# Efficient Parameter Estimation for Information Retrieval using Black-box Optimization

Alberto Costa, Emanuele Di Buccio, Massimo Melucci, and Giacomo Nannicini

**Abstract**—The retrieval function is one of the most important components of an Information Retrieval (IR) system, because it determines to what extent some information is relevant to a user query. Most retrieval functions have "free parameters" whose value must be set before retrieval, significantly affecting the effectiveness of an IR system. Choosing the optimum values for such parameters is therefore of paramount importance. However, the optimum can only be found after a computationally expensive process, especially when the generalization error is estimated via cross-validation. In this paper, we propose to determine free parameter values by solving an optimization problem aimed at maximizing a measure of retrieval effectiveness. We employ the black-box optimization paradigm, since the analytical expression of the measure of effectiveness with respect to the free parameters is unknown. We consider different methods for solving the black-box optimization problem: a simple grid-search over the whole domain, and more sophisticated techniques such as line search and surrogate model based algorithms. Experimental results on several test collections not only provide useful insight about effectiveness, but also about efficiency: they indicate that with appropriate optimization techniques, the computational cost of parameter optimization can be greatly reduced without compromising retrieval effectiveness, even when taking generalization into account.

**Index Terms**—Information Retrieval, Optimization, Parameter Estimation.

## 1 Introduction

Information Retrieval (IR) is the complex of activities that represent information as data and retrieves the data representing information relevant to the user's information needs. An IR system is a computer system performing indexing and retrieval activities. Current technology requires a retrieval model to be effective and allow prediction. A retrieval model is a set of algebraic structures that describes documents and queries. The model's core is a retrieval function that maps these information structures to the numeric real field. Choosing the right model is therefore a key step of the deployment of an IR system.

Leaving indexing aside, we focus on the parametric retrieval function. Many such functions involve an array of parameters that are called "free" because their values can in principle be set irrespective of the specific set of documents and queries. This allows researchers to tune the retrieval function to maximize retrieval effectiveness, i.e., the proportion of documents that are relevant to an information need and are retrieved from a query representing this need. Indeed, [1] states that "traditional retrieval functions have few free parameters, but nevertheless these parameters need to be set to some value, and this choice affects the resulting performance". Examples of free parameters in retrieval functions are the $\lambda$ parameter when performing smoothing with the Jelinek-Mercer method, the $\mu$ parameter in the Dirichlet smoothing method [2], the $c$ parameter in Information-based Models [3], or the $\alpha, \beta, \gamma$ parameters in the Rocchio Feedback Model [4]. Another widely used retrieval function is Best Match 25 (BM25) [5]. This paper employs BM25 and a field-based version of it called BM25F [6]. The latter function in particular has several free parameters, and is well-suited for the investigation of algorithms to set free parameters.

IR researchers often compare parameter configurations and struggle to obtain the most effective configuration. We claim that efficient algorithms to select free parameters of retrieval functions are needed in several practically relevant scenarios. We now provide examples of such scenarios.

A parameter configuration that performs well can typically be found by testing many values for every parameter on some training set, and selecting the values that maximize a certain retrieval measure. To this end, a common approach is to divide each parameter range into many equal-size subranges and pick one value from each subrange. Exploring all these values is called grid search. If the subranges are very small, the grid is fine-grained and this type of search is quite exhaustive. Although grid search for the optimal parameter values may become inefficient, it has two main advantages: it is simple to implement, and it can be applied to any retrieval function and retrieval effectiveness measure. The problem is that the degree of inefficiency grows exponentially with the number of parameters, since dividing $n$ parameters into $k$ subranges would require testing $O(k^n)$ different configurations. This can be very inefficient for as little as three parameters. In fact, studies in machine learning reveal that even random search (testing several random values sampled from a uniform distribution) can work better than grid search [7].

Furthermore, parameters are typically optimized only at the time of construction and indexing of the document collection, but optimization should be performed whenever

- *A. Costa is with the National University of Singapore and ETH Zurich, Future Resilient Systems Program, Singapore.*
  *E-mail: costa@lix.polytechnique.fr*
- *E. Di Buccio and M. Melucci are with the University of Padova, Italy.*
- *G. Nannicini is with IBM Research, T.J. Watson research center, NY, USA.*

document collections grow, or queries evolve from one style to the other: for example, three query styles – informational, navigational and transactional – were found in the context of the Web [8]. Therefore, the overall effectiveness of a retrieval system is harmed by the inefficiency of performing repeated parameter optimization for every situation, i.e., collection or query set.

Finally, to address the rapid evolution of data and variety of collections and queries, parameter values should be chosen taking generalization performance into account, to avoid overfitting. Generalization can be estimated e.g., using cross-validation. Unfortunately, cross-validation compounds the problem of inefficient search for parameter values of the retrieval function, increasing the computational effort required.

All these examples support our claim that efficient algorithms to select free parameters of retrieval functions are needed. This paper aims to address the crucial issue of efficiently finding the parameter values of retrieval functions that yield the most effective system. The main contribution of this paper consists in casting the parameter optimization problem as a mathematical program solved by black-box optimization methods that do not require an analytical description of the objective function, and in empirically showing that (i) black-box optimization is significantly more efficient than grid search on a benchmark set, and (ii) black-box optimization reliably and efficiently (i.e., in a short time) finds effective parameter values even when generalization is a requirement and/or the retrieval function has many parameters. The rest of the paper is organized as follows. Section 2 provides an overview of existing literature relevant to our work. Section 3 introduces black-box optimization and the techniques used in the our experiments. Section 4 provides more detail on how black-box optimization can be used in the context of IR, in particular discussing the choice of objective function and how to evaluate it. A computational evaluation is discussed in Section 5 to compare efficacy and efficiency of the proposed methods. Finally, conclusions are presented in Section 6.

## 2 Related Work

The subject of this paper is optimization in the context of IR. This area has been extensively studied since the early sixties, when researchers started to investigate retrieval functions and their optimization, suggesting algorithms that can automatically refine queries and update document ranking by leveraging parameters optimized on the basis of the user's feedback, see [9].

A limitation of such algorithms is the difficulty of dealing with increasing number of parameters: there could be as many parameters as the documents assessed by users, and optimizing many parameters may lead to computational inefficiency, thus making the considered algorithm too expensive for practical applications. Another limitation is the risk of overfitting, due to the computational burden of assessing generalization. Therefore, even though the emphasis of optimization in the context of IR has been effectiveness rather than efficiency, the latter has become an important aspect to consider when dealing with complex retrieval functions. An example of the importance of this aspect is found in the context of automatic query expansion. A crucial step of automatic query expansion is the ranking of candidate expansion features [10]: at a high level, an algorithm for automatic query expansion has three parameters, namely the number of documents that provide terms, the number of terms provided by each document, and the number of terms related to each term. Even with just three parameters, evaluating all their combinations in order to find the optimal one would be too computationally challenging.

Another prominent application domain of optimization in IR is that of parametric retrieval functions. Most retrieval functions have parameters that stem from design choices or mathematical properties. For example, language modeling consists of ranking documents by a mixture of probabilities interpolated by way of parameters [11]. Binary independence retrieval models are enhanced by smoothing parameters that reduce the side-effects of lack of term-occurrence statistics, e.g., undesired $0/0$ situations, as illustrated in [12]. In the case of Bernoulli or binomial retrieval functions, these smoothing parameters may be beta random variables. When based on maximum likelihood, optimization can leverage the continuity of retrieval functions. However, in almost all relevant cases the assumption of continuity does not hold; an example is optimization based on maximum retrieval effectiveness, which is a discrete function and requires algorithms such as those illustrated in [1] and [13]. The emergence of learning-to-rank raised the issue of parameter optimization at a scale larger than that encountered in ad-hoc retrieval, see e.g., [14]. An interesting application of learning-to-rank related to our work is presented in [15], where the authors propose a method to optimize the configuration of an IR system and to deal with discrete values for system setting, while in this paper, we deal with continuous values. The approach by [15] can choose values in a continuous space only insofar as a finite set of possible values for the continuous is preselected, for example via discretization. The aim of our paper is to study the intrinsic parameter optimization of a retrieval model (e.g., BM25) rather than the whole IR system. However, these approaches may be integrated to further improve the overall performance.

## 3 Black-box Optimization

Without loss of generality, a single-objective mathematical optimization problem can be written as follows[1]:

$$\max \quad f(\mathbf{x}) \tag{1}$$
$$\text{s.t.} \quad \mathbf{x} \in \mathcal{X}. \tag{2}$$

In the specific case studied in this paper, $\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^n \cap [\mathbf{x}^L, \mathbf{x}^U]\}$ and $\mathbf{x}^L, \mathbf{x}^U$ are the vectors of lower and upper bounds on the parameters. In general $\mathcal{X}$ may involve other constraints, such as integrality requirements, but this is not the case for the problem studied in this paper. Problem (1)-(2) is called *black-box* when the analytical expression of the objective function $f$ is not known. The only assumptions about $f$ is that given $\mathbf{x} \in \mathcal{X}$, we can evaluate $f(\mathbf{x})$ through an oracle (convergence proofs for specific algorithms typically require further assumptions,

1. Bold symbols represent vectors.

e.g., some inherent smoothness). Black-box optimization is also known as *derivative-free*.

The black-box optimization framework applies naturally to the problem of parameter tuning in IR: in this case, $\mathbf{x}$ is the vector of parameters of an IR model, e.g., $b$ and $k_1$ in BM25, and these parameters are usually defined within some range. Given a collection of documents and a set of queries, one usually wants to maximize some performance measure such as Mean Average Precision (MAP) [16] or Normalized Discounted Cumulative Gain (NDCG) [17]. This measure can be viewed as the unknown objective function of a black-box problem: we do not know how to express it in terms of the parameters that we want to tune, but when $b$ and $k_1$ are set, MAP or NDCG can be evaluated on the ranking lists given as output by a system that uses BM25. More details are provided in Section 4. Here we discuss the black-box optimization framework in general, and present specific algorithms that can be used to solve Problem (1)-(2).

Many approaches to solve black-box optimization problems have been proposed. As mentioned above, simple grid search does not scale well and is often impractical, especially when the evaluation of the oracle for $f$ requires a significant amount of time. Some methodologies rely on metaheuristics, such as genetic algorithms, simulated annealing, and particle swarm. However, in many applications such methods present a slow convergence, i.e., the quality of the best solution found improves very slowly with respect to the number of function evaluations. The mathematical optimization community regards metaheuristics as a method of last resort [18], and prefers other types of approaches, in particular direct search methods [19] or methods based on surrogate models.

The defining characteristic of methods based on surrogate models is that, unlike direct search method, they try to build an approximate model for the function $f$. This model can then be used in many ways, e.g., to estimate gradients or determine candidate optima. Among the approaches to implementing surrogate modes there are the kriging-based EGO (Efficient Global Optimization) method [20], Gutmann's Radial Basis Function (RBF) method [21], and the stochastic RBF method [22]. These approaches build a global model of the function $f$, in contrast to the local models employed by other methodologies such as trust regions [18]. This guarantees convergence to the global optimum from a theoretical standpoint if the number of iterations is large enough, and the capability of identifying solutions of good quality within a limited number of iterations from a practical point of view.

In this paper we compare the performance of three methods to solve the problem of parameter tuning in IR:

1) grid search, arguably the most popular methodology;
2) line search, a direct search method already employed in [1];
3) an approach based surrogate models, and more specifically, we employ the variant of the Metric Stochastic Response Surface Method (MSRSM) [22] implemented in the open-source library RBFOpt [23].

We choose RBFOpt among the many approaches proposed by the mathematical optimization community, because it can handle potentially nonconvex problems, it has already shown its effectiveness in empirical comparisons with many existing methods on several types of problems (e.g., mathematical benchmarks [23], neural networks [24], and architectural design, see in particular the recent benchmarks [25], [26]), and because its implementation is open-source. We now discuss the main features of line search and the MSRSM algorithm as implemented in RBFOpt.

## 3.1 Line search

Line search is a simple optimization methodology that can be employed to solve black-box optimization problems, because it only requires zero-order information (i.e., it does not require gradients). In this paper, we refer to the version of line search applied to IR in [1], which will be our benchmark. We now provide a brief description of the algorithm.

For $j = 1, \dots, n$, we denote by $x_j$ the $j$-th component of the point $\mathbf{x} \in \mathcal{X}$, and by $\mathbf{e}_j$ the $j$-th unit vector. Line search tries to iteratively improve the objective function value by determining an ascent direction. At a given iteration, let $\mathbf{x}$ be the current point. Starting from $\mathbf{x}$, the algorithm samples along each direction $\mathbf{e}_j$ according to a given step size, and records the value $x_j^p$ corresponding to the best point found along direction $\mathbf{e}_j$, $j = 1, \dots, n$. Then, the point $\mathbf{x}^p = (x_1^p, \dots, x_n^p)$ is used to define the next sampling direction: we perform another round of sampling along the line connecting $\mathbf{x}$ to $\mathbf{x}^p$. If a point better than $\mathbf{x}$ is found, $\mathbf{x}$ is updated and the whole procedure, called *epoch* in [1], is repeated. The algorithm stops when we either reach a maximum of 24 epochs, or there are 3 consecutive epochs without improvement. The computation of the initial step size is not specified in [1]. In our implementation we sample 10 points along each direction $j = 1, \dots, n$, and the initial step size for direction $j$ is $\Delta_j^0 = \frac{x_j^U - x_j^L}{9}$. The update rule for the step size, i.e., the formula to compute the step size $\mathbf{\Delta}^{i+1}$ at epoch $i+1$, is $\mathbf{\Delta}^{i+1} = 0.85\mathbf{\Delta}^i$, following [1].

Given a starting point $\mathbf{x}^0$, Line Search can be summarized as follows:

```
Require: Starting point $\mathbf{x}^0$
1:  for all dimension $j = 1, \dots, n$ do
2:      $\Delta_j^0 \leftarrow (x_j^U - x_j^L) / 9$
3:  end for
4:  $i \leftarrow 0$
5:  $\mathbf{x} \leftarrow \mathbf{x}^0$
6:  repeat
7:      for all dimension $j = 1, \dots, n$ do
8:          Starting from $\mathbf{x}$ sample along the direction $\mathbf{e}_j$ with
            step size $\Delta_j^i$
9:          save the best point found in $x_j^p$
10:     end for
11:     Sample along the line between $\mathbf{x}$ and $\mathbf{x}^p$ with step size
        $(||\mathbf{x}^p - \mathbf{x}||) / 9$
12:     Set $\mathbf{x}$ to the best point found in the current cycle
13:     if $i \geq 2$ AND $\mathbf{x}$ has not changed in the last 3 epochs
        then
14:         BREAK
15:     end if
16:     for all dimension $j = 1, \dots, n$ do
17:         $\Delta_j^{i+1} \leftarrow 0.85\Delta_j^i$
```

```
18:     end for
19:     $i \leftarrow i+1$
20: until $i = 24$
21: return $\mathbf{x}$
```

## 3.2 Optimization using a RBF surrogate model

Optimization algorithms based on surrogate models typically build, at every major iteration, a model of the unknown objective function using the points evaluated so far. In the specific approach employed in this paper, and in several others discussed in the literature, this interpolant is built by means of RBFs. RBFs are functions of the form $\phi(||\mathbf{x}-\mathbf{y}||) : \mathbb{R}^+ \rightarrow \mathbb{R}$ where $\mathbf{y}$ is a given vector, i.e., they depend solely on the Euclidean distance $r = \|\mathbf{x}-\mathbf{y}\|$ of the argument $\mathbf{x}$ from the point $\mathbf{y}$ that is called the center of the RBF. There are different types of RBFs; the most commonly employed are the cubic RBF $\phi(r) = r^3$ and the thin plate spline RBF $\phi(r) = r^2 \log(r)$. Given $k$ distinct points $\mathbf{x}^1, \ldots, \mathbf{x}^k \in [\mathbf{x}^L, \mathbf{x}^U]$ and corresponding values $f(\mathbf{x}^i)$, the RBF surrogate model $s_k$ that interpolates the points $(\mathbf{x}^1, f(\mathbf{x}^1)), \ldots, (\mathbf{x}^k, f(\mathbf{x}^k))$ can be expressed as:

$$s_k(\mathbf{x}) = \sum_{i=1}^{k} \lambda_i \phi(||\mathbf{x}-\mathbf{x}^i||) + p(\mathbf{x}), \tag{3}$$

where $\phi$ is the chosen RBF, $\boldsymbol{\lambda} \in \mathbb{R}^k$, and $p(\mathbf{x})$ is a polynomial in $\mathbf{x}$. The polynomial guarantees existence and uniqueness of an interpolant of the form (3), and its degree depends on the type of RBF $\phi$ employed, see [21] for details. For the commonly used cubic and thin plate spline RBFs, $p$ is of degree 1, i.e., it is of the form $\mathbf{a}^\top \mathbf{x} + b$ with $\mathbf{a} \in \mathbb{R}^n$ and $b \in \mathbb{R}$. From now on we assume that this is the form of the interpolant. All the coefficients $\boldsymbol{\lambda}, \mathbf{a}, b$ can be found by solving a linear system imposing that $s_k(\mathbf{x}^i) = f(\mathbf{x}^i)$ with additional technical conditions, see [23].

Let $k$ be the function evaluation counter, and let MAX_EVAL be the maximum number of allowed function evaluations. The MSRSM method used by default by RBFOpt can be summarized as follows.

```
Require: $m$ starting points $\mathbf{x}^1, \ldots, \mathbf{x}^m$
 1: $S \leftarrow \{(\mathbf{x}^1, f(\mathbf{x}^1)), \ldots, (\mathbf{x}^m, f(\mathbf{x}^m))\}$
 2: $k \leftarrow m$
 3: repeat
 4:     Compute the RBF interpolant $s_k$ to the points in $S$ according to (3)
 5:     Determine the trade-off $\alpha \in [0,1]$ between exploration and exploitation
 6:     Find a point $\mathbf{x}^{k+1}$ with large value of $\min_{i=1,\ldots,k} \|\mathbf{x}^{k+1} - \mathbf{x}^i\|$ and large value of $s_k(\mathbf{x}^{k+1})$, using the weight $\alpha$ to balance these two criteria
 7:     Evaluate $f$ at $\mathbf{x}^{k+1}$ to obtain $f(\mathbf{x}^{k+1})$
 8:     $S \leftarrow S \cup \{(\mathbf{x}^{k+1}, f(\mathbf{x}^{k+1}))\}$
 9:     $k \leftarrow k+1$
10: until $k =$ MAX_EVAL
11: return the best solution from among those in $S$
```

We remark that RBFOpt implements additional optimization algorithms, in particular Gutmann's RBF method [21], but we limit our discussion to MSRSM because it is the default setting and it is employed in our experiments. We now provide more details on steps 1, 5 and 6.

### 3.2.1 *Initial sample points*

To guarantee existence of a unique interpolant of the form (3), we need at least $n+1$ distinct interpolation points. There are several ways to choose these initial points. In the context of optimization algorithms based on RBF surrogate models, a simple strategy proposed in early work [21] consists in using the $2^n$ vertices of the hypercube $[\mathbf{x}^L, \mathbf{x}^U]$. Such a strategy, labeled *all_corners* in Section 5.4, is only practical when $n$ is small, but empirical evidence suggests that it is a stable approach whenever practical. The default approach of RBFOpt is to use a Latin hypercube design [27] with exactly $n+1$ points, generated as follows: 50 Latin hypercube designs are constructed at random, and the one that maximizes the minimum Euclidean distance between the sample points is selected. This strategy is labeled *lhd_maximin* in Section 5.4.

### 3.2.2 *Selection of the next search point*

The choice of the next point at which the objective function $f$ should be evaluated is based on two criteria: the distance from previously evaluated points, and the objective function value of the surrogate model, both of which are to be maximized. Typically these criteria are conflicting, as the surrogate model is unlikely to attribute very large objective function values to regions of the search space that are very far from existing interpolation points, in particular those with high values. Prioritizing the distance criterion leads to an *exploration* phase, because it results in evaluating points in a previously unexplored region, whereas prioritizing the objective function value estimated by the surrogate model leads to an *exploitation* phase.

The algorithm uses a cyclic strategy that prioritizes each criterion in turn, by choosing a weight that determines the relative trade-off between distance and objective function value. More precisely, the parametrization of RBFOpt used for the computational evaluation presented in this paper works as follows. Let $\alpha \in [0,1]$ be a given weight and suppose we are at iteration $k$. We apply a genetic algorithm on a bi-objective optimization problem with the two objective functions:

$$g(\mathbf{x}) := \min_{i=1,\ldots,k} \|\mathbf{x}-\mathbf{x}^i\|, \qquad h(\mathbf{x}) := s_k(\mathbf{x}),$$

where $s_k$ is defined as in (3). After an initial population $P$ of points is generated, each point $\mathbf{x} \in P$ is assigned the score:

$$\alpha \frac{g(\mathbf{x}) - \min_{\mathbf{y} \in P} g(y)}{\max_{\mathbf{y} \in P} g(y) - \min_{\mathbf{y} \in P} g(y)} + \frac{h(\mathbf{x}) - \min_{\mathbf{y} \in P} h(y)}{\max_{\mathbf{y} \in P} h(y) - \min_{\mathbf{y} \in P} h(y)},$$

which is a weighted sum of a normalization of the values of the two objective functions $g$ and $h$. Larger scores are better. A fraction of the population with large score is retained and randomly mutated or combined, whereas the remaining part of the population is discarded and replaced by new points generated uniformly at random within the box $[\mathbf{x}^L, \mathbf{x}^U]$. The weight $\alpha$ varies from 0 to 1 in a periodic way with a period of length 6. Details on the implementation can be found in [28]. We remark that this approach is slightly different from MSRSM as described in [22], but the underlying ideas are similar and in our experience the version proposed here works better in practice.

## 4 Optimization of Retrieval Functions

The black-box optimization framework presented in the previous section is general. We now specialize it to the optimization of retrieval functions, discussing several choices for the objective function $f$ of Problem (1)-(2) in the context of retrieval functions with free parameters.

### 4.1 Retrieval Functions with Free Parameters

In this section, we provide a brief overview of BM25 with the goal of explaining the meaning of its free parameters, which will be thoroughly analyzed in the remainder of the paper. We refer the reader to [5] for a detailed description of the derivation of BM25.

The description of the retrieval functions reported below uses the notation introduced in [29]. Let $t$ be a term appearing in document $d$; according to BM25, the weight $w_{\text{BM25}}(t,d)$ assigned to the term $t$ in the document $d$ is:

$$w_{\text{BM25}}(t,d) = \frac{n'_L(t,d)}{k_1 + n'_L(t,d)}\, w_{\text{RSJ}}(t,c)\, \frac{(k_3+1)\, n_L(t,q)}{k_3 + n_L(t,q)}, \tag{4}$$

where:

- $n_L(t,q)$ is the number of locations at which the term $t$ occurs in the query $q$, i.e., the *term frequency* of term $t$ in query $q$.
- $n_L(t,d)$ is the number of locations at which the term $t$ occurs in the document $d$, i.e., the *term frequency* of term $t$ in document $d$.
- $n'_L(t,d)$ is the normalized term frequency. The normalization is with respect to the length of the document, $N_L(d) = \sum_{t \in d} n_L(t,d)$:

$$n'_L(t,d) = \frac{n_L(t,d)}{(1-b) + b\frac{N_L(d)}{\text{avgdl}}}, \tag{5}$$

  where avgdl is the average document length of the collection. The free parameter $0 \le b \le 1$ controls the extent of document length normalization: when $b = 1$ we consider a full document length normalization since $n'_L(t,d) = \frac{n_L(t,d)}{N_L(d)} \cdot \text{avgdl}$, whereas when $b = 0$ no document length normalization is adopted since $n'_L(t,d) = n_L(t,d)$.
- $k_1$ is a free parameter that controls the *saturation* of the (document) term frequency. Saturation is introduced to derive a retrieval function such that the score assigned to a term $t$ increases monotonically with $n_L(t,d)$ but asymptotically approaches a maximum value as $n_L(t,d) \to \infty$; in other words, the term score "cannot exceed a saturation point (the asymptotic limit), however frequently it occurs in the document" [5].
- $k_3$ is a free parameter that controls the saturation of the query term frequency, $n_L(t,q)$.
- $w_{\text{RSJ}}(t,c)$ is the Robertson-Sparck Jones weight that in absence of relevance information is the Inverse Document Frequency (IDF) [30] and is computed as:

$$w_{\text{RSJ}}(t,c) = \frac{N_D(c) - n_D(t,c) + 0.5}{n_D(t,c) + 0.5}, \tag{6}$$

  where $N_D(c)$ is the total number of documents in the collection $c$, and $n_D(t,c)$ is the number of documents in the collection where the term $t$ occurs.

Despite its popularity, BM25 has some limitations. In many application domains, documents are structured, since they are composed of *multiple fields* such as title, abstract, and body, or result from *multiple streams*; for example webpages can be characterized by title, body, terms occurring in the URL and anchor text. The BM25 retrieval function was not intended to deal with information on multiple fields or streams. In such a context, BM25 can be still adopted by merging all document fields/streams into a non-structured form, i.e., obtaining a unique stream as a concatenation of the multiple streams. The limitation of this approach is that we cannot exploit structure and associate a weight to each stream. BM25F was proposed to address this issue, and this paper studies the optimization of BM25F as well as BM25.

The simplest version of BM25F [6] employs a weighted variant of the total term frequency and of the document length, see Section 3.6.3 of [5] for details. If $s \in \{1, \ldots, T\}$ refers to the available streams, the free parameters in this simple BM25F version are the stream weights $\{v_1, \ldots, v_T\}$, $b$, and $k_1$. The number of free parameters is therefore $T + 2$.

Another version of BM25F was proposed in [1]. Therein, a stream-specific value of $b$ is adopted:

$$\widetilde{n}_L(t,d) = \sum_{s=1}^{T} v_s \frac{n_L(t,s,d)}{(1-b_s) + b_s\, \frac{N_L(s,d)}{\text{avgdl}(s)}} \quad 0 \le b_s \le 1, \tag{7}$$

where:

- $n_L(t,s,d)$ is the number of locations at which the term $t$ occurs in stream $s$ of document $d$;
- $N_L(s,d)$ is the length of stream $s$ in document $d$, i.e., the total number of locations for that stream;
- avgdl$(s)$ is the average length of stream $s$ in the collection.

The BM25F weight for a term $t$ in a document $d$ is therefore:

$$w_{\text{BM25F}}(t,d) = \frac{\widetilde{n}_L(t,d)}{\widetilde{n}_L(t,d) + k_1}\, w_{\text{RSJ}}(t,c). \tag{8}$$

When multiple streams are present and no relevance information is available at the level of document fragments, $w_{\text{RSJ}}(t,c)$ can be computed as the IDF on the entire collection disregarding streams [1], [6]; this is also the approach adopted in this paper. Notice also that $k_3$ does not appear in (8), since BM25F assumes that each term appears only once in the query. The number of free parameters in the second version of BM25F is $2T + 1$: $k_1$, plus two free parameters for each stream, $b_s$ and $v_s$. Our paper focuses on the $(2T+1)$-parameter version of BM25F: the wide array of free parameters provides additional degrees of freedom to researchers who aim to improve retrieval effectiveness, and it becomes a ground on which efficiency can significantly be improved.

### 4.2 Choice of Objective Function

In this section, the choice of the objective function for the optimization of BM25 or BM25F is discussed. In this discussion, we ultimately map a given value of the parameters for the retrieval functions to a single real number, so as to have a problem of the form (1)-(2). Hence, we need a measure of retrieval effectiveness, which in turn requires an evaluation on a given set of queries and documents. Given a set of queries $\mathcal{Q}$, a corpus of documents, and a set of relevance judgements about the documents with respect to each query, there are three natural approaches to estimate a measure of retrieval effectiveness:

(a) The entire set $\mathcal{Q}$ is used to discover the best free parameter values for the ranking function considered. This procedure has the severe drawback of being likely to overfit the training set, yielding poor generalization. However, in our experimental evaluation this is a reasonable procedure to investigate the best parameters values in a "best case" situation.

(b) $\mathcal{Q}$ is randomly partitioned into $k$ groups; then, for $i = 1, \ldots, k$, the $i$th part is used for test, and the remaining $k - 1$ parts are used for training. This approach is called *cross-validation*. Effectiveness can then be computed as the average of the effectiveness measure over the $k$ distinct test sets. This procedure is significantly more robust than the previous approach, but it has a higher computational cost.

(c) $\mathcal{Q}$ is split into a training set consisting of $x\%$ of the queries, and a test set consisting of the remaning $(100 - x)\%$ of the queries. The retrieval function is tuned on the training set using cross-validation as in (b), but the performance of the best parameter settings found using cross-validation is assessed on the (unseen) test set.

In the experiments reported in this paper, we exploit all three approaches to tune BM25 with two free parameters, while (a) is used to tune BM25F as well as BM25 with three parameters. Although (a) is simpler, approaches (b) and (c) allows us to better estimate generalization capabilities of the models discovered by the parameter optimization methods, thus investigating whether some methodologies are more susceptible to overfitting.

## 5 Experiments

### 5.1 Research Questions

We evaluate the performance of an optimization method to tune the free parameters of retrieval functions in terms of effectiveness and efficiency.

A method should be able to discover values of the free parameters that yield the highest value of the adopted performance metric, e.g., MAP or NDCG, on a given test collection. Hence, a natural research question is to investigate the *effectiveness* of the optimization methods discussed in Section 3. While Grid Search and Line Search[2] are "traditional" approaches in the IR community, RBFOpt has experimentally shown to be a very effective black-box optimization tool [23], [24] in other contexts. Therefore, our first research question is:

RQ1 Is there anything to gain in terms of effectiveness of the IR function, by switching from the traditional search methodologies to a more sophisticated black-box approach?

If the effectiveness of the tested methodologies is comparable, the next research question concerns their *efficiency*. When tuning IR functions, efficiency is measured in terms of the number of objective function evaluations, or "runs",[3] needed to discover the values of the free parameter yielding the highest value of the adopted effectiveness measure. This measure of efficiency is independent of computer architecture and widely used as a metric in the black-box optimization literature. As for the problem under consideration, each evaluation of the objective function requires training and testing a retrieval function on a large dataset; it is therefore very time consuming, whereas the optimization algorithm itself requires an amount of computing time that is negligible in comparison. Hence, a method that is more efficient in terms of number of function evaluations is also faster in terms of CPU time.[4] For this reason, if two or more methodologies have the same effectiveness, we should prefer the one that discovers the (supposedly) optimal free parameter values with fewer evaluations, because it requires fewer computational resources. Our second research question is therefore:

RQ2 Which parameter optimization method is the most efficient?

Of course, while we would like to determine the most efficient parameter optimization method overall, our empirical evaluation is limited by its computational cost, and we cannot afford to test all optimization methodologies described in the literature. Hence, we focus on the three approaches discussed in this paper as representatives of different classes of optimization algorithms.

### 5.2 Test Collections and Effectiveness Measures

The experiments are carried out on a set of diverse IR test collections designed to measure the effectiveness in ad hoc settings, where "ad hoc requests are searched against a fixed document collection" [31]. Each test collection consists of a document corpus, a set of topics and a set of relevant judgements provided for (a subset of) topic-document pairs. Each topic refers to a user information need; three textual descriptions are available for each topic: title, description, and narrative. In ad hoc IR settings, it is customary to use the topic title to simulate the query submitted by the user to express the information need related to the topic; when other topic descriptions are used, it will be explicitly stated. A brief description and some statistics on the adopted test collections are reported in Table 1.

Effectiveness is measured in terms of MAP and NDCG@20. The Average Precision (AP) for a topic is computed as the average of the precision values (the ratio of

2. We use capital letters to specify the particular implementation of these methodologies employed in this paper.

3. It is useful to note that here, "evaluation" has a different meaning than the one usually considered in IR. In the context of this paper, and in the black-box optimization literature, "evaluation" refers to the process of computing the value of the objective function.

4. In our experiments, even the algorithm that has the slowest per-iteration time spends at least 97% of the total CPU time evaluating the objective function. More specifically, RBFOpt requires approximately 0.15 seconds per iteration, while LineSearch requires a few milliseconds, but training and testing the retrieval function takes several seconds, which accounts for the vast majority of the CPU time.

relevant documents over retrieved documents) at each of the relevant documents in the ranked sequence; MAP is the mean of the AP computed over all the topics. The second measure, NDCG@20, is adopted to investigate the model capability to rank highly relevant documents at high rank positions. Unlike MAP, NDCG can take into account not only binary relevance (i.e., if a retrieved document is relevant or not relevant to the topic), but also information on relevance grades: for instance, if a retrieved document is marginally or highly relevant to the topic. We use the NDCG variant adopted in the TREC2010 Web Track: NDCG@$k$ = DCG@$k$ / IDCG@$k$, where

$$\mathrm{DCG@}k = \sum_{i=1}^{k} (2^{\mathrm{rel}_i} - 1)/\log_2(i+1),$$

and IDCG@$k$ is the DCG@$k$ computed on the top $k$ relevant documents in the pool, ranked by their (descending) degree of relevance; rel$_i$ denotes the relevance grade of documents at the $i$-th position. The relevance grades rel$_i$ vary according to the test collection: for instance, in *wt00-01* relevant documents have relevance grade rel$_i = 1$ and highly relevant documents have relevance grade rel$_i = 2$. The NDCG cutoff $k$ was set to 20 in accordance with the guideline of the TREC 2010 Web Track [32].

The number of results $m$ to compute the MAP is set to $m = 1000$ for the *robust-04* and the *wt00-01* test collections; $m = 10000$ is employed for the *tb04-06* and the *wt10-12* test collections, following guidelines reported respectively in [33] and [32].

## 5.3 Experimental Methodology

We optimize the free parameters of the two retrieval models described in Section 4: BM25 and BM25F.

BM25 has three free parameters: $b$, $k_1$ and $k_3$. The default values are $b = 0.75$, $k_1 = 1.2$ and $k_3 = 0$. We carry out two sets of experiments on the optimization of BM25 parameters. In the first set, we optimize only $b$ and $k_1$, for the following reasons:

- $k_3$ determines the effect of the frequency of a term $t$ in the query, $n_L(t,q)$; however, in ad hoc settings a query coincides with the title of the topic, which is usually short and results in $n_L(t,q) = 1$. For example, in the *robust-04* test collections there are no title queries with $n_L(t,q) > 1$, while in *wt00-01* there are only three such title queries.
- By focusing on two parameters only, we can perform optimization via Grid Search with fine granularity. Since Grid Search is a popular approach, this allows us to verify if this popularity is justified in an ideal setting.

In the second set of experiments we optimize all three free parameters. Since $k_3$ affects the BM25 weight only if $n_L(t,q) > 1$, and this is a rare occurence in our test collections when considering title descriptions, we consider two different test collections. The first one, labeled *robust-04n*, is the *robust-04* test collection with narrative description: in 195 of the 250 topic narrative descriptions $n_L(t,q) > 1$, and the range for the maximum values of $n_L(t,q)$ for term $t$ among those topics is $[2,6]$. The second one, labeled *robust-04v*, is the *robust-04* test collection developed in [34] to investigate verbose queries. A verbose query is defined as one that is "too long, detailed, uses or is expressed in more words than are needed" [34]. This test collection relies on a subset of the topics of *robust-04*: 30 topics are selected and verbose queries are built by extracting some passages from the documents that are relevant to the topic; these passages contain information relevant to the topic, that is, these passages are what an assessor would read if he or she assessed a document as relevant. Additional details on the *robust-04v* test collection and the verbose queries can be found in [34]. In this test collection $n_L(t,q) > 1$ for 29 of the 30 topics, and the range for the maximum values of $n_L(t,q)$ for the term $t$ among those topics is $[2,10]$.

The ranges of the free parameters of BM25 are:

$$b \in [0,1] \qquad k_1 \in [0,10] \qquad k_3 \in [0,1000].$$

The range for $k_1$ is selected in accordance with the objective function contours for the 2-parameter model (BM25) reported in [1]. The range for $k_3$ is suggested in [35].

In the set of experiments involving only 2 free parameters, we perform a grid search on the range $[0,1]$ with a step of 0.01 for the parameter $b$, and the range $[0,10]$ with a step 0.1 for the parameter $k_1$. This leads to $(1+1/0.01)\cdot(1+10/0.1) = 10201$ tested configurations for each test collection.

The BM25F version adopted in [1] has $2S+1$ free parameters[5]; in this paper the following ranges are investigated:

$$b_s \in [0,1] \qquad v_s \in [0,100] \qquad k_1 \in [0,10].$$

The range for $k_1$ is selected as for BM25. Ranges for $v_s$ are not explicitly discussed in [6] and [1]; however, in [6] the contour plots in the case of BM25F with title, body and anchor streams suggest $v_s \in [0,100]$ for the "simple" BM25F version and $v_s \in [0,1]$ for the second version[6]; moreover, when considering title and body fields, [6] states that "the range of weights explored is between 0 and 3". We investigate a wide range $[0,100]$, since the literature exhibits little agreement in the range of free parameters when certain stringent mathematical properties such as the range of the parameters $b_s$ cannot be considered. The experiments on BM25F are carried out on the *wt00-01* test collection; each document is represented using four streams: title and body extracted from the Web page, anchor text, and terms extracted from the URL.

## 5.4 Experimental System and Settings

All experiments are carried out via Apache Lucene version 4.7.2 [36]. Retrieval models are re-implemented using the index API, and Porter stemmer is adopted both for indexing and retrieval. No stopping list is adopted to build the index; the stoplist available in the Lemur Toolkit is adopted at retrieval time. The anchors are extracted through the Galago [37] functionality to harvest links.

5. The number of parameters could be reduced by 1 as suggested in [1] by setting one $v_s$ arbitrarily to 1.

6. Remarks reported on the second version suggest $[0,1]$ as a good range for searching $v_s$ values.

TABLE 1
Description of the test collections adopted in the experiments.

| Test Collection | Corpus | Number of documents | Topic sets |
|---|---|---|---|
| *robust-04* | Financial Times (FT), Federal Register (FR), Foreign Broadcast Inf. Serv. (FBIS), the LA Times (LATIMES) | 528,155 | 301-450; 600-700 |
| *wt00-01* | WT10g | 1,692,096 | 451-550 |
| *tb04-06* | Gov2 | 25,205,179 | 700-850 |
| *wt10-12* | ClueWeb09 Cat B | 50,220,423 | 51-200 |

The value of the measures of effectiveness is computed through the Lemur Toolkit Utility *ireval*, since it can be easily included in our pipeline for free parameter tuning of IR functions.

For the optimization, the Line Search methodology described in Section 3.1 and based on [1] is implemented in Python. The initial point is set to $\mathbf{x}^0 = \mathbf{0}$. We use RBFOpt version 2.0.0, available at [28]. All parameters for RBFOpt are left to their default values, except for the initialization strategy, for which we try *all_corners* and *lhd_maximin* as described in Section 3.2.1.

## 5.5 Results

### 5.5.1 BM25 with 2 free parameters

The results for the optimization of $b$ and $k_1$ are reported in Table 2 and Table 3 for the MAP and the NDCG@20 effectiveness measure, respectively. The column labels are explained in the caption of the tables.

We adopt the paired t-test to check whether the differences among the results obtained are statistically significant at a significance level of $\alpha = 0.05$. In the context of this paper, the t-test is applied to the population determined by the performance measure (MAP or NDCG@20) recorded on individual queries. Thus, the number of samples is the number of queries in the experiment, and each algorithm has the same sample size. When marked with an asterisk, a result exhibits a statistically significant difference (p-value $< 0.05$) from the result with the default parameter values.

In terms of MAP, the three optimizations methods are able to improve the effectiveness over the baseline: the differences from the results with the default parameter values are statistically significant for all test collections except *wt10-12*. The same results are observed when optimizing in terms of NDCG@20: differences from the results with the default parameter values are statistically significant for all test collections except *wt10-12*.

Looking at e@h$Y$, i.e., the number of evaluations required by algorithm "$Y$" (where "G", "L", and "R" refer to Grid Search, Line Search, and RBFOpt, respectively) to find the highest value of MAP and NDCG@20, neither Line Search nor RBFOpt is able to clearly outperform the other across all the collections. However, Tables 2 and 3 report only the evaluation at which the highest value is observed for each algorithm. A more comprehensive picture on the efficiency of the methods can be obtained by comparing, after every objective function evaluation, the highest value of the effectiveness measure reached until that point. Fig. 1 reports the highest value of NDCG@20 obtained by BM25 at each evaluation. We focus on NDCG@20 since the analysis of the two-dimensional contour plots ($b$, $k_1$) showed a similar behavior for both MAP and NDCG@20 in most collections (*robust-04, wt00-01, tb04-06*), while the contour plot of NDCG@20 was relatively rough in *wt10-12* (see Figure 2a), thus providing a challenging environment for assessment of optimization algorithms. Fig. 1 shows that RBFOpt reaches its maximum quickly when the initial sample points are selected by the default strategy (RBFOpt lhd_maximin, dotted curve in the plots).

To further investigate this claim, Table 4 reports: the number of evaluations e@h$Y$ needed to obtain the highest NDCG@20 value, h$Y$; the number of evaluations e@fY≈hY needed to obtain a value of NDCG@20 for which no statistically significant difference is detected from the highest value obtained with the same optimization method; the number of evaluations e@fL≈hR needed by Line Search to obtain a value of NDCG@20 for which no statistically significant difference is detected from the highest value obtained by RBFOpt at the same evaluation[7]. Looking at these results, more specifically e@fL≈hL and e@fR≈hR, we conclude that RBFOpt reaches close-to-maximum NDCG@20 values not later, and at times even earlier, than Line Search. RBFOpt's ability to reach near-optimal parameter values much earlier than Line Search indicates that it can be more efficient than Line Search in an IR context that requires adapting a retrieval function to diverse queries and collections.

To further investigate RBFOpt's capabilities of finding effective parameter values within a small number of objective function evaluations, we compare the two initialization strategies *lhd_maximin* and *all_corners*: results in terms of NDCG@20 are reported in Table 5. There is no significant difference between the two in terms of the value of NDCG@20 attained. In three out of four test collections (*wt00-01*, *tb04-06* and *wt10-12*), *all_corners* finds a maximum within fewer evaluations than *lhd_maximin*; however, looking at the curves in Fig. 1, a greater number of evaluations is needed by *all_corners* to obtain a NDCG@20 value close to the highest one, see the dashed curve.

In terms of the actual points evaluated in the 2-dimensional free parameter space $(b, k_1)$, *lhd_maximin* and *all_corners* behave differently: Fig. 2 reports the points evaluated by the Line Search (Fig. 2b), RBFOpt with *lhd_maximin* (Fig. 2c), and RBFOpt with *all_corners* (Fig 2d). Interestingly, *all_corners* is able to find a different region ($b \in [0.05, 0.25]$,

7. We remark that not detecting a difference at a given significance level does not imply that the values being compared are the same (i.e., that the null hypothesis is true), but it gives an indication that their distribution may look similar.

TABLE 2
MAP in a "best case" situation when tuning BM25 parameters $b$ and $k_1$. hG, hL and hR denote the highest values obtained respectively by Grid Search (G), Line Search (L) and RBFOpt (R). e@hL and e@hR denote the number of evaluations needed to obtain hL and hR. #e is the total number of evaluation performed by the optimization methods. The values in parentheses are the number of evaluations needed to obtain a MAP value equal or greater than that obtained by Grid Search. Results marked with an asterisk are significantly different from the results with the default parameter values.

| Collection | Default | Grid Search | | | Line Search | | | | | RBFOpt | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $b$ | $k_1$ | hG | $b$ | $k_1$ | hL | e@hL | #e | $b$ | $k_1$ | hR | e@hR | #e |
| *robust-04* | 0.2352 | 0.30 | 0.7 | 0.2545* | 0.3066 | 0.6976 | 0.2546* | 38 (38) | 90 | 0.3050 | 0.6962 | 0.2546* | 21 (20) | 165 |
| *wt00-01* | 0.1720 | 0.21 | 0.6 | 0.2101* | 0.2480 | 0.4960 | 0.2096* | 14 (-) | 60 | 0.2206 | 0.6041 | 0.2099* | 165 (-) | 165 |
| *tb04-06* | 0.2670 | 0.33 | 0.9 | 0.3166* | 0.3488 | 0.6976 | 0.3158* | 15 (-) | 65 | 0.3292 | 0.8860 | 0.3167* | 27 (26) | 162 |
| *wt10-12* | 0.0885 | 0.00 | 3.3 | 0.0903 | 0.0000 | 3.4450 | 0.0903 | 40 (40) | 100 | 0.2919 | 4.6228 | 0.0900 | 9 (-) | 165 |

TABLE 3
NDCG@20 in a "best case" situation when tuning BM25 parameters $b$ and $k_1$.

| Collection | Default | Grid Search | | | Line Search | | | | | RBFOpt | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $b$ | $k_1$ | hG | $b$ | $k_1$ | hL | e@hL | #e | $b$ | $k_1$ | hR | e@hR | #e |
| *robust-04* | 0.3975 | 0.39 | 1.0 | 0.4177* | 0.3976 | 0.9395 | 0.4171* | 42 (-) | 95 | 0.3920 | 0.9931 | 0.4181* | 39 (25) | 165 |
| *wt00-01* | 0.2941 | 0.33 | 2.2 | 0.3297* | 0.3312 | 2.322 | 0.3285* | 67 (-) | 131 | 0.3157 | 2.1683 | 0.3283* | 117 (-) | 165 |
| *tb04-06* | 0.3865 | 0.43 | 1.4 | 0.4444* | 0.4244 | 1.6976 | 0.4450* | 16 (16) | 70 | 0.4298 | 1.3899 | 0.4445* | 156 (27) | 165 |
| *wt10-12* | 0.1248 | 0.03 | 0.1 | 0.1262 | 0.0000 | 1.4555 | 0.1265 | 56 (11) | 105 | 0.0015 | 0.0001 | 0.1282 | 160 (44) | 165 |

TABLE 4
Comparison between RBFOpt and Line Search in terms of number of evaluations. hL and hR denote the highest value of NDCG@20 obtained respectively by Line Search (L) and RBFOpt (R); e@hL and e@hR denote the number of evaluations needed to obtain those values. fY≈hY denotes the first value obtained by method $Y$ and not statistical different from hY; e@fY≈hY denotes the number of evaluations needed by method $Y$ to obtain fY≈hY.

| Collection | Line Search | | | | | | RBFOpt | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | hL | e@hL | fL≈hL | e@fL≈hL | fL≈hR | e@fL≈hR | hR | e@hR | fR≈hR | e@fR≈hR |
| *robust-04* | 0.4171 | 42 | 0.4155 | 16 | 0.4158 | 17 | 0.4181 | 39 | 0.4149 | 9 |
| *wt00-01* | 0.3285 | 67 | 0.3159 | 14 | 0.3159 | 14 | 0.3283 | 117 | 0.3221 | 11 |
| *tb04-06* | 0.4450 | 16 | 0.4397 | 15 | 0.4321 | 14 | 0.4445 | 156 | 0.4402 | 13 |
| *wt10-12* | 0.1265 | 59 | 0.1252 | 7 | 0.1252 | 7 | 0.1282 | 156 | 0.1234 | 1 |

TABLE 5
Comparison in terms of NDCG20 in a "best scenario" situation when tuning BM25 parameters $b$ and $k_1$ with two different starting point strategies: *all_corners* and *lhd_maximin* (RBFOpt default). Differences between the NDCG@20 values obtained by the two strategies are not statistically significant.

| Collection | RBFOpt lhd_maximin | | | | | RBFOpt all_corners | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $b$ | $k_1$ | hR | e@hR | #e | $b$ | $k_1$ | hR | e@hR | #e |
| *robust-04* | 0.3920 | 0.9931 | 0.4181 | 39 | 165 | 0.3913 | 0.9781 | 0.4183 | 105 | 154 |
| *wt00-01* | 0.3157 | 2.1683 | 0.3283 | 117 | 165 | 0.3269 | 2.3268 | 0.3285 | 80 | 154 |
| *tb04-06* | 0.4298 | 1.3899 | 0.4445 | 156 | 165 | 0.4371 | 1.3095 | 0.4444 | 75 | 154 |
| *wt10-12* | 0.0015 | 0.0001 | 0.1282 | 160 | 165 | 0.2099 | 6.1563 | 0.1259 | 20 | 154 |

$k_1 \in [4, 6.2]$) where the contour plot shows high values of NDCG@20; this is evident also from the values of $(b, k_1)$ reported in Table 3 and Table 5 for *wt10-12*: while the optimal parameter values obtained by Grid Search, Line Search and RBFOpt with *lhd_maximin* are approximately in the same area ($b \approx 0$ and small values of $k_1$), RBFOpt with *all_corners* provides parameter values in a different part of the space ($b = 0.2099, k_1 = 6.1563$).

In the remainder of this paper we rely on *lhd_maximin*, because Fig. 1 suggests that it is more efficient than *all_corners*, and *all_corners* is impractical when optimizing a large number of parameters.

We conclude this section devoted to the optimization of two-parameter BM25 and summarize our results with respect to the two research questions RQ1 and RQ2 stated earlier in the paper, relative to the tuning of $b$ and $k_1$ only.

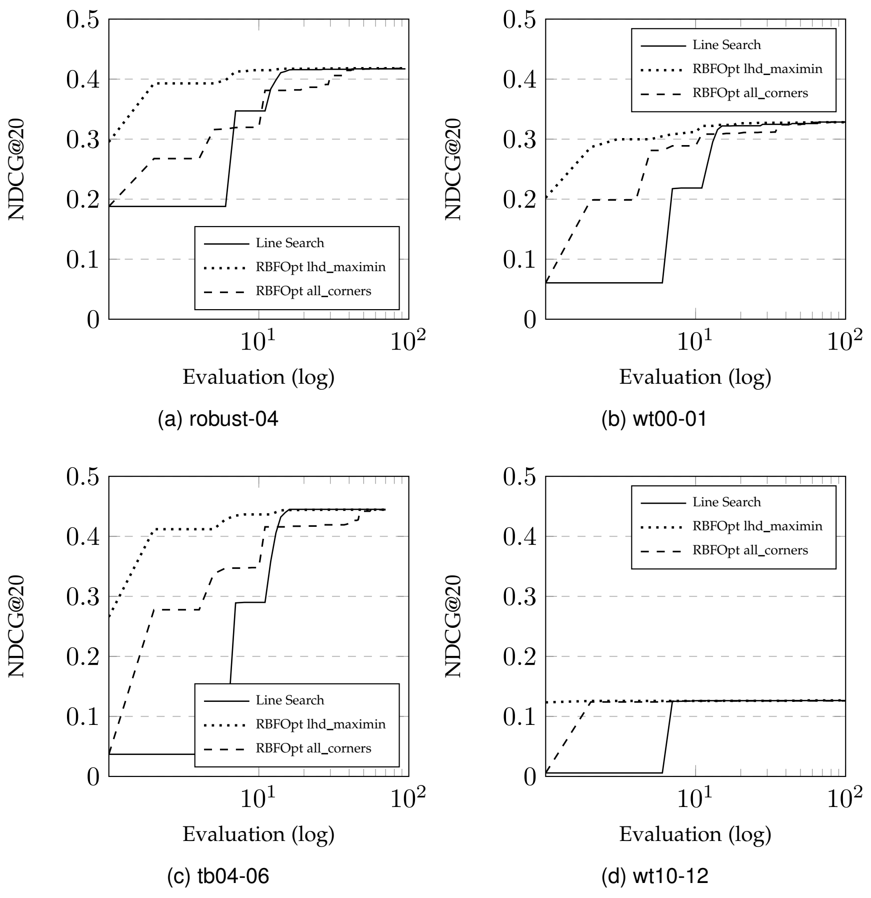


Fig. 1. Highest NDCG@20 obtained at a given evaluation by BM25 for the diverse test collections; plots stop at the maximum number of evaluations performed by Line Search.

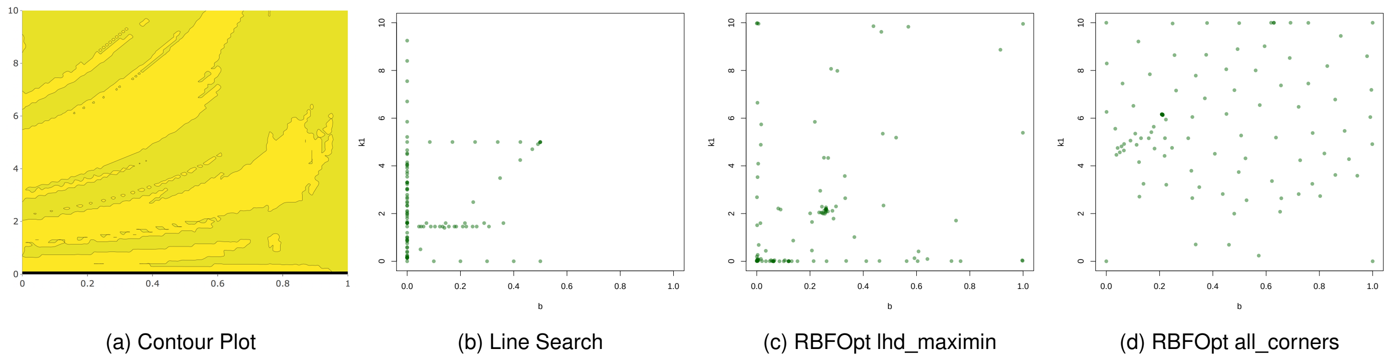


Fig. 2. Contour plot of NDCG@20 obtained with Grid Search (Fig. 2a), points (b,$k_1$) evaluated by Line Search (Fig. 2b), RBFOpt with *lhd maximin* (Fig. 2c) and RBFOpt with *all the corners* (Fig. 2d) when tuning BM25 in the *wt10-12* test collection. The plot in Fig. 2a was obtained using 26 contour levels; lighter colors denote areas with higher NDCG@20; for $k1 \in [0, 0.1]$ contour lines are spaced close to each other, meaning that NDCG@20 values change rapidly. Plots in Fig. 2c and Fig. 2d are based on the first 105 evaluations, i.e. the total number of evaluations performed by Line Search for *wt10-12* — see Table 3.

RQ1 Empirical results show that there is no significant gain in terms of effectiveness of the retrieval function when switching from the traditional search methodologies to a more sophisticated black-box approach.

RQ2 Both Line Search and RBFOpt can be efficient, since they are able to find effective free parameter settings even with a small number of evaluations.

We now proceed to investigate these questions in the context of retrieval functions with a larger number of free parameters.

### 5.5.2 BM25 with three free parameters and BM25F

Table 6 reports results obtained when optimizing $b$, $k_1$ and $k_3$. There is no significant difference among the highest MAP and NDCG@20 values obtained by Line Search and RBFOpt. On the other hand, the number of evaluations required by RBFOpt to obtain the highest value of the effectiveness measure is lower than the number of evaluations required by Line Search, both when considering the highest value hY, and when considering the first value fL greater than the highest obtained by RBFOpt. Fig. 3 reports the maximum MAP and NDCG@20 obtained at a given evaluation: these plots show trends similar to those observed when optimizing two parameters, but a higher number of evaluations is needed by Line Search to achieve the same MAP/NDCG@20 values obtained by RBFOpt.

Results for the optimization of nine free parameters of BM25F are reported in Table 7. The differences between the highest value of MAP and NDCG@20 values obtained by Line Search and RBFOpt are significant ($p\text{-value} < 0.05$); however, there is no significant difference between the highest value hR obtained by RBFOpt, and the first value fL greater than hR and obtained by Line Search, or the highest value within the first 200 evaluations (the maximum number of evaluations performed by RBFOpt). However, notice that there is a large difference between the total number of evaluation performed by Line Search and RBFOpt: 999 vs 200 when optimizing in terms of NDCG@20. Thus, we investigate if better results in terms of NDCG@20 can be obtained by increasing the maximum number of allowed function evaluations in RBFOpt. To this end, we performed another experiment, allowing RBFOpt to perform as many evaluations as Line Search.[8] The highest value of NDCG@20 obtained with these settings was 0.3251, only marginally smaller than Line Search, and required 355 evaluations; the difference with the highest value obtained by Line Search was not statistically significant (p-value=0.3666), and the number of evaluations required (355) was smaller than that required by Line Search (682), similarly to the previous experiment.

In Fig. 4 we plot the maximum value of MAP and NDCG@20 obtained at a given evaluation for the optimization of BM25F. The plots show that RBFOpt attains high values of the performance metrics very quickly, much faster than Line Search, although in some cases Line Search eventually finds parameters yielding slightly larger MAP/NDCG@20 values.

To conclude this section, we go back to our research questions, now taking into account the optimization of a larger number of free parameters.

RQ1 We reach a similar conclusion: even when optimizing a larger number of parameters, there is no significant gain in terms of effectiveness of the IR function by switching from a more traditional search methodology, i.e., Line Search, to a more sophisticated black-box approach such as RBFOpt.

RQ2 RBFOpt may provide effective parameter settings in a much smaller number of evaluations.

Our findings concerning RQ2 indicate that for application scenarios in which each evaluation of a parameter configuration is time-consuming, Line Search may not be a viable option. As previously mentioned, a situation in which this scenario may occur is that of using a training/test split of the query set or a cross-validation approach, to avoid overfitting; in such a case, each evaluation may require long computing times, and efficiency is a key requirement. Experiments reported in the next section investigate precisely this scenario.

8. The settings adopted for RBFOpt were: max_evaluations=1000; max_iterations=1000.

### 5.5.3 *Evaluation by cross-validation*

Table 8 reports results when performance is measured by 5-fold cross validation. Line Search and RBFOpt were comparable in terms of effectiveness: the only significant difference is that between the results on fold 3, where RBFOpt was able to outperform Line Search. Table 9 reports results when optimization is performed using 5-fold cross validation over a training set, and performance assessment takes place on a separate test set. The free parameter settings obtained by both optimization methods outperformed the default values, and RBFOpt performed slightly better than Line Search (differences are statistically significant). We remark that the partitioning of the training set into folds is the same for both optimization methods.

Results in this section are in line with our previous analysis: all the tested methods are similarly effective (RQ1), but the plots of the maximum value of NDCG@20 at a given evaluation show that RBFOpt is more efficient (RQ2).

We conclude our analysis with final remarks. Our experiments indicate that performance metrics and datasets studied in this paper seem to yield objective functions with few, if any, local optima with large basin of attraction: even a simple local ascent method such as line search always manages to find a point with near-optimal (as far as we know) objective function value. This indicates that black-box optimization methods that only perform local search may be a good choice for these problems. Of course, structural properties are unknown a priori, hence a global search method remains a safer choice. Furthermore, while tested methods achieve comparable values of the performance metrics, RBFOpt is considerably more efficient, quickly finding parameterizations of the retrieval function yielding high performance – much faster than line search.

## 6 Conclusion

In this paper, we reported our findings on the problem of automatically tuning the free parameters of retrieval functions with the aim of improving the effectiveness of an IR system. We addressed this problem in terms of mathematical optimization with a computable but not analytically available objective function.

After comparing the performance of several optimization approaches to solving this problem, we showed that there is little – if any – room for improvement when it comes to identifying more effective parameters of the retrieval function by using more sophisticated algorithms. However, our experiments indicate that there is a lot to gain in terms of efficiency. In particular, a black-box optimization methodology can quickly find parameters yielding near-optimal (as far as we know) performance: our tests used the algorithm implemented in the open-source library RBFOpt.

For applications in which efficiency plays an important role, the computational evaluation provided in this paper shows that black-box optimization can yield great benefits

TABLE 6
BM25 Optimization of $b$, $k_1$ and $k_3$ for the *robust-04v* and the *robust-04n* test collections. hL and hR denote the highest value of the effectiveness measure obtained by Line Search (L) and RBFOpt (R), while e@hL and e@hR the evaluation at which the hL and hR were obtained. #e is the total number of evaluations performed by the optimization method. fL is the first value of the effectiveness measure obtained by Line Search and greater than hR, and e@fL is the evaluation at which the value fL was attained. Differences among MAP and NDCG@20 are not statistically significant.

| Test Collection | Metric | Line Search | | | | | RBFOpt | | |
|---|---|---|---|---|---|---|---|---|---|
| | | hL | e@hL | #e | fL | e@fL | hR | e@hR | #e |
| *robust-04n* | MAP | 0.1270 | 124 | 228 | 0.1270 | 124 | 0.1262 | 72 | 170 |
| | NDCG@20 | 0.2498 | 325 | 403 | 0.2461 | 60 | 0.2457 | 40 | 170 |
| *robust-04v* | MAP | 0.2230 | 237 | 470 | – | – | 0.2233 | 102 | 170 |
| | NDCG@20 | 0.5497 | 599 | 903 | 0.5487 | 466 | 0.5484 | 162 | 170 |

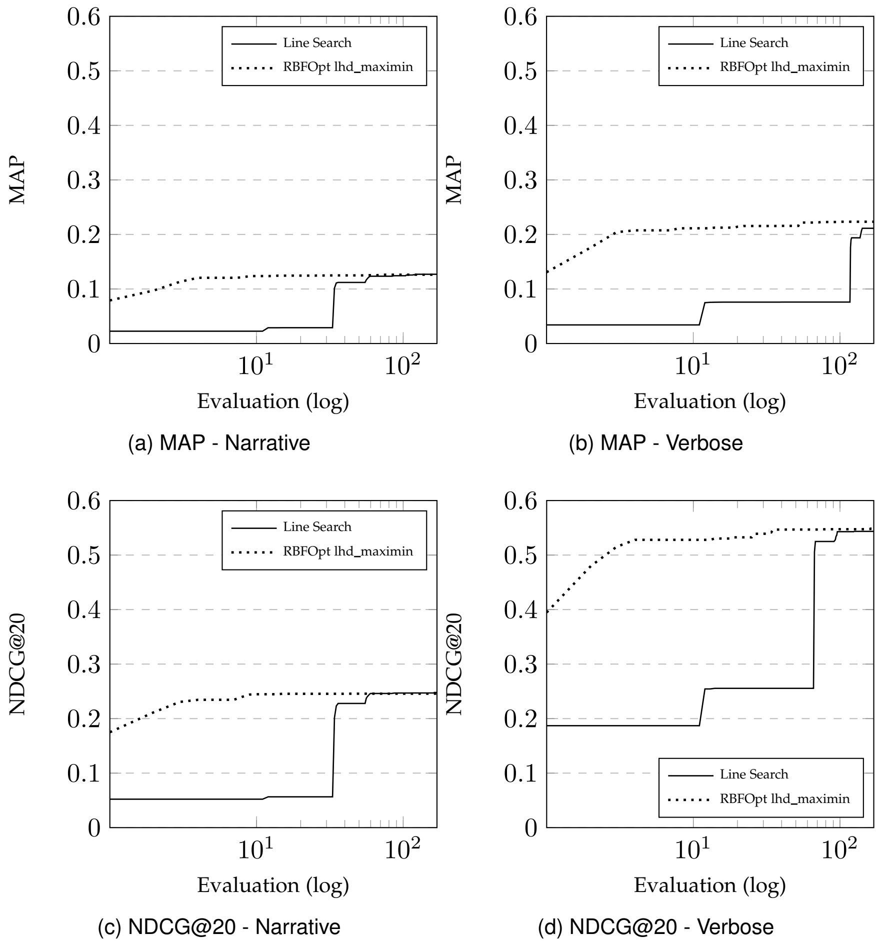
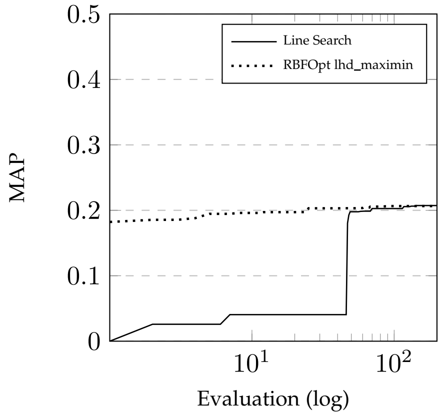


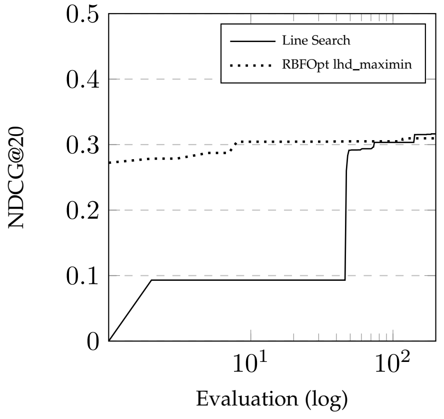
Fig. 3. Maximum MAP and NDCG@20 obtained at a given evaluation by BM25 with three free parameters.

TABLE 7
Optimization of BM25F. hY denotes the value of the effectiveness measure by method $Y$. e@hY denotes the evaluation at which the highest value hY of the effectiveness measure was obtained. #e is the total number of evaluation performed by the optimization method. fL is the first value of the effectiveness measure obtained by Line Search and greater than the highest obtained by RBFOpt, and e@fL is the evaluation at which fL was attained. Results marked with an asterisk are significantly different both from fL and hR.

| Metric | Line Search | | | | | RBFOpt | | |
|---|---|---|---|---|---|---|---|---|
| | hL | e@hL | #e | fL | e@fL | hR | e@hR | #e |
| MAP | 0.2105* | 1716 | 2084 | 0.2073 | 143 | 0.2065 | 90 | 200 |
| NDCG@20 | 0.3270* | 682 | 999 | 0.3152 | 143 | 0.3096 | 126 | 200 |



(a) MAP



(b) NDCG@20

Fig. 4. Maximum MAP and NDCG@20 obtained at a given evaluation by BM25F for *wt00-01*.

TABLE 8
NDCG@20 using 5-fold cross-validation on *robust-04*. A subset of 245 topics is extracted at random from the 249 topics of the *robust-04* test collections. The subset is randomly split in 5 parts: 4 parts were used for training and the remaining one for testing. The best value for the $i$th training set (best ts$i$) is evaluated on the corresponding test set (test), i.e. the part (fold) not in the $i$th training set. The asterisk denotes a NDCG@20 value statistically different from default with p-value $< 0.05$.

| Opt | Configuration | | | #e | NDCG@20 | |
|---|---|---|---|---|---|---|
| | | b | $k_1$ | | training | test |
| LS | best ts1 | 0.3899 | 0.9395 | 40 | 0.4239 | 0.3852 |
| | best ts2 | 0.3984 | 0.9395 | 39 | 0.4049 | 0.4619 |
| | best ts3 | 0.2076 | 0.9950 | 91 | 0.4166 | 0.4050 |
| | best ts4 | 0.3976 | 0.9395 | 42 | 0.4183 | 0.4083 |
| | best ts5 | 0.3975 | 0.9395 | 43 | 0.4189 | 0.4064 |
| | average | | | | | 0.4134 |
| RBFOpt | best ts1 | 0.3851 | 1.1232 | 31 | 0.4251 | 0.3861 |
| | best ts2 | 0.2246 | 1.1028 | 99 | 0.4069 | 0.4473 |
| | best ts3 | 0.3751 | 1.0521 | 38 | 0.4166 | 0.4183* |
| | best ts4 | 0.3826 | 1.1381 | 117 | 0.4200 | 0.4052 |
| | best ts5 | 0.3965 | 0.9015 | 111 | 0.4190 | 0.4060 |
| | average | | | | | 0.4126 |

TABLE 9
NDCG@20 on *robust-04* using 2 random splits, one for training using 5-fold cross-validation and one for testing. Each fold consists of 40 topics, each training set consists of 160 topics and each validation consists of 40 topics. The test set comprises 49 topics. The configuration with highest average NDCG@20 over all the validation folds (h_avg_val) is evaluated on the test set (last column). $d$ and $l$ denote an NDCG@20 value statistically different (p-value $< 0.05$) respectively from the default and from Line Search.

| Opt | Configuration | | | #e | NDCG@20 | |
|---|---|---|---|---|---|---|
| | | b | $k_1$ | | h_avg_val | test |
| - | default | 0.75 | 1.2 | - | - | 0.3699 |
| LS | best ts1 | 0.4006 | 0.8488 | 37 | 0.4231 | - |
| | best ts2 | 0.4244 | 0.8488 | 16 | 0.4218 | - |
| | best ts3 | 0.3939 | 1.1119 | 130 | 0.4235 | $0.3930^{d}$ |
| | best ts4 | 0.3988 | 0.9395 | 60 | 0.4234 | - |
| | best ts5 | 0.3939 | 0.8488 | 70 | 0.4225 | - |
| RBFOpt | best ts1 | 0.2911 | 1.0336 | 155 | 0.4230 | - |
| | best ts2 | 0.2123 | 1.1717 | 39 | 0.4229 | - |
| | best ts3 | 0.3915 | 1.1304 | 59 | 0.4242 | $0.3943^{l,d}$ |
| | best ts4 | 0.3983 | 1.1733 | 39 | 0.4234 | - |
| | best ts5 | 0.384 | 1.0743 | 144 | 0.4238 | - |

over commonly used approaches such as grid search and line search. Although we cannot give a definitive answer to the ultimate question of what is the best methodology to choose a retrieval function and its parameters, we presented compelling evidence that state-of-the-art mathematical optimization techniques can help take such decision in an efficient and effective way.

## Acknowledgments

The authors would like to thank the editor and the reviewers for the useful comments which helped to improve significantly the quality of the paper. The research of A. C. was partially conducted at the Future Resilient Systems at the Singapore-ETH Centre (SEC). The SEC was established as a collaboration between ETH Zurich and National Research Foundation (NRF) Singapore (FI 370074011) under the auspices of the NRF's Campus for Research Excellence and Technological Enterprise (CREATE) programme.

The research of E. Di B. and M. M. has partially been funded by the European Union's Horizon 2020 Research and Innovation Programme under the Marie Skłodowska-Curie Grant Agreement No. 721321.

## References

[1] M. Taylor, H. Zaragoza, N. Craswell, S. Robertson, and C. Burges, "Optimisation methods for ranking functions with multiple parameters," in *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, ser. CIKM '06. New York, NY, USA: ACM, 2006, pp. 585–593.

[2] C. Zhai and J. Lafferty, "A study of smoothing methods for language models applied to information retrieval," *ACM TOIS*, vol. 22, no. 2, pp. 179–214, 2004.

[3] S. Clinchant and E. Gaussier, "Information-based models for ad hoc ir," in *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '10. New York, NY, USA: ACM, 2010, pp. 234–241.

[4] J. J. Rocchio, *Relevance Feedback in Information Retrieval*, 1971, pp. 313–323.

[5] S. Robertson and H. Zaragoza, "The Probabilistic Relevance Framework: BM25 and Beyond," *Foundations and Trends® in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.

[6] S. Robertson, H. Zaragoza, and M. Taylor, "Simple BM25 extension to multiple weighted fields," in *Proceedings of the Thirteenth ACM conference on Information and knowledge management - CIKM '04*. New York, New York, USA: ACM Press, 2004, p. 42.

[7] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.

[8] A. Broder, "A taxonomy of web search," *SIGIR Forum*, vol. 36, no. 2, pp. 3–10, 2002.

[9] J. J. Rocchio and G. Salton, "Information search optimization and interactive retrieval techniques," in *Proceedings of the November 30–December 1, 1965, Fall Joint Computer Conference, Part I*, ser. AFIPS '65 (Fall, part I). New York, NY, USA: ACM, 1965, pp. 293–305.

[10] C. Carpineto and G. Romano, "A survey of automatic query expansion in information retrieval," *ACM Computing Surveys*, vol. 44, no. 1, pp. 1–50, 2012.

[11] W. Croft and J. Lafferty, Eds., *Language Modeling for Information Retrieval*, ser. Kluwer International Series on Information Retrieval. Kluwer Academic Publishers, 2002, vol. 13.

[12] S. Robertson and K. Spärck Jones, "Relevance weighting of search terms," *Journal of the American Society for Information Science*, vol. 27, pp. 129–146, 1976.

[13] M. Taylor, J. Guiver, S. Robertson, and T. Minka, "SoftRank: Optimizing Non-smooth Rank Metrics," in *Proceedings of the 2008 International Conference on Web Search and Data Mining*, ser. WSDM '08. New York, NY, USA: ACM, 2008, pp. 77–86.

[14] K. Z. Hongyuan Zha Yi Chang and G.-R. Xue, "Learning the gain values and discount factors of discounted cumulative gains," *IEEE TKDE*, vol. 26, no. 2, pp. 391–404, 2014.

[15] R. Deveaud, J. Mothe, and J.-Y. Nie, "Learning to Rank System Configurations," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management - CIKM '16*. New York, New York, USA: ACM Press, 2016, pp. 2001–2004.

[16] C. Buckley and E. Voorhees, "Retrieval System Evaluation," in *TREC: Experiment and Evaluation in Information Retrieval*. The MIT Press, 2005, ch. 3.

[17] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of IR techniques," *ACM TOIS*, vol. 20, no. 4, pp. 422–446, 2002.

[18] A. Conn, K. Scheinberg, and L. Vicente, *Introduction to Derivative-Free Optimization*, ser. MOS-SIAM series on optimization. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2009.

[19] T. G. Kolda, R. M. Lewis, and V. Torczon, "Optimization by direct search: New perspectives on some classical and modern methods," *SIAM Review*, vol. 45, no. 3, pp. 385–482, 2003.

[20] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[21] H.-M. Gutmann, "A radial basis function method for global optimization," *Journal of Global Optimization*, vol. 19, no. 3, pp. 201–227, 2001.

[22] R. G. Regis and C. A. Shoemaker, "A stochastic radial basis function method for the global optimization of expensive functions," *INFORMS Journal on Computing*, vol. 19, no. 4, pp. 497–509, 2007.

[23] A. Costa and G. Nannicini, "RBFOpt: an open-source library for black-box optimization with costly function evaluations," Optimization Online, Tech. Rep. 2014-09-4538, 2017.

[24] G. I. Diaz, A. Fokoue, G. Nannicini, and H. Samulowitz, "An effective algorithm for hyperparameter optimization of neural networks," *IBM Journal of Research and Development*, vol. 61, no. 4/5, 2017.

[25] T. Wortmann, A. Costa, G. Nannicini, and T. Schroepfer, "Advantages of surrogate models for architectural design optimization," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 29, pp. 471–481, 2015.

[26] T. Wortmann, C. Waibel, G. Nannicini, R. Evins, T. Schroepfer, and J. Carmeliet, "Are genetic algorithms really the best choice for building energy optimization?" in *Symposium on Simulation for Architecture & Urban Design (SimAUD)*, 2017.

[27] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.

[28] "Rbfopt library for black-box optimization." [Online]. Available: https://github.com/coin-or/rbfopt/releases/tag/2.0.0

[29] T. Roelleke, *Information Retrieval Models: Foundations and Relationships*, 1st ed. Morgan & Claypool Publishers, 2013.

[30] K. Spärck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[31] D. Harman, *Information Retrieval Evaluation*, 1st ed. Morgan & Claypool Publishers, 2011.

[32] C. L. A. Clarke, N. Craswell, I. Soboroff, and G. V. Cormack, "BIT at TREC 2010 blog track: Faceted blog distillation," in *Proceedings of The Nineteenth Text REtrieval Conference, TREC 2010, Gaithersburg, Maryland, USA, November 16-19, 2010*, vol. Special Publication 500-294. NIST, 2010.

[33] S. Büttcher, C. L. A. Clarke, and I. Soboroff, "The TREC 2006 terabyte track," in *Proceedings of the Fifteenth Text REtrieval Conference, TREC 2006, Gaithersburg, Maryland, USA, November 14-17, 2006*, vol. Special Publication 500-272. NIST, 2006.

[34] E. Di Buccio, M. Melucci, and F. Moro, "Detecting verbose queries and improving information retrieval," *Information Processing and Management*, vol. 50, no. 2, pp. 342–360, 2014.

[35] S. Walker, S. E. Robertson, M. Boughanem, G. J. F. Jones, and K. S. Jones, "Okapi at TREC-6 automatic ad hoc, VLC, routing, filtering and QSDR," in *Proceedings of The Sixth Text REtrieval Conference, TREC 1997, Gaithersburg, Maryland, USA, November 19-21, 1997*, vol. Special Publication 500-240. NIST, 1997, pp. 125–136.

[36] "Apache Lucene Core." [Online]. Available: http://lucene.apache.org/core/

[37] "Lemur Project Components: Galago." [Online]. Available: http://lemurproject.org/galago.php

**Alberto Costa** is a Postdoctoral Researcher at the National University of Singapore. He holds B.Eng and M.Eng degrees from the University of Padova, Italy, and a Ph.D. in Operations Research from École Polytechnique, France. Dr. Costa's research interests include black-box and robust optimization, clustering in complex networks, and information retrieval.

**Emanuele Di Buccio** is a Postdoctoral Researcher at University of Padova, Italy. He holds B.Eng and M.Eng degrees in Computer Engineering, and he received a PhD in Information Engineering in 2011 from the University of Padova, Italy. His research interests include information access and retrieval models, distributed search, digital geolinguistics, and computational social science.

**Massimo Melucci** received the PhD in Computer Engineering in 1996 from the University of Padova, Italy. Associate Professor at the University of Padova, his scientific research ranges in modeling and experimenting advanced methods for indexing, retrieving and ranking documents.

**Giacomo Nannicini** received a PhD in Computer Science from École Polytechnique, France, in 2009. He is a Research Staff Member at the IBM T. J. Watson research center, NY. His research spans several areas of optimization, including network optimization, integer and dynamic programming, derivative-free optimization. He is the recipient of several awards, including the 2012 Glover-Klingman prize, the 2015 Robert Faure prize, and the 2016 COIN-OR Cup.