a voting rule r is consistent if, when considering two profiles p_1 and p_2 with disjoint sets of voters, who vote over the same candidates, such that $r(p_1) = r(p_2)$, we have $r(p_1 \cup p_2) = r(p_1)$.

Theorem 4.2.6 If all the local voting rules are consistent, then the sequential voting procedure is consistent as well.

Proof. If all the local rules are consistent, at every step i of the sequential procedure, applied to profile $p_1 \cup p_2$, the result for variable v_i is the same as the result in profile p_1 (and also in profile p_2), so the overall result $(d_1, \ldots, d_n)$ will be the same as the result obtained by the sequential procedure in profile p_1 and in profile p_2. ■

Participation. Participation assures that voters have an incentive to vote.

Unfortunately, it is possible for all local voting rules to be participative, while the sequential voting procedure is not.

Theorem 4.2.7 If all the local voting rules are participative, the sequential voting procedure may not be participative.

Proof. Consider the profile where $\texttt{Var} = \{X, Y\}$, $D = (\{a, b, c\}, \{a, b\})$, and P is a sequence of two fuzzy SCSPs which coincide and contain a unary constraint on X (associating preference value 1 to a, 0.8 to b, and 0.6 to c), a binary constraint on X and Y (associating preference value 1 to (a, b), 0.9 to (a, a), 0.8 to (b, a), 0.7 to (b, b), 0.6 to (c, a), and 0.5 to (c, b)), and a unary constraint over Y (associating preference value 1 to both a and b). It is easy to see that the SCSPs of this soft

profile are DAC. Assume also that variables are ordered $X \prec_O Y$ and that r_1 is the scoring rule with score vector $(3,2,0)$ and r_2 is the majority rule. In this profile, $\text{Seq}_{O,R}(\texttt{Var}, D, P) = (X = a, Y = b)$. We now consider a third voter, with a fuzzy SCSP with a unary constraint on X (associating preference value 0.8 to a, 1 to b, and 0.9 to c), a binary constraint on X and Y (associating preference value 0.8 to (a,b), 0.5 to (a,a), 0.7 to (b,a), 1 to (b,b), 0.6 to (c,a), and 0.5 to (c,b)), and a unary constraint over Y (associating preference value 1 to both a and b). In this new profile P', $\text{Seq}_{O,R}(\texttt{Var}, D, P') = (X = b, Y = a)$. However, the third voter prefers $(X = a, Y = b)$ to $(X = b, Y = a)$. Thus the third voter would be better off not participating to the sequential voting process. ■

Efficiency. It is possible that all local voting rules are efficient, but the sequential voting procedure is not so. However, if we add the condition that there is a single candidate which is optimal for all agents, then the sequential voting procedure is efficient as well.

Theorem 4.2.8 If all the local voting rules are efficient, and there is a single candidate which is strictly preferred to all other candidates for all voters, then the sequential voting procedure is efficient.

Proof. If there is a single candidate, say $d = (d_1, \ldots, d_n)$ that is optimal for all agents, then we have that, after DAC, for each agent i and for each variable j, $top(v_j, P_i|_{v_1=d_1,\ldots,v_{j-1}=d_{j-1}}) = d_j$. Thus since each rule is efficient it will elect the value of d assigned to its variable. Thus $\text{Seq}_{O,R}(\texttt{Var}, D, P) = d$. ■

We note that for profiles where there is a unique candidate that is optimal for all agents, efficiency coincides with Condorcet consistency. Thus, given such profiles, the Condorcet consistency of the local rules implies the Condorcet consistency of the sequential rule.

Monotonicity and strong monotonicity. As defined above, a voting rule is monotonic if, when a candidate wins, and one or more voters improve their vote in favor of this candidate, then the same candidate still wins.

Theorem 4.2.9 If each local voting rule is monotonic, so is the sequential rule.

Proof. Let us assume that the sequential rule is not monotonic. This means that there is at least one soft profile $(\texttt{Var}, D, P)$ such that $\text{Seq}_{O,R}(\texttt{Var}, D, P) = d$ and another soft profile $(\texttt{Var}, D, P')$, where d has been moved up in the preference orderings of some agents, but $\text{Seq}_{O,R}(\texttt{Var}, D, P') = d' \neq d$. Let us denote with d_i, resp. d'_i, the value assigned to variable v_i in d, resp. d'. We note that there must be at least one value on which they differ. For each SCSP in P' and for each variable v_i, d_i has either improved w.r.t. d'_i or remained as in P. Let v_j be any of the variables such that $d_j \neq d'_j$. Then r_j is not monotonic since d_j has either improved or remained the same as d'_j but applying r_j to the profile over variable v_j has elected d'_j as a winner. ■

We note that, in contrast to the corresponding result proven for CP-nets in Proposition 6 of [65], in the case of soft constraints monotonicity is not determined only by the last voting rule but by any

of the voting rules. The same results can be proven for strong monotonicity with similar proofs: all local voting rules are strongly monotonic iff the sequential procedure is so.

Independence of Irrelevant Alternatives. The sequential procedure may be not IIA, even starting from IIA local rules.

Theorem 4.2.10 If all the local rules are IIA, the sequential procedure may be non-IIA.

Proof. Consider the fuzzy profile $(\texttt{Var}, D, P)$ in Figure 4.4, with the variable ordering $\langle X, Y\rangle$ and where r_i = Approval (which is IIA) for each i. In such a scenario the elected solution is $s = (X = a, Y = b)$.

Consider now a different profile, where the winner is $s' = (X = b, Y = a)$. This new soft profile is defined as $(\texttt{Var}, D, P')$ where P' is obtained from P by swapping the preference values of $(X = a, Y = a)$ and $(X = b, Y = b)$ in P_1 and P_2. If we run the sequential election on P', the sets of approved values for X are: $\{b\}$ for the first two voters and as in P for the other voters. Thus, b is chosen for variable X and, given this, the sets of approved values for Y are: $\{b\}$ for the first two voters and $\{a\}$ for the remaining five voters. Thus the winning solution is $s' = (X = b, Y = a)$. Thus, despite the fact of using IIA local rules on all variables, the sequential procedure is not IIA. ■

Non-dictatorship. If all the local rules are non-dictatorial the sequential approach is non-dictatorial as well.

Theorem 4.2.11 If all local rules are non-dictatorial, then the sequential procedure is non-dictatorial.

Proof. Take any rule r_i which, by assumption, is non-dictatorial. Thus, for each voter j there exists a profile over variable v_i, say p_{ij}, such that $r_i(p_{ij})$ is not a top element for j. Let us now consider the soft profile $Ext(p_{ij})$. Since $\text{Seq}_{O,R}(Ext(p_{ij})) \downarrow v_i = r_i(p_{ij})$, it is not one of j's top candidates. Therefore $\text{Seq}_{O,R}(Ext(p_{ij}))$ is not the top candidate of any voter. ■

By Theorem 4.2.1 we also know that if the sequential procedure is dictatorial then at least one of the local rules is dictatorial and we note that the dictator is the same for the sequential and local rule.

Strategy-proofness. From Theorem 4.2.1 we know that, the presence of a local rule that is manipulable jeopardizes the strategy-proofness of the sequential approach. We now prove that the strategy-proofness of all local rules is not sufficient for ensuring the strategy-proofness of the sequential procedure.

Theorem 4.2.12 If all the local rules are strategy-proof, the sequential procedure may be non-strategy-proof.

Proof. Consider the fuzzy profile with $\texttt{Var} = \{X, Y\}$, $D = \{a, b\}$, and $P = \{P_1, P_2, P_3\}$, as shown in Figure 4.5.

Consider the sequential procedure where $O = \langle X, Y\rangle$ and r_i=Plurality for $i = 1, 2$ (Plurality over two candidates is strategy-proof [6]). The sequential winner is $(X = a, Y = b)$. However, if the first

$X=a, Y=a$	1
$X=b, Y=a$	0.9
$X=a, Y=b$	0.8
$X=b, Y=b$	0.7

(a) P_1.

$X=a, Y=b$	1
$X=b, Y=a$	0.9
$X=a, Y=a$	0.8
$X=b, Y=b$	0.7

(b) P_2.

$X=b, Y=a$	1
$X=b, Y=b$	0.9
$X=a, Y=b$	0.8
$X=a, Y=a$	0.7

(c) P_3.

Figure 4.5: A fuzzy profile.

voter lies by swapping the preference values of $(X=a,Y=a)$ and $(X=b,Y=a)$ in P_1, then in the first step $x=b$ wins, and the final winner is $(X=b,Y=a)$, which is better for him. Thus, the sequential procedure is not strategy-proof. ■

A similar result has been shown for acyclic CP-nets with a common order over issues in [113].

We turn our attention to the complexity of a special kind of manipulation, called coalitional constructive manipulation. Coalitional constructive manipulation, denoted by $CCM(d,C,r,p)$, is the problem of deciding if a set (called coalition) of voters C can make candidate d win in profile p via voting rule r [6, 25].

We can prove that, if CCM (coalitional constructive manipulation) is easy for all the local rules, it remains so for the sequential approach. Actually, when obtaining the desired manipulation is easy at the local level, our result gives also a polynomial algorithm to set the preference values of the coalition in order to manipulate at the sequential level. Conversely, if CCM is difficult for at least one of the local rules, then it is so for the sequential procedure.

Theorem 4.2.13 If CCM is in P for all the local rules, then it is in P also for the sequential procedure.

Proof. Consider a DAC soft profile $(\texttt{Var}\ ,,P)$ with n variables and m voters such that all the SCSPs, except thoDse of agents in the coalition, have been specified. The goal of the coalition C is to make candidate $d=(d_1,d_2,\ldots,d_n)$ win. The sequential procedure $\text{Seq}_{O,R}$ used is such that $CCM(d_i,C,r_i,P)$ is in P for every r_i in R.

We consider the rules one at a time, following O. It is possible to determine in polynomial time if d_1 can win the election on the domain of v_1 and, if so, how the coalition should vote on the domain values of v_1 to achieve this. If d_1 cannot win the local election, then candidate d cannot win the sequential election. If instead d_1 can win on v_1, then we add for each agent in the coalition a unary constraint on v_1 to his SCSP simulating the ordering that that agent in the coalition must give in order to manipulate successfully. In order to so, the best elements in the ordering are given preference 1 and the following values are given any preference that respects the ordering. Then, $v_1=d_1$ is fixed in the SCSPs of all the agents not in the coalition, the result is propagated, and DAC is restored.

Next, in polynomial time the possibility of a constructive manipulation for d_2 given r_2 is

checked, and we proceed as above, until all the variables have been considered. After (at most) n polynomial steps, we will either have determined that d cannot win, or we will have defined the SCSPs of the agents in the coalition so that d does win. We notice that such SCSPs have only unary constraints and thus can be solved in polynomial time. ■

Conversely, if coalitional constructive manipulation is difficult for at least one of the local rules, then it is so for the sequential procedure. This result is a consequence of Theorem 4.2.1. Since, unlike the other properties, the complexity is involved in the formulation of this property, and for the sake of clarity we write the explicit proof below.

Theorem 4.2.14 If all the local rules are polynomially computable and CCM is NP-complete for one of the local rules, then it is so also for the sequential procedure.

Proof. We reduce polynomially an instance of CCM for a voting rule r to an instance of CCM for the sequential procedure where one of the local rules is r. An instance of CCM for r consists of a candidate d, a coalition C, and a profile p, written as $CCM(d,C,r,p)$. From such an instance, we construct an instance of CCM for the sequential procedure $\text{Seq}_{O,R}$ where R is any finite sequence of voting rules $(r_1,\ldots,r_n)$ including r, say in the i-th position, and O is any ordering of variables $v_1,\ldots,v_n$, all having the same domain, containing all the candidates appearing in p. Also, the soft profile considered for such an instance is $Ext(p)$, the coalition remains C, and the candidate is $E(d) = (d_1,\ldots,d_n)$ where $d_i = d$ and d_j for $j \neq i$ has preference value 1 in the SCSPs of $Ext(p)$. We will write it $CCM(E(d),C,\text{Seq}_{O,R},Ext(p))$.

If $CCM(d,C,r,p) = true$, then, in the sequential instance, coalition C can set its preference values in profile $Ext(p)$ to make $E(d)$ win. In fact, such agents can modify their preference values over the domain of v_i in the same way as needed to make d win in the non-sequential instance. Conversely, if $CCM(E(d),C,\text{Seq}_{O,R},Ext(p)) = true$, then we compute the orderings over the domain of v_i when $v_1 = d_1,\ldots v_{i-1} = d_{i-1}$ (via DAC). By using such orderings, the coalition can make d win in the non-sequential instance. ■

Computational analysis

In this section we provide a theoretical analysis of the computational costs of the voting rules we considered, both in their sequential and in their non-sequential version.

All the sequential voting rules have the same upper bound on the time complexity: $O(m \cdot n^2 \cdot d^2)$, where m is the number of agents in the profile, n is the number of features and d is the cardinality of the features domains. The part $n \cdot d^2$ corresponds to the cost of constraints propagation for a tree-shaped agent, but during the sequential procedure we need to propagate the constraints every time that we assign a feature, so n times for each agent, obtaining $O(m \cdot n^2 \cdot d^2)$.

For the non-sequential voting rules we have the following computational costs:

- *Plurality*: $O(m \cdot n \cdot d^2)$. It needs to find the optimal solution for all the agents. The computational cost for finding the optimal solution for one tree-shaped agent is $n \cdot d^2$.
- *Approval*: $O(m \cdot d^n)$. For each possible outcome (there are d^n possible outcomes), it needs to check if is approved by an agent, and then it need to repeat for all the agents.
- *Borda*: $O(m \cdot d^n)$. Each agent gives a score for each possible outcomes.

- *Copeland*: $O(m \cdot d^{2n})$. Each agent gives a dominance answer for each possible pair of outcomes.

Therefore, all the sequential voting rules can be computed in polynomial time in the input size, while the non-sequential ones, except Plurality, have an exponential computational cost. The sequential voting rule seq(Plurality), is slightly worse than the non-sequential Plurality, but still polynomial.

4.2.3 Experimental results

The theoretical results of Section 4.2.2 seem to suggest that the impact of the information loss caused by using a sequential approach may be non-negligible, since many properties are not transferred from the local to the sequential level. We now evaluate this from the experimental point of view.

If r is a voting rule, we denote by $seq(r)$ the sequential procedure where r is applied at each step. We will compare r and $seq(r)$ for several voting rules r in terms of computational cost, satisfaction of voters, sensitivity to the change of the order used for the sequential vote, and how they perform on real profiles.

Our experimental results, performed with both synthetic and real data, show that the winners provided by r and $seq(r)$ are very close in terms of voters' satisfaction, and that the time saving achieved by the sequential rule is substantial.

Experimental setting for synthetic preference profiles

We have randomly generated profiles with tree-shaped FCSPs based on the following parameters:

- the number of voters m,
- the number of variables n,
- the number of domain elements d, and
- the tightness t (the percentage of tuples with preference 0 in each fuzzy constraint).

These are common parameters when generating random fuzzy CSPS.

Given values to such parameters, we generate a profile with m tree-shaped FCSPs defined over n variables, with d elements in the domain of each variable, and where each fuzzy constraint involves two variables and has a number of tuples with preference 0 which is $t\%$ of the maximum number of tuples (that is, d^2). For the other tuples, we uniformly generate preference values in $(0,1]$. For each rule r that is considered, the soft profile generated in this way is given in input to $seq(r)$. Also, the profile obtained by computing the agents' induced preference orderings over the solutions is given in input to the rule r.

In each set of tests, we fix all parameters except one, and let this last parameter vary. When a parameter is fixed, we use the following values: $m = 25$, $n = 5$, $d = 5$, and $t = 20\%$. Also, each result is the average over 100 fuzzy profiles with the same parameters' values. All the results show also the average error, but, since we are averaging over 100, most of the time these errors are marginal.

The experiments were conducted on a machine with an Eight-Core Intel(R) Xeon(R) 2.40GHz with 256GB of RAM. We show detailed results for Borda, Plurality, Approval, Copeland and their

corresponding sequential versions.

Computation time

We compare r and $seq(r)$ in terms of their computation time.

In Figure 4.6, Figure 4.7, Figure 4.8 and Figure 4.9 we provide the results for Plurality, Approval, seq(Plurality), seq(Approval), seq(Borda) and seq(Copeland). In Figure 4.10, Figure 4.11, Figure 4.12 and Figure 4.13 we show separately the results for Borda since the computational time for Borda has a different order of magnitude. For the same reason we provide the results for Copeland in Figure 4.14, Figure 4.15, Figure 4.16 and Figure 4.17.

As it can be seen in the figures Figure 4.6, Figure 4.7, Figure 4.8, Figure 4.9, Figure 4.10, Figure 4.11, Figure 4.12, Figure 4.13, Figure 4.14, Figure 4.15, Figure 4.16 and Figure 4.17 the sequential procedure substantially outperforms the non-sequential rule in terms of the time needed to find the winner for Borda and Copeland. For Approval and Plurality the performance is approximately equal, but the non-sequential voting rules performs slightly better, as predicted by the theoretical analysis of the computational cost (Section 4.2.2).

However, the time for the non-sequential approach varies depending on the rule that is considered. The time for Approval and Plurality is similar since the task is that of computing all or one optimal solution. One may wonder why computing all optimal solutions is not much more expensive than computing just one. We recall that we are considering tree-shaped fuzzy SCSPs, where the first optimal solution in lex order can be computed in polynomial time. Once such a solution is found, all the other optimal solutions can be computed efficiently by exploiting the tree-structure [16]. Borda and Copeland instead require the computation of the entire solution ordering. Borda needs to extract the complete ordering between all the outcomes, and Copeland needs also the overhead of considering all possible pairwise competitions among all the solutions. The computational time for the sequential voting rules has a similar trend and order of magnitude to Approval and Plurality (Figure 4.6, Figure 4.7, Figure 4.8, Figure 4.9, Figure 4.10, Figure 4.11, Figure 4.12, Figure 4.13, Figure 4.14, Figure 4.15, Figure 4.16 and Figure 4.17).

An increase in the values of all the parameters, except tightness, causes a non-trivial growth in the computation time of all the approaches (sequential and non-sequential). However, an increase in the values of the domain size and in the number of features induces an exponential growth of the computational time, but increasing the number of agents we obtain a linear increasing of the computational time. On the other hand, by varying the tightness the computational time does not change. This is not surprising since the amount of computational steps remains the same.

Solution quality

To assess the quality of a solution, we have measured:

1. *the preference*: the value of the preference of the winner outcome for each voter, averaged over all voters;
2. *the error*: the distance, for each voter, between his preference value for the winner and the preference value of his optimal solutions, averaged over all voters, computed as the difference between the two values of preference.

We call the second measure the “error" of the winner outcome.

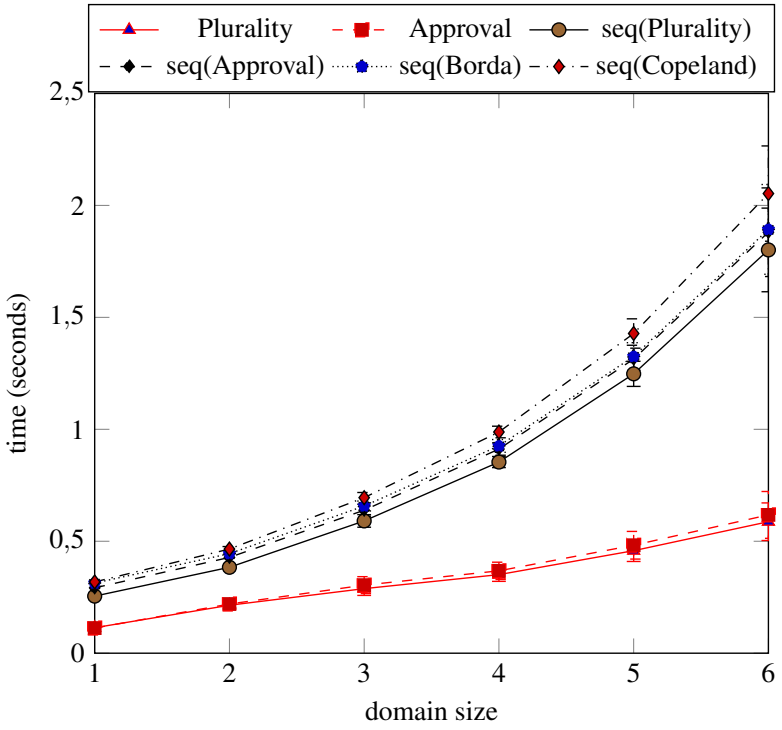


Figure 4.6: Computational time for Approval, Plurality, seq(Plurality), seq(Approval), seq(Borda), seq(Copeland), varying the domain size.

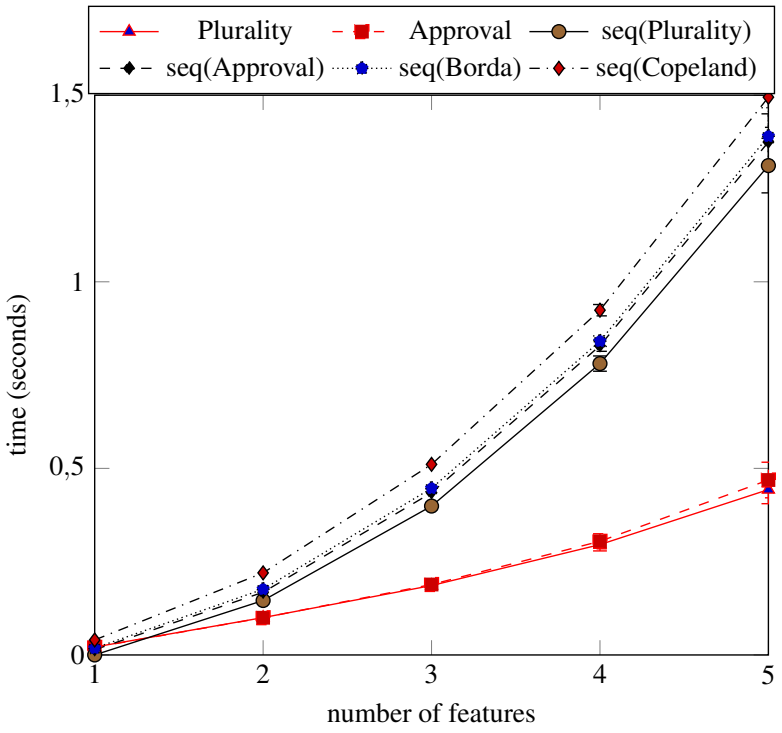


Figure 4.7: Computational time for Approval, Plurality, seq(Plurality), seq(Approval), seq(Borda), seq(Copeland), varying the number of features.

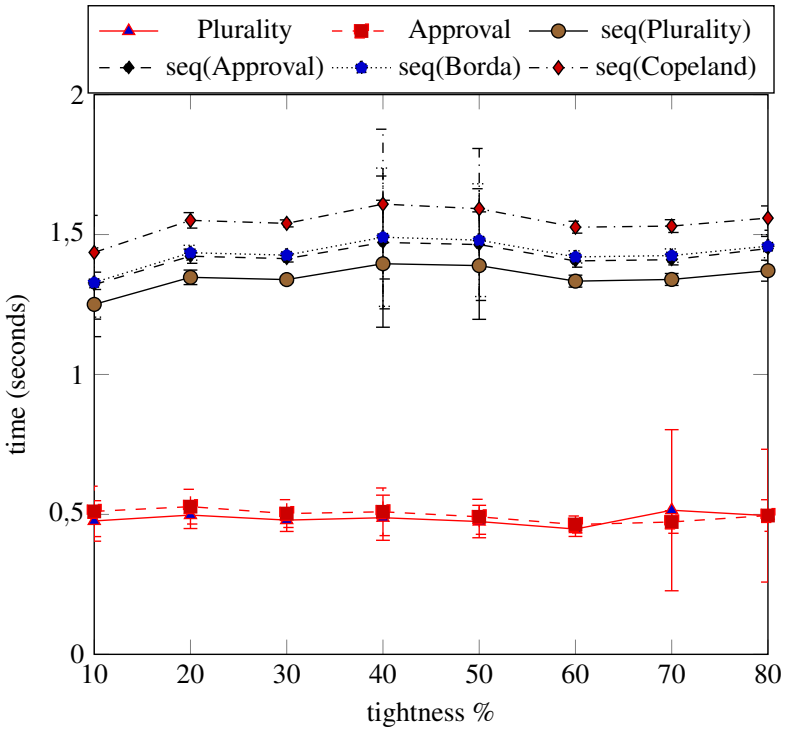


Figure 4.8: Computational time for Approval, Plurality, seq(Plurality), seq(Approval), seq(Borda), seq(Copeland), varying the tightness.

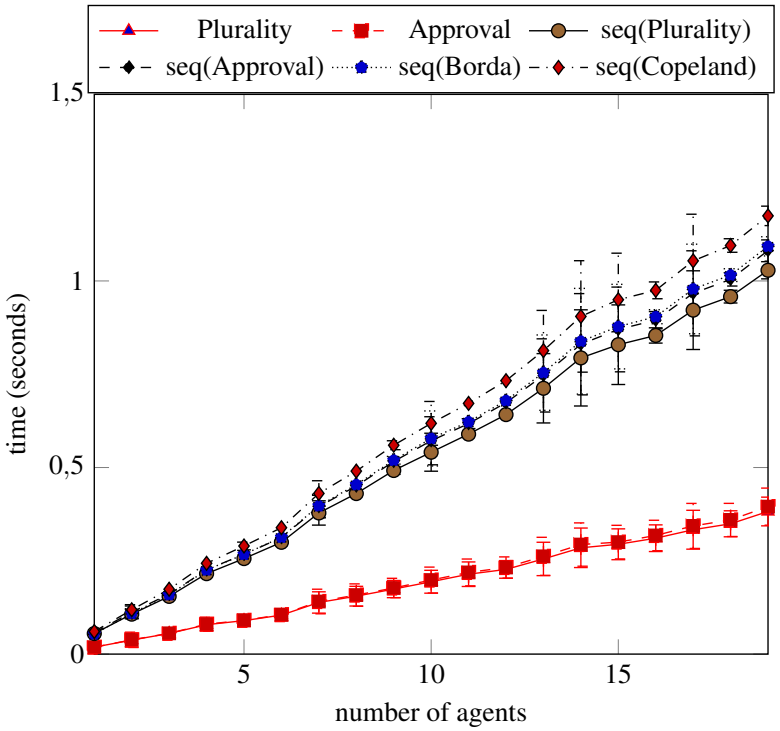


Figure 4.9: Computational time for Approval, Plurality, seq(Plurality), seq(Approval), seq(Borda), seq(Copeland), varying the number of agents.

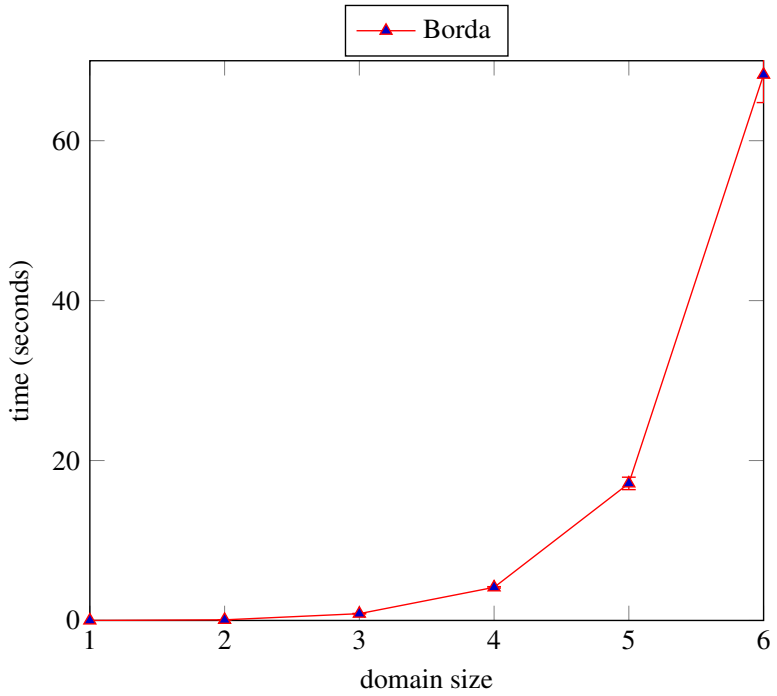


Figure 4.10: Computational time for Borda, varying the domain size.

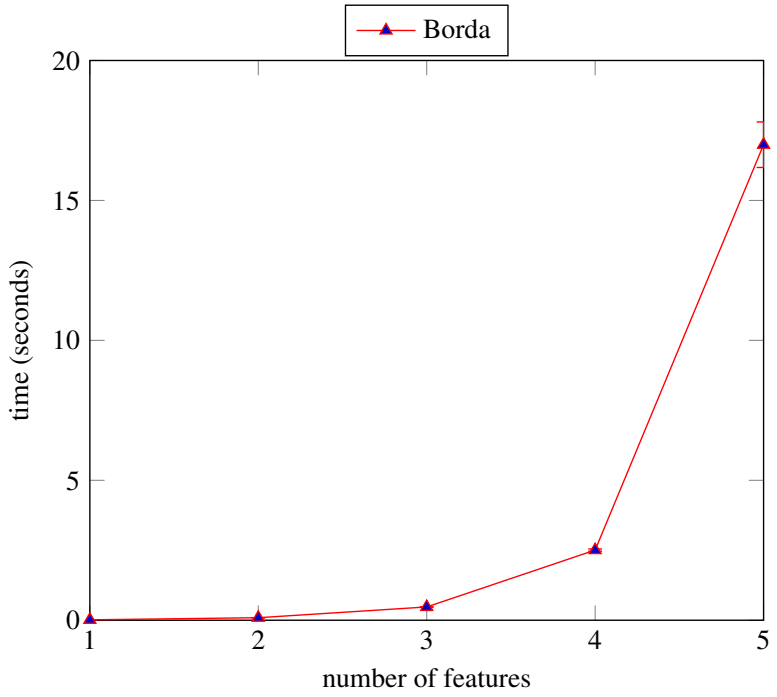


Figure 4.11: Computational time for Borda, varying the number of features.

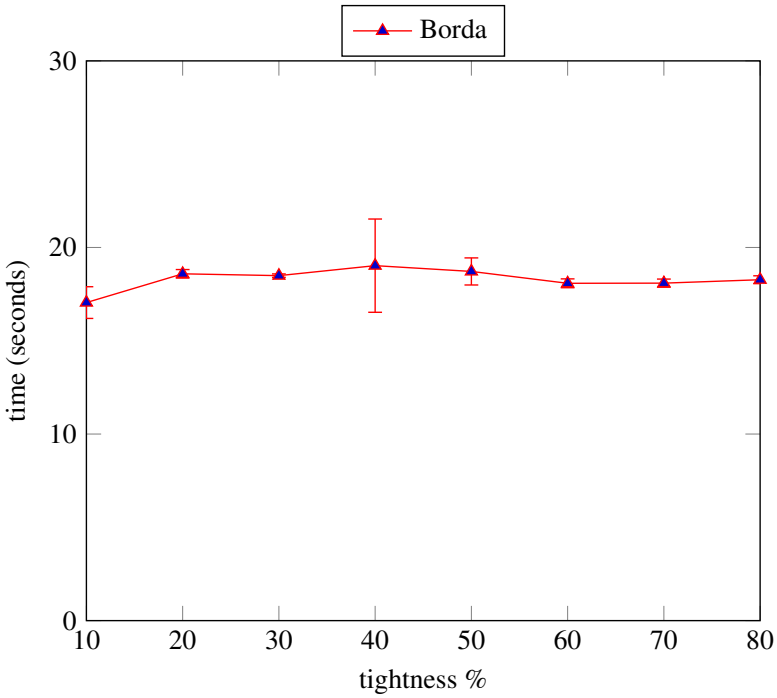


Figure 4.12: Computational time for Borda, varying the tightness.

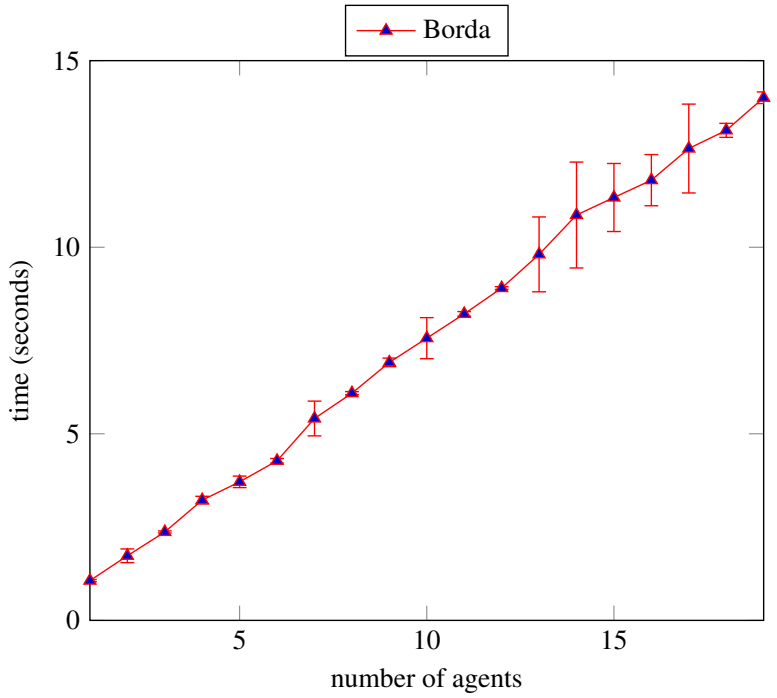


Figure 4.13: Computational time for Borda, varying the number of agents.

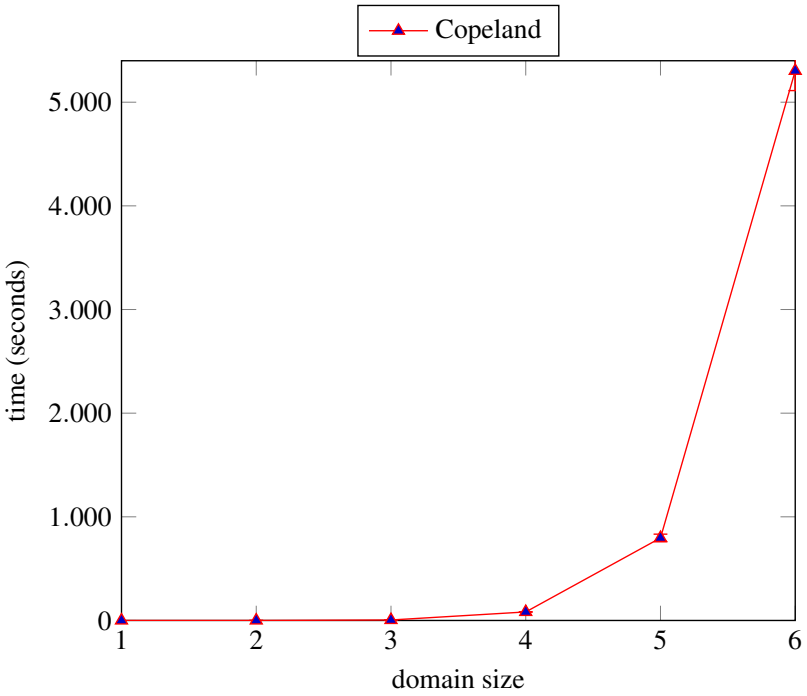


Figure 4.14: Computational time for Copeland, varying the domain size.

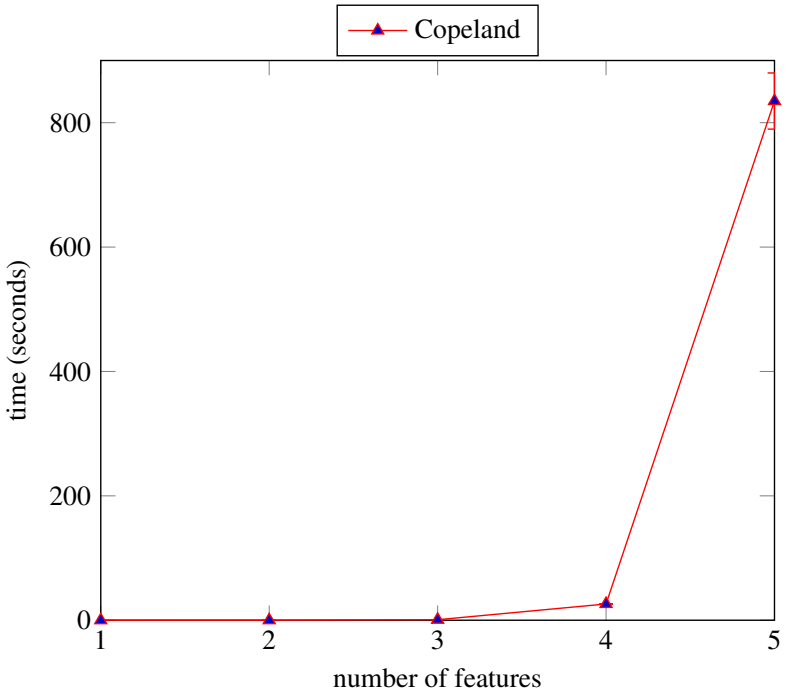


Figure 4.15: Computational time for Copeland, varying the number of features.

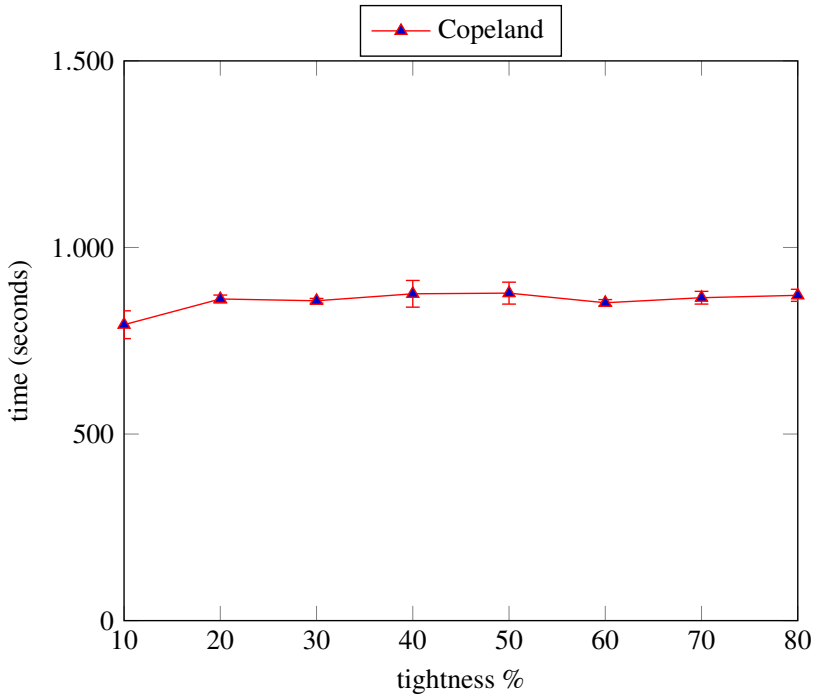


Figure 4.16: Computational time for Copeland, varying the tightness.

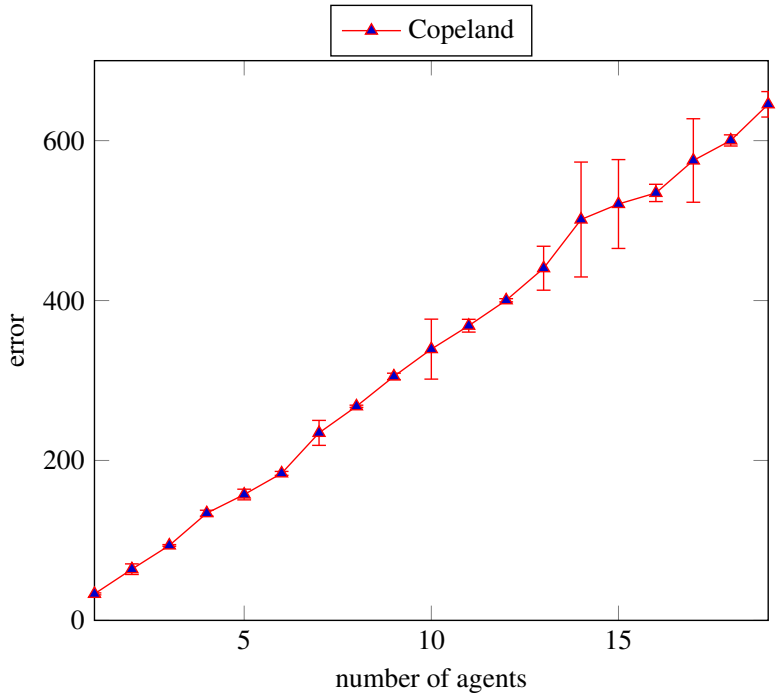


Figure 4.17: Computational time for Copeland, varying the number of agents.

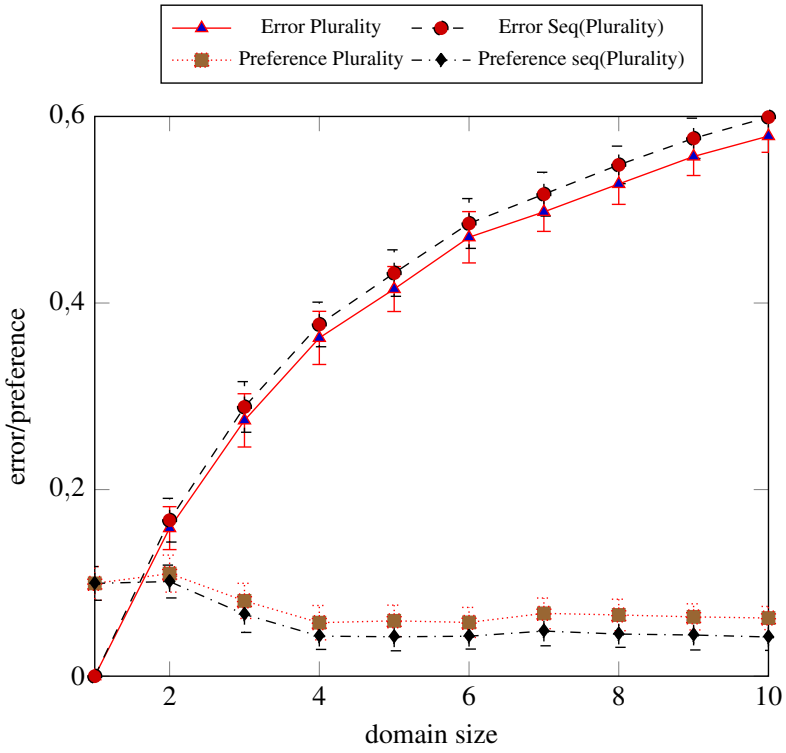


Figure 4.18: Error and Preference for Plurality and seq(Plurality), varying the domain size.

In Figure 4.18, Figure 4.19, Figure 4.20 and Figure 4.21 we show the preference and the error for Plurality and seq(Plurality). The results for Borda, seq(Borda), Approval, seq(Approval), Copeland and seq(Copeland) are very similar: the winner of the sequential version of each voting rule has error and preference slightly worse than the winner of the non-sequential voting rule, but exactly the same trend. Moreover, all the four voting rules have the same trend, as shown in Figure 4.18, Figure 4.19, Figure 4.20 and Figure 4.21. Quite surprisingly, for all rules, the error is small, meaning that the agents are on average almost equally satisfied with the winner elected by the two approaches.

When the number of values in the domains, or the number of variables, grows, the error grows as well. The reason for this is that an increase in the number of solutions makes it less likely for the winner to have a high preference in each FCSP.

The same trend can be observed when the number of voters grows. In this case, however, the higher error is due to an increased amount of disagreement among the larger number of voters. This is supported also by the fact that the preference decreases. Moreover, we observe that, when the number of voters increases, with more than 10 voters, the values of preference and error become stable.

When tightness grows, instead, the error decreases. This is to be expected, since, the less solutions with a non-zero preference each FCSP has, the more likely it is that the winner has a preference close to the optimal one for each voter.

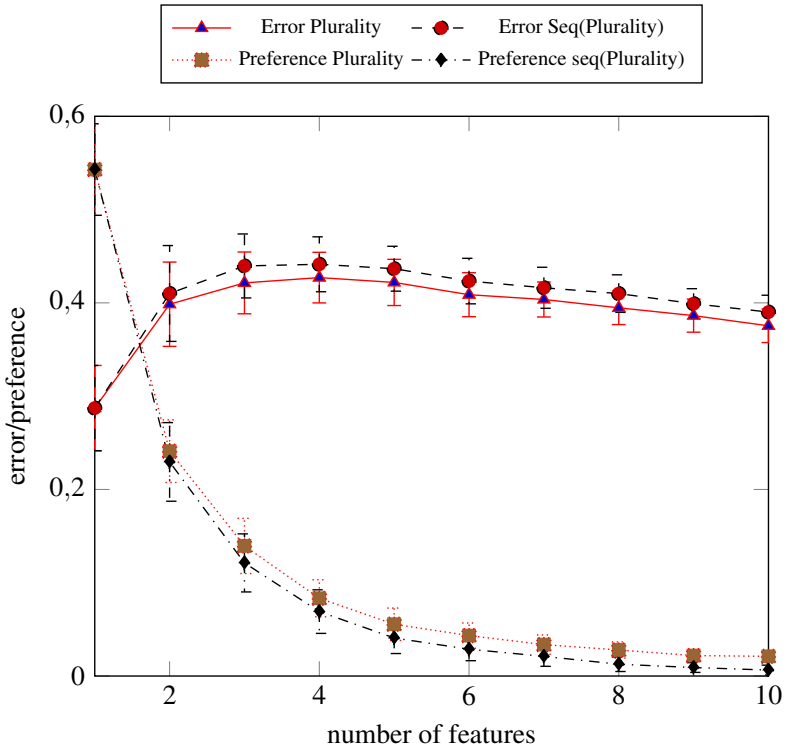


Figure 4.19: Error and Preference for Plurality and seq(Plurality), varying the number of features.

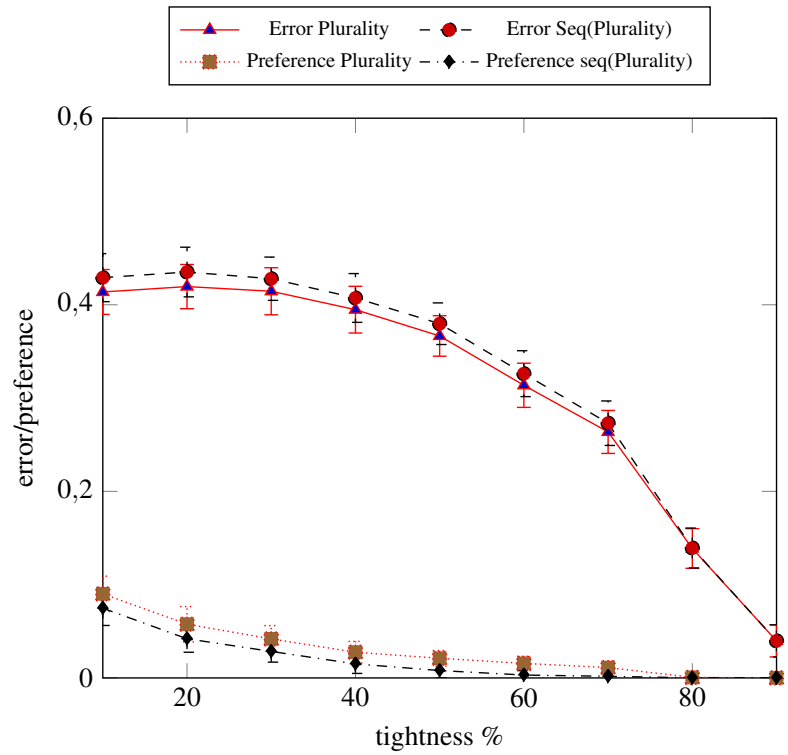


Figure 4.20: Error and Preference for Plurality and seq(Plurality), varying the tightness.

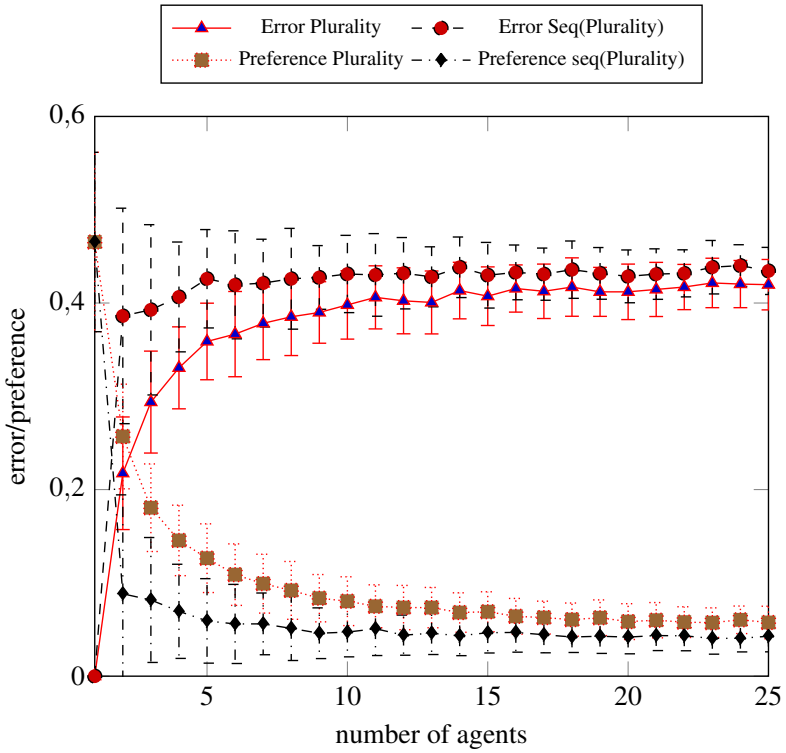


Figure 4.21: Error and Preference for Plurality and seq(Plurality), varying the number of agents.

Rules Comparison

To compare the four considered voting rules, we look at their preference, error, and computation time on the following parameters values (Table 4.4): 25 voters, 5 issues, 5 domain element per issue, and 20% tightness. We show this in Table 4.4.

Rule	*Preference*	*Error*	*Computation Time*
Seq. Plurality	0.046	0.43	0.012
Seq. Approval	0.072	0.40	0.013
Seq. Borda	0.053	0.42	0.013
Seq. Copeland	0.047	0.43	0.014
Plurality	0.058	0.41	0.004
Approval	0.068	0.40	0.005
Borda	0.082	0.39	17.139
Copeland	0.096	0.38	796.107

Table 4.4: Rules Comparison.

We can see that:

- Plurality, as expected, is the worst voting rule in terms of preference and error, but it is the best in terms of execution time, due to its naive scoring system.
- Approval is a good voting rule in terms of preference, error, and execution time. It has better preference and error results than Plurality, probably because for each vote the Approval's scoring system considers the best top candidates, contrarily to Plurality, which considers only

the first top candidate according to lexicographic ordering.

- Borda and Copeland are good voting rules in terms of preference and error, but they are very bad in terms of execution time, due to their complex scoring system.

Moreover, our results show that the sequential approach performs similarly to the non-sequential approach in terms of preference and error. This is good news, since it suggests that the loss of information in using the sequential approach does not push outliers further apart. However, the performance of seq(Borda) and seq(Copeland) are considerably better than the corresponding non-sequential voting rules.

Variable orderings and worst/best cases

The experimental results shown so far assume a fixed variable ordering, neglecting the fact that different orders may give different results for the sequential approach.

We prove experimentally that changing the variable order in the sequential voting rule does not alter the mean preference of the winning solution over the agents in the profile. We computes the difference of the mean (over the agent) preference of the sequential approaches on 100 different pairs of variable orderings (randomly generated) on a profile, averaged on 100 different profiles. We provide the results about the difference in the preference value, changing the variable order, in Table 4.5.

Rule	*Difference in preference value (%)*	*Variance(%)*
Seq. Plurality	1.6 %	1.3 %
Seq. Approval	1.4 %	0.8 %
Seq. Borda	1.5 %	1.1 %
Seq. Copeland	1.5 %	1.2 %

Table 4.5: Rules Comparison.

The sequential approach is basically not influenced by the change in the variable ordering since there is a difference in the preference value of about 1.5% with a variance of about 1%. Thus, a change in the variable ordering may imply a change in the elected candidate, but not in the average satisfaction of the agents.

TripAdvisor profiles

In our synthetic profiles, described in the previous sections, each agent's FCSP is generated randomly. Thus the probability that two voters vote equally is very small and this implies a large amount of disagreement among the agents. This is often not the case in real life situations, where profiles may show some form of correlation among voters' preferences. To consider also more realistic profiles , we have taken real data from profiles extracted from TripAdvisor.

More precisely, we used a dataset from PrefLib [76], a reference library of preference data about computational social choice, recommender systems, data mining, machine learning, and combinatorial optimization. The dataset used contains $675,069$ reviews of $1,851$ hotels across the world scraped from Trip Advisor. The reviews consists of numerical ratings (integers between 1 and 5) provided by the users about several characteristics of the hotel.

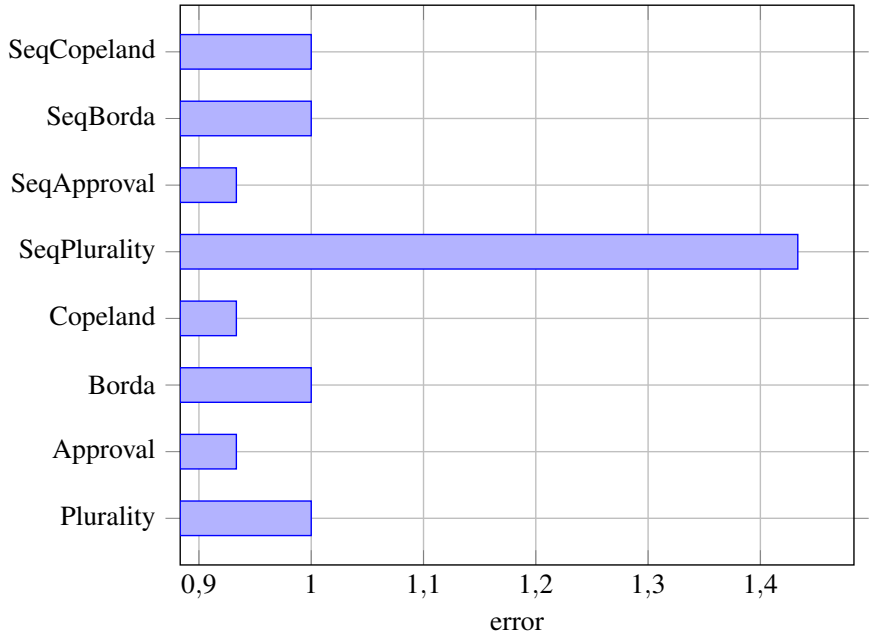


Figure 4.22: Error comparison on TripAdvisor data.

From the dataset we extracted the ratings of 30 agents over 5 characteristics of 6 hotels. The characteristics are: overall, rooms, location, cleanliness, and front desk. This are the variables in our setting. If a user has some missing value but the overall rating is specified, we completed the missing values with the overall rating. If a user has no rating for a hotel we completed the missing values with the mean rating between 1 and 5 (3). In the dataset considered we were not able to select a dense subset of 30 agents that rated a common subset of hotels. This is not surprising also for large dataset. In this way, we obtain profiles in which each agent has a preference structure with 5 independent nodes (thus, with no constraints between them), each one corresponding to a characteristic of the hotels. The domain of the features has 6 elements corresponding to the 6 hotels and the preferences are the rating extracted from the dataset.

Figure 4.22 and Figure 4.23 shows the preference and the error of the elected candidate for the sequential and non-sequential approaches. The results about the computation time of the eight voting rules are shown in Table 4.6.

Plurality	Approval	Borda	Copeland	SeqPlurality	SeqApproval	SeqBorda	SeqCopeland
0.018	0.26	21.5	502.63	0.00017	0.00069	0.0009	0.0022

Table 4.6: Computation time (seconds) comparison on TripAdvisor data.

The results presented in Figure 4.22, Figure 4.23 and Table 4.6 are in line with the results of the experiments on synthetic data of Section 4.2.3. Thus our experimental setting, with a random generation of profiles and preferences, corresponds to the results of a real-life dataset. We can observe that the sequential voting rules are the best in terms of execution time. Copeland is a good voting rule in terms of preference and error, but it is very bad in terms of execution time, due to

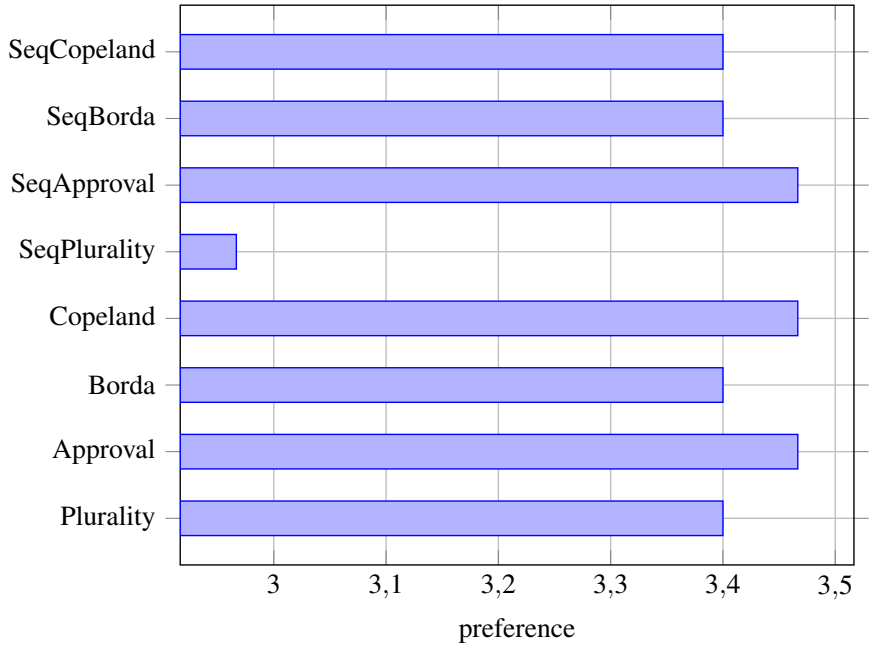


Figure 4.23: Preference comparison on TripAdvisor data.

their complex scoring system. Approval instead is a good voting rule in terms of preference, error, and also execution time.

4.3 Summary and Discussion

In this chapter we analyzed the use of PCP-nets as a compact representation language for the preferences of a set of agents. We showed that, given an empirical probability distribution over a set of CP-nets, there may exist no PCP-net inducing it, even if they have the same dependency graph. Thus we defined and compared two polynomial aggregation methods. The first one (*proportion method*, PR) extracts the probability distribution directly from the CP-nets and the second one (*least square method*, LS) minimizes the error between the aggregated structure and the initial set of CP-nets. We compared them in a theoretical and experimental evaluation, and the results suggest that using the PR method in the input profile to construct a PCP-net is accurate with respect to answering both optimality and dominance queries.

We then combined these two methods with the two ways of extracting an optimal outcome from a PCP-net, obtaining four approaches: four voting rules that take as input a profile of individual CP-nets and output a winning candidate. We then compared these voting rules with plurality and sequential voting with majority, since our experimental setting is based on binary features. We showed theoretically that the output of one of the four methods we defines coincides with the output of the sequential voting method with majority, thus our proposed aggregation method is a direct generalisation of this setting. We then showed experimentally that our voting rules performs better than the plurality voting method.

By generating a compact representation of the full preference profile using PCP-nets, we are also able to perform either exact or approximate dominance reasoning on a profile of individual

CP-nets. Moreover, for the case of polytree PCP-nets, we showed that our proposed approximation technique for dominance, applied with PR, yields results that are very close to the probability of dominance in the initial profile of individual CP-nets.

We then considered an alternative way to represents preferences in a multi-agent contexts: profiles of users described by fuzzy profiles of soft constraints. Soft constraints profile are a framework that is less restrictive with respect to CP-nets and allows a more general formalization: with soft constraints we are not forced to use a particular topological order when we consider sequential procedures since soft constraints have an undirected preference graph. Moreover, the dominance task is easy in soft constraints profiles contra CP-nets profiles.

We studied a sequential preference aggregation procedure based on voting rules for settings where several agents express their preferences over a common set of variable assignments via soft constraints. We analyzed this approach from a theoretical point of view providing a general result which shows that if the sequential voting procedure satisfies a given property, so do all the local voting rules. But the opposite holds only for anonymity, consistency, efficiency, (strong) monotonicity, and non-dictatorship.

We then studied this approach by providing experimental results, evaluating the performance of our approach considering several widely used voting rules such as Plurality, Approval, Borda and Copeland. We proved that both from a theoretical point of view and an experimental point of view, the sequential approach is convenient in terms of computation time, while satisfying the agents preferences just as much as the non-sequential approach. We performed another set of experiments to prove that the quality of the returned solution is not affected by the ordering of the features. We provided also results on real preferences obtained from the PrefLib database. The results of the real data application are in line with the results of the experiments on synthetic data.

We compared our method to the one performed in [65] for CP-nets, in which they shows that a sequential single-feature voting protocol can find a winner in polynomial time, and has several other desirable properties, but only if the CP-nets satisfy certain conditions: the CP-nets must have an acyclic dependency graph and all the dependency graphs in the profile must all be compatible with a given graph ordered according to the feature ordering in the voting procedure. Also other more specific approaches to sequential composition of voting rules with CP-nets that have been proposed: in [114] acyclicity and O-legality are relaxed via a procedure that exploits a graph defined on complete assignments, in [115] they based the procedure on maximum likelihood estimators, on maximum criterion in [87], or considering variants of CP-nets in [26, 64].

5. A logical model for conditional preferences

Until now we have focused mainly on CP-nets, but this framework is limited in expressiveness since cp-rules may specify a preference for exactly one value over another. A lot of extension and variants of CP-net have been developed such as *GCP-nets* [52], *CP-theories* [111], *Comparative preference theories* [112] and *CP-nets with constraints* [12] [42], each one with ad hoc syntax, semantics and algorithms. We develop in this chapter a unification model for all these different frameworks for conditional preferences following the idea that conditional preferences can be directly expressed in standard first order logic, as constrained Datalog theories [19, 59, 105].

The fundamental advantage of introducing conditional preference theories as Datalog programs is that Datalog's rich semantic, algorithmic and implementation framework is now available in service of conditional preferences. The semantics of our model is that of (constrained) first-order logic theories. The framework is rich enough to express CP-nets and each of its extensions discussed above, including algorithms for the main tasks: consistency, dominance and optimality. Using outcomes rather than assertions of preference over individual features permits the formalization of the semantics (e.g. *ceteris paribus* or indifference) internally, as just a certain pattern of quantification over variables.

Constraints fit in naturally in our theory and do not have to be introduced after the fact in an *ad hoc* fashion as in [12, 42, 84]. In our formulation the constraints are already built into basic definitions (constraints are additional goals in the body of preference clauses, and in the body of the clause defining outcomes) and no changes are necessary.

Additionally, recursive LCP-rules offer a powerful new form of *dependent* conditional preference statements, particularly useful in multi-agent contexts [72, 90]. They support rules such as "If Alice prefers to drive to Oxford today, Bob will prefer to fly to Manchester tomorrow".

The rich complexity theory developed for Datalog [30, 47, 55, 109] applies *inter alia* to

conditional preference theories – in particular we discuss the notion of *data-complexity*. General results about (linear) Datalog programs lead to complexity bounds for consistency, dominance and optimization extending current known bounds. Further, tabled Prolog systems such as XSB Prolog[101, 102] implement constrained Datalog with sophisticated features such as partial order answer subsumption that are directly usable in an implementation of LCP.

One of the reasons that CP-nets are popular in practice is that useful special cases have been identified (acyclic nets, tree-structured nets) which can be implemented efficiently. We show how some of these special cases can be extended to the richer language we consider. Further, we provide a compiler for LCP theories that can recognize these special cases and generate custom code for consistency, dominance and optimization.

We study also different semantics for the dominance task that are computationally easier and we compare them with the classical notion of dominance.

Our formalization of extensions of CP-nets permits an integrated treatment of preferences in constraint (logic) programming, leading to more powerful reasoning systems which can deal with both preferences and hard constraints.

Chapter structure and related publications

The work presented in this chapter has appeared in the proceedings of the following conferences and international workshops.

- ⋆ **"Logical conditional preference theories"**, *C. Cornelio, A. Loreggia, and V. Saraswat*, Proceedings of the MPREF workshop of the International Joint Conference on Artificial Intelligence 2015, MPREF-15.
- ⋆ **"Models for Conditional Preferences as extensions of CP-nets"**, *C. Cornelio*, Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (extended abstract), Doctoral Consortium, IJCAI-15.

The chapter is organized as follows.

- In Section 5.1 we introduce some background notions regarding logic programming and in particular Datalog. We also define several techniques used in this context such as tabling and answer subsumption.
- In Section 5.2 we introduce Flat LCP-Theories, a logical formulation for conditional preferences, defined as a linear Datalog theory. We show that the frameworks for conditional preferences studied in the background chapter (Section 2.2 and Section 2.3) can be expressed using this theory. We provide algorithms for the main tasks: consistency checking, outcome optimization and dominance testing, with also a computational complexity analysis.
- In Section 5.3 we introduce a generalization of Flat LCP-Theories using Datalog recursive clauses. We define Recursive LCP-Theories and we provide some application examples. We then introduce procedures for the main tasks: consistency checking, outcome optimization and dominance testing, with also a computational complexity analysis.
- In Section 5.4 we study the notion of completeness for LCP-theories and we perform an experimental evaluation of the outcome optimization task.
- In Section 5.5 we present three alternative definitions for the dominance task. We provide a

theoretical comparison of these three new semantics between them and in comparison to the classical dominance semantics.

- In Section 5.6 we provide a summary of the chapter and a discussion of the results.

5.1 Datalog and logic programming

A Datalog program consists of a collection of definite clauses in a language with no function symbols, hence a finite Herbrand domain. Datalog programs can be implemented using *tabled Logic Programming (TLP)*. Tabling maintains a memo table of subgoals produced in a query evaluation and their answers. If a subgoal is reached again then the information in the table can be reused, without recomputing the subgoal. This method ensures termination and improves the computational complexity for a large class of problems [102] (at the expense of additional space consumption). Answer subsumption extends the functionality of tabling. *Answer variance* adds a new answer to a table only if the new answer is not a variant of any other answer already in the table. *Partial order answer subsumption* adds a new answer to a table only if the new answer is maximal with respect to the answers in the table, given a partial order `po/2`:

```
:- table predicate(_,_,partialOrder(po/2)).
```

Traditionally, predicates are divided into *extensional* and *intensional* predicates. The extensional predicates define a database, and intensional predicates define (possibly recursively defined) queries over the database. In our context, `d/2` and `outcome/1` will be considered extensional predicates (in Flat LCP) and other predicates such as `dom/2, inconsistent` and user-defined predicates are considered intensional. Given an intensional program P, database D and query q, *data-complexity* [109] is the complexity of answering $P, D \vdash q$ as a function of the size of D and q (thus the program is considered fixed). *Combined complexity* is the complexity of answering $P, D \vdash q$ as a function of the size of P, D and q (thus nothing is taken to be fixed). The basic results are that for data-complexity, general Datalog programs are PTIME-complete, and linear programs are NLOGSPACE-complete [55]. For combined complexity, general Datalog programs are EXPTIME-complete, and linear programs are PSPACE-complete.

5.2 Flat LCP-Theories (FLCP-Theories)

We assume given a set of N features, and a logical vocabulary $\mathscr{V}$ with unary predicates `d1`, ..., `dN` (corresponding to the domains of the features), constants for every value in the domains `di`, a single function symbol `o/N` that corresponds to the description of an outcome, and a single binary predicate `d/2` that indicates that the first argument is preferred to the second one. The user specifies preferences between two outcomes `S,T` (expressed as `o/N` terms) by supplying clauses for the atom `d(S,T)`:

```
d(S,T) :- c.
```

where `c` is a constraint (possibly involving equality). This clause is a *Flat Logical Conditional Preference (FLCP) rule*. The user can specify hard constraints on features by providing clauses for the `outcome/1` predicate, of the form:

```
outcome(o(X1,...,Xn)) :- C,
                         d1(X1), ..., dn(Xn).
```

where `C` is a constraint and the `di` are domain predicates.

The LCP runtime supplies the following definition for the `dom/2` predicate, expressing (tabled) transitive closure over `d/2`, and for consistency and optimal outcomes:

```
:- table(dom(_,_)).
dom(X,Y):- d(X,Y), outcome(X), outcome(Y).
dom(X,Y):- d(X,Z), dom(Z,Y).
consistent :- \+ dom(X,X).
:- table(optimal(po(dom/2))).
optimal(X):- outcome(X).
```

Observation 5.1 Note that the clauses above are linear. Below, given an LCP theory (Datalog program) P, by $\mathscr{L}(P)$ we will mean P together with the LCP runtime clauses specified above. ■

Given these definitions, the problem solver may use `consistent` to determine whether the supplied preference clauses are consistent, `dom(S,T)` to determine whether outcome `S` is preferred to `T`, and `optimal(S)` (where `S` may be a partially instantiated `o/N` structure) to determine an optimal completion of `S`.

■ **Example 5.1 — Dinner, modified from (11).** Two components of a meal are the soup (`fish` or `veg`) and wine (`white` or `red`). I prefer `fish` to `veg` all else being equal. If I am having `fish`, I prefer `white` wine to `red`. I simply do not want to consider `veg` with `red`. This may be formulated as the LCP theory:

```
soup(fish). soup(veg). wine(white). wine(red).
outcome(o(X,Y)):- soup(X), wine(Y), (X\== veg; Y\==red).
d(o(fish, X), o(veg, X)).
d(o(fish, white),o(fish, red)).
```

On this theory, the query `?-consistent.` returns `yes`. The `dom/2` predicates order outcomes as:

```
o(fish,white) > o(fish,red)    o(fish,white) > o(veg,white)
```

Note because of hard constraints the outcomes are not totally ordered. The query `optimal(X)` returns the instantiation of `O` that is highest in the `dom/2` order, in this case it returns the single answer `X=o(fish,white)`. ■

■ **Example 5.2 — Holiday Planning, (111).** There are three features: `time`, with values later `l` and now `n`; `place`, with values Manchester `m` and Oxford `o`, and `mode` with values fly `f` and drive `d`. The rule "I would prefer to fly rather than drive, unless I go later in the year to Manchester." translates to clauses 1-2. The CP-theory rule "I would prefer to go next week, regardless of other choices." corresponds translated to clause 3, and the comparative preference rule "All other things being equal, I would prefer to fly now, rather than to drive later." to clause 4:

```
time(n). time(l). place(o).
place(m). mode(f). mode(d).
outcome(o(T,P,M)):- time(T), place(P), mode(M).
(1) d(o(T,P,f),o(T,P,d)):- T=n;P=o.
(2) d(o(l,m,d),o(l,m,f)).
(3) d(o(n,_,_),o(l,_,_)).
(4) d(o(n,X,f),o(l,X,d)).
```

■

Proposition 5.2.1 — Normal form for FLCP rules. Let R be a FLCP-rule. For appropriate choices of disjoint index sets J, K, M and Z s.t. $\{1,\ldots,n\} = J \cup K \cup M \cup Z$, and given L and U disjoint subsets of M and given constants $v_j(j \in J)$, $a_m, a'_m(m \in M, a_m \neq a'_m)$, $a_l(l \in L)$ and $a_u(u \in U)$, R is logically equivalent to the clause `d(o(`$S_1,\ldots,S_n$`), o(`$T_1,\ldots,T_n$`))`. where S_i, T_i are defined by: $S_j = T_j = v_j(j \in J)$, $S_k = T_k, k \in K$, $S_m = a_m, T_m = a'_m(m \in M)$, $S_z = X_z, T_z = Y_z(z \in Z)$, $S_l = X_l, T_l = a_l(l \in L)$, $S_u = a_u, T_u = Y_u(u \in U)$. The set J corresponds to the parent variables, K to the ceteris paribus variables, M to the variables that change the value from S to T and Z to the variables that are less important than the variables in M. We call L the *lower-bound* set, and U the *upper-bound* set.

A GCP-net rule is simply a FLCP-rule such that $Z = \emptyset$, $|M| = 1$, $L \cup U = \emptyset$. A CP-theory rule is a FLCP-rule with $|M| = 1$, $L \cup U = \emptyset$. A comparative preference rule is a FLCP-rule with $L \cup U = \emptyset$.

The following theorems establish that LCP-theories conservatively extend these sub-languages. Proofs are straightforward, we have essentially just used standard logical notions to formalize the sub-languages:

Theorem 5.2.2 — Logical characterization of *ceteris paribus and general ceteris paribus*. Given a CP-net $\mathscr{R}$, consider the set P of FLCP-rules which represent all rows of its CP-tables. Then, for any two outcomes s and t, $s \succ t$ in $\mathscr{R}$ iff $\mathscr{L}(P) \vdash$ `dom(s,t)`.

Theorem 5.2.3 — Logical characterization of *CP-theories* and *comparative preference languages*. Given a CP-theory $\mathscr{R}$, consider the set P of FLCP rules modeling all the rules of the CP-theory. Given two outcomes s and t, $\mathscr{R} \vdash s \succ t$ iff $\mathscr{L}(P) \vdash$ `dom(s,t)`.

We now show that the conditional preference rules discussed in the literature can be expressed as special cases of FLCP rules. First we prove that CP-nets, GCP-nets, and CP-theories are particular cases of comparative preference theories, then we reformulate a Comparative preference rule using a FLCP rule.

Suppose we have a CP-net or GCP-net rule $u : x > \bar{x}$ where x and $\bar{x}$ are distinct literals for some variable $X \in$ `Var` where `Var` is the set of variables, u is the assignment of a set $U \in$ `Var` $\setminus \{X\}$. This rule can be seen as a comparative preference rule `p>q||T` where $P = Q = \{X\} \cup U$ and $p = ux$, $q = u\bar{x}$ and $T =$ `Var` $\setminus (U \cup \{X\})$. Given a CP-theory rule $u : x > \bar{x}[W]$ where x and $\bar{x}$ are elements in the domain of a variable $X \in$ `Var`, u is the assignment of a set $U \in$ `Var` $\setminus \{X\}$ and

W is a set of features from `Var` s.t. the sets $U, \{X\}, W$ are mutually disjoint. This rule can be seen as a comparative preference rule `p>q||T` such that $P = Q = \{X\} \cup U$ and $p = ux$, $q = u\bar{x}$ and $T = \texttt{Var} \setminus (U \cup \{X\} \cup W)$.

Comparative preference theories. Let $R \equiv p > q||T$ be a rule in a comparative preference theory. Recall that this means that p is an assignment to a set of features $P \subseteq \texttt{Var}$, and q is an assignment to a set of features $Q \subseteq \texttt{Var}$, and T is a set of features from `Var`. Let $[p]$ represent the constraint obtained from p by conjoining $X_i = w_i$ for every feature $i \in P$ (where p assigns the value w_i to feature i). Similarly $[q]$ for the set Q. Let t stand for the conjunction of constraints $X_i = Y_i$ for every feature i in T. Then R can be represented by the LCP rule: `d(o(`$X_1,\ldots,X_n$`), o(`$Y_1,\ldots,Y_n$`)) :-` $[p],[q],t$. The following two translations can be derived from the translation above because its special cases.

CP-net and GCP-net. Let $R \equiv u : l > l'$ be a conditional preference rule where l, l' are distinct literals for some variable X_i. Let $[u]$ represent the constraint obtained from u by conjoining $X_j = Y_j, X_j = w_j$ for every feature $j \in U$ (where u assigns the value w_j to feature j). This rule can be represented by the LCP rule: `d(o(`$X_1,\ldots,X_n$`), o(`$Y_1,\ldots,Y_n$`)) :-` $X_i = l, Y_i = l', [u]$.

CP-theories. Similarly, let $R \equiv u : l \succ l'[W]$ be a rule in a conditional preference theory . Recall that this means that u is an assignment to a set of features $U \subseteq \texttt{Var}$, l, l' are values for a feature i, and W is a set of features from `Var` s.t. the sets $U, \{i\}, W$ are mutually disjoint. As before let $[u]$ stand for the constraint obtained from u by conjoining $X_j = Y_j, X_j = w_j$ for every feature $j \in U$ (where u assigns the value w_j to feature j). Let c stand for the conjunction of constraints $X_i = Y_i$ for every feature i not in $U \cup \{i\} \cup W$. Then R can be represented by the LCP rule: `d(o(`$X_1,\ldots,X_n$`), o(`$Y_1,\ldots,Y_n$`)) :-` $[u], X_i = l, Y_i = l', c$.

Thus we have the following theorem and corollaries, that establish that LCP-theories conservatively extend these sub-languages. Proofs are straightforward, we have essentially just used standard logical notions to formalize the sub-languages:

Theorem 5.2.4 — Logical characterization of comparative preference theories. Given a comparative preference theory Γ, consider the set P of FLCP rules representing the rules of Γ as described above. Given two outcomes s and t, $\Gamma \vdash s \succ t$ iff $\mathscr{L}(P) \vdash$ `dom(s,t)`.

Corollary 5.2.5 — Logical characterization for GCP-nets. Given a GCP-net $\mathscr{R}$, consider the set P of FLCP rules representing the rules of $\mathscr{R}$ as described above. Given two outcomes s and t, $\mathscr{R} \vdash s \succ t$ iff $\mathscr{L}(P) \vdash$ `dom(s,t)`.

Corollary 5.2.6 — Logical characterization of CP-theories. Given a CP-theory Γ, consider the set P of FLCP rules representing the rules of Γ as described above. Given two outcomes s and t, $\Gamma \vdash s \succ t$ iff $\mathscr{L}(P) \vdash$ `dom(s,t)`.

Summarizing FLCP rules can translate all the language described in background section. LCP-rules are also a generalization since they can handle hard constraints.

	General structure	Acyclic	Tree
Dominance			
CP-nets:	PSPACE-complete [52]	NP or harder [11]	Polynomial [9, 11]
CP-theories:	PSPACE-complete [111]	PSPACE-complete [111]	**Polynomial**
FLCP-theories:	**PSPACE-complete**	**PSPACE-complete**	**Polynomial***
Consistency / Optimality			
CP-nets:	PSPACE-complete [52]	Polynomial [11]	Polynomial [11]
CP-theories:	PSPACE-complete [111]	Polynomial [111]	Polynomial [111]
FLCP-theories:	**PSPACE-complete / ?**	**? / Polynomial***	**? / Polynomial ***

Table 5.1: Computational complexity of Dominance, Consistency and Optimality. *Bold: our results, *: with some constraint on the form of the rule, ?: open problem, —: dependency graph not defined.*

5.2.1 Algorithmic properties

The main algorithmic tasks regarding a preference theory are *dominance queries*, *consistency checking*, and *outcome optimization*. Below we fix a set of features `Var` with cardinality N and an LCP-theory P over `Var`.

Checking the **dominance** over a pair of outcomes corresponds to finding a swapping sequence in CP-theories or a flipping sequence in CP-nets. For LCP-theories this is determined by first-order derivability: the dominance query `(s,t)` succeeds iff $\mathscr{L}(P) \vdash$ `dom(s,t)`. **Consistency checking** verifies whether there is any outcome `s` such that $\mathscr{L}(P) \vdash$ `dom(s,s)`. **Outcome optimization** corresponds to find the most preferred outcome given an assignment to a (possibly empty) subset of features. Formally, an *optimal outcome* is an outcome `s` such that there is no other outcome `t` such that $\mathscr{L}(P) \vdash$ `dom(s,t)`. (i.e.`s` is an undominated outcome). Given a (possibly non-ground) term `s` (partial outcome) an *optimal completion* of `s` is a ground term `t` instantiating `s` s.t. for no other ground term `t1` instantiating `s` is it the case that $\mathscr{L}(P) \vdash$ `dom(t1,t)`. Note that an acyclic CP-net or CP-theory (consistent and complete) has a unique optimal outcome, but an LCP-theory may have several optimal outcomes.

Our algorithmic approach is based on analyzing whether the input LCP-theory corresponds to a special case (e.g. acyclic or tree-structured nets). If so, optimal algorithms for the special case are used. Otherwise general Datalog procedures are used. We define the dependency graph for FLCP-theories as the generalized dependency graph G described in [111].

In the following sections we analyze the algorithms for dominance, consistency and optimality from a general point of view, and then consider the special cases in which the algorithms are faster. We take the viewpoint of *combined complexity*, i.e. assuming N, the clauses of the LPC theory, and the given query are supplied at runtime. The results for Flat LCP-theories are summarized in Table 5.1.

Dominance

Theorem 5.2.7 Given a FLCP theory P over N features, deciding `dom(s,t)` is PSPACE-complete in N.

Proof. Since FLCP theories can encode GCP-nets, the dominance problem is at least PSPACE-hard. That the problem is in PSPACE can be established in a form similar to the proof of Theorem 4.4 in [55]. Since P has but a single rule, and the rule is linear, we can build up the proof non-deterministically using a polynomial amount of space. In fact, we need to keep space only for two ground facts of the form `dom(s, t)` where the `s,t` are constants. We start by using the base clause for `dom/2` to non-deterministically establish a `dom/2` fact, using some fact for `d/2`, and scratch space linear in N. Then we use the recursive clause for `dom/2` to non-deterministically generate a new `dom/2` fact. From this new fact, we can generate another, and delete the old fact. We stop when `dom(s,t)` is established. ■

Dominance for acyclic and tree dependency graphs. We note that the computational complexity for dominance for acyclic FLCP-theories is PSPACE-complete. This because dominance for an acyclic FLCP-theory is easier than a general FLCP-theory but harder than an acyclic CP-theory. Both of them are PSPACE-complete thus also dominance for acyclic FLCP-theories is PSPACE-complete.

A dominance query in tree structured CP-nets can be computed in time linear in N [9, 11]. We observe that a procedure similar to [9] can be used for tree structured CP-theories. We generalized this procedure [9] also for tree structured FLCP theories. We follow the same algorithmic structure with some modifications. (1) We partition the set of nodes in subsets: each subset contains nodes that change their value simultaneously in a LCP-rule and the nodes that belongs to the direct paths between them, if it exists. Suppose we have 4 nodes X_i and two edges (X_1,X_2), (X_2,X_3). The nodes X_1 and X_4 change their values simultaneously in a LCP-rule and X_3 and X_4 change their values simultaneously in another rule. Thus all the nodes are in the same subset. (2) We start the procedure from the leafs to the root, and when we process a node we have to process simultaneously the whole subset of nodes to which it belongs in the partition. (3) The *change* function is replaced by the function $C(S_X)[v_1,\cdots,v_m]$ where X is a node, S_X is the subset of nodes to which X belongs and v_i are assignments or partial assignments of the nodes in S_X. Given a dominance query over two outcomes o and o', this function returns $True$ if and only if there exists a worsening path from $o \restriction_X$ to $o' \restriction_X$ equal to $o \restriction_X = o_1 \restriction_X, \cdots, o_l \restriction_X = o' \restriction_X$ such that $\forall i \exists j | v_i = o_j \restriction_X$. Each value v could appear several time and in that case must exists a path such that there exist $j_1,\cdots,j_l$such that $v = o_{j_1} \restriction_X$ and $v = o_{j_l} \restriction_X$ and $j_i \neq j_k \forall i,k$. The function $C(X)[v_1,\cdots,v_m]$ follow the following rule with respect to the conjunction operator $\wedge$: $C(X)[v_1,\cdots,v_m] \wedge C(X)[v'_1,\cdots,v'_{m'}] = C(X)[$ where $[v''_1,\cdots v''_{m''}] = [v_1,\cdots,v_m,v'_1,\cdots,v'_{m'}]$ and when a $v'' = v'_i = v_j$ the multiplicity of v'' is equal to the maximum between the multiplicities of v'_i and of v_j . It is important to notice that the function C in the case of subset of cardinality more then 1, corresponds to a dominance query on partial outcomes.

Observation 5.2 It is not important the order $v_1 \ldots v_m$, every permutation of these values is ok, since we are conditioning on the parent nodes, but we are considering only trees so the parent is only one. ■

The computational complexity of this procedure depends on the maximum size of the subsets of nodes, since on the subsets we use the general procedure for dominance, that is exponential in the cardinality of the subsets. If we consider fixed the maximum cardinality of the subsets, that is a reasonable assumption (in the case of CP-net or CP-theory the cardinality is fixed to 1), then the procedure become polynomial. The correctness proof of the procedure follow directly from the proof of Theorem 2 in [9].

Consistency

Consistency is determined by invoking the query `?-consistent.` This takes advantage of the tabling of the `dom/2` predicate.

The following theorem affirms that consistency remains in PSPACE even when the language for preferences is extended beyond CP-nets to FLCP rules, and it is a generalization of Theorem 3 in [52].

Theorem 5.2.8 Given a FLCP theory P over N features, deciding consistency is PSPACE-complete in N.

The proof follows directly from the proof of Theorem 5.2.7 since consistency is reduced to checking entailment.

Outcome optimization

For optimality, the user invokes the query `?- optimal(s).` Note that `optimal/1` uses partial order answer subsumption . In theory this may result in an exponential number of calls to `dom/2` atoms; each check takes exponential time. For now we leave as open the corresponding problem for FLCP theories (note that this problem is PSPACE-complete for CP-nets).

Optimization for acyclic and tree dependency graphs.

In acyclic CP-nets the sweep-forward procedure [11] finds the unique optimal outcome (or completion) in polynomial time. A similar result holds for [111]. Under the assumptions of consistency and acyclicity, an optimal outcome (and completion) can also be found for FLCP theories in polynomial time, using the following algorithm (*Acyclic-LCP-Opt*) generalizing sweep-forward : (1) We compute a linearization of the topological order defined by G(the dependency graph) over `Var`: $\mathscr{O} = \{X_1, \ldots, X_N\}$. (2) For each variable X_i, chosen following $\mathscr{O}$, we consider a set $W_i \subseteq$ `Var` such that it contains all the variables that change the value jointly with X_i in at least one rule (W_i could contain only X_i). Using the rules in the LCP-theory that involve the variables in W_i, and given the assignments for the variables $X_1, \ldots, X_{i-1}$, we generate an ordering over the partial outcomes defined on the variables in W_i. We assign to X_i the X_i value of a top element (following a certain tie breaking rule) of the ordering that satisfies `outcome/1` for at least one completion (the completion has the given assignment to $X_1, \ldots, X_{i-1}$ and an arbitrary assignment to $\{X_{i+1}, \ldots, X_N\} \setminus W_i$). (3) We repeat the previous step for all the features in `Var` (following $\mathscr{O}$).

Theorem 5.2.9 The outcome obtained with the *Acyclic-LCP-Opt* procedure is an optimal outcome for acyclic LCP-theories.

Proof. We prove it by contradiction, supposing that exists an outcome o' such that $o' \succ o$. We suppose by contradiction that exists an outcome o' such that $o' \succ o$. This implies that exists a chain of outcomes $o' = o_1, \cdots, o_m = o$ such that $\forall i \in \{1, \cdots, m-1\}$ exists a rule R that implies $o_i \succ o_{i+1}$. Considering a linearization $X_1, \cdots, X_N$ of the graph G associated to our set of rules $\mathscr{C}$, we take the first variable X in this order such that $o' \restriction_X \neq o \restriction_X$. Thus there exist a rule R and an index $i \in \{1, \cdots, m-1\}$ such that $o' \restriction_X = o_1 \restriction_X = \cdots = o_i \restriction_X \neq o_{i+1} \restriction_X = \cdots = o_m \restriction_X = o \restriction_X$ and $o_i \succ o_{i+1}$. This implies that $o \restriction_X$ is not the maximal element in the set of rules that involve X, that is a contradiction. ■

LCP-theories may have multiple optimal outcomes: we can obtain the whole set of optimal outcomes using the *Acyclic-LCP-Opt* procedure. If there is more than one optimal outcome then there exists some variable X that in the second step of the algorithm has more than one top element. Running in parallel all these possible assignments we obtain the whole set of optimal outcomes.

The complexity of the procedure is $O(d^w * N)$ where $w = \max_i |W_i|$: the second step of the procedure could involve all the partial outcomes defined on W_i. If the program bounds w, the procedure becomes linear in N. Note that if the LCP-theory corresponds to a CP-net or a CP-theory then $|W_i| = 1 \ \forall i$ and the algorithm coincides with the sweep-forward procedure for CP-nets, and the procedure introduced in [111] for CP-theories.

■ **Example 5.3 — Holiday planning.** Following the usual holiday planning example, we consider the following three rules: "All else being equal, I would prefer to fly now then drive later", " If I have to go now, I prefer to fly, all else being equal", " If I have to go later, I prefer to drive, all else being equal" and "If I have to go now I prefer to go to Oxford". These rules correspond to:

```
d(o(n,X,f),o(l,X,d)).    d(o(n,X,f),o(n,X,d)).
d(o(l,X,d),o(l,X,f)).    d(o(n,o,X),o(n,m,X)).
```

The graph G has two edges: from *time* to *mode* and from *time* to *place*, so, a possible linearization of G is: *(time, place, mode)*. Following this order we compute sequentially the optimal outcome as: `o(n,_,_)` (considering *time*), `o(n,o,_)` (considering *place*), `o(n,o,f)` (considering *mode*). Thus the optimal outcome is `o(n, o, f)`. ■

We can use this procedure also to compute the optimal completion, considering the input partial outcome as pre-assigned values for a subset of features. Recursive LCP-theories may require m dominance queries, where m is the number of dominance goals in the body of the input preference rules. Since we use tabling, the time cost is amortized over all calls from the problem-solver (in lie of space). With this change, the procedure described above can be used. Because of the dominance queries, the optimality procedure is PSPACE-complete. We note that if evidence for some variables is given, the resulting simplified LCP theory may have the structure of one of the special cases discussed above, and hence we can use specialized, polynomial procedures. To this

end, our implementation maintains a dynamic dependency graph that ignores features for which values have been provided.

5.3 Recursive LCP-theories

Once we have defined a unification theory of preferences written directly in Datalog, we explored the potentials of this translation. Using this model a user can specify also recursive or dependent preferences between two outcomes `S,T` by supplying clauses for the atom `d(S,T)`:

```
d(S,T) :- c, g1, ..., gk.
```

where `c` is a constraint and `g1`, ..., `gk` are possibly recursively defined predicates involving `d/2`. If $k > 0$, this clause is called *Recursive LCP rule*. The user can also specify hard constraints on features in a similar way as for FLCP-rules. Recursive or dependent rules are particularly useful in multi-agent contexts, where different agents may influence each other by stating their preferences depending on the preferences of some other agent [72].

■ **Example 5.4** Let us consider two agents ranking features "main dish" (pasta or fish) and dessert (tiramisu or muffin). We can formulate "If Alice doesn't prefer pasta, I would like to take pasta" as:

```
d(o(AM, AD, pasta, ID),
        o(AM, AD, fish, ID)) :-
                dom(o(fish,_,_,_),
                o(pasta,_,_,_)).
```

■

We do not assume acyclicity in variable ordering. Recursive LCP theories have the full power of Datalog (any Datalog program can be expressed as a recursive LCP theory).

The recursive rules need to be **self consistent** to not introduce contradictions.

Definition 5.3.1 A rule R is *self-consistent* if given any fact F all conclusions generated by the rule R jointly the fact F are consistent with F.

■ **Example 5.5** Given two features, with domains $\{A1, A2\}$ and $\{B1, B2\}$ respectively, the following rule R is self-inconsistent: `d(o(A2,X),o(A1,X)):- d(o(A1,Y),o(A2,Y)).` Considering the fact F =`d(o(A1,B1),o(A2,B1)).` jointly to R we obtain the facts F_1 =`d(o(A2,B1),o(A1,B1)).` and F_2 =`d(o(A2,B2),o(A1,B2)).`. F_1 is inconsistent with F. ■

We might also want to represent a more general set of rules, in which we universally quantify the variable that are present in the body of the rule.

■ **Example 5.6 — Quantification in the body of the rule..** Given two agents `M` mother and C the child, we translate the sentence "The child has the same opinions as his mother" as:

```
d(o(M, C1), o(M, C2)) :-
    ∀C dom(o(M1,C), o(M2,C)).
```

■

To represent this kind of rules we need an extension of Datalog that allows universal quantifications also in the body of the rules and not only in the head.

	General structure	Acyclic	Tree
Dominance			
CP-nets:	PSPACE-complete [52]	NP or harder [11]	Polynomial [9, 11]
CP-theories:	PSPACE-complete [111]	PSPACE-complete [111]	**Polynomial**
FLCP-theories:	**PSPACE-complete**	**PSPACE-complete**	**Polynomial***
Recursive LCP-theories:	**EXPTIME-complete**	**—**	**—**
Consistency / Optimality			
CP-nets:	PSPACE-complete [52]	Polynomial [11]	Polynomial [11]
CP-theories:	PSPACE-complete [111]	Polynomial [111]	Polynomial [111]
FLCP-theories:	**PSPACE-complete / ?**	**? / Polynomial***	**? / Polynomial ***
Recursive LCP-theories:	**EXPTIME-complete**	**—**	**—**

Table 5.2: Computational complexity of Dominance, Consistency and Optimality. *Bold: our results, *: with some constraint on the form of the rule, ?: open problem, —: dependency graph not defined.*

5.3.1 Algorithmic properties

The main algorithmic tasks regarding a preference theory are *dominance queries*, *consistency checking*, and *outcome optimization*.

The results for Recursive LCP-theories are summarized in Table 5.2.

It is important to notice that for Recursive LCP-theories optimality and consistency procedures never have a lower computational complexity than the dominance procedure, since a recursive LCP rule also contains a dominance query. We note in passing that for FLCP theories, data-complexity is also of interest. Recall that data-complexity for a Datalog programs is the complexity of determining, for a fixed program P, and input database D and query q, whether $P, D \vdash q$ (as a function of the size of D and q). What is the distinction between P and D for LCP theories? For FLCP theories, P is simply the clauses for `dom/2`, and `consistent/0`. Once N, the number of features is fixed, this program is fixed. Thus data complexity for consistency of LCP theories corresponds to the complexity of determining for *fixed* N, whether $P, D \vdash$ `inconsistent`, as a function of the number of rules in the program. For FLCP (=linear Datalog) the data-complexity is NLogSpace-complete (see e.g. [55]).

Dominance

Theorem 5.3.1 Given a recursive LCP theory P over N features, deciding `dom(s,t)` is EXPTIME-complete in N.

Proof. The proof is as above, except that the `d/2` clauses may no longer be linear, hence the combined complexity for full Datalog comes into play. ■

We note in passing that the connection with Datalog allows for a simple and direct proof of the PSPACE-hardness of dominance for CP-nets. We show that the PSPACE-HARD problem of determining whether a deterministic Turing Machine can accept the empty string without ever moving out of the first k tape cells can be reduced to checking dominance queries for CP-nets by

modifying slightly the proof for Datalog in [55, Theorem 4.5]. In practice, the solutions of the dominance problem can be found using tabling on the `dom/2` predicate.

Ordering

Given a logical conditional preference program `P` and a pair of outcomes `(S1, ..., Sn)` and `(T1, ..., Tn)`, this query succeeds iff the first is *not* preferred to the second by `P`.

Since all queries for Datalog programs terminate, this query can be implemented simply by running the query `?- (S1, ..., Sn) > (T1, ..., Tn).` If this query succeeds return false, else return true.

Consistency

Consistency is determined by invoking the query `?-consistent.` This takes advantage of the tabling of the `dom/2` predicate.

The following theorem affirms that consistency remains in PSPACE even when the language for preferences is extended beyond CP-nets to FLCP rules, and it is a generalization of Theorem 3 in [52].

Theorem 5.3.2 Given a recursive LCP theory P over N features, deciding consistency is EXPTIME-complete in N.

As above, noting that the combined complexity for full Datalog is EXPTIME.

Outcome optimization

For optimality, the user invokes the query `?- optimal(s).` Note that `optimal/1` uses partial order answer subsumption. In theory this may result in an exponential number of calls to `dom/2` atoms; each check takes exponential time. This leads to:

Theorem 5.3.3 Given a recursive LCP theory P over N features, deciding optimality is in EXPTIME over N.

The proof follows directly from the proof of Theorem 5.3.1 since the computational complexity for the optimality for recursive rule is greater then the complexity of dominance, and the fact that enumerate the whole set of outcomes pair is exponential.

5.4 Reasoning with LCP-theories

5.4.1 Completeness of LCP-theories

In this section we analyze the relation between the LCP-theories and two notion of completeness.

First we considered the classical notion of *completeness*: a preference theory is complete if and only if given an order, it can be represented with this theory. FLCP-theories trivially satisfies this condition. Given a partial order we can express each relation between two outcome `o` and `o'` with the following FLCP-rule: `d(o,o').` Thus we enumerate all the relations between pairs of outcomes (that are facts in our theory) and we add the corresponding rule in the database. The theory that we obtain corresponds to the order given in input.

We considered also other notion of completeness: a theory is complete if and only if we can describe using this theory all the possible transformations from a database to other one.

Definition 5.4.1 A theory is complete if given two databases `D` and `D'` exists a set of rules `R`, described in this theory, such that we can obtain `D'` applying the rules `R` to `D`.

In our case, since we are describing the theory using a logic formulation, in particular Datalog, we have that the database `D'` is always a superset of `D`: a logic rule can only add information and not delete information. With this restriction, we observe that LCP-theories satisfies also this notion of completeness. Thus, given two databases `D` and `D'` in input, we can construct the set of rules `R` as follow: the body of the rules is a tautology and the head correspond to a fact in the set `D' \ D`. In this way we can obtain every extension `D'` of the initial database `D`.

5.4.2 Experimental evaluation

We have developed a compiler for LCP theories, also called LCP. The compiler and associated tooling will be made available on Github as an open source project under the Eclipse Public Licence. The compiler reads a LCP-theory, builds the dependency graph and checks whether it represents an acyclic CP-net. If so, it performs a linearization of the dependency graph, and produces a pre-digested representation of the theory. Otherwise it emits the clauses unchanged so that the standard default (tabled) algorithms optimality, consistency and dominance can be used. In more detail, the compiler captures (a linearization of) the dependency order in a clause `dependency([`$a_1,a_2,\ldots,a_N$`]).` where $a_i \in 1\ldots N$ (features are implemented as Prolog integers), and if a_j depends on a_i, then $i < j$. Suppose a_p depends on $a_{i_1},\ldots,a_{i_k}$. If the input LCP program specifies that if each of the features $a_{i_1},\ldots,a_{i_k}$ had values $x_{i_1},\ldots,x_{i_k}$ respectively, then the known order of values of a_p is given by (best) $w_1,\ldots,w_r$ (worst), then the compiler emits the fact `preference([`$x_{p-1},\ldots,x_1$`], [`$w_1,\ldots,w_r$`]).` with $x_{i_1},\ldots,x_{i_k}$ as constant, and the remaining x_i as unique variables (occur only once in the clause). Thus we use Prolog unification to select the correct preference clause to use, given the current partial outcome `[v_p, ..., v_1]` specifying values for the first `p` attributes (in reverse dependency order). To improve the performance, our implementation exploits some specific input known structures such as tree or acyclic graph. In such a way the compiler can reach high performances when the input falls into one of the considered special cases. On the other hand, if the input is not a special case than the algorithm uses the state-of-the-art approaches to find a solution for the given instance. For the best of our knowledge this is the first implementation which have this property.

The following code for `optimize/1` uses these clauses and implements *Acyclic-LCP-Opt*:

```
optimize(O) :- O=o(_,_), dependency(D),
               reorder(D, O, AList),
               optimize_a([], AList).
reorder([], _, []).
reorder([X| Xs], O, [V | Vs]) :-
               arg(X, O, V),
               reorder(Xs, O, Vs).
```

```
optimize_a(_, []).
optimize_a(Upargs, [Xs|R]) :-
                select(Upargs, Xs),
                append(Xs, Upargs, Upargs1),
                optimize_a(Upargs1, R).
select(Us, []).
select(Us, [X | R]) :-
                nonvar(X),
                select(Us, R).
select(Us, [X | R]) :- var(X),
                preference(Upargs, [X|_]),
                select(Us ,R).
```

We would test the developed LCP-theories compiler against a dataset of CP-nets, to check performances and also to check whether the tool can give the right answer in a reasonable amount of time. Due to the fact that these dataset type are not available we have also implemented a CP-net generator. Generating CP-nets i.i.d. is non-trivial [2] and therefore we use an approximation method that randomly generates acyclic CP-nets with N features, given a maximum in-degree k for each feature. We consider a fixed ordering $X_1, \ldots, X_N$ of features. We first generate the acyclic dependency graph: for each feature X_i, we randomly choose its in-degree $d \in 0..\min\{k, i-1\}$. Next, we randomly choose d parents from the features $\{X_1, \ldots, X_{i-1}\}$. When the graph is built, we fill in the CP-tables choosing randomly one element of the domain (since the domain is binary). The resulting CP-net is written out as an LCP theory, using XSB Version 3.5.0 syntax [103]. We have run two different kinds of experiment. In the first we varied the number of features from 5 to 200 fixing the maximal number of dependencies for each feature and measure the running time for optimality queries, In the second experiment, fixing the number of features, we varied the upper bound of dependencies from 1 to 10. In both experiments we used the CP-net generator that we implemented to generate the CP-nets and we asked for the optimal outcome of the CP-nets 100 times and then we computed the average elapsed time to output the result.

Figure 5.1 shows the results for the experiment of the first type where the upper bound for the number of dependencies is fixed to 6. The elapsed time to compute the optimal outcome grows quadratically in the number of features. This is in line with our computational results, notice that the runtime is not linear because each step involves checking for the value of parents using unification on $O(N)$ terms. Different tests of the second type gave similar results even varying the upper bound of dependencies, for a fixed number of features.

5.5 Semantics for the dominance

In this section we introduce three new semantics for the dominance. The computational complexity of these new tasks is polynomial (in time) in the CP-net description size contra the exponential time of the classical notion of dominance (also for acyclic structures). This is the main advantage of these new semantics respect to the usual definition of dominance.

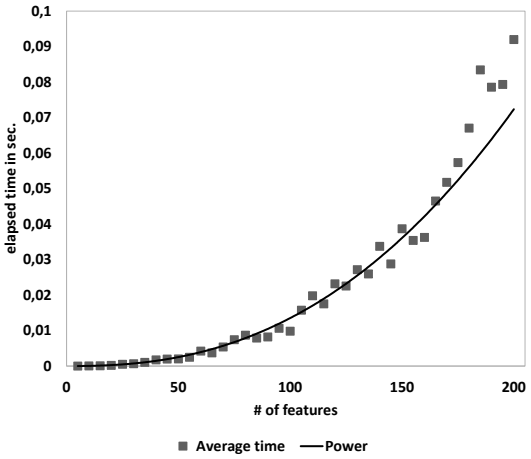


Figure 5.1: Optimal outcome performances.

First we provide some notions that we need for the definition of the three semantics.

Definition 5.5.1 Given a program P that defines a CP-net $\mathscr{C}$ and given a complete or partial solution (outcome) S we define as $P(S)$ the set of constraints generated by $\mathscr{C}$ given the assignment of the variables S.

■ **Example 5.7** Given a CP-net $\mathscr{C}$ with 3 variables A, B and C and the edges $A \to B$, $A \to C$ and $B \to C$ and the following CP-tables:

- **A=** $a \succ \bar{a}$
- **B=** $a : b \succ \bar{b}$, $\bar{a} : \bar{b} \succ b$
- **C=** $ab : c \succ \bar{c}$, $a\bar{b} : \bar{c} \succ c$, $\bar{a}b : \bar{c} \succ c$, $\bar{a}\bar{b} : c \succ \bar{c}$

and an outcome $S = \bar{a}b\bar{c}$ then the program P defined by $\mathscr{C}$ produces the set:

$$P(S) = \{a \succ \bar{a}, \bar{b} \succ b, \bar{c} \succ c\}$$

Given a partial outcome $T = ac$, the program produces the set:

$$P(T) = \{a \succ \bar{a}, b \succ \bar{b}\}$$

■

5.5.1 Alternative dominance semantics

Given two solutions (outcomes) S and T defined over a set of feature `Var` and a program P corresponding to a CP-net, we propose the following three different semantics for the dominance.

Definition 5.5.2 — S1. The semantics $S1$ associates a score $Score(S)$ to each solution S. Given two outcomes S and T, if $Score(S) < Score(T)$ then $S \succ T$.

Given an outcome S, to compute $Score(S)$, first we have to compute $P(S)$. Then, with the resulting preferences, we calculate for each dimension i (associated to the feature X_i) the distance between the position of $S \restriction_{X_i}$ and the max in the ordering defined by $P(S)$ over the domain D_i of

the feature X_i. Then we aggregate the scores of each component with an aggregation function f. Many choices for the function f are possible, for example lexicographic, point wise, sum, maximum, etc.

Definition 5.5.3 — S2. Given two outcomes S and T, if the preference order $\succ$ generated by $P(S)$ equals that generated by $P(T)$ ($P(S)$ and $P(T)$ generate the same sets of constraints), then $S \succ T$ if in each dimension i (associated to the feature X_i), $S \restriction_{X_i} \succ T \restriction_{X_i}$.

Intuition: We will compare S and T only if they agree on the generated preference order. Two ways in which S and T can be different and still generate the same preference order:

- They differ on variables which are not decision variables (i.e. variables that influence preferences on other variables)
- They differ on the values of decision variables, but that does not affect the generated preference order.

Definition 5.5.4 — S3. The semantic $S3$ is parametric in the choice of variables in V that is a subset of `Var` . If the preference order $\succ$ generated by $P(S \restriction_V)$ equals that generated by $P(T \restriction_V)$ then $S \succ T$ if in each dimension $i \in V$, $S \restriction_{X_i} \succ T \restriction_{X_i}$.

■ **Example 5.8** Given a CP-net with 3 variables A, B and C and the edges $A \to B$, $A \to C$ and $B \to C$ and the following CP-tables:

- **A=** $a \succ \bar{a}$
- **B=** $a : b \succ \bar{b}$, $\bar{a} : \bar{b} \succ b$
- **C=** $ab : c \succ \bar{c}$, $a\bar{b} : \bar{c} \succ c$, $\bar{a}b : \bar{c} \succ c$, $\bar{a}\bar{b} : c \succ \bar{c}$

and given two outcomes S and T such that $S = a\bar{b}c$ and $T = \bar{a}bc$ we obtain that:

$$P(S) = \{a \succ \bar{a}, b \succ \bar{b}, \bar{c} \succ c\}$$

$$P(T) = \{a \succ \bar{a}, \bar{b} \succ b, \bar{c} \succ c\}$$

We obtain the following results for the three semantics:

- **S1 (using $f =$ sum):** $Score(S) = 0+1+1 = 2$, and $Score(T) = 1+1+1 = 3$. $Score(S) > Score(T)$ thus $S \succ T$.
- **S2:** $P(S) \neq P(T)$ thus $S \bowtie T$
- **S3:**
  - if $V = \{A, C\}$, then $P(S \restriction_V) = P(T \restriction_V)$. $S \restriction_A \succ T \restriction_A$ and $S \restriction_C = T \restriction_C$ thus $S \succ T$;
  - if $V = \{C\}$, then $P(S \restriction_V) = P(T \restriction_V)$. $S \restriction_C = T \restriction_C$ thus $S \succ T$;
  - if $V = \{B\}$, then $P(S \restriction_V) \neq P(T \restriction_V)$ thus $S \bowtie T$.

■

5.5.2 Properties of the semantics

We observe that the three semantics satisfy the following intuitive properties:

- S1 is transitive
- S2 is transitive

- S3 is transitive i.e. $S \preceq_V T$ and $T \preceq_V U$ implies $S \preceq_V U$.
- S2 is obtained from S3 by taking V to be $\texttt{Var}$.

Observation 5.3 It is important to notice that the assumption to have an aggregation function for the scores (for S1) that is monotonic not decrescent with the notion of weakly Pareto dominance between the outcomes, is a restriction, but it is very common and all the most common aggregation functions (i.e. sum, max, etc.) have this property. ■

Definition 5.5.5 Given a CP-net described by a program P and two outcomes S and T, we can generate the sets $P(S)$ and $P(T)$. We define as $\tilde{V}$ the maximal subset of $\texttt{Var}$ such that $P(S\restriction_{\tilde{V}}) = P(T\restriction_{\tilde{V}})$. This set could be empty.

The set $\tilde{V}$ has the following properties:

- if $S \succ T$ in $\tilde{V}$, for all V subset of $\tilde{V}$ then $S \succ T$ in V;
- for all subset Y of $\texttt{Var}$ that contain at least one element of $\texttt{Var} \setminus \tilde{V}$ we obtain $S \bowtie T$ on Y.

We prove now three propositions (Proposition 5.5.1, Proposition 5.5.2 and Proposition 5.5.3)that describe the relations between the three semantics.

Proposition 5.5.1 If $S \succ T$ in S1 then:

1. in S2 it could be $S \succ T$ or $S \bowtie T$ but never $S \prec T$
2. in S3 it could be $S \succ T$, $S \prec T$ or $S \bowtie T$

Proof. 1.
- $[\prec]$: $S \succ T$ in S1 and we assume by contradiction that $S \prec T$ in S2. This means that $P(S) = P(T)$ and that $\forall X \in \texttt{Var}\ \ S\restriction_X \prec T\restriction_X$. This implies that $Score(S\restriction_X) > Score(T\restriction_X)$, and than $f(Score(S)) > f(Score(T))$ (with the assumption of monotonic function f). Thus $S \prec T$ in S1 that is a contradiction.
- $[\succ]$: Given a CP-net with only a feature A such that $a \succ \bar{a}$ and given $S = a$ and $T = \bar{a}$ then we obtain that $S \succ T$ in S1 and also in S2.
- $[\bowtie]$: Given a CP-net with two features A and B such that $a \succ \bar{a}$, $a : b \succ \bar{b}$ and $\bar{a} : \bar{b} \succ b$, and given $S = ab$ and $T = \bar{a}b$ then we obtain that $S \succ T$ in S1 but $S \bowtie T$ in S2.

2. Given a CP-net with three features A, B and C such that $a \succ \bar{a}$, $b \succ \bar{b}$, $b : c \succ \bar{c}$ and $\bar{b} : \bar{c} \succ c$, given $S = \bar{a}bc$ and $T = a\bar{b}c$ we obtain that $S \succ T$ in S1 and:
   - $[\prec]$: if $V = \{A\}$ then $S \prec T$
   - $[\succ]$: if $V = \{B\}$ then $S \succ T$
   - $[\bowtie]$: if $V = \{A, C\}$ then $S \bowtie T$

■

Proposition 5.5.2 Given two outcomes S and T, if $S \succ T$ in S2 then:

1. in S1 it could be $S \succ T$ or $S \bowtie T$ if we use an aggregation function f for the scores that is monotonic not decrescent with the notion of weakly Pareto dominance between the outcomes. If we use a monotonic crescent function $S \succ T$.
2. in S3 it could be only $S \succ T$.

Proof. 1. If $S \succ T$ in S2 then the two sets $P(S)$ and $P(T)$ contains the same constraints and

$S \restriction_{X_i} \succ T \restriction_{X_i} \ \forall i$. This implies that $Score(S \restriction_{X_i}) < Score(T \restriction_{X_i}) \ \forall i$. If we use an aggregation function f for the scores that is monotonic crescent with the notion of Pareto dominance between the outcomes, this means that given two outcomes o and o', if

$$\forall i \ Score(o \restriction_{X_i}) > Score(o' \restriction_{X_i}) \text{ then } f(Score(o)) > f(Score(o')) \ .$$

In our case this implies that $f(Score(S)) < f(Score(T))$ and then we obtain that $S \succ T$ also in S1.

2. We suppose by contradiction that $S \succ T$ in S2 and $S \prec T$ in S3. This implies that exists a subset V of `Var` such that $\forall X \in V$ we have $S \restriction_X \prec T \restriction_X$. This is a contradiction because if $S \succ T$ we have that $\forall X \in$ `Var` $S \restriction_X \prec T \restriction_X$.

   We suppose by contradiction that $S \succ T$ in S2 and $S \bowtie T$ in S3. We have two cases:

   - $S \bowtie T$ in S3 because $P(S \restriction_V) \neq P(T \restriction_V)$. But this implies that also $P(S) \neq P(T)$ and thus that $S \bowtie T$ also in S2, that is a contradiction.
   - $P(S \restriction_V) = P(T \restriction_V)$ and $S \bowtie T$ in S3 because exists $X_1, X_2 \in V$ such that $S \restriction_{X_1} \prec T \restriction_{X_1}$ and $S \restriction_{X_2} \prec T \restriction_{X_2}$ or viceversa. But this implies also that $S \bowtie T$ in S2 and this is a contradiction.

■

Proposition 5.5.3 If $S \succ T$ in S3 then:

1. in S1 it could be $S \succ T$ or $S \bowtie T$
2. in S2 it could be $S \succ T$, $S \prec T$ or $S \bowtie T$

Proof. 1. Given a CP-net with three independent features A, B and C such that $a \succ \bar{a}$, $b \succ \bar{b}$ and $c \succ \bar{c}$:

   - $[\prec]$: given $S = \bar{a}bc$ and $T = a\bar{b}\bar{c}$ we obtain that $S \succ T$ in S1 and if $V = \{B\}$ then $S \succ T$ in S3
   - $[\succ]$: given $S = a\bar{b}\bar{c}$ and $T = \bar{a}bc$ we obtain that $S \prec T$ in S1 and if $V = \{A\}$ then $S \succ T$ in S3

   Given a CP-net with two independent features $A$and B such that $a \succ \bar{a}$ and $b \succ \bar{b}$:

   - $[\bowtie]$: given $S = a\bar{b}$ and $T = \bar{a}b$ we obtain that $S \bowtie T$ in S1 and if $V = \{A\}$ then $S \succ T$ in S3

2. Given a CP-net with two independent features $A$and B such that $a \succ \bar{a}$ and $b \succ \bar{b}$:

   - $[\succ]$: given $S = a\bar{b}$ and $T = \bar{a}b$ if $V = \{A\}$ then $S \succ T$ in S3 but $S \bowtie T$ in S2
   - $[\bowtie]$: given $S = ab$ and $T = \bar{a}\bar{b}$ then $S \succ T$ in S3 and also $S \bowtie T$ in S2

   We prove by contradiction the remaining case:

   - $[\prec]$: We suppose by contradiction that $S \succ T$ in S3 and $S \prec T$ in S2. If $S \prec T$ in S2, this implies that $P(S) = P(T)$ and also that $\forall X \in$ `Var` $S \restriction_X \prec T \restriction_X$. Considering the set V used by S3, because $S \succ T$ in S3 we have that $\forall X \in V \ S \restriction_X \succ T \restriction_X$. But V is a subset of `Var` so it is a contradiction.

■

5.5.3 Properties of the semantics comparing to the classical dominance

We compare now the three semantics with the classical dominance semantic.

Theorem 5.5.4 If $S \succ T$ in S2 then $S \succ T$ also using the classical notion of dominance.

Proof. Given S and T such that $S \succ T$ in S2, we want to prove that exist a worsening flip sequence from S to T. If $S \succ T$ in S2, then we have that $P(S) = P(T)$ and $S \restriction_X \succ T \restriction_X \ \forall X \in \texttt{Var}$. Thus if we consider the variables from the leafs of the structure, we obtain an order over the variables that have a different value on S and on T: $X_1, X_2, \cdots, X_k$. Thus considering the sequence $P = S, P_1, P_2, \cdots, P_k = T$, P is a worsening sequence because for each step i we make a worsening flip for the feature X_i given the assignment of the parents in S. ■

Corollary 5.5.5 Given a PCP-net and two outcomes S and T then the value of the dominance using S2 is a lower bound for the the value of the classical dominance.

Proposition 5.5.6 If $S \succ T$ in S3 and in S1, in the classical dominance we could have $S \succ T$, $S \bowtie T$ or also $S \prec T$.

Proof.

- If $S \succ T$ in S1 we may have the same in the classical dominance:
  - $[\succ]$: Given a CP-net with a independent feature A such that $a \succ \bar{a}$ and given the outcomes $S = a$ and $T = \bar{a}$ then in both S1 and real dominance we have $S \succ T$.
  - $[\bowtie]$: Given a CP-net with three independent features A and B and C such that $a \succ \bar{a}$, $b \succ \bar{b}$ and $c \succ \bar{c}$ and two outcome $S = x\bar{y}z$ and $T = \bar{x}y\bar{z}$. We obtain that $Score(S) = 1 < Score(T) = 2$ and thus $S \succ T$ in S1 but $S \bowtie T$ in the classical dominance.
  - $[\prec]$: Given a CP-net with five feature A, B, C, D, E and the dependencies $A \to B$, $B \to C$, $C \to D$ and $C \to E$ such that: $a \succ \bar{a}$, $a : b \succ \bar{b}$, $\bar{a} : \bar{b} \succ b$, $b : c \succ \bar{c}$, $\bar{b} : \bar{c} \succ c$, $c : d \succ \bar{d}$, $\bar{c} : \bar{d} \succ d$, $c : e \succ \bar{e}$, $\bar{c} : \bar{e} \succ e$, and given two outcomes $S = \bar{a}\bar{b}\bar{c}de$ and $T = abcd\bar{e}$. We suppose to aggregate the scores with the sum and thus we obtain: $Score(S) = 1$ and $Score(T) = 2$ thus $S \succ T$ in S1. But $S \prec T$ in the classical dominance because exist a worsening flip sequence from T to S:

$$T = abcd\bar{e} \succ ab\bar{c}d\bar{e} \succ a\bar{b}\bar{c}d\bar{e} \succ \bar{a}\bar{b}\bar{c}d\bar{e} \succ a\bar{b}\bar{c}\bar{d}e \succ \bar{a}\bar{b}\bar{c}de = S.$$

- If $S \succ T$ in S3 we may have the same in the classical dominance:
  - $[\succ]$: Given a CP-net with two independent feature A and B such that $a \succ \bar{a}$ and $b \succ \bar{b}$ and given the outcomes $S = ab$ and $T = \bar{a}b$ then in both S3 with $V = \{A\}$ and real dominance we have $S \succ T$.
  - $[\bowtie]$: Given a CP-net with two independent features A and B such that $a \succ \bar{a}$ and $b \succ \bar{b}$ and two outcome $S = a\bar{b}$ and $T = \bar{a}b$ and we take $V = A$, we obtain that $S \succ T$ in S3 but $S \bowtie T$ in the classical dominance.
  - $[\prec]$: Given a CP-net with three features A, B and C such that $a \succ \bar{a}$, $b \succ \bar{b}$ and $ab : c \succ \bar{c}$, $a\bar{b} : \bar{c} \succ c$, $\bar{a}b : \bar{c} \succ c$, $\bar{a}\bar{b} : c \succ \bar{c}$ and two outcome $S = \bar{a}\bar{b}c$ and $T = ab\bar{c}$. Then $S \succ T$ in S3 if we take $V = C$ but $T \succ S$ in the classical dominance because exists a sequence of

worsening flips from T to S:

$$T = ab\bar{c} \succ a\bar{b}\bar{c} \succ a\bar{b}c \succ \bar{a}\bar{b}c = S.$$

■

Proposition 5.5.7 If $S \succ T$ in the classical dominance, in S2 could be $S \succ T$ or $S \bowtie T$ but not $S \prec T$.

Proof. If $S \succ T$ in the classical dominance, in S2 it can't be $S \prec T$ otherwise, using Theorem 5.5.6, we must have that also $S \prec T$ in the classical dominance. ■

Theorem 5.5.8 The optimal outcome using S1 is the same as using the classical notion of dominance, in an acyclic CP-net.

Proof. The optimal outcome in S1, given the sum or the maximum as score aggregation function, is the outcome that has the minimum score, this score is 0. This means that $\forall X \in \texttt{Var}\ \mathit{Score}(\texttt{Var}) = 0$. This condition holds only if for each feature of the outcome, its assignment it is in the top position in the statements of the corresponding line in the CP-net, given the assignment of the parents of the outcome. But this correspond to the outcome resulting from the sweep forward procedure that determines the optimal outcome for the classical dominance. ■

5.6 Summary and Discussion

Since CP-nets framework and its generalizations/variants are limited in expressiveness and are specialized formalisms with their own ad hoc syntax and semantics, in this chapter we presented a new unification theory for qualitative conditional preferences, namely LCP-theories. We defined this new theory directly in standard first order logic, as constrained Datalog theories. This is an important advantage since we now make Datalog's rich semantic, algorithmic and implementation framework available in service of conditional preferences.

First we introduced Flat LCP-theories, conditional preferences defined as *linear* Datalog theories. We compared Flat LCP-theories with a set of models representing conditional preferences such as CP-nets, CP-theories, GCP-nets and comparative preferences and we showed that Flat LCP-theories can express each of them: are all particular cases of LCP theories. We then provided algorithms for the main tasks: consistency checking, outcome optimization and dominance testing, with also a computational complexity analysis. We compared our complexity results with the results concerning other models as CP-nets and CP-theories, considering three different structure types for the dependency graph: general structures, acyclic structures and tree-shaped structures. The results show that considering Flat LCP-theories does not increase, in the most of the cases, the computational time.

We then generalized Flat LCP-Theories allowing also Datalog recursive clauses into the formulation. This defines a new notion of preferences: recursive conditional preferences. Thus, based on this idea, we defined Recursive LCP-Theories. We introduced procedures for the main

reasoning tasks: consistency checking, outcome optimization and dominance testing, with also a computational complexity analysis. Obviously, increasing the power of the model expression, we observed an increase of the computational complexity of the tasks procedures. We also implemented in modern tabled Prolog systems such as XSB Prolog, the main tasks, providing an experimental analysis that confirmed the theoretical study.

Focusing on the direct expression of our formalism in Datalog we reinterpretated the notion of dominance, presenting three more efficient alternative definitions for the dominance task. We provided a theoretical comparison of these three new semantics between them and in comparison to the classical dominance semantics.

In conclusion the power of LCP-theories, in addiction to unify the state-of-the-art formalisms for conditional preferences, is that LCP-theories extend them (introducing the new notion of recursive conditional preferences), preserving the state-of-the-art computational complexities.

6. A real-life scenario: kidney transplant protocol

The role of kidneys is to filter waste from blood. Kidney failure leads to death in months. One option is dialysis, but it is a invasive procedure and it leads to a fast worsening of the quality of the patients life. The best treatment for the kidney disease is transplantation. Kidneys for transplantation may be obtained from deceased donors or from living donors. This second option allows a patient to receive the kidney of a friend or a relative, or in same cases, an altruistic donor. But in many cases the patient is not compatible with the relative or friend, for a blood incompatibility or tissue incompatibility. Thus there is also another possibility for a living transplantation: kidney exchange [107]. Kidney exchange is a powerful lifesaving alternative for patients that are waiting in the deceased-donor waiting list, since it permits to swap the living donors of a patient with another donor-patient pair. The pairs that join this program are not necessary only the incompatible ones, but it is encouraged the participation of all the pairs that could have an improvement in the quality of the transplantation or in the expected life.

Kidney exchange is widely used in many countries, such as UK, US, Spain, etc. The success of this procedure depends on the amount of pairs that participate to the program. In Italy, and in particular in our context of Padua, the patients do not take a lot of advantage from this program since only a very small subset of the patients join the kidney exchange program. The results is that there are few pairs and thus few compatible matchings. Our purpose is to encourage people and also the medical structures to participate. The principal way to obtain our scope is to guarantee a sure advantage in the participation. For example we force the system to produce matching that increase considerably the quality of the transplantation of all the pairs. In the case of incompatible pairs the advantage is to obtain a compatible matching and for the compatible pairs the advantage corresponds to a considerable increasing of the expected life, the decrease of the age delta between the pairs or the increase of the compatibility.

The procedures used to compute these matching, find cycles of exchange or chains that starts from an altruistic donor. Our idea consists into combine these two options with also chains that starts from a deceased donor kidney. In this way, given a kidney from a deceased donor, we can obtain more than one transplantation. We implemented the algorithm and we tested it in the context of the Padua medical center, obtaining encouraging results.

In the medical literature it is a well known problem that a large percentage of the matches computed by the exchange algorithms are not performed [35], for many reason: varying levels of sensitization between candidates and donors in the pool, illness, uncertainty in medical knowledge, logistical problems or a donor in a chain reneging. We provide some preliminary ideas to minimize this percentage to increase the quality of the transplantations, introducing also patients preferences in the computation of the matching. Adding the preferences obviously reduces the number of the feasible matchings, but minimizing the number of the failure we obtain an increase of the global number of the transplantations effectively performed.

Chapter structure

The chapter is organized as follows.

- In Section 6.1 we introduce several approaches present in the literature.
- In Section 6.2 we present our procedure for kidney exchange. This procedure consider both cycles and chains starting from a deceased donor kidney. We provide the algorithm description as a system of constraints, then formulated in ILP (integer linear programming). We show examples of the procedure running interface.
- In Section 6.3 we perform an experimental analysis of our procedure applied to the Padua hospital dataset. We provide a results discussion.
- In Section 6.4 we introduce the future direction of introducing preferences in our algorithm, with the purpose of improving the results.

6.1 Background and Related works

6.1.1 Allocation algorithms detecting cycles

In a recent work Abraham et al. [1], provide an algorithm to solve the clearing problem maximizing cycle kidney exchanges. A cycle consists in a swapping between agents, where each agent is represented by a pair donor/patient and each agent receives a kidney from the next agent in the cycle and donates a kidney to the previous agent. In this work they fix a value as the maximum length of a cycle, since all transplants in a cycle must be performed simultaneously and thus long cycles are forbidden. Shorter cycles are preferred also because in long cycles if one agent drops out of the cycle more agents are involved not receiving the kidney. They also proved that detecting cycles with this cycle-length constraint is NP-hard. In this paper they use a basic formulation of the problem as a integer linear programming problem (ILP) and then they use techniques to improve both runtime and memory usage.

6.1.2 Allocation algorithms detecting chains with an altruistic donor

An alternative to the detection of cycles of kidney exchange, are the chains of exchanges: sequences of transplants initiated by an altruistic kidney donor. In practice this alternative is a highly successful practice, and widely used in United States. Dickerson et al. [34] provide an analysis the efficacy of chains proving the optimal length of a chain is of 3 exchanges, in very large dataset. This results is in contradiction with real-world results (in United States) since they show in [34] that in United States instead the solution quality improves by increasing the chain length exceeds 13 exchanges. This gap is explained by the fact that in reality the number of altruistic donor is smaller.

6.1.3 Other approaches

In another work of Dickerson et al. [35] they analyze the following problem concerning the combined formulation of kidney exchange with cycles and kidney exchange with chains: the number of transplantations failures. With failures we mean failure before that the transplant procedure takes place (not during or after it). The show that most of the planned matches fail: in the case of the UNOS exchange, 93% of matches fail (in 2012). They propose, to ensure this condition, to consider not the maximum cardinality matching but the transplant plan with maximum expected value. They move away from the deterministic clearing model (currently used) into a probabilistic model where we have to consider the failure probabilities on possible plans. They proved, running simulations first on theoretical data (on random graph models and on simulation generated via a model of dynamic kidney exchange) and then on real data (from kidney exchange match runs between 2010 and 2012), that their algorithm increases the number of expected transplants dramatically.

Gentry et al. in [94] proved that adding also compatible pairs and not only incompatible pairs doubles the number of matching (from 28.2% to 64.5% for single-center program and from 37.4% to 75.4% for national program). But in this study the compatible pairs play an altruistic role in the pool. Thus only pairs with altruistic intentions would be interested in participating in such a program. In the work of Nicolo' et al. [81] incentives the compatible pairs to participate to the program ensuring that also their matching is an advantage: they provide to a compatible pair a younger donor. Thus the advantage for compatible pairs is that they receive a kidney with a better quality that is reflecting in a higher expected graft survival.

Dickerson et. al in [36] focus their attention to a particular class of patient: the sensitized patient, that are highly-sensitized with a very low probability that their blood will pass a crossmatch test with a random organ. For these patients, finding a kidney is quite difficult. These kind of patients are about the 17% of the adult patients on the waiting list for deceased donor kidneys. They propose different notion of *fairness* to increase the number of sensitized patient that are allocated by the matching algorithm.

6.2 Our algorithm: detection of cycles and chains starting from deceased donors

The idea behind the algorithm is to maximize the number of transplantations given a kidney from a deceased donor, and additionally to look for cycles between the pairs patient/donor in the pool. Thus our idea is to consider a deceased donor kidney as an altruistic donors and thus to start a chain of exchanges from a deceased donor kidney. Usually a living donor is always preferred to a deceased donor, but in this way we maximize the number of transplantation for each received kidney and thus we increase the probability for a highly-sensitized patient to receive a graft.

Given a kidney from deceased donor, the procedure find all the chains that can starts from a deceased donor kidney and that continue in the pool of the pairs that participates to the exchange program. A chain always finish with a patient in the deceased-donor waiting list, where the sensitized patients have the priority. We add the constraint that if we are processing a kidney that we would have allocated to an sensitized patient, then the chain have to finish with another sensitized patient. In this way the total number of allocated sensitized patients is never lower to the number of sensitized patients allocated without using our procedure. We add in our pool also a subset of compatible pairs: pairs that become compatible through desensitization. This means that these pairs are not completely compatible but need desensitization to become fully compatible, a procedure that has some risks especially in the case of critical patient. Thus we propose, to this kind of pairs, a sure advantage in the participation, because they can find a match in which they do not need desensitization. In this way we are encouraging them to participate into the program. This idea is similar to the approach explored in the work of Nicolo' et al. in [81].

We provide a formal description of the procedure in the following sections.

6.2.1 Our exchange procedure

Our procedure take as input a pool composed by:

- living donor/patient pairs;
- highly-sensitized patients;
- kidneys;

and returns a matching composed by cycles between pairs, and chains, that start from a kidney and end or in a pair or in a sensitized patient.

We represent each element of the pool as a node of a graph, namely the compatibility graph. We have three different types of nodes, the pair nodes, the kidney nodes and the sensitized patient nodes. The edges in this graph represent the compatibility between the nodes. We add an edge (X,Y) from a node X to another node Y only if the donor/kidney of the first node X is compatible with the patient of the second node Y. The nodes that represent a kidney will have only outgoing edges and the sensitized patients nodes will have only ingoing edges. The pair nodes could have both outgoing and ingoing edges.

Furthermore each node has a set of constraints, that are different for the three types of nodes:

- *Kidney nodes.* This type of node can have either only an outgoing edge activated, or all the edges inactive, because each kidney can be grafted only to one patient or to no one.

- *Pair nodes.* This type of node can have either only an ingoing edge activated, or one ingoing and one outgoing edge activated or all the edges inactive. This because a pair can not donate a kidney without receive a kidney, but can receive and not donate because in this case we consider that the kidney of the donor will be donated to the patients in the deceased-donor waiting list.
- *sensitized patient nodes.* This type of node can have either only an ingoing edge activated, or all the edges inactive. This because each patient can either receive only one kidney or no one.

Formally, we partition the set of nodes in three sets: P the set of the pair nodes, IM the set of sensitized patient and K the set of kidney. Given a node X, we call E_X the set of its edges, partitioned in two subsets E_X^- of the ingoing edges and E_X^+ of the outgoing edges. An edge $e \in E$, where E is the set of all the edges, has value in $\{0,1\}$: 1 if active and 0 if inactive.

We translate the constraints described above in a formal way as:

$$\begin{cases} e \in \{0,1\} \quad \forall e \in E & \text{active/inactive} \\ \sum_{e \in E_X} e \in \{0,1\} \quad \forall X \in K & \text{kidney nodes} \\ \sum_{e \in E_X^-} e - \sum_{e \in E_X^+} e \in \{0,1\} \quad \forall X \in P & \text{pair nodes} \\ \sum_{e \in E_X} e \in \{0,1,2\} \quad \forall X \in P & \text{pair nodes} \\ \sum_{e \in E_X} e \in \{0,1\} \quad \forall X \in IM & \text{sensitized patient nodes} \end{cases}$$

If we are processing kidneys that we would have allocated to a sensitized patient (set IM), then we add also this constraint:

$$\sum_{X \in IM} \Big(\sum_{e \in E_X^-} e \Big) - \sum_{X \in K} \Big(\sum_{e \in E_X^-} e \Big) \geq 0$$

Our algorithm is divided into two main parts: *cycle detection* and *kidneys processing*. First we look for cycles between the pairs and then we process one by one the kidneys, following the inter-arrival sequence in the deceased-donor program, considering only optimal kidneys.

- *Cycle detection.* We build the compatibility graph using only pair nodes. Here we look for the maximum subset of active edges. In this step we can create only solution that are cycles between the pairs.
- *Kidneys processing.* We build the compatibility graph using all the kind of nodes, but adding only one kidney at iteration, following the inter-arrival sequence in the deceased-donor program. Here we look for the maximum subset of active edges. In this step we can create only solution that are chains between the nodes and each chain start with the processed kidney and can finish or with a pair node or with a sensitized patient node.

At each iteration of the algorithm we obtain a solution, the maximum subset of active edges. Thus we are maximizing the following objective function:

$$\sum_{e \in E} e$$

Once obtained a solution, we compute the nodes that corresponds to the patients involved in the solution matching, so the nodes that are touched by the active edges in the solution and we remove

this nodes/patients for the next iteration.

To solve this problem we use a integer linear programming (ILP) implementation. We maximize the sum of the variables e with the following constraints:

$$\begin{cases} \sum_{e\in E_X} e > -1 \quad \forall X \in K & \text{kidney nodes} \\ \sum_{e\in E_X} e < 2 \quad \forall X \in K & \text{kidney nodes} \\ \sum_{e\in E_X^-} e - \sum_{e\in E_X^+} e > -1 \quad \forall X \in P & \text{pair nodes} \\ \sum_{e\in E_X^-} e - \sum_{e\in E_X^+} e < 2 \quad \forall X \in P & \text{pair nodes} \\ \sum_{e\in E_X} e > -1 \quad \forall X \in P & \text{pair nodes} \\ \sum_{e\in E_X} e < 3 \quad \forall X \in P & \text{pair nodes} \\ \sum_{e\in E_X} e > -1 \quad \forall X \in IM & \text{sensitized patient nodes} \\ \sum_{e\in E_X} e < 2 \quad \forall X \in IM & \text{sensitized patient nodes} \\ \sum_{X\in IM}(\sum_{e\in E_X^-} e) - \sum_{X\in K}(\sum_{e\in E_X^-} e) > -1 & \text{kidneys allocated to sensitized patients} \end{cases}$$

We add also the integer constraints for all the edges e (the *active/inactive* constraint).

We implemented the algorithm in python using the opensource library *lp_solve*. To solve a ILP problem using this package we have to provide a constraint matrix M, a sign vector s, a known values vector v and a vector for the objecting function f. In our algorithm we provide the following structures M, s, v and f:

$$M = \begin{bmatrix} K^+ & IM^- & P^+ & P^- \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 1 & 0 & 0 \end{bmatrix} \quad v = \begin{bmatrix} -1 \\ 2 \\ -1 \\ 2 \\ -1 \\ 3 \\ -1 \\ 2 \\ -1 \end{bmatrix} \quad s = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \end{bmatrix} \quad f = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

The procedure returns the set of transplantation in each iteration and plots each solution into a graphical representation of the nodes:

- the green node are the nodes that represent not sensitized patient or kidney that we would not have allocated to an sensitized patient in the deceased donor list;
- the blue nodes are the nodes that represent sensitized patient or kidney that we would have allocated to an sensitized patient in the deceased donor list;
- the nodes are identified by a number that is the id of the node and by a label that is *D* for kidneys, *D/R* for the pairs and *R* for the sensitized patients;
- the active edges are red and the non active edges are black.

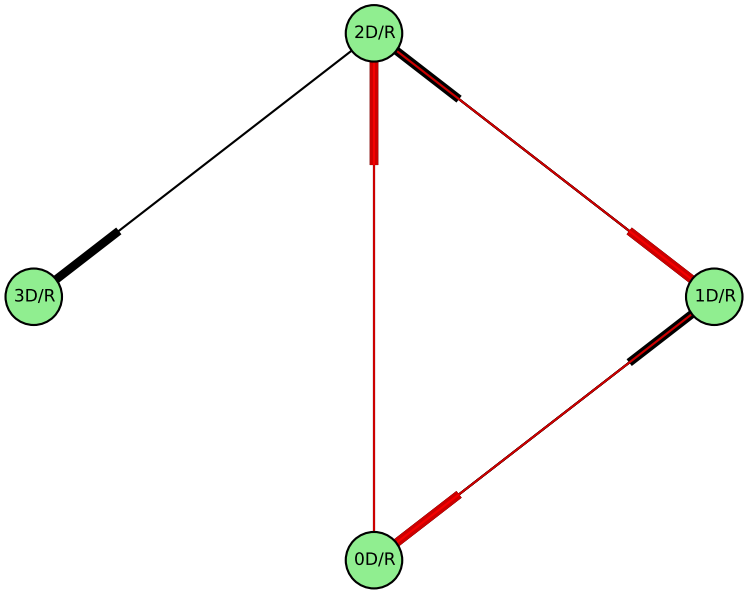


Figure 6.1: Cycle detection.

■ **Example 6.1** We show an example of the *cycle detection* step of the procedure in Figure 6.1 and an example of the *kidneys processing* step in Figure 6.2.

In the *cycle detection* example (Figure 6.1) we have four pairs $0D/R$, $1D/R$, $2D/R$ and $3D/R$. The pair $0D/R$ is compatible with the pairs $1D/R$ and $2D/R$, the pair $1D/R$ is compatible with the pairs $0D/R$ and $2D/R$, the pair $2D/R$ is compatible with the pairs $1D/R$ and $3D/R$ and the pair $3D/R$ is not compatible with any pair. We can see that the algorithm detects the cycle with the maximal number of trasplantations that is between nodes $0D/R$, $1D/R$ and $2D/R$.

In the *kidneys processing* example(Figure 6.2) we have four nodes: two pairs $0D/R$ and $1D/R$, uno kidney $4D$ and two patients, one sensitized $2R$ and one not sensitized $3R$. The pair $0D/R$ is compatible with the pair $1D/R$, the pair $1D/R$ is compatible with the receivers $3R$ and $2R$ and the donor $4D$ is compatible with the pairs $0D/R$ and $1D/R$. We can see that the algorithm detects the longest chain that maximize the number of transplantations that starts from $4D$ then reach in order $0D/R$, $1D/R$ and $2R$. We can observe that since the processed kidney is a "blue" node, a kidney that we would have allocated to an sensitized patient in the deceased donor list, the chain ends with another "blue" node, an sensitized patient. The constraint about the conservation of the number of sensitized patients/donors is satisfied.

■

6.3 Experiments

We performed experiments to test our procedure on a dataset extracted and anonymized from the Padua hospital database. We consider three years (from January 2012 until December 2014) of:

- living donor/patient pairs (or incompatible pairs or compatible with desensitization);
- sensitized patients;

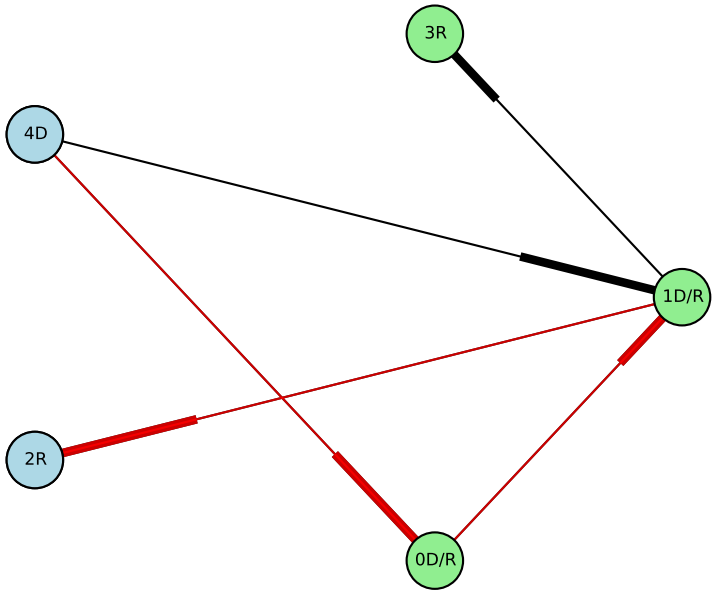


Figure 6.2: Kidneys processing.

- optimal kidney from deceased donors (optimal means that the age of the donor is less than 60, the kidney is suitable for a single transplantation thus not a double-kidney transplantation, the kidney has an immunological low risk).

The living donor/patient pairs are divided in two types: transplanted pairs with desensitization and not transplanted pairs since incompatible. Our purpose is to maximize the number of transplants of the not transplanted pairs and minimize the number of desensitizations because this procedure has some risks especially in the case of critical patient.

The Padua hospital dataset, that we have considered, has 51 pairs donor/patient (22 desensitization pairs and 29 incompatible pairs), 68 optimal kidneys and 21 sensitized patients.

We performed three type of experiments, in which we vary the composition of the pool.

6.3.1 First experiment

The pool is composed by:

- transplanted pairs with desensitization
- not transplanted pairs
- sensitized patients
- all the optimal kidney from cadaver

The results are described in Table 6.1.

6.3.2 Second experiment

The pool is composed by:

- transplanted pairs with desensitization
- not transplanted pairs
- sensitized patients

Number of:	Result
transplantation in the algorithm	40
transplantation in cycles	4 (2 cycles of length 2)
sensitized patient allocated by the algorithm	9
pairs previously with desensitization and now without	16
incompatible pairs now transplanted	9
allocated kidneys	23
kidney from deceased donors that we would not have allocated to an sensitized patient in the deceased donor list, that are allocated by the algorithm	1

Table 6.1: First experiment results.

- optimal kidney from deceased donors that we would not have allocated to an sensitized patient in the deceased donor list

The results are described in Table 6.2.

Number of:	Result
transplantation in the algorithm	39
transplantation in cycles	4 (2 cycles of length 2)
sensitized patient allocated by the algorithm	9
pairs previously with desensitization and now without	15
incompatible pairs now transplanted	9
allocated kidneys	22

Table 6.2: Second experiment results.

6.3.3 Third experiment

The pool is composed by:

- not transplanted pairs
- sensitized patients
- optimal kidney from deceased donors that we would not have allocated to an sensitized patient in the deceased donor list

The results are described in Table 6.3.

6.3.4 Results analysis

As we can see in Table 6.1, Table 6.2 and Table 6.3 the effective number of additional transplantations corresponds to the number of incompatible pairs now transplanted. This number is 9 for the first two experiments and 8 for the third. Considering that the initial pool contains 29 incompatible pairs, we are transplanting now around the 31% and the 27% of the incompatible pairs. Furthermore we are transplanting without desensitization 16 patients in the first experiment

Number of:	Result
transplantation in the algorithm	14
transplantation in cycles	0
sensitized patient allocated by the algorithm	6
incompatible pairs now transplanted	8
allocated kidneys	7

Table 6.3: Third experiment results.

(76% of the pairs with desensitization) and 15 patients in the second experiment (68% of the pairs with desensitization). We are also transplanting 9 sensitized patient in the first two experiments and 6 in the third.

We expected results with less impact. The percentages of incompatible pairs now transplanted and of the pairs that do not need desensitization anymore, are strongly encouraging. These results are even more interesting considering the dimension of the considered pool. If we move to a national scenario, and thus not only local as the Padua context, the number of matchings surely increase since for each patient the probability of finding a compatible kidney increases.

Another unexpected fact is the high percentage of the highly-sensitized patients that are now allocated. This percentage is highly significant: the 42% of sensitized patient receive a kidney, that is near to the half of the sensitized pool. Since for this kind of patients have a very low probability to receive a compatible kidney waiting in the deceased-donor waiting list, this result will have a strong impact.

The use of kidney from deceased donors that we would have allocated to an sensitized patient in the deceased donor list, seems not affect the results. Only one kidney of this typology is used and the other results remains almost unchanged. Also the introduction of pairs that need desensitization seems not influence the other results, but this fact probably depends on the limited dimension of the dataset. Extending the analysis to a national context, the behavior of the system could change significantly.

6.4 Introduction of Preferences

The preferences of the patients are important in kidney transplantation. For example patients favor calculating waiting time from the initiation of dialysis therapy, rather than from time of listing; patients favor giving more weight to waiting time than to HLA matching; and patients do not favor giving preference to younger versus older recipients. Geddes et al. in [24] proved that patients do not agree with the current allocation criteria in the UK Transplant (UKT). Only 6.0% of the patients agreed with the current protocol in which the allocation to a patient not yet on dialysis therapy who had been on the transplant waiting list longer than a patient already on dialysis therapy. Only 24.6% of participants agreed with the usage of HLA matching discriminant with respect to the waiting time: a patient with a good HLA match that is waiting from 2 years in the list is advantaged respect to a patient with a bad match but waiting from 7 years. Patients also were opposed to the use of recipient age. The majority agreed that recipient sex should not be used to allocate kidneys and

allocation should favor recipients who have waited longer. In conclusion of this study they discover that patients disagreed with several aspects of current allocation systems.

Another problem that could be solved with the using of patient preferences are the discarding of a large amount of kidneys (more than 10% of cadaveric kidneys) because considered of marginal quality. Su et al. in [100] show that allowing to the patients to accept also this kind of kidney (following their preferences) cause an increase of 6% in QALY (quality-adjusted life expectancy that is an index of survival times estimated) , a 12% decrease in median waiting time, a 39% increase in the likelihood of transplantation, and a 56% reduction in the number of discarded kidneys.

Using the patients' preferences also can help to avoid the number of matching failure as described by Dickerson et al. [35] (they show that in the UNOS exchange program the 93% of matches fail in 2012). For instance, in the case of logistical problems, we can include the patients preferences about the location of the hospitals. Many chains are broken because a donor in a chain reneging (i.e., backing out after his patient received a kidney) [35]. We can estimate the risk aversion of each patient and the corresponding donor, to minimize these kind of situations. To this purpose we need to use effective frameworks to represent preferences and to be able to elicit the patients preferences.

In conclusion adding patients preferences could reduce the number of the feasible matchings, but minimizing the number of the failure we obtain an increase of the global number of the transplantations effectively performed. Considering instead the introduction of preferences about marginal kidneys we instead are increase the number of organs in the pool, allowing to use also kidneys that otherwise would have been discarded.

6.4.1 Future work

We plan to follow the models described above, adding preferences to the patients description in the algorithms. It is really important to understand which kind of preferences must taken into account. We decided to consider the risk aversion of the patients to estimate the typology of kidney that a patient could accept or not. To learn this parameter we are administering a questionnaire to a set of patients, asking questions related to the risk aversion (following the studies in [4, 5, 50]).

Moreover we plan to follow the approach introduced by Dickerson et al. in [35], using a probabilistic model to find the transplant plan with maximum expected value.

In the kidney exchange protocol we want also to add preferences regarding the minimum threshold for a patient for the received advantage in a matching. For example a compatible pair could become interested in the participation to the program if the program will provide a matching with a younger donor with also a better match HLA. We can also consider a set of different possible advantages and the patient provide a ranking of them. In this way we encourage a massive participation into the program and also we are minimizing the number of transplantation failure, with the meaning of number of rejected proposed matchings.

6.5 Summary and Discussion

In this chapter we analysed the kidney transplant protocol and how to improve it. There exists many possible improvements of the currently used algorithm proposed in the literature: incentivizing people to partecipate, maximizing the number of effective done transplantation, minimizing the failure of an exchange, or increasing the pool of possible kidneys allowing also marginal organs.

We proposed a first improvement method that is based on the idea of considering both cycles of exchanges and chains of exchanges which start from a deceased donor kidney. Usually chains start only from altruistic donors. We instead maximize the number of transplantations for each received deceased donor kidney, allowing to the chains to start from a deceased donor kidney.

We implemented the procedure using a simple ILP (integer linear programming) description of the matching algorithm, and we tested the procedure in the Padua pool of patients and patient/donor pairs. The results provided, show that we increase the number of incompatible pairs that are transplanted of around the 30%. Moreover we avoided desensitization to the 76% of the patient/donor pairs and we allocated the 42% of the highly-sensitized patients currently present in the Padua pool, patients having a very low probability to receive a compatible kidney.

We then introduced a second possible strategy to improve the currently used procedures, but only as a future direction. The basic idea is to consider the patient preferences in order to minimize the number of matching failure. For instance, in the case of logistical problems, we can include the patients preferences about the location of the hospitals. Moreover, many chains are broken because a donor in a chain reneges, thus we plan to estimate the risk aversion of each patient and the corresponding donor, to minimize these kind of situations. To this purpose we need to use effective frameworks to represent and manage preferences and to be able to elicit the patients preferences.

III Conclusions and Future work

7. Summary and Conclusions

7.1 PCP-net framework

We have defined and shown how to reason with a generalized version of CP-nets, called PCP-nets, which can model probabilistic uncertainty and be updated without recomputing their entire structure. PCP-nets can be seen as a way to bring together BNs and CP-nets, thus allowing to model preference and probability information in one unified structure. We have studied how to reason with these new structures in terms of optimality and dominance, the two main task regarding conditional preferences. We have defined and compared different notions of optimality: the most probable induced CP-net, the most probable optimal outcome and the optimal outcome of the most probable induced CP-net. We are also able to perform exact dominance, but, since this task is computationally hard we have defined also an algorithm that compute an approximate value for the dominance. This procedure is computable in polynomial time and we have proved experimentally that the result is really close to the exact value.

We have defined a procedure for learning a PCP-net in the preliminary case of separable dependency structure.

We have studied also a generalization of PCP-nets, using a different notion of probability: the Dempster-Shafer theory of probability.

7.2 Multi-agent context

We have also evaluated the use of PCP-nets as a compact representation language for the preferences of a set of agents. Starting from a profile of individual CP-nets, we introduced and evaluated two aggregation methods for the definition of a PCP-net, a first one based on relative frequencies of pairwise preferences (*PR*) and a second one based on the exact distribution of CP-nets (*LS*). Our theoretical and experimental results suggest that using the *PR* method in the input profile to

construct a PCP-net is accurate with respect to answering both optimality and dominance queries. Since optimality queries under this method can be shown to be equivalent to performing sequential voting, our proposed aggregation method is a direct generalisation of this setting. By generating a compact representation of the full preference profile using PCP-nets, we are also able to perform either exact or approximate dominance reasoning on a profile of individual CP-nets. Moreover, for the case of polytree PCP-nets, we showed that our proposed approximation technique for dominance yield results that are very close to the probability of dominance in the initial profile of individual CP-nets.

We have considered also an alternative way to represents preferences in a multi-agent contexts: profiles of users described by fuzzy profiles of soft constraints. We have studied a sequential preference aggregation procedure based on voting rules for settings where several agents express their preferences over a common set of variable assignments via soft constraints. We have studied this approach by providing both theoretical and experimental results. In particular, we have shown several theoretical properties of the approach and we have evaluated its performance experimentally by considering several widely used voting rules.

7.3 Logical formulation

Since CP-nets and its generalizations/variants are limited in expressiveness and are specialized formalisms with their own ad hoc syntax and semantics, we have presented a new unification theory for qualitative conditional preferences, namely LCP-theories. The idea behind the LCP-theories new framework is based on expressing preferences using Datalog. We have proved that existing conditional preference model as CP-nets, CP-theories, GCP, comparative preferences, are all particular cases of LCP theories. We also have shown that LCP-theory is a powerful framework since, in addiction to unify the state-of-the art formalisms for conditional preferences, it also extend them: LCP-theories introduce the new notion of recursive conditional preferences. We have shown how dominance, consistency and optimality queries (that are the main tasks regarding conditional preferences) can be formulated directly in Datalog and implemented in modern tabled Prolog systems such as XSB Prolog. We have also analyzed the complexity of the different algorithms, developed efficient procedures for some common use cases, and implemented a translator that exploits these algorithms.

We have studied three new notion of dominance for CP-nets computationally more efficient. We have analyzed the properties of these new semantics and we have compared them with the classical formulation of dominance.

7.4 Kidney exchange

We have studied the real-life scenario of kidney exchanges.

We have proposed an innovative procedure that starts the chains of exchanges from a deceased donor kidney contra the existing procedures that use only altruistic donors for the chains. Our procedure also encourage compatible pairs, that need desensitization, to participate into the program,

ensuring a sure improvement: a matching that not needs desensitization. We have provided some encouraging results performed in the context of the Padua pool.

Then we have analysed how to improve the existing algorithms to increase the number of transplantations and the expected life duration. We have studied related works on how to incorporate preferences in the matching procedures currently used and we have proposed a plan to minimize the transplantation failure (the number of rejected proposed matchings).

8. Future directions

In this chapter we discuss some future directions of research, both from a theoretical and an application point of view.

8.1 Breaking the independence assumption in PCP-nets

Conventionally, in the context of conditional preferences, authors usually consider the dependencies only in the preference graph (i.e. we consider dependencies between the features), they instead consider the cp-statements independent to each other [9, 28]. This means, in our scenario of PCP-nets, that the preferences of an user over a feature, given an assignment to the parents nodes, are independent to her preferences over another feature or over the same feature with another assignment of the parent nodes. We plan to study the case in which the cp-statements are not independent.

In our definition of PCP-net, this independence assumption appears prominent in the structure of the *G-net* (see Section 3.4.1). The *G-net* structure expresses the relation between the cp-statements and, in our formulation, has a separable graph, without edges (contra the graph of the PCP-net that have edges between the features). Thus, in PCP-nets, we are considering all the cp-statements independent. Also the probability of an induced CP-net is computed using this independence assumption, computing the joint probability of an assignment of the separable structured *G-net*. For this reason, an exact aggregation of a profile of CP-nets the most of the times is not possible (see Theorem 4.1).

Therefore a possible generalization of PCP-nets follows from the idea of removing this independence assumption, converting the structure of the *G-net* to a not necessarily separable graph.

8.2 Generalizations and extensions of LCP-theories

We have developed LCP-theories as an unification framework for conditional preferences based on a logical description written directly in Datalog. However our model still has some drawbacks: it does not support uncertainty, probabilistic information, and the changing of the preferences over time.

8.2.1 A probabilistic logical framework for conditional preferences

LCP-theories model has the drawback of not supporting uncertainty and probabilistic information. Our idea is to introduce probabilities in this framework in a union between PCP-net and LCP-theories. We plan to define a new framework, *PLCP-theories*, using an extension of Datalog that supports probabilities. We focus on *ProbLog* [86], that similar to Datalog with the difference that some of the facts are annotated with probabilities. ProbLog is based on a conversion of programs, queries and evidence to weighted Boolean formulas [48].

In our context the idea is to associate with each `d/2` fact a probability. Considering the Example 5.1 in Section 5.2, we will have for example the following probabilistic formulation:

```
soup(fish). soup(veg). wine(white). wine(red).
outcome(o(X,Y)):- soup(X), wine(Y), (X\== veg; Y\==red).
0.5::d(o(fish, X), o(veg, X)).
0.6::d(o(fish, white),o(fish, red)).
```

In this way we produce a semantics defined in terms of a probability distribution of the facts over possible worlds.

We plan to analyse this direction and to study the computational complexity of the main tasks in this scenario: optimization (about outcome or about users description) and dominance.

We hope also to use the implemented LCP system in real-life applications to determine the adequacy of the system, and consequently to implement PLCP-theories framework and test it in real-life scenarios.

8.2.2 A dynamic logical framework for conditional preferences

Real life scenarios are often dynamic, users may change their preferences over time. LCP-theories can not handle this kind of informations, thus we need a structure that can updates information in a computationally tractable costs.

LCP-theories are not dynamic because the number of features is fixed. Since we specify preferences as a clauses set, we can only add preferences to our model and we can not replace or delete informations.

We have a dynamic component in our model, but it is produced only by the effects of evidence (on the theory dependency graph). When we are computing a task (optimality, dominance or consistency) and we receive some evidence (assignment of a subset of features) we compute a new dependency graph (that changes dynamically) taking in account the provided information. In this way the computational complexity of the tasks could decrease, since we could move from a general dependency structure to a particular case, such as trees or acyclic graphs.

8.3 Kidney exchange

Considering the kidney exchange scenario, we plan to follow two main research directions:

1. we plan to provide an alternative algorithm to NITK3 (currently used in the Padua) to improve the quality of the kidney allocation from cadavers, maximizing the number of transplants and increasing the average expected life, by considering patients' preferences;
2. we also plan to show with a simulation, that the insertion of the patients in the kidney exchange program, produce not only an increase of transplants but also an increase of the expectation of life and quality of the transplant. We also plan to implement the patient preferences in the kidney exchange protocol to prove that in a scenario simulated following the Padua context we effectively minimize the number of rejection.

Bibliography

[1] David J. Abraham, Avrim Blum, and Tuomas Sandholm. “Clearing Algorithms for Barter Exchange Markets: Enabling Nationwide Kidney Exchanges”. In: *Proceedings of the 8th ACM Conference on Electronic Commerce*. 2007 (cited on page 176).

[2] T.E. Allen, J. Goldsmith, and N. Mattei. “Counting, Ranking, and Randomly Generating CP-nets”. In: *Proc. of the 8th Multidisciplinary Workshop on Advances in Preference Handling (MPREF)*. 2014 (cited on pages 39, 88, 121, 167).

[3] S. de Amo, M.L.P. Bueno, G. Alves, and N.F. Silva. “CPrefMiner: An Algorithm for Mining User Contextual Preferences Based on Bayesian Networks.” In: *Proceedings of IEEE 24th International Conference on Tools with Artificial Intelligence, ICTAI 2012*. 2012 (cited on page 61).

[4] Steffen Andersen, Glenn W. Harrison, Morten I. Lau, and E. Elisabet Rutstrom. “Eliciting Risk and Time Preferences”. In: *Econometrica* (2008) (cited on page 185).

[5] James Andreoni and Charles Sprenger. “Risk Preferences Are Not Time Preferences”. In: *American Economic Review* (2012) (cited on page 185).

[6] K. J. Arrow, A. K. Sen, and K. Suzumura. *Handbook of Social Choice and Welfare.* North-Holland, 2002 (cited on pages 52–54, 135, 136).

[7] F. Bacchus and A.J. Grove. “Graphical models for preference and utility”. In: *Proceedings of UAI 1995*. 1995, pages 3–10 (cited on page 16).

[8] C. Bessiere. “Constraint Propagation”. In: *Handbook of Constraint Programming*. Edited by P. Van Beek F. Rossi and T. Walsh. Elsevier, 2005 (cited on page 51).

[9] D. Bigot, H. Fargier, J. Mengin, and B. Zanuttini. "Probabilistic Conditional Preference Networks". In: *Proc. of the 29th International Conference on Uncertainty in Artificial Intelligence (UAI)*. 2013 (cited on pages 39, 62, 75, 76, 159–161, 164, 193).

[10] S. Bistarelli, U. Montanari, and F. Rossi. "Semiring-based constraint satisfaction and optimization". In: *Journal of the ACM (JACM)* (1997) (cited on page 16).

[11] C. Boutilier, R.I. Brafman, C. Domshlak, H.H. Hoos, and D. Poole. "CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements". In: *Journal of Artificial Intelligence Research* 21 (2004), pages 135–191 (cited on pages 16, 21, 26, 34, 36, 38, 39, 47, 62, 76, 111, 156, 159–161, 164).

[12] C. Boutilier, I. Brafman, C. Domshlak, H. Hoos, and D Poole. "Preference-Based Constrained Optimization with CP-Nets". In: *Computational Intelligence* 20.2 (2004), pages 137–157 (cited on pages 16, 19, 26, 47, 48, 153).

[13] Craig Boutilier, Fahiem Bacchus, and Ronen I. Brafman. "UCP-Networks: A Directed Graphical Representation of Conditional Utilities". In: *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, University of Washington, Seattle, Washington, USA, August 2-5, 2001*. 2001, pages 56–64 (cited on pages 26, 44, 45).

[14] Sylvain Bouveret, Ulle Endriss, and Jérôme Lang. "Conditional Importance Networks: A Graphical Language for Representing Ordinal, Monotonic Preferences over Sets of Goods". In: *Proceedings of the 21st International Jont Conference on Artifical Intelligence*. IJCAI'09. 2009, pages 67–72 (cited on pages 26, 42).

[15] R. Brafman and C. Domshlak. "Introducing Variable Importance Tradeoffs into CP-Nets". In: *Proceedings of UAI '02*. 2002, pages 69–76 (cited on pages 26, 40).

[16] R. I. Brafman, F. Rossi, D. Salvagnin, K. B. Venable, and T. Walsh. "Finding the Next Solution in Constraint- and Preference-Based Knowledge Representation Formalisms". In: *Proceedings of KR 2010*. 2010 (cited on page 139).

[17] R.I. Brafman and C. Domshlak. "Preference handling-an introductory tutorial". In: *AI Magazine* 30.1 (2009), page 58 (cited on page 15).

[18] D. Braziunas and Craig Boutilier. "Elicitation of factored utilities." In: *AI Magazine* (2008) (cited on page 93).

[19] S. Ceri, G. Gottlob, and L. Tanca. "What You Always Wanted to Know About Datalog (And Never Dared to Ask)". In: *IEEE Trans. on Knowl. and Data Eng.* 1 (1989), pages 146–166 (cited on pages 19, 153).

[20] S. Ceri, G. Gottlob, and L. Tanca. "What You Always Wanted to Know About Datalog (And Never Dared to Ask)". In: *IEEE Transactions on Knowledge and Data Engineering* 1.1 (1989), pages 146–166. ISSN: 1041-4347. DOI: `http://doi.ieeecomputersociety.org/10.1109/69.43410` (cited on page 19).

[21] Y. Chevaleyre, U. Endriss, J. Lang, and N. Maudet. “Preference handling in combinatorial domains: From AI to social choice”. In: *AI Magazine* 29.4 (2008), page 37 (cited on page 15).

[22] Yann Chevaleyre, Frédéric Koriche, Jérôme Lang, Jérôme Mengin, and Bruno Zanuttini. “Preference Learning”. In: Springer Berlin Heidelberg, 2011. Chapter Learning Ordinal Preferences on Multiattribute Domains: The Case of CP-nets (cited on pages 94, 97, 106).

[23] Arthur Choi, Hei Chan, and Adnan Darwiche. “On Bayesian network approximation by edge deletion”. In: *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*. 2005 (cited on page 126).

[24] Christopher Smith Colin C. Geddes R. Stuart C. Rodger and Anita Ganai. “Allocation of Deceased Donor Kidneys for Transplantation: Opinions of Patients With CKD”. In: *American Journal of Kidney Diseases* (2005) (cited on page 184).

[25] V. Conitzer, T. Sandholm, and J. Lang. “When are elections with few candidates hard to manipulate”. In: *JACM* 54.3 (2007), pages 1–33 (cited on page 136).

[26] V. Conitzer and L. Xia. “Paradoxes of Multiple Elections: An Approximation Approach”. In: *Proceedings of KR 2012*. 2012 (cited on pages 53, 127, 152).

[27] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. Edited by 3rd. The MIT Press, 2009 (cited on page 124).

[28] C. Cornelio, J. Goldsmith, N. Mattei, F. Rossi, and K.B. Venable. “Updates and Uncertainty in CP-nets”. In: *Proc. of the 26th Australasian Joint Conference on Artificial Intelligence (AUSAI)*. 2013 (cited on pages 62, 193).

[29] B. D’Ambrosio. “Inference in Bayesian Networks”. In: *AI Magazine* 20.2 (1999), page 21 (cited on pages 27, 31, 33).

[30] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. “Complexity and Expressive Power of Logic Programming”. In: *ACM Comput. Surv.* 33.3 (Sept. 2001), pages 374–425. ISSN: 0360-0300. DOI: 10.1145/502807.502810. URL: http://doi.acm.org/10.1145/502807.502810 (cited on page 153).

[31] R. Dechter. “Bucket elimination: A unifying framework for reasoning”. In: *Artificial Intelligence* 113.1-2 (1999), pages 41–85 (cited on pages 27, 31–33).

[32] R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003 (cited on page 33).

[33] R. Dechter. “Tractable Structures for CSPs”. In: *Handbook of Constraint Programming*. Edited by P. Van Beek F. Rossi and T. Walsh. Elsevier, 2005 (cited on page 51).

[34] John P. Dickerson, Ariel D. Procaccia, and Tuomas Sandholm. “Optimizing kidney exchange with transplant chains: theory and reality.” In: *Proceedings of AAMAS-12*. 2012 (cited on page 177).

[35] John P. Dickerson, Ariel D. Procaccia, and Tuomas Sandholm. “Failure-aware kidney exchange”. In: *Proceedings of ACM Conference on Electronic Commerce, EC ’13, Philadelphia, PA, USA,* 2013 (cited on pages 176, 177, 185).

[36] John P. Dickerson, Ariel D. Procaccia, and Tuomas Sandholm. “Price of Fairness in Kidney Exchange”. In: *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014)*. 2014 (cited on page 177).

[37] Y. Dimopoulos, L. Michael, and F. Athienitou. “Ceteris paribus preference elicitation with predictive guarantees.” In: *Proceedings of IJCAI-09*. 2009 (cited on page 94).

[38] J. Dombi, Imreh C., and Vincze N. “Learning lexicographic orders.” In: *Eur. J. of Operational Research* (2007) (cited on page 94).

[39] C. Domshlak and R.I. Brafman. “CP-nets: Reasoning and consistency testing”. In: *Proc. 8th International Conference on Principles of Knowledge Representation and Reasoning (KR)*. 2002 (cited on pages 15, 47).

[40] C. Domshlak and T. Joachims. “Efficient and non-parametric reasoning over user preferences.” In: *Proceedings of User Modeling and User Adapted Interaction (UMUAI)*. 2007 (cited on page 93).

[41] C. Domshlak, F. Rossi, K.B. Venable, and T. Walsh. “Reasoning about soft constraints and conditional preferences: complexity results and approximation techniques”. In: *Proc. of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*. 2003 (cited on pages 15, 26, 46).

[42] C. Domshlak, S. Prestwich, F. Rossi, K. Venable, and T. Walsh. “Hard and soft constraints for reasoning about qualitative conditional preferences”. In: *J. Heuristics* 12.4-5 (2006), pages 263–285 (cited on pages 16, 19, 153).

[43] Alan Eckhardt and Peter Vojtáš. “How to learn fuzzy user preferences with variable objectives.” In: *Proceedings of IFSA/EUSFLAT*. 2009 (cited on page 94).

[44] Alan Eckhardt and Peter Vojtáš. “Learning user preferences for 2CP-regression for a recommender system.” In: *Proceedings of SOFSEM*. 2010 (cited on page 94).

[45] P. Faliszewski, E. Hemaspaandra, and L. A. Hemaspaandra. “How Hard Is Bribery in Elections?” In: *JAIR* 35 (2009), pages 485–532 (cited on page 127).

[46] Boi Faltings, Marc Torrens, and Pearl Pu. “Solution generation with qualitative models of preferences”. In: *Computational Intelligence* 20.2 (2004), pages 246–263 (cited on pages 17, 113).

[47] T. Feder and M. Vardi. “The Computational Structure of Monotone Monadic SNP and Constraint Satisfaction: A Study Through Datalog and Group Theory”. In: *SIAM J. Comput.* 28.1 (1999), pages 57–104 (cited on page 153).

[48] Daan Fierens, Guy Van den Broeck, Ingo Thon, Bernd Gutmann, and Luc De Raedt. “Inference in Probabilistic Logic Programs using Weighted CNF’s”. In: *UAI 2011, Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*. 2011 (cited on page 194).

[49] J. Fürnkranz and E. Hüllermeier. *Preference Learning: An Introduction*. Springer, 2010 (cited on pages 15, 16).

[50] Alex Imas Gary Charness Uri Gneezy. “Experimental methods: Eliciting risk preferences”. In: *Journal of Economic Behavior & Organization* (2013) (cited on page 185).

[51] J. Goldsmith and U. Junker. “Preference handling for artificial intelligence”. In: *AI Magazine* 29.4 (2009) (cited on page 15).

[52] J. Goldsmith, J. Lang, M. Truszczynski, and N. Wilson. “The Computational Complexity of Dominance and Consistency in CP-nets”. In: *Journal of Artificial Intelligence Research* 33.1 (2008), pages 403–432 (cited on pages 15, 16, 21, 26, 38, 39, 46, 47, 76, 153, 159, 161, 164, 165).

[53] C. Gonzales and P. Perny. “GAI Networks for Utility Elicitation.” In: *Proc. of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR)*. 2004 (cited on pages 16, 93).

[54] C. Gonzales, P. Perny, and S. Queiroz. “Preference Aggregation with Graphical Utility Models”. In: *Proceedings of AAAI 2008*. 2008, pages 1037–1042 (cited on page 16).

[55] G. Gottlob and C. Papadimitriou. “On the Complexity of Single-rule Datalog Queries”. In: *Inf. Comput.* 183.1 (May 2003), pages 104–122. ISSN: 0890-5401. DOI: 10.1016/S0890-5401(03)00012-9. URL: http://dx.doi.org/10.1016/S0890-5401(03)00012-9 (cited on pages 153, 155, 160, 164, 165).

[56] Joshua T. Guerin, Thomas E. Allen, and Judy Goldsmith. “Learning CP-net Preferences Online from User Queries”. In: *Proceedings of AAAI-13, Twenty-Seventh AAAI Conference on Artificial Intelligence*. 2013 (cited on page 94).

[57] Bradley W. J., Hodge J. K., and Kilgour D.M. “Separable discrete preferences.” In: *Mathematical Social Sciences* (2005) (cited on pages 93, 106).

[58] S. Jung, J. Hong, and T. Kim. “A statistical model for user preference”. In: *Proceedings of TKDE*. 2005 (cited on page 60).

[59] P.C. Kanellakis, G.M. Kuper, and P.Z. Revesz. “Constraint Query Languages”. In: *Journal of Computer and System Sciences* 51.1 (1995), pages 26 –52 (cited on pages 19, 153).

[60] Frédéric Koriche and Bruno Zanuttini. “Learning Conditional Preference Networks with Queries”. In: *Proceedings of IJCAI-09, 21st International Joint Conference on Artificial Intelligence*. 2009 (cited on page 94).

[61] Chen L. and Pu P. *Survey of preference elicitation methods.* Technical report. 2004 (cited on page 93).

[62] Emerson M. S. Niou Lacy Dean. “A Problem with Referendums”. In: *Journal of TheoreticalPolitics* (2000) (cited on page 53).

[63] J. Lang. “Vote and aggregation in combinatorial domains with structured preferences.” In: *Proc. of the Twentieth International Joint Conference on Artificial Intelligence*. 2007 (cited on page 112).

[64] J. Lang, J. Mengin, and L. Xia. “Aggregating Conditionally Lexicographic Preferences on Multi-issue Domains”. In: *Proceedings of CP 2012*. 2012, pages 973–987 (cited on pages 127, 152).

[65] J. Lang and L. Xia. “Sequential composition of voting rules in multi-issue domains”. In: *Mathematical Social Sciences* 57.3 (2009), pages 304–324 (cited on pages 18, 19, 53, 110, 111, 117, 118, 127, 132, 134, 152).

[66] Jérôme Lang and Jérôme Mengin. “The Complexity of Learning Separable ceteris paribus Preferences”. In: *Proceedings of IJCAI-09, 21st International Joint Conference on Artificial Intelligence*. 2009 (cited on pages 94, 97, 106).

[67] M. Li, Q.B. Vo, and R. Kowalczyk. “An Efficient Procedure for Collective Decision-making with CP-nets”. In: *Proc. of the 19th European Conference on Artificial Intelligence (ECAI)*. 2010 (cited on page 18).

[68] M. Li, Q.B. Vo, and R. Kowalczyk. “Majority-rule-based preference aggregation on multi-attribute domains with CP-nets”. In: *Proc. of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2011 (cited on page 18).

[69] J. Liu, Y. Xiong, C. Wu, Z. Yao, and W. Liu. “Learning conditional preference networks from inconsistent examples.” In: *Proceedings of TKDE*. 2012 (cited on page 94).

[70] Thomas Lukasiewicz, Maria Vanina Martinez, and Gerardo I. Simari. “Probabilistic Preference Logic Networks”. In: *Proceedings of the 21st European Conference on Artificial Intelligence, ECAI-14*. 2014 (cited on page 61).

[71] Thomas Lukasiewicz, Maria Vanina Martinez, Gerardo I. Simari, and Oana Tifrea-Marciuska. “Preference-Based Query Answering in Probabilistic Datalog+/–Â Ontologies.” In: *Journal on Data Semantics* (2014) (cited on page 60).

[72] A. Maran, N. Maudet, M. S. Pini, F. Rossi, and K. B. Venable. “A Framework for Aggregating Influenced CP-Nets and Its Resistance to Bribery”. In: *Proceedings of AAAI-27*. 2013, pages 668–674 (cited on pages 18, 127, 153, 163).

[73] John I Marden. *Analyzing and Modeling Rank Data*. CRC Press, 1995 (cited on page 17).

[74] N. Mattei, M. S. Pini, F. Rossi, and K. B. Venable. “Bribery in Voting Over Combinatorial Domains Is Easy”. In: *Proc. 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2012 (cited on pages 18, 127).

[75] N. Mattei, M.S. Pini, F. Rossi, and K.B. Venable. “Bribery in voting with CP-nets”. In: *Annals of Mathematics and Artificial Intelligence* 68.1-3 (2013), pages 135–160 (cited on pages 18, 127).

[76] Nicholas Mattei and Toby Walsh. “PrefLib: A Library of Preference Data”. In: *Proceedings of ADT 2013*. Springer, 2013 (cited on page 149).

[77] N. Maudet, M. S. Pini, K. B. Venable, and F. Rossi. “Influence and aggregation of preferences over combinatorial domains”. In: *Proceedings of AAMAS 2012*. 2012, pages 1313–1314 (cited on page 127).

[78] P. Meseguer, F. Rossi, and T. Schiex. “Soft constraints”. In: *Handbook of Constraint Programming*. Edited by P. Van Beek F. Rossi and T. Walsh. Elsevier, 2005 (cited on pages 16, 26, 50, 51).

[79] S. Mittal and F. Frayman. “Toward a generic model of configuration tasks”. In: *Proceedings of IJCAI 1989*. 1989 (cited on page 15).

[80] H. Moulin. *Axioms of Cooperative Decision Making*. Cambridge University Press, 1991 (cited on pages 117, 118).

[81] Antonio Nicolò and Carmelo Rodríguez Álvarez. “Age-Based Preferences: Incorporating Compatible Pairs into Paired Kidney Exchange”. In: *Journal of Economic Theory* (2011) (cited on pages 177, 178).

[82] M. S. Pini, F. Rossi, and K. B. Venable. “Bribery in Voting With Soft Constraints”. In: *Proceedings of AAAI 2013*. 2013 (cited on page 127).

[83] M. S. Pini, F. Rossi, and K. B. Venable. “Resistance to bribery when aggregating soft constraints”. In: *Proceedings of AAMAS 2013*. 2013, pages 1301–1302 (cited on page 127).

[84] S. Prestwich, F. Rossi, K. B. Venable, and T. Walsh. “Constrained CP-nets”. In: *Proceedings of CSCLP,Äô04*. 2004 (cited on pages 19, 26, 47, 153).

[85] Robert Price and Paul R Messinger. “Optimal recommendation sets: Covering uncertainty over user preferences”. In: *Proc. 20th AAAI Conference on Artificial Intelligence*. 2005, pages 541–548 (cited on pages 17, 113).

[86] *ProbLog*. URL: `https://dtai.cs.kuleuven.be/problog/` (cited on page 194).

[87] K. Purrington and E. H. Durfee. “Making social choices from individuals’ CP-nets”. In: *Proceedings of AAMAS 2007*. 2007, pages 1122–1124 (cited on pages 127, 152).

[88] M. Regenwetter, J. Dana, and C.P. Davis-Stober. “Transitivity of preferences.” In: *Psychological Review* 118.1 (2011) (cited on page 17).

[89] Michel Regenwetter, Bernard Grofman, A. A. J. Marley, and Ilia Tsetlin. “Behavioral Social Choice: Probabilistic Models, Statistical Inference, and Applications.” In: Cambridge University Press., 2006 (cited on page 60).

[90] F. Rossi, K.B. Venable, and T. Walsh. “mCP nets: representing and reasoning with preferences of multiple agents”. In: *Proc. of the 19th AAAI Conference on Artificial Intelligence (AAAI)*. 2004 (cited on pages 18, 26, 45, 153).

[91] F. Rossi, K.B. Venable, and T. Walsh. *A Short Introduction to Preferences: Between Artificial Intelligence and Social Choice*. Morgan & Claypool Publishers, 2011 (cited on pages 15, 16).

[92] A.E. Roth and J.H. Kagel. *The handbook of experimental economics*. Volume 1. Princeton University Press Princeton, 1995 (cited on page 16).

[93] S. Russell and P. Norvig. *Artificial Intelligence - A Modern Approach*. Prentice Hall, 1994 (cited on page 27).

[94] M. Simmerling S. E. Gentry D. L. Segeva and R. A. Montgomery. “Expanding Kidney Paired Donation Through Participation by Compatible Pairs”. In: *American Journal of Transplantation* (2007) (cited on page 177).

[95] N. Wilson S. Moral. “Markov Chain Monte-Carlo Algorithms for the Calculation of Dempster-Shafer Belief”. In: *Proceedings of the Twelfth National Conference on Artificial Intelligence*. 1994 (cited on pages 31, 104).

[96] G. R. Santhanam, S. Basu, and V. Honavar. “Dominance Testing via Model Checking.” In: *Proc. of the 24th AAAI Conference on Artificial Intelligence (AAAI)*. 2010 (cited on page 18).

[97] T. Schiex. “Possibilistic Constraint Satisfaction Problems or "How to Handle Soft Constraints?"”. In: *Proceedings of UAI 1992*. 1992, pages 268–275 (cited on page 50).

[98] M. Schmitt and L. Martignon. “On the complexity of learning lexicographic strategies.” In: *J. Mach. Learn. Res.* (2006) (cited on page 94).

[99] Glenn Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976 (cited on page 100).

[100] Xuanming Su, Stefanos A. Zenios, and Glenn M. Chertow. “Incorporating Recipient Choice in Kidney Transplantation”. In: *Journal of the American Society of Nephrology* (2004) (cited on page 185).

[101] T. Swift and D. S. Warren. “Tabling with Answer Subsumption: Implementation, Applications and Performance”. In: *Proceedings of the 12th European Conference on Logics in Artificial Intelligence*. 2010, pages 300–312 (cited on page 154).

[102] T. Swift and D. S. Warren. “XSB: Extending Prolog with Tabled Logic Programming”. In: *Theory Pract. Log. Program.* 12.1-2 (2012) (cited on pages 154, 155).

[103] T. Swift and D. S. Warren. *XSB Home page*. `http://http://xsb.sourceforge.net/`. 2012 (cited on page 167).

[104] A.D. Taylor. *Social Choice and the Mathematics of Manipulation*. Cambridge University Press, 2005 (cited on page 53).

[105] D. Toman. “Memoing Evaluation for Constraint Extensions of Datalog”. In: *Constraints* 2 (1998) (cited on pages 19, 153).

[106] T.T. Truyen, D.Q. Phung, and S. Venkatesh. “Preference networks: Probabilistic models for recommendation systems”. In: *Proceedings of AusDM*. 2007 (cited on page 60).

[107] *UNOS. United Network for Organ Sharing (UNOS).* http://www.unos.org/ (cited on page 175).

[108] Ha V. and Haddawy P. “Problem-focused incremental elicitation of multi-attribute utility models.” In: *Proceedings of UAI-97*. 1997 (cited on page 93).

[109] M. Vardi. “The complexity of relational query languages (extended abstract”. In: *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*. 1982, pages 137–146 (cited on pages 153, 155).

[110] P. Viappiani, B. Faltings, and P. Pu. “Preference-based search using example-critiquing with suggestions.” In: *JAIR* (2006) (cited on page 93).

[111] N. Wilson. “Extending CP-Nets with Stronger Conditional Preference Statements”. In: *Proceedings of AAAI-04*. 2004, pages 735–741 (cited on pages 16, 21, 26, 48, 153, 156, 159, 161, 162, 164).

[112] N. Wilson. “Efficient Inference for Expressive Comparative Preference Languages”. In: *Proceedings of IJCAI-09*. 2009 (cited on pages 16, 21, 26, 41, 42, 49, 153).

[113] L. Xia and V. Conitzer. “Strategy-Proof Voting Rules over Multi-issue Domains with Restricted Preferences”. In: *Proceedings of WINE 2010*. 2010, pages 402–414 (cited on page 136).

[114] L. Xia, V. Conitzer, and J. Lang. “Voting on Multiattribute Domains with Cyclic Preferential Dependencies.” In: *Proc. of the 23rd AAAI Conference on Artificial Intelligence (AAAI)*. 2008 (cited on pages 18, 127, 152).

[115] L. Xia, V. Conitzer, and J. Lang. “Aggregating Preferences in Multi-Issue Domains by Using Maximum Likelihood Estimators”. In: *Proc. of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2010 (cited on pages 127, 152).

[116] L. Xia, V. Conitzer, and J. Lang. “Hypercubewise Preference Aggregation in Multi-issue Domains”. In: *Proc. of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*. 2011 (cited on page 18).

[117] F. Yaman, T. Walsh, M. Littman, and M. desJardins. “Democratic approximation of lexicographic preference models.” In: *Proceedings of ICML-08*. 2008 (cited on page 94).