Journal of Complex Networks, 2025, **6**, cnaf042
https://doi.org/10.1093/comnet/cnaf042
Advance access publication 7 November 2025
Article

An efficient network-aware direct search method for influence maximization

Matteo Bergamaschi [1,*], Sara Venturini [2], Francesco Tudisco[3], Francesco Rinaldi[1]

[1]Department of Mathematics "Tullio Levi-Civita", University of Padua, Via Trieste, 63, Padova, 35121, Italy
[2]Senseable City Laboratory, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA, 02139, United States
[3]School of Mathematics, The University of Edinburgh, Edinburgh, EH93FD, United Kingdom

*Corresponding author. Department of Mathematics "Tullio Levi-Civita", University of Padua, Via Trieste, 63, Padova, 35121, Italy. E-mail: bergamas@math.unipd.it

ABSTRACT

Influence Maximization (IM) is a pivotal concept in social network analysis, involving the identification of influential nodes within a network to maximize the number of influenced nodes, and has a wide variety of applications that range from viral marketing and information dissemination to public health campaigns. IM can be modeled as a combinatorial optimization problem with a black-box objective function, where the goal is to select B seed nodes that maximize the expected influence spread. Direct search methods, which do not require gradient information, are well-suited for such problems. Unlike gradient-based approaches, direct search algorithms, in fact, only evaluate the objective function at a suitably chosen set of trial points around the current solution to guide the search process. However, these methods often suffer from scalability issues due to the high cost of function evaluations, especially when applied to combinatorial problems like IM. This work, therefore, proposes the Network-aware Direct Search (NaDS) method, an innovative direct search approach that integrates the network structure into its neighborhood formulation and is used to tackle a mixed-integer programming formulation of the IM problem, the so-called General Information Propagation model. We tested our method on large-scale networks, comparing it to existing state-of-the-art approaches for the IM problem, including direct search methods and various greedy techniques and heuristics. The results of the experiments empirically confirm the assumptions underlying NaDS, demonstrating that exploiting the graph structure of the IM problem in the algorithmic framework can significantly improve its computational efficiency in the considered context.

KEYWORDS: influence maximization; network analysis; derivative-free optimization.

1. INTRODUCTION

Influence Maximization (IM) is a key concept in social network analysis. It entails identifying a group of influential nodes (seeds) in a network, with the goal of maximizing the resulting number of influenced nodes [1–4]. This concept has numerous applications, one of the most recognized applications of IM is viral marketing [5], which involves a company trying to increase the adoption of a new product through the social networks of initial users. IM also forms the basis for numerous other applications, such as network monitoring [6], rumour control [7, 8], and social recommendation [9]. The two most popular models to simulate how influence spreads through

Received: 11 December 2024; **Accepted:** 28 September 2025

a network are the Independent Cascade (IC) Model and the Linear Threshold (LT) Model [10, 11]. In the IC model, each influenced node has a single chance to influence its neighbors, and if it succeeds, those neighbors are then activated and have a chance to influence their neighbors in subsequent rounds. Indeed, in the LT model, each node is influenced based on the fraction of its neighbors that are influenced, and a node becomes influenced if the weighted sum of its influenced neighbors exceeds a certain threshold.

IM is typically formulated as a combinatorial optimization problem. Given a budget of B (the number of seed nodes to be selected), the objective is to find the set of B nodes that maximizes the expected influence spread. Here, we use as our information propagation model the General Information Propagation (GIP) model presented in [12]. Such a model has the following features: it considers continuous variables, enables feedback between nodes, and encompasses two well-known variants of IC and LT models as special cases. Furthermore, the authors in [12] formulate the influence maximization problem as a mixed integer nonlinear programming problem with a black-box objective function (i.e. a function whose analytical expression is not available) and propose a direct search approach, called Customized Direct Search (CDS), to solve it. However, the method has computational limitations and does not fully exploit the structure of the graph under analysis. See Section 3 for more details.

Direct Search Methods are well-known and well-suited zeroth-order/derivative-free optimization algorithms, and hence do not rely on derivative information to drive the optimization process (see, e.g. [13–16] and references therein for further details on derivative-free and zeroth-order optimization algorithms). Instead, they make use of function evaluations to guide such a process, making them well suited for non-smooth, noisy, or black-box optimization problems where derivative information is unavailable or unreliable. In particular, they sample the objective function at a finite number of points at each iteration and decide which actions to take next solely based on those function values, either by function value comparison through ranking or based on numerical values and without any explicit or implicit derivative approximation or model building. These methods are, hence, particularly effective in problems where gradient estimation is computationally expensive or not available, such as IM. Since function evaluations can be time-consuming and resource-intensive, direct search methods, however, lack scalability and are computationally inefficient when dealing with large-scale problems. Therefore, balancing the trade-off between exploration and computational cost is crucial for practical application.

In order to overcome these limitations, in this work, we propose the Network-aware Direct Search (NaDS) method, an innovative direct search approach that exploits the network structure knowledge thus reducing the computational effort to find good solutions. This is possible thanks to a novel neighborhood definition based on the connections in the graph related to the current iterate. We provide extensive numerical tests that empirically demonstrate the better efficiency of our NaDS method, showing that direct search methods can tackle large-scale networks with thousands of nodes, outperforming baseline approaches in terms of influence score and computational time.

The remainder of the paper is organized as follows. In Section 2, we discuss the existing methods for the IM problem. In Section 3, we present the problem, focusing on the mathematical formulation, and introduce the CDS method, which is the main direct search method we use for comparison. In Section 4, we present our novel direct search method NaDS. Finally, in Section 5, we discuss numerical experiments, and in Section 6, we draw some conclusions and discuss possible future work.

2. RELATED WORK

The IM problem gained significant attention due to its applications in marketing, information dissemination, and the spread of health behaviors [17–21]. Due to its large number of applications and complex optimization formulation, it presents many research challenges. On the theoretical modelling side, there are different ways to define a diffusion model that captures information diffusion. On the algorithm side, solving IM optimally is proven to be NP-hard in most diffusion

models. Its complexity extends to the evaluation of influence spread from any given seed set, which is also computationally intensive. Consequently, much of the existing research in this area is dedicated to finding approximate solutions. Over the past decade, a variety of models and methods have been proposed to tackle this problem, and there are available comprehensive surveys that explore different facets of these advancements [22–28]. In the following, using the taxonomy proposed in [25], we attempt to review and summarize some of these models and algorithms.

Different models employ distinct mechanisms for capturing how a user transitions from an inactive to an active status due to their neighbors' influence. Diffusion models can be broadly categorized into progressive and non-progressive diffusion models, depending on whether activated nodes can be deactivated in subsequent steps or not, respectively. Examples of typical non-progressive models include the SIR/SIS model [29] and the Voter model [30]. We will focus on the first class of progressive models and we will briefly present the three most popular models: the Independent Cascade (IC) model, the Linear Threshold (LT) model, and the Triggering (TR) model.

All these three models are diffusion models based on stochastic processes where information propagates from one node to its neighbors at each time step according to different probabilistic rules. In the IC model [31], each activated node has a fixed probability of influencing its neighbors in each time step. Once a node is activated, it cannot be deactivated, and each neighbor's activation is independent of other neighbors' activations. In the LT model [32, 33], each node has a threshold that must be exceeded by the sum of the influence from its neighbors to become active. Each node is activated based on the weighted sum of its neighbors' states compared to its own threshold. In the TR model [10], each node independently selects a random "triggering set" from its neighbors based on a specified distribution, and a node becomes active if it has a neighbor in its chosen triggering set that is active.

These foundational models have paved the way for numerous extensions and adaptations in the literature. For instance, IC, LT, and TR are time-unaware models where diffusion stops when no more nodes can activate. However, real-world propagation campaigns often require maximizing influence within time constraints. This has led to the development of Time-Aware Diffusion Models, categorized as discrete-time [34–36] and continuous-time models [37, 38]. In particular, in this work, we refer to [12], which introduces the GIP model, which extends the IC and LT to incorporate continuous variables and dynamic feedback mechanisms (see Section 3.2).

State-of-the-art methods for IM include greedy algorithms, which have been shown to be effective since the seminal work of Kempe *et al.* [10]. These methods iteratively select nodes that provide the largest marginal gain in influence spread. They can be categorized into three primary categories depending on their goals: simulation-based approaches involve Monte-Carlo simulations to evaluate the influence spread and gain model generality [10]; proxy-based approaches utilize proxy models to approximate the influence function obtaining practical efficiency [18, 39–42]; sketch-based approaches focus on theoretical efficiency by initially constructing theoretically grounded sketches under a diffusion model, which are subsequently used to expedite the evaluation of the influence function [43–46]. In addition, methods based on percolation processes or on the concept of k-core centrality have also been proposed as alternatives for identifying influential nodes [47–50].

3. PROBLEM SETTING

We define our social network starting from a graph, denoted as $\mathcal{G}(\mathcal{N}, \mathcal{E}, W)$, which is undirected, connected, and weighted. Here, $\mathcal{N} = \{1, 2, \ldots, n\}$ represents the set of nodes, and $\mathcal{E} = \{(i, j) \mid$ there exists a connection from node$i$to node$j\}$ represents the set of edges.

The weight matrix $W = [W_{ij}]$ encodes the strength or level of trust between nodes, where each weight satisfies $W_{ij} \geq 0$, $W_{ij} \in \mathbb{R}$. Specifically, $W_{ij} > 0$ if $(i, j) \in \mathcal{E}$, and $W_{ij} = 0$ otherwise, indicating the absence of a direct connection. In this context, we employ the notation $x_i(t)$ to denote the state of node i at discrete time step t.

Following [12], in this section we formalize the influence maximization (IM) problem as well as the associated GIP model, and we review the CDS method which we will then use as a baseline for our improved NaDS scheme.

3.1. Problem formulation

The IM problem revolves around the objective of maximizing the collective impact on network nodes at the culmination of the propagation process. Our particular focus lies on an essential constraint: the initiation of a finite number of nodes determined by a budget size. This constraint represents the notion of allocating limited resources to influence a wider audience.

Given a defined information propagation process and an associated function $s_j = \sum_{t=0}^{\infty}(1-\gamma)^t x_j(t)$ gauging the overall influence on each node j, the cumulative influence across the network emerges naturally as:

$$s(x) = \sum_{j=1}^{n} s_j(x(0)), \tag{3.1}$$

where $x(0)$ represents the initial state vector. Consequently, the IM problem revolves around maximizing $s(x)$, while ensuring adherence to the constraint of activating a limited number of nodes:

$$|\{j : x_j(0) > 0\}| \leq B, \tag{3.2}$$

where $B \in \mathbb{N}$ represents the budget size.

Based on the objective function (3.1) and constraint (3.2), the IM problem can be formulated as a mixed-integer nonlinear program (MINLP):

$$\begin{aligned} &\max_{x,z} s(x) \\ &s.t. x_j \leq h_{j,0}\, z_j, \\ &\quad x_j \geq l_{j,0}\, z_j, \\ &\quad \sum_j z_j \leq B, \\ &\quad x_j \in \mathbb{R}, \quad z_j \in \{0,1\}, \quad \forall j \in [n], \end{aligned}$$

where $0 < l_{j,0} \leq h_{j,0}$ define a lower and an upper bound on the initial influence level of node j. The objective function $s(\cdot)$ measures the aggregate influence across the network, as illustrated in (3.1). The vector x comprises initial state values, while vector z, of the same dimensions, signifies the decision to set positive initial state values ($z_j = 1$) or not ($z_j = 0$), thus enforcing the constraint (3.2).

Taking advantage of the non-decreasing nature of $s(x)$, the task of maximizing the objective $s(x)$ with respect to both x and z in the MINLP (3.3) can be simplified to the task of maximizing $s(x)$ while x and z are set to their highest attainable values. Specifically, we set $x_j = h_{j,0}\, z_j$ and $\sum_j z_j = B$, effectively reducing the problem to the following form with respect to the binary vector z:

$$\begin{aligned} &\max_{z} s(\mathbf{h_0} \odot z) \\ &s.t. \sum_j z_j = B, \\ &\quad z_j \in \{0,1\}, \quad \forall j, \end{aligned} \tag{3.4}$$

where $\mathbf{h_0} = (h_{j,0})$ and $\odot$ denotes the element-wise Hadamard product. As a result, the domain Ω^B becomes a natural mesh [1] for searching at every iteration of the algorithm.

$$\Omega^B = \{z \in \{0,1\}^n : \sum_j z_j = B\}. \tag{3.5}$$

The constraints are inherently embedded within the domain, and they are managed using the extreme barrier approach s_{Ω^B}, where $s_{\Omega^B}(z) = s(\mathbf{h_0} \odot z)$ if $z \in \Omega^B$ and $-\infty$ otherwise. We define the neighborhood function for binary variables z as:

$$N(z,d) = \{y \in \Omega^B : \|y - z\|_1 \leq d\}, \tag{3.6}$$

where $d \in \mathbb{N} \setminus \{1\}$, as the L_1 distance between y and z can reach a minimum of 2 when $y \neq z$ and $y, z \in \Omega^B$. This minimal distance of 2 is achieved when only one element of positive value is exchanged with an element of value 0.

Given $d \in \mathbb{N} \setminus \{1\}$, we can now formally define a d-local maximum of Problem (3.4) as a point z^* satisfying the following condition:

$$s_{\Omega^B}(z^*) \geq s_{\Omega^B}(z), \quad \forall z \in N(z^*, d). \tag{3.7}$$

3.2. The GIP model

Here, we review the General Information Propagation (GIP) model proposed in [12], which unifies the fundamental mechanisms underlying the two classic propagation models. This model encompasses two primary aspects:

(i) Each node i independently attempts to influence its neighbors, proportional to the edge weight and its current state value $x_i(t)$. This characteristic aligns with the principles of the IC model.
(ii) The actual impact on each node j stems from the collective behavior of its entire neighborhood.

The latter point is achieved by applying a nonlinear transformation to

$$y_j(t) = \sum_i W_{ij}\, x_i(t-1), \tag{3.8}$$

enabling us to capture how the cumulative influence attempts from all neighbors translate into a state change for node j. Intuitively, this equation captures the idea that the current state or output $y_j(t)$ is influenced by a linear combination of prior inputs, where each input $x_i(t-1)$ contributes according to its associated weight W_{ij}. This concept draws parallels not only with the LT model but also with nonlinear models applied to opinion dynamics. Specifically, at each time step $t > 0$, we introduce a lower bound $l_{j,t}$ representing the threshold required to initiate propagation. This implies that $x_j(t) = 0$ if $y_j(t) < l_{j,t}$. Additionally, an upper bound $h_{j,t}$ accounts for the saturation effect, signifying that $x_j(t) = h_{j,t}$ if $y_j(t) \geq h_{j,t}$.

Overall, the GIP model can be described as a bounded discrete dynamical system:

$$x_j(t) = f_{j,t}\left[\sum_i W_{ij} x_i(t-1)\right], \quad \text{for all} t > 0,\ j \in V, \tag{3.9}$$

[1] In the context of mesh adaptive direct search methods, the mesh is a discrete subset of the parameter space that is adaptively refined or coarsened throughout the optimization process. It provides a framework for generating candidate points around the current best solution.

Algorithm 1 Propagation Algorithm for the GIP Model

Require: A undirected, connected, and weighted network $\mathscr{G}(\mathscr{N}, \mathscr{E}, W)$ with non-negative weights, parameters $\{l_{j,t}, h_{j,t}\}$ in the GIP model, time-discounting factor γ, initial state $x(0)$ where $x_j(0) \in [l_{j,0}, h_{j,0}]$ if and only if $j \in A_0$ (0 otherwise), and the tolerance ε.

Ensure: The value of the objective function in the MINLP, s.

Set $t \leftarrow 0$, $x(t) \leftarrow x(0)$, and $s \leftarrow 0$.
Set $N_t \leftarrow \bigcup_{j \in A_0} N^{\text{out}}(j)$, where $N^{\text{out}}(j) := \{k \in V \mid (j,k) \in E\}$.
while $|(1-\gamma)^t x(t)| > \varepsilon$ **do**
  $A_{t+1}, N_{t+1} \leftarrow \varnothing$, and $x(t+1) \leftarrow 0$.
  for each potentially activated node $j \in N_t$ **do**
    $x(t+1)_j \leftarrow f_{j,t}\left(\sum_{i \in A_t} W_{ij} x(t)_i\right)$.
    if $x(t+1)_j > 0$ **then**
      $A_{t+1} \leftarrow A_{t+1} \cup \{j\}$.
      $N_{t+1} \leftarrow N_{t+1} \cup N^{\text{out}}(j)$.
      $s \leftarrow s + (1-\gamma)^{t+1} x(t+1)_j$.
    end if
  end for
  $t \leftarrow t+1$.
end while

where $f_{j,t}(x)$ takes the form:

$$f_{j,t}(x) = \begin{cases} 0, & x < l_{j,t}, \\ x, & l_{j,t} \le x < h_{j,t}, \\ h_{j,t}, & x \ge h_{j,t}, \end{cases} \tag{3.10}$$

representing the time-dependent bounds for each node j. $\{l_{j,t}\}$ and $\{h_{j,t}\}$ denote the time-dependent lower and upper bounds for each node j respectively, where $0 \le l_{j,t} \le h_{j,t}$. These bound values offer flexibility in characterizing the underlying population. Furthermore, it is worth noting that the GIP model can replicate the classic models by setting specific bound values. The initial states $x(0)$ are predefined, with $x_j(0) \in \{0\} \cup [l_{j,0}, h_{j,0}]$ and $l_{j,0} > 0$.

Specifically we will use the following threshold-type bounds for $t > 0$ and $j \in V$:

$$\begin{aligned} l_{j,t} &= (\theta_{l,j}\alpha)^t l_{j,0}, \\ h_{j,t} &= \theta_{h,j}\,\theta_{l,j}^{t-1}\alpha^t h_{j,0}, \end{aligned} \tag{3.11}$$

where $\alpha = \frac{1}{|E|}\sum_{(i,j)\in E} W_{ij}$ is the average across edge weights, while $\theta_{l,j}$ and $\theta_{h,j}$ are the two parameters that characterize the propagation dynamics in the GIP model, as they decide the trend for the lower and the upper bound. In order to guarantee convergence of the propagation algorithm and ensure that the bounds tend to zero, it is required that $\theta_{l,j}\alpha < 1$.

For a detailed explanation of the propagation process, we refer to Algorithm 1.

3.3. CDS method

In [12], after introducing the GIP Model (see Section 3.2), the CDS method is proposed to solve it.

The algorithm starts with the node set that maximizes the Katz centrality [51], which represents an exact solution in certain GIP settings. At each iteration r, we start with a **SEARCH step** that suitably explores the points in the domain Ω^B in order to possibly find a new iterate that improves the objective function value. If this is the case, we directly switch to the **Termination check**. Otherwise, we perform a **POLL step**. Such a step explores the local neighborhood of the current

Algorithm 2 Customized Direct Search (CDS) Method

1: **Initialization:** Set $0 < \zeta, \delta < 1, d > 1$. Let $z(0) \in \Omega^B$ such that $z(0)_j = 1$ if node $j \in A = \{j_1, \ldots, j_k\}$ where $h_{i,0}\, c_i = h_{j,0}\, c_j, \forall i \notin A, j \in A$, and $c = \{[I - (1-\gamma)W]^{-1} - I\}\mathbf{1}$ is the Katz centrality. Set iteration $r = 0$.
2: **SEARCH step (optional):** Evaluate s_{Ω^B} on a finite subset of trial points on the domain Ω^B, until a sufficiently improved point z is found, where $s_{\Omega^B}(z) > (1+\zeta) s_{\Omega^B}(z^{(r)})$, or all points have been exhausted. If an improved point is found, then the SEARCH step may terminate; skip the next POLL step and go directly to step 4.
3: **POLL step:** Evaluate s_{Ω^B} on the set $N(z^{(r)}, d) \subset \Omega^B$ as in (3.6), until a sufficiently improved point z is found, where $s_{\Omega^B}(z) > (1+\zeta) s_{\Omega^B}(z^{(r)})$, or all points have been exhausted.
4: **Termination check:** If an improvement is found, set $z^{(r+1)}$ as the improved solution, while decreasing $\zeta \leftarrow \delta\zeta$ if a sufficient improvement has not been found, increment $r \leftarrow r+1$, and go to step 2. Otherwise, output the solution $z^{(r)}$.

candidate $z(r)$ until a point with a sufficient improvement in the objective value is found or all points have been examined. In the termination check, if an improved point is identified, the algorithm will start a new iteration but will decrease the required improvement if a sufficiently improved point has not been discovered. If no improvement is found, the algorithm outputs the current iterate and terminates. We refer to Algorithm 2 for a detailed description of steps.

Hence, the termination step directly guarantees convergence to local minima. Even though convergence to global optimal solutions could be achieved through a carefully developed search step and a better understanding of the objective function's landscape to avoid poor local optima, the drawback of global optimization methods is their time complexity. Thus, we maintain the search step as an optional component. The CDS method takes advantage of the problem's characteristics and mitigates worst-case complexity by initializing with an exact solution when the GIP model reaches the extreme of the EIC model. As conjectured in [12], the local optima in the vicinity of this particular solution are expected to provide sufficiently good values of the objective function.

From the current CDS method, two avenues for enhancing output quality emerge: (i) brute force, where all node sets of size k are evaluated to select the optimal one, and (ii) random sampling, which exhibits asymptotic global convergence if it samples densely enough. Two improvement strategies inspired by those methods can hence be used in our context. Firstly, by expanding the neighborhood definition, more points are searched within the feasible domain. If the neighborhood encompasses the entire domain, we get back to the brute-force method. Secondly, the search process (steps 2, 3, and 4 in Algorithm 2) can be restarted from other randomly generated unexplored points, mirroring the logic of the search step. This strategy achieves asymptotic convergence to global minima, akin to the random sampling approach.

4. NADS METHOD

In this section, we present the NaDS method, which takes advantage of the inherent structure of the network to reduce the computational cost of the exploration strategy, thus speeding up the search process in CDS.

In the context of direct search methods, a central challenge revolves around the selection of an appropriate neighborhood, a crucial factor that significantly influences the way solution space is explored. In the preceding section, we reviewed the CDS method, which employs a rather straightforward search strategy. Specifically, it populates the neighborhood with all points located at a specified L_1 distance, commonly set at 2. Moreover, the CDS method incorporates the graph structure by utilizing Katz centrality to guide the initial point selection. However, these measures do not adequately take into account the underlying graph structure. In this section, we introduce

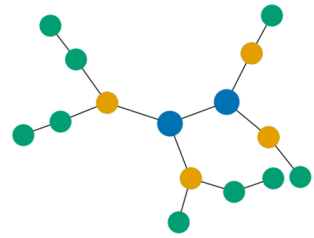

Figure 1. Simple example of how nodes are selected in CDS and NaDS. Given a set of activated nodes (in blue), CDS selects potential swap nodes from all the remaining nodes of the graph (in green and orange), while NaDS only considers the ones connected to the starting nodes (in orange).

a variation of CDS that directly exploits the network structure in the neighborhood construction phase. We call the resulting algorithm Network-aware Direct Search (NaDS) method.

The proposed NaDS method deviates primarily from CDS in its trial point selection. This new methodology aims to better leverage the underlying graph structure during neighborhood construction. The core notion is centered around a more conservative selection of trial points, emphasizing connections within the graph. The fundamental premise of the NaDS method is to exclusively consider points that are interconnected through graph edges. Unlike the CDS inclusive approach of exploring all points at a set L_1 distance, the NaDS method narrows down its focus to only those points that are directly linked through edges in the graph. More specifically, in the main poll step, NaDS filters the trial points through the set C, defined as follows:

$$C(z) = \{y \in \Omega^B \,|\, y_i = 1 \Rightarrow \exists\, j \text{ s.t. } (i,j) \in E \vee z_i = 1\}. \tag{4.1}$$

Here, $C(z)$ represents the set of points that can be constructed using only nodes that are either:

(i) already active in z (i.e. $z_i = 1$),
(ii) connected to an active node in z (i.e. $\exists\, j$ such that $(i,j) \in E$).

Figure 1 provides an example that illustrates the differences in neighborhood construction between CDS and NaDS.

The NaDS method operates within a more constrained neighborhood, leading to a reduction in the number of function evaluations required at each iteration. Importantly, this reduction in neighborhood size does not compromise the quality of the search. This is due to the fact that the NaDS method hones in solely on the most significant and meaningful points within the solution space.

This approach ensures a more focused exploration by considering points that are directly linked in the graph rather than exhaustively covering a broader L_1 distance as done by the CDS method. By deliberately selecting only interconnected points, the NaDS method strikes a balance between the efficiency and depth of the search.

In Algorithm 3, we present the detailed scheme of the NaDS method. In general, we do not start from a solution based on Katz centrality like CDS. In fact, our experiments showed that more effective and faster heuristics can be used as starting points, depending on the graph's characteristics and size—for example, Single Discount and Simple Greedy—as demonstrated in Section 5. Then, at each iteration r, we start with an optional **SEARCH step** that suitably explores the points in the mesh Ω^B in order to possibly find a new iterate that improves the objective function value. If this is the case, we directly switch to the **Termination check**; otherwise, we perform the **POLL step**.

In the first phase of the poll step, we explore the refined neighborhood defined in (4.1). This is the core of the poll step, and as our tests empirically confirm, almost every improving point is found in this phase. In case this phase cannot find an improving point, NaDS performs **two additional POLL**

Algorithm 3 Network-aware Direct Search (NaDS) Method

1: **Initialization:** Set $0 < \zeta, \delta < 1, d = 2, d_{max} > 1$. Let $z(0) \in \Omega^B$ be a suitably chosen starting point. Set iteration $r = 0$.
2: **SEARCH step (optional):** Evaluate s_{Ω^B} on a finite subset of trial points on the domain Ω^B, until a sufficiently improved point z is found, where $s_{\Omega^B}(z) > (1 + \zeta)s_{\Omega^B}(z^{(r)})$, or all points have been exhausted. If an improved point is found, then the SEARCH step may terminate, skip the next POLL step and go directly to step 6.
3: **POLL step (Phase 1):** Evaluate s_{Ω^B} on the set $N(z^{(r)}, d) \cap C(z^{(r)}) \subset \Omega^B$ as in (**??**) with distance d, until a sufficiently improved point z is found, where $s(\Omega^B)(z) > (1+\zeta)s(\Omega^B)(z^{(r)})$, or all points have been exhausted. $C(z^{(r)})$ is the set of points that are connected by an arc with $z^{(r)}$. If an improved point is found, then the POLL step may terminate, and go directly to step 6.
4: **POLL step (Phase 2):** Expand dynamically $N(z^{(r)}, d) \cap C(z^{(r)})$ to $N(z^{(r)}, d)$. If an improved point is found, then the POLL step may terminate, skip Phase 3, and go directly to step 6.
5: **POLL step (Phase 3 - Optional):** If $d < d_{max}$ then increase d.
6: **Termination check:** If an improvement is found, set $z^{(r+1)}$ as the improved solution, while decreasing $\zeta \leftarrow \delta\zeta$ if a sufficient improvement has not been found, increment $r \leftarrow r + 1$, and go to step 2. Otherwise, output the solution $z^{(r)}$.

step phases, which also ensure local convergence, as in the case of the CDS method. In the second phase, the refined neighborhood (4.1) is dynamically expanded to match the L_1 neighborhood defined in (3.6). We remark that in this phase, we set the distance d to the minimum, which is 2. The third phase consists of expanding the aforementioned neighborhood by increasing the distance d. In general, this third phase is optional due to the high computational cost it requires, but it is probably useful in smaller networks.

The local convergence of the NaDS method is guaranteed by construction since it operates within a specific neighborhood and will, therefore, find a local minimum within that area. Precisely, we have

Theorem 4.1. Let $\{z_k\}$ and $\{s_k\}$ be the sequences of solutions and reference values generated by NaDS. Then, the algorithm produces a d-local maximum for any fixed d in finite time.

Proof. Let $\{z_k\}$ and $\{s_k\}$ be the sequences of solutions and reference values generated by NaDS. Since z_k is the output of NaDS, the **POLL step** phase ensures that every point z_t in the neighborhood $N(z_k, d)$ has been evaluated, and there is no point such that $s_t > s_k$. This guarantees that z_k is a d-local maximum. To prove that NaDS produces the sequence $\{z_k\}$ in finite time, we observe that $s_0 < s_1 < \ldots < s_k$, which implies that NaDS does not visit previously evaluated points twice. Furthermore, since the bounded domain Ω^B contains a finite number of points, it follows that the sequence $\{z_k\}$ must also be finite. □

5. NUMERICAL EXPERIMENTS

In this section, we evaluate the NaDS method through extensive experiments, showing its superiority over existing approaches. Our results indeed show that NaDS consistently outperforms both the CDS method introduced in [12] and classic heuristics across all tested networks, achieving higher influence scores in 23 over 24 test cases. The network-aware neighborhood construction is particularly effective, reducing the search space by 35%–60%, while maintaining solution quality. The method shows robust performance across diverse network structures and budget constraints, from small collaboration networks to large social graphs. The following sections detail our experimental setup on six real-world networks (Table 1) with budgets $B \in \{5, 10, 15, 20\}$, comparing against five baseline methods while evaluating both influence spread and computational efficiency. We

Table 1. Basic statistics for the real-world datasets, including the reference, number of nodes, and number of edges for each dataset

Dataset	*Ref.*	No. of nodes	No. of edges
Arxiv Astro	[52]	18 772	198 110
Arxiv Gr-Qc	[52]	5242	14 496
Arxiv Hep-Ph	[52]	12 008	118 521
Lastfm-Asia	[53]	7624	27 806
email-Enron	[54, 55]	36 692	183 831
Facebook	[56]	4039	88 234

also conducted experiments on artificial graphs [57]; see Section A of the Appendix for a detailed overview.

The complete code for the implementation of methods and experiments is available at https://github.com/Berga53/Network-aware-Direct-Search. We acknowledge the use of the open-source libraries NetworkX [58], NumPy [59], SciPy [60], and Matplotlib [61] in the development of our methods and experiments.

Given that greedy algorithms and heuristics are widely used for this problem, we also incorporate several of these methods into our experiments. The objectives of these experiments are outlined below:

(i) to compare the performance of NaDS against the CDS method and show that NaDS achieves better results;
(ii) to illustrate that greedy methods are not suitable for every IM setting, and specifically, demonstrate that these methods lack guarantees and yield worse results in our particular setting;
(iii) to show that NaDS provides better and more reliable results compared to the best heuristics available in the literature for the IM problem.

The six networks used for our experiments details are reported in Table 1. We selected networks from various applications of the IM problem, which include the science of science (Arxiv Astro, Arxiv Gr-Qc, Arxiv Hep-Ph networks) and opinion dynamics (Lastfm-Asia, email-Enron, Facebook). All the datasets were selected from the SNAP dataset [62]. Below we give a brief description:

1. **Arxiv Astro** [52]: collaboration network from the e-print arXiv restricted to papers in the Astro Physics category. Nodes represent authors, and edges indicate collaborations between pairs of authors. The data covers papers in the period from January 1993 to April 2003.
2. **Arxiv Gr-Qc** [52]: collaboration network from the e-print arXiv restricted to papers in the General Relativity and Quantum Cosmology category. Nodes represent authors, and edges indicate collaborations between pairs of authors. The data covers papers in the period from January 1993 to April 2003.
3. **Arxiv Hep-Ph** [52]: collaboration network from the e-print arXiv restricted to papers in the High Energy Physics—Phenomenology category. Nodes represent authors, and edges indicate collaborations between pairs of authors. The data covers papers in the period from January 1993 to April 2003.
4. **Lastfm-Asia** [53]: social network from the LastFM music website. Nodes are LastFM users from Asian countries and edges are mutual follower relationships between them. The dataset was collected from the public API in March 2020.
5. **email-Enron** [54, 55]: communication network containing around half a million emails generated by employees of the Enron Corporation. This data was originally made public, and posted to the web, by the Federal Energy Regulatory Commission during its investigation.

Nodes represent email addresses, and edges indicate email communication between pairs of addresses.

6. **Facebook** [56]: social network consisting of "circles" (or "friends lists") from Facebook. The dataset was collected from survey participants using the Facebook app. Nodes represent Facebook users, and edges are mutual follower relationships between them.

The methods considered in our experiments are:

- **NaDS Method**: the direct search method proposed in this paper;
- **CDS Method** [12]: the direct search method presented in Section 3.3;
- Other Non-Direct Search Methods:
  - **Single Discount (SD)** [18]: a heuristic that selects nodes based on their centrality, adding a penalty for those that are connected to other high-centrality nodes.
  - **Simple Greedy (SG)** [10]: a classic greedy algorithm for IM, which iteratively selects nodes to maximize the spread of influence. At each step, it evaluates the marginal gain in influence spread for each candidate node and adds the node with the highest gain to the seed set.
  - **Katz Centrality (KC)** [12]: a heuristic that selects the nodes with the highest value of Katz centrality score, a centrality measure that accounts for the number of paths from a node to all others, weighted by path length.
  - **K-core Centrality (CC)** [47]: a heuristic that selects nodes by their core number, defined as the largest k-core in which the node is a member. A k-core refers to a maximal subgraph where each node has a degree that is equal to or greater than k.
  - **Collective Influence (CI)** [48]: a heuristic that selects nodes with the highest value of Collective Influence, a centrality measure designed to identify nodes whose removal most efficiently fragments the network. CI accounts for both a node's local degree and the degrees of nodes at the boundary of a ball of fixed radius.

We remark that classic methods and heuristics have a significantly lower computational cost, albeit at the expense of solution quality, as demonstrated by the results presented below. Before presenting the results, we define the settings for the GIP model used in our experiments. As shown in Equation (3.11), the parameters θ_l and θ_h fully define the behavior of the GIP model. We have chosen the following values for our experiments:

- $\theta_l = 2$, i.e. it requires at least 2 nodes to activate another node;
- $\theta_h = 50$: which ensures variance in node behavior.

Using this setting, the first observation is that KC generally performs poorly compared to other heuristics, both in terms of computation time and solution quality as we can see in Table 2. Consequently, we decided to use the SD heuristic as a starting point for both CDS and NaDS. We then conducted two sets of experiments on the six networks mentioned above, varying the budget across four values: $B = 5, 10, 15, 20$, which is connected with the complexity of the problem.

In the first set of experiments, we used the SD heuristic as the starting point for both methods. Table 2 shows the best results achieved and the method that achieved them, in terms of cumulative influence as defined in (3.1). This approach ensures that CDS and NaDS consistently achieve the best solutions across all the methods considered. One of the strengths of the direct search approach is its ability to be combined with heuristics to achieve better results, as demonstrated in this study.

In the second set of experiments, we aimed to test the robustness of CDS and NaDS. For each network, we selected 10 pseudo-random feasible points to use as starting points for both direct search approaches. These pseudo-random points were chosen to be similar to the SD solution: the starting points were constructed by sampling B random nodes from the list of the top $4B$ nodes, where B is the budget for the specific problem. This approach ensures that the starting points are

Table 2. Performance comparison based on cumulative influence from the first set of experiments, evaluated across different datasets and budget levels. The best results are highlighted in bold.

Dataset	B	NaDS			CDS			SD	SG	KC	CC	CI
		Mean	Best	Disc	Mean	Best	Disc					
Arxiv Astro	5	3751.43	**3815.8**	**3815.8**	3373.38	3749.06	3597.15	3571.59	2049.12	357.35	1934.67	3186.73
	10	5068.22	**5083.1**	**5083.1**	4404.69	4592.02	4989.82	4853.77	4031.24	459.94	2602.13	4560.46
	15	5986.41	**5989.09**	5977.78	5221.54	5472.44	5909.25	5796.86	5315.98	795.46	3008.36	5501.35
	20	6660.95	**6671.49**	**6671.49**	5761.73	5972.2	6558.47	6481.9	6304.03	1149.91	3325.22	6284.10
Arxiv Gr-Qc	5	199.31	199.97	198.74	203.1	**220.22**	198.74	186.45	4.29	34.18	184.29	172.78
	10	280.86	**307.13**	277.39	271.9	292.97	276.94	275.9	35.53	74.69	245.79	237.71
	15	375.32	**379.4**	340.92	341.73	350.05	**379.4**	326.85	188.29	100.49	307.29	295.79
	20	446.26	**460.46**	396.38	410.38	430.1	438.01	382.93	270.98	121.66	359.82	340.20
Arxiv HeP-Ph	5	2329.97	**2339.73**	2339.73	2263.14	2314.29	2317.71	2293.24	2109.6	85.91	2300.30	2020.11
	10	3008.04	**3043.62**	3000.11	2851.42	2899.88	2989.11	2965.51	2822.33	456.01	2965.91	2532.34
	15	3519.25	**3533.07**	3500.4	3275.9	3340.8	3490.11	3479.19	3380.15	628.78	3465.91	2915.90
	20	3939.2	**3946.2**	3936.05	3657.36	3722.2	3934.66	3910.11	3842.82	695.57	3822.11	3282.54
Lastfm-Asia	5	452.68	**487.06**	406.41	351.21	449.76	407.8	188.00	178.70	126.54	409.80	99.40
	10	773.71	**815.85**	806.61	695.56	735.28	697.32	571.75	318.26	415.93	534.06	557.20
	15	1036.49	**1077.62**	1090.26	866.06	944.14	976.72	913.53	410.2	587.65	662.31	873.06
	20	1280.09	**1297.63**	1280.41	1003.63	1085.63	1207.27	1158.69	618.8	657.01	747.04	1065.35
email-Enron	5	7679.31	**7689.81**	**7689.81**	7001.57	7505.15	6435.79	5960.03	879.02	4183.08	5138.12	4862.94
	10	9955.37	**10 018.82**		9995.02	8824.24	9153.67	9480.59	9359.18	8066.82	5276.44	7221.76	7355.98
	15	11 467.5	**11 589.03**		11 588.8	9993.2	10 482.51	11 197.63	11 189.28	10 162.89	6219.94	8267.10	9143.80
	20	12 738.43	**12 925.81**		12 841.32	11 056.62	11 500.55	12 429.74	12 429.74	11 869.34	7025.14	9262.36	10 843.41
Facebook	5	**1570.67**	**1570.67**	**1570.67**	1323.08	**1570.67**	1424.51	380.37	890.93	297.08	589.48	380.37
	10	2032.22	**2037.63**	2028.97	1825.81	1974.28	1998.38	1601.01	1537.71	447.29	781.12	1066.63
	15	2368.61	**2639.21**	2361.72	2071.97	2472.67	2561.29	2065.6	1800.75	555.14	954.58	1761.60
	20	2966.17	2972.91	**2999.34**	2197.34	2588.8	2738.75	2239.41	2413.19	1013.8	1084.30	1934.85

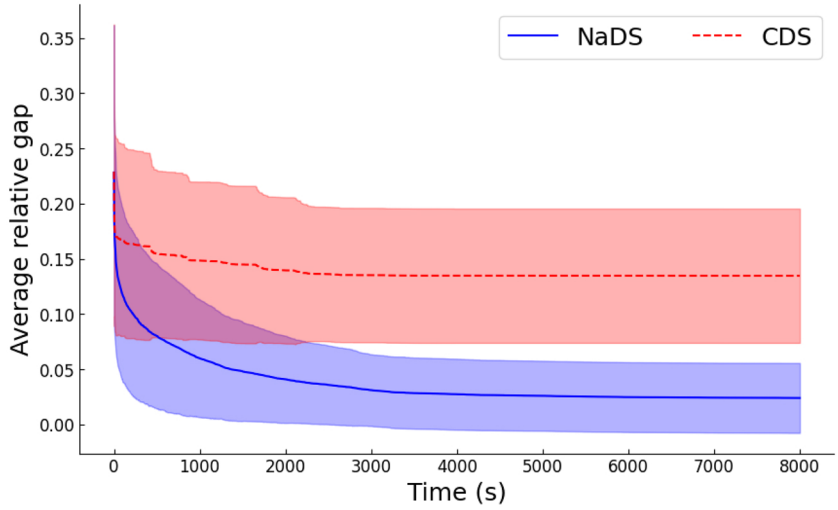

Figure 2. Results of the second set of experiments where we compare the average relative gap over time (with shaded standard deviation) of the NaDS and CDS methods, respectively, in solid and dashed lines.

random yet still possess a good value in terms of influence. Additionally, we used a time-stopping condition for every problem, depending on the network size. For the larger networks, namely Arxiv Astro and email-Enron, the time budget was 400 seconds multiplied by the budget B, while for the other networks, it was 200 seconds multiplied by the budget B. In Fig. 2, we report the average relative gap of the solutions of CDS and NaDS, where the relative gap at time t for a specific problem is defined as:

$$g(t) = \frac{m - s(t)}{m}, \tag{5.1}$$

with $s(t)$ the solution found at time t by the method and m is the best solution found among all methods.

Table 2 presents more detailed results from both sets of experiments and summarize the best outcomes achieved by classic models and heuristics (namely SD, SG, KC, CC, and CI) as well as the NaDS and CDS methods. For NaDS, we provide results from the first set of experiments, labeled as NaDS (disc), where the method was initialized using the solution from SD. Additionally, results from the second set of experiments, using 10 pseudo-random starting points, are included. For these, we use the labels NaDS (mean) for the average results across a single problem and NaDS (best) for the best result obtained. For more comprehensive results from the second set of experiments, we refer to Section B of the Appendix.

Table 3 summarizes the observed time complexity across NaDS and CDS algorithms, revealing that NaDS maintains a better scaling behavior than CDS while giving much better results in terms of solution quality. Notably, while greedy heuristics show good time complexity in theory, their actual cpu-time cost is much larger than our methods (and solution quality is not as good as NaDS). As for the other heuristics, while they achieve faster computation times, our experiments reveal this often comes at the cost of solution quality.

We can thus draw the following conclusions from the numerical experience we carried out. More specifically,

1. Classic methods do not perform well in this setting. While the SD yields good results, it does not guarantee success across all datasets. Notably, in this setting, the submodularity of the

Table 3. Average relative gaps for NaDS and CDS at different percentages of the time budget. Relative gaps are expressed in %.

Dataset	No. of nodes	B	Time Budget	NaDS					CDS				
				15%	30%	50%	75%	100%	15%	30%	50%	75%	100%
Arxiv Astro	18 772	5	2000	2.931	0.100	0	0	0	10.118	10.118	10.118	10.118	10.118
		10	4000	6.743	2.033	0.066	0	0	13.708	13.708	13.708	13.708	13.088
		15	6000	7.385	3.249	0.759	0	0	13.152	13.152	13.152	12.775	12.775
		20	8000	8.517	3.144	0.815	0.051	0	13.667	13.667	13.542	13.542	13.499
Arxiv Gr-Qc	52 442	5	1000	2.976	2.976	2.976	2.976	2.976	3.541	3.040	2.655	1.802	1.323
		10	2000	2.241	1.328	1.328	1.328	1.328	6.972	6.692	6.635	4.816	4.515
		15	3000	0.320	0.116	0.001	0.001	0	11.625	11.084	11.072	9.087	8.954
		20	4000	0.296	0.207	0.128	0	0	10.979	10.533	10.134	8.140	8.023
Arxiv Hep-Ph	12 008	5	1000	1.165	0.046	0	0	0	2.864	2.864	2.864	2.864	2.864
		10	2000	2.456	0.829	0.047	0	0	5.445	5.445	5.445	5.208	5.208
		15	3000	4.415	1.990	0.402	0.013	0	7.352	7.352	7.072	6.916	6.916
		20	4000	4.925	2.261	0.435	0.015	0	7.407	7.202	7.202	7.163	7.154
Lastfm-Asia	7624	5	1000	0.052	0.052	0.052	0.052	0.052	30.914	30.914	25.174	25.174	22.687
		10	2000	1.282	1.187	0.425	0.220	0	21.788	16.684	14.248	12.787	9.909
		15	3000	1.664	0.797	0.155	0.095	0	20.747	19.650	18.520	16.949	16.478
		20	4000	1.663	0.738	0.120	0	0	24.426	23.057	22.050	21.672	21.554
Email-Enron	36 692	5	2000	7.911	4.246	1.363	0.104	0	8.818	8.818	8.818	8.818	8.818
		10	4000	10.098	6.845	3.266	1.068	0	11.378	11.378	11.378	11.378	11.378
		15	6000	10.072	7.993	4.459	1.715	0	12.848	12.848	12.848	12.848	12.848
		20	8000	11.407	9.291	4.746	1.794	0	13.197	13.197	13.197	13.197	13.197
Facebook	4039	5	1000	8.514	2.981	1.861	0	0	24.188	23.839	21.063	18.261	15.763
		10	2000	9.221	3.965	1.527	0.289	0	14.532	13.893	13.604	10.600	10.167
		15	3000	17.274	10.729	5.392	2.149	0	21.401	20.723	20.718	12.832	12.515
		20	4000	24.546	16.406	9.475	2.119	0	30.943	30.936	30.876	25.920	25.915

objective functions does not hold, which was a crucial assumption for the guarantees provided by greedy methods. This implies that in this context, direct search methods have the upper hand, consistently showing better results. As we can see in Table 2, all the best results were achieved by direct search methods. In particular, NaDS achieved the best results for every problem except one.

2. NaDS outperforms CDS in almost every dataset, both with SD and pseudo-random start. We also observe that the performance difference increases with the complexity of the problems, particularly represented by the budget B. This confirms that the improved neighborhood formulation of NaDS scales well with the dimensions of the graphs. We note that in every problem, except for one single case, NaDS achieves better results than CDS. Figure 2 highlights the improved computational efficiency of the NaDS method.

6. CONCLUSIONS

In this paper, we introduced NaDS, a novel direct search method for solving the IM problem in networks. The NaDS method leverages the structure of the network to enhance computational efficiency and effectiveness in identifying influential nodes. Our approach integrates the GIP model to better capture the dynamics of influence spread, providing a more robust and scalable solution compared to traditional direct search methods.

Extensive numerical experiments demonstrate that NaDS outperforms many classic methods, like, e.g. Single Discount [18] Simple Greedy (SG) [10], Katz Centrality [12], K-core Centrality [47] and Collective Influence [48] in terms of quality of the solutions. In particular, in our GIP setting, some of those heuristics, like, e.g. greedy algorithms, lose their theoretical guarantees, especially those based on the submodularity of the objective function. This makes the use of direct search methods a valuable approach in practice. So, while those classic methods have proven successful in settings where the objective function exhibits submodularity [20], our work highlights their limitations in those complex scenarios, where such properties may not hold. This observation is consistent with recent critiques of heuristic methods in non-submodular settings [12, 22]. Furthermore, the superior performance of NaDS over the Customized Direct Search (CDS) method [12] highlights the importance of incorporating network structure into the optimization process. By refining the neighborhood definition to focus on graph-connected nodes, NaDS indeed achieves significant gains in terms of computational efficiency and highly improved solution quality. Our results thus complement recent efforts to unify propagation models [12], demonstrating that tailored direct search methods work pretty well when considering more sophisticated diffusion dynamics.

Future works will focus on refining the computational efficiency of the algorithmic framework, as the implementation still has many possible improvements. The current implementation of NaDS could indeed be optimized by, e.g. parallelizing the neighborhood evaluation phase, leveraging modern multi-core architectures. This will likely allow for further experimentation and application of the NaDS method to even larger networks, on the scale of hundreds of thousands or even millions of nodes.

Additionally, an important extension to the IM problem can be made by considering higher-order networks, i.e. hypergraphs and multilayer graphs. In particular, extending NaDS to hypergraphs would require redefining the neighborhood to account for hyperedges, thus capturing the influence of node groups; while adapting it to multilayer networks would involve integrating cross-layer dependencies into the neighborhood formulation.

SUPPLEMENTARY DATA

Supplementary data is available at *COMNET Journal* online.

CONFLICT OF INTEREST

None declared.

FUNDING

None declared.

REFERENCES

1. Bakshy E, Rosenn I, Marlow C, Adamic L. The role of social networks in information diffusion. In: *Proceedings of the 21st International Conference on World Wide Web*. 2012, 519–28.
2. Centola D. The spread of behavior in an online social network experiment. *Science* 2010;**329**:1194–7.
3. Erkol S, Castellano C, Radicchi F. Systematic comparison between methods for the detection of influential spreaders in complex networks. *Sci Rep* 2019;**9**:15095.
4. Nekovee M, Moreno Y, Bianconi G *et al.* Theory of rumour spreading in complex social networks. *Phys A Stat Mech Appl* 2007;**374**:457–70.
5. Domingos P, Richardson M. Mining the network value of customers. In: *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2001, 57–66.
6. Leskovec J, Krause A, Guestrin C *et al.* Cost-effective outbreak detection in networks. In: *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2007, 420–9.
7. Budak C, Agrawal D, El Abbadi A. Limiting the spread of misinformation in social networks. In: *Proceedings of the 20th International Conference on World Wide Web*. 2011, 665–74.
8. He X, Song G, Chen W, Jiang Q. Influence blocking maximization in social networks under the competitive linear threshold model. In: *Proceedings of the 2012 SIAM International Conference on Data Mining*, 2012, 463–74.
9. Ye M, Liu X, Lee W. Exploring social influence for recommendation: a generative model approach. In: *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2012, 671–80.
10. Kempe D, Kleinberg J, Tardos É. Maximizing the spread of influence through a social network. In: *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2003, 137–46.
11. Shakarian P, Bhatnagar A, Aleali A *et al.* The independent cascade and linear threshold models. In: *Diffusion in Social Networks*. 2015, 35–48.
12. Tian Y, Lambiotte R. Unifying information propagation models on networks and influence maximization. *Phys Rev E* 2022;**106**:034316.
13. Audet C, Hare W. *Derivative-Free and Blackbox Optimization*. Springer, 2017.
14. Conn A, Scheinberg K, Vicente L. *Introduction to Derivative-Free Optimization*. SIAM, 2009.
15. Dzahini K, Rinaldi F, Royer C, Zeffiro D. Revisiting theoretical guarantees of direct-search methods. arXiv, arXiv:2403.05322, 2024, preprint: not peer reviewed.
16. Larson J, Menickelly M, Wild S. Derivative-free optimization methods. *Acta Numer* 2019;**28**:287–404.
17. Bovet A, Makse H. Influence of fake news in Twitter during the 2016 US presidential election. *Nat Commun* 2019;**10**:7.
18. Chen W, Wang C, Wang Y. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In: *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2010, 1029–38.
19. Leskovec J, Adamic L, Huberman B. The dynamics of viral marketing. *ACM Trans Web* 2007;**1**:5.
20. Mossel E, Roch S. Submodularity of influence in social networks: from local to global. *SIAM J Comput* 2010;**39**:2176–88.
21. Pastor-Satorras R, Castellano C, Van Mieghem P *et al.* Epidemic processes in complex networks. *Rev Mod Phys* 2015;**87**:925–79.
22. Arora A, Galhotra S, Ranu S. Debunking the myths of influence maximization: an in-depth benchmarking study. In: *Proceedings of the 2017 ACM International Conference on Management of Data*. 2017, 651–66.
23. Chen W, Castillo C, Lakshmanan L. *Information and Influence Propagation in Social Networks*. Springer Nature, 2022.
24. Guille A, Hacid H, Favre C *et al.* Information diffusion in online social networks: a survey. *Sigmod Rec* 2013;**42**:17–28.
25. Li Y, Fan J, Wang Y *et al.* Influence maximization on social graphs: a survey. *IEEE Trans Knowl Data Eng* 2018;**30**:1852–72.
26. Sun J, Tang J. A survey of models and algorithms for social influence analysis. *Soc Netw Data Analytics*. 2011;177–214.

27. Tejaswi V, Bindu P, Thilagam P. Diffusion models and approaches for influence maximization in social networks. In: *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. 2016, 1345–51.
28. Zhang H, Mishra S, Thai M *et al.* Recent advances in information diffusion and influence maximization in complex social networks. *Opportunistic Mobile Soc Netw* 2014;**37**:37.
29. Kermack W, McKendrick A. A contribution to the mathematical theory of epidemics. *Proc R Soc Lond Ser A Contain Papers Math Phys Charact* 1927;**115**:700–21.
30. Clifford P, Sudbury A. A model for spatial conflict. *Biometrika* 1973;**60**:581–8.
31. Goldenberg J, Libai B, Muller E. Talk of the network: a complex systems look at the underlying process of word-of-mouth. *Mark Lett* 2001;**12**:211–23.
32. Granovetter M. Threshold models of collective behavior. *Am J Sociol* 1978;**83**:1420–43.
33. Schelling T. *Micromotives and Macrobehavior*. WW Norton & Company, 2006.
34. Chen WEI, Lu WEI, Zhang N. Time-Critical Influence Maximization in Social Networks with Time-Delayed Diffusion Process. *AAAI* 2021;**26**:591–8.
35. Kim J, Lee W, Yu H. CT-IC: continuously activated and time-restricted independent cascade model for viral marketing. *Knowl Based Syst* 2014;**62**:57–68.
36. Liu B, Cong G, Xu D, Zeng Y. Time constrained influence maximization in social networks. In: *2012 IEEE 12th International Conference On Data Mining*. 2012, 439–48.
37. Gomez-Rodriguez M, Balduzzi D, Schölkopf B. Uncovering the temporal dynamics of diffusion networks. In: *Proceedings of the 28th International Conference on International Conference on Machine Learning*. 2011, 561–8.
38. Xie M, Yang Q, Wang Q *et al.* Dynadiffuse: a dynamic diffusion model for continuous time constrained influence maximization. *Proc AAAI Conf Artif Intell* 2015;**29**:346–52.
39. Chen W, Wang Y, Yang S. Efficient influence maximization in social networks. In: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2009, 199–208.
40. Jung K, Heo W, Chen W. Irie: scalable and robust influence maximization in social networks. In: *2012 IEEE 12th International Conference on Data Mining*. 2012, 918–23.
41. Kimura M, Saito K. Tractable models for information diffusion in social networks. In: *European Conference on Principles of Data Mining and Knowledge Discovery*. 2006, 259–71.
42. Liu Q, Xiang B, Chen E *et al.* Influence maximization over large-scale social networks: a bounded linear approach. In: *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*. 2014, 171–80.
43. Borgs C, Brautbar M, Chayes J, Lucier B. Maximizing social influence in nearly optimal time. In: *Proceedings of the Twenty-fifth Annual ACM-SIAM Symposium on Discrete Algorithms*. 2014, 946–57.
44. Cheng S, Shen H, Huang J *et al.* Staticgreedy: solving the scalability-accuracy dilemma in influence maximization. In: *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*. 2013. 509–18.
45. Cohen E, Delling D, Pajor T, Werneck R. Sketch-based influence maximization and computation: scaling up with guarantees. In: *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. 2014, 629–38.
46. Ohsaka N, Akiba T, Yoshida Y *et al.* Fast and accurate influence maximization on large networks with pruned Monte–Carlo simulations. *Proc AAAI Conf Artif Intell* 2014;138–44.
47. Kitsak M, Gallos L, Havlin S *et al.* Identification of influential spreaders in complex networks. *Nat Phys* 2010;**6**:888–93.
48. Morone F, Makse H. Influence maximization in complex networks through optimal percolation. *Nature* 2015;**524**:65–8.
49. Patwardhan S, Radicchi F, Fortunato S. Influence maximization: divide and conquer. *Phys Rev E* 2023;**107**:054306.
50. Pei S, Wang J, Morone F *et al.* Influencer identification in dynamical complex systems. *J Complex Netw* 2020;**8**:cnz029.
51. Katz L. A new status index derived from sociometric analysis. *Psychometrika* 1953;**18**:39–43.
52. Leskovec J, Kleinberg J, Faloutsos C. Graph evolution: densification and shrinking diameters. *ACM Trans Knowl Discov Data* 2007;**1**:2.2-es 3)
53. Rozemberczki B, Sarkar R. Characteristic functions on graphs: birds of a feather, from statistical descriptors to parametric models. In: *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*. 2020, 1325–34.
54. Leskovec J, Lang K, Dasgupta A, Mahoney M. Community structure in large networks: natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*. 2008;6.
55. Klimt B, Yang Y. The enron corpus: a new dataset for email classification research. *Mach Learn ECML*. 2004, 217–26.

56. McAuley J, Leskovec J. Learning to discover social circles in ego networks. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems – Volume 1*. 2012. 539–47.
57. Lancichinetti A, Fortunato S, Radicchi F. Benchmark graphs for testing community detection algorithms. *Phys Rev E* 2008;**78**:10.
58. Hagberg A, Schult D, Swart P. Exploring network structure, dynamics, and function using NetworkX. In: *Proceedings Of The 7th Python In Science Conference*. 2008, 11–15.
59. Harris C, Millman K, Van Der Walt S *et al.* Others Array programming with NumPy. *Nature* 2020;**585**:357–62.
60. Virtanen P, Gommers R, Oliphant T, Others SciPy *et al.* 1.0: fundamental algorithms for scientific computing in Python. *Nat Methods* 2020;**17**:261–72.
61. Hunter J. Matplotlib: a 2D Graphics Environment. *Comput Sci Eng* 2007;**9**:90–5.
62. Leskovec J, Krevl A. *SNAP Datasets: Stanford Large Network Dataset Collection*. 2014. http://snap.stanford.edu/data