Finding the Best Parameter Setting
Particle Swarm Optimisation

Javier Parapar, María M. Vidal, and José Santos

Information Retrieval Lab
Computer Science Department
University of A Coruña, Spain
{javierparapar,irlab,santos}@udc.es

Abstract. Information Retrieval techniques traditionally depend on the setting of one or more parameters. Depending on the problem and the techniques the number of parameters can be one, two or even dozens of them. One crucial problem in Information Retrieval research is to achieve a good parameter setting of its methods. The tuning process, when dealing with several parameters, is a time consuming and critical step. In this paper we introduce the use of Particle Swarm Optimisation for the automatic tuning process of the parameters of Information Retrieval methods. We compare our proposal with the Line Search method, previously adopted in Information Retrieval. The comparison shows that our approach is faster and achieves better results than Line Search. Furthermore, Particle Swarm Optimisation algorithms are suitable for parallelisation, improving the algorithm behaviour in terms of time convergence.

1 Introduction

There are very few retrieval parameter free methods. Most of state-of-the-art retrieval methods depend on one or more parameters which have to be set to some values. A good parameter setting is generally a crucial factor in the performance of the algorithms. The traditional methodology in Information Retrieval (IR) implies the optimisation of the parameter values in a training sub-set for a certain evaluation metric, and then testing that set of parameter values in the test scenario. When dealing with one or two parameters, the exhaustive search for the best parameter values is expensive but affordable; however, the computational cost of the optimisation process grows exponentially with the number of parameters. Models with very few free parameters have limited effectiveness in tasks where many sources of evidence exist like web search. Because of this, the number of parameters in the current retrieval methods is increasing, difficulting the process of searching for a good parameter setting.

This problem has been previously addressed in the literature. In [13], Taylor et al. studied this issue for retrieval methods containing up to 375 parameters. They agreed that the lack of efficient algorithms for parameter optimisation is one of the bottlenecks of current IR research. In this paper we will directly

address this particular problem proposing the use of the global search method provided by Particle Swarm Optimisation, as an efficient and effective method for parameter optimisation for retrieval models. Particle Swarm Optimisation (PSO) [8] is a population based stochastic optimisation technique, inspired by social behaviour of bird flocking or fish schooling, and included in swarm intelligence techniques [9]. PSO has been previously used for optimisation problems in other areas. We want to evaluate in this paper how this technique can be applied in the context of parameter optimisation in several retrieval scenarios, where the analytic form of the retrieval functions is not necessarily known.

The remaining of the paper is as follows: Section 2 presents some related work in the parameter optimisation task, in Section 3 we introduce our proposal and the greedy search technique traditionally used as baseline, called Line Search; Section 4 presents the results of the different experiments addressed and finally conclusions are reported in Section 5.

2 Related Work

We can classify existing methods based on the necessary knowledge of the retrieval functions: there exist blind methods that only depend on the evaluation metric to optimize [11] or methods that take advantage of knowing the analytic form of the retrieval models or functions [13, 10]. The former methods usually evaluate the objective function under different parameter settings and it is normally an expensive procedure. The cost of such process can be ameliorated by applying some heuristics, allowing a non-exhaustive exploration of the dimensions [2, 15]. Although these methods are expensive, they do not have any application restriction and no extra information is needed in order to be successfully applied. On the opposite, the methods that take advantage of the internal retrieval methods details have the advantage that the calculation of the overall ranking (that is the most expensive operation) is not mandatory. Knowing the analytical form of the retrieval methods allows to apply algorithms such as the gradient descent [3]. As explained in [13], these methods, although theoretically much faster, can suffer of a mismatch between the objective function used and the ranking base metric that is finally evaluated, resulting in poor performance. In this paper, we will be focus on the blind methods, that are more broadly usable, considering how to improve their efficiency.

3 Optimisation Methods

As explained above, this paper proposes a new approach to parameter optimisation for Information Retrieval problems. This proposal has to be compared with standard existing approaches. As we are presenting a method that does not require neither takes advantage of the analytical form of the retrieval methods, we have chosen Line Search as baseline [11], as in [13]. Hence, in this Section we briefly present the basis of Line Search and our proposal: Particle Swarm Optimisation.

3.1 Line Search

Line Search is a general class of optimisation methods which first find a descent direction along which the objective function will be reduced and then computes a step size that decides how far the parameter values should move along that direction [11]. Here, we used the Line Search procedure followed by Taylor et al. [13], where the authors conclude that Line Search and Gradient Descent (which takes advantage of the knowledge about the analytical form of the retrieval function) perform similarly in terms of effectiveness.

In Line Search, from an initial random point in the parameter space, a search in each dimension is performed, moving each time the parameter value in one dimension while fixing the values of the other dimensions. The procedure is summarized in the pseudo-code of Algorithm 1. For each dimension, N sample points are selected with equal inter-distance in its axis and around the initial parameter value (taking into account the limits of the parameter space in the sampling). In order to assess the optimality of each point a *fitness* value is calculated for each of these sample points, storing the point with the best fitness. The fitness function will be dependent on the problem and will measure the quality of a parameter setting for the given task. This basic procedure is repeated for each of the dimensions or parameters (Step 1 of the algorithm).

Algorithm 1 Line Search Algorithm.

N=Number of sample points in each dimension, D=number of dimensions, I=Sampling interval.
Select an initial random point.
Step 1 of the algorithm (for dimension d)
$min \leftarrow max(0, initial\_position[d] - \frac{I}{2}); (0\ is\ the\ parameter\ lower\ limit)$
$max \leftarrow min(1, initial\_position[d] + \frac{I}{2}); (1\ is\ the\ parameter\ upper\ limit)$
$increment \leftarrow \frac{(max-min)}{N}$;
$best\_position[d] \leftarrow min; (Best\ initial\ position)$
for $n \leftarrow 1$ ***to*** N **do**
  $p \leftarrow min + increment * n$;
  $new\_position[d] \leftarrow p$;
  if $(fitness(new\_position[d]) < fitness(best\_position[d]))$ **then**
    $best\_position[d] \leftarrow new\_position[d]$;
  end if
end for
return best_position[d]
Step 2 of the algorithm
for $d \leftarrow 1$ **to** D **do**
  $max\_dim[d] \leftarrow max(initial\_point[d], best\_position[d])$;
  $min\_dim[d] \leftarrow min(initial\_point[d], best\_position[d])$;
  $increment[d] \leftarrow \frac{(max\_dim[d]-min\_dim[d])}{N}$;
end for
$best\_position \leftarrow initial\_position$
for $n \leftarrow 1$ ***to*** N **do**
  for $d \leftarrow 1$ ***to*** D **do**
    $new\_position[d] \leftarrow min\_dim[d] + increment[d] * n$;
  end for
  if $(fitness(new\_position) < fitness(best\_position))$ **then**
    $best\_position \leftarrow new\_position$
  end if
end for
return $best\_position$

In a second step of the algorithm, it is defined a line between the original point and the new computed point. This new point is built by taking for each dimension the parameter value with the best fitness in the first step. This line represents the "promising" direction. Again, the same procedure is repeated, selecting a number of equidistant points (samples) between the two extremes of the promising segment. The point with the best fitness is selected, and if this new point is better in terms of fitness than the original one, being the new starting point in the next iteration of the algorithm.

So, an "iteration" is defined as one cycle through all parameters, plus the final search along the promising direction. Thus, if there are P parameters, then the procedure performs $P + 1$ line search operations per iteration. Finally, the scale over which the samples are taken is reduced by a factor of 0.85 at the beginning of each new iteration, allowing higher exploration in the first iterations and higher exploitation in the final iterations.

3.2 Particle Swarm Optimisation

Particle Swarm Optimisation (PSO) [8] is a population based stochastic optimisation technique, inspired by social behaviour of bird flocking or fish schooling, and included in swarm intelligence techniques [9]. The potential solutions, called "particles", fly through the problem space following the current optimum particles. The movements of the particles are guided by the best known position of each particle in the search space as well as the entire swarm's best known position. The process is repeated until a satisfactory solution is discovered.

The basic PSO algorithm is summarized in Algorithm 2. Each particle i stores its current position x_i^t, velocity v_i^t and its best known position pb_i^t at time t. Moreover, the algorithm considers the best known position of the entire swarm (gb^t).

Algorithm 2 PSO basic Algorithm.

Initialise all particles i with random positions x_i^0 in search space as well as random velocities v_i^0.
Initialise the particle's best known position (pb_i^0) to its initial position.
Calculate the initial swarm's best known position gb^0.
repeat
  for all Particle i in the swarm **do**
    Pick random numbers: $r_p, r_g \in (0, 1)$
    Update the particle's velocity: $v_i^{t+1} = a * v_i^t + b * r_p * (pb_i^t - x_i^t) + c * r_g * (gb^t - x_i^t)$
    Compute the particle's new position: $x_i^{t+1} = x_i^t + v_i^{t+1}$
    if $fitness(x_i^{t+1}) < fitness(pb_i^t))$ **then**
      Update the particle's best known position: $pb_i^{t+1} = x_i^{t+1}$
    end if
    if $fitness(pb_i^{t+1}) < fitness(gb^t))$ **then**
      Update the swarm's best known position: $gb^{t+1} = pb_i^{t+1}$
    end if
  end for
until termination criterion is met
return The best known position: gb.

In the algorithm, a, b and c are constants that separately control the importance of the three directions which determine the next velocity and position of the particle. The three components are usually referred as "inertia" (v_i^t), "personal influence" ($pb_i^t - x_i^t$) and "social influence" ($gb^t - x_i^t$). By updating the velocities with some element of randomness it is enabled the exploration by the particles of novel areas of the search space, avoiding stagnation in local minima. This is incorporated by means of the random values, in the (0,1) interval, of the terms r_p and r_g, which imply a region of uncertainty around both positions pb_i^t and gb^t. Recommended values for the parameters a, b and c were used [4].

One of the reasons PSO is an interesting method in many optimisation or search problems, in comparison with the simulated evolution methods largely used, like classic genetic algorithms, is the reduced number of parameters that are needed to define their implementations. In a standard genetic algorithm it is difficult to control the balance between exploration and exploitation. Even with low selective pressures there is a high probability that the population converges to a local optimum in few generations. Therefore, there is no guaranty that different areas that could be good in the space are simultaneously searched. The niche and speciation techniques based on "fitness sharing" [5] or the distribution of the population in races or islands that evolve independently and simultaneously, which periodically interchange their best genetic material, are alternatives to minimize that problem. On the contrary, PSO intrinsically explores the search space with a concentration of the population around the promising areas found.

4 Evaluation

In order to properly evaluate the feasibility of our proposal and comparing it with standard techniques currently used in IR, we performed three different experiments. The three experiments presented here will share the same evaluation metric: Mean Reciprocal Rank. Next, we introduce the evaluation metric, the TREC [14] tasks used in the evaluation together with the experiments and their commented results. At the end of this section we also show the advantages in terms of parallelisation of our proposal, so an additional experiment is presented in this regard.

Known item search task is a retrieval task where the objective is to retrieve, from the whole collection, a particular document for a given query. Mean Reciprocal Rank (MRR) is a commonly used evaluation metric for the known item search task. The MRR is the average of the reciprocal rank of each processed query. The reciprocal rank for a particular query will be higher the higher the desired document for that query is in the ranking list. Analytically, the MRR is computed as:

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i}$$

where Q is the set of queries and $rank_i$ is the position in the ranking of the unique relevant document for the query i.

In our evaluation we will use, for both optimisation techniques (PSO and LS), this particular evaluation metric as quality measure for assessing the goodness of the parameter settings. *The TREC 5 Confusion Track* [7] was designed in order to evaluate the effectiveness of the retrieval systems when dealing with degraded information (output of OCR processes). In this track, three different versions of the same collection (55,533 documents from the 1994 Federal Register) were used by the participants. Each version corresponds with a level of degradation: 0% the original collection, 5% and 20% the other two. The designed task followed the *know item search* paradigm, with a set of 49 queries with only one relevant document assessed. MRR was used as evaluation metric for the participant runs. Particularly, the participating teams produced 5 different runs for the original collection, 8 different runs for 5% degraded collection and 7 different runs for the 20% degraded collection.

The TREC 2001 Web Track [6] was designed in that time to provide a reference for evaluation of retrieval methods in large web tasks. Particularly, the track participants used the well known WT10g corpus composed of 1,692,096 web pages and with an approximate size of 10GB. In this track, two different tasks were proposed: the traditional topic relevance task and the *homepage finding task*. Home page finding task is another example of *known item search* where the objective is to locate in the collection the entry page of a website. In order to do this, 145 homepage finding queries were distributed to the participants. Again, MRR was used as primary evaluation metric. In the homepage finding task the teams provided with 43 different runs.

4.1 TREC 5 Confusion Track, Combining Evidences Task

Recently, a method [12] which outperformed the best results of the TREC Confusion Track was presented. In [12], the authors introduced a method that produces five different tokenisations of the query and the documents in order to ameliorate the term matching problem that occurs in the degraded collections. The five different tokenisations were words, 5-grams, 4-grams, 3-grams and 2-grams of characters. In order to combine the five different evidences weights are assigned in the combination to adjust the importance of each tokenisation. In [12] the authors used a traditional tuning of the parameters achieving good results. We believe that this is a good scenario to test our approach and compare with the traditional ones.

We compared the proposed PSO method against the baseline method Line Search. In every experiment for PSO we used 90 particles and 100 generations, meanwhile for LS we used 15 samples per dimension and 100 iterations or epochs. This way, the number of computations of the overall ranking for both approaches is similar (Remember that LS needs $(5+1) \times 15$ fitness calculations, being 5 the number of parameters). Given the stochastic nature of both methods all the results are an average of five independent runs. Results are presented in Table

1. For both methods the range of permitted values for every weight parameter was limited between 0 and 1[1].

Table 1. Mean Reciprocal Rank (MRR) for the n-gram based method presented in [12] for retrieval over degraded information on the Confusion Track Collections, when applying the best parameters reported in [12], when tuning the parameters with Particle Swarm Optimisation (PSO) and when tuning with Line Search (LS). Between brackets is the number of parameters to optimize. Best and worst values for every collection obtained by the TREC Confusion Track participants are shown as reference.

Collection	*Method*	*MRR*
Original (5)	PSO	0.8075
	LS	0.7992
	[12]	0.7686
	Best TREC	0.7353
	Worst TREC	0.3039
Degraded 5% (5)	PSO	0.7306
	LS	0.7255
	[12]	0.7276
	Best TREC	0.5737
	Worst TREC	0.1900
Degraded 20% (5)	PSO	0.4712
	LS	0.4670
	[12]	0.4708
	Best TREC	0.4978
	Worst TREC	0.1174

The first comment about the results is that, in terms of effectiveness, PSO achieved for the three collections the best results, i.e., PSO achieved the parameter stetting with which the method achieve the best performance. Another important point is that LS was unable to obtain better results than the ones reported in [12] with the traditional tuning methodology for two of the collections. Anyway, in this task, the differences in terms of effectiveness between LS and PSO are negligible. Nevertheless, when checking the efficiency we have to remark that the processing time needed for each method to achieve the best parameter setting differs significantly. We will thoroughly analyse this aspect later in the paper.

4.2 TREC Confusion Track, Metasearch Task

After presenting the task of adjusting the parameters of a retrieval model which combines different evidences we introduce another different task. In order to

[1] In both methods, each set of five weight parameters, is normalised in order to have a sum of 1. This is done before the fitness calculation. So it is not necessary to incorporate this restriction in the optimisation problem.

reduce the description of the experimental settings we decided to use the Confusion Track collection to present the metasearch task. Given the ranked lists of documents returned by multiple search engines or retrieval models, metasearch [1] combines these lists in a way which optimises the performance of the combination. One traditional way of metasearch is the *Linear Combination Model*, where the relevance scores given to each document d by each model $rel_i(d)$ are combined so as to obtain an overall relevance score $rel(d)$:

$$rel(d) = \sum_i \alpha_i rel_i(d)$$

We tested the effectiveness of the optimisation methods to adjust the weights α_i, when combining the different runs provided by the TREC 5 Confusion Track participants. The results of such experiments are reported in Table 2.

Table 2. Mean Reciprocal Rank (MRR) for the metasearch task with the TREC-5 Confusion Track runs, when tuning the weight parameters of the linear combination with Particle Swarm Optimisation (PSO) and with Line Search (LS). Between brackets is the number of parameters to optimise. For every collection PSO achieves statistical significant improvements according to Wilcoxon test with $p - value = 0.01$. Best and worst values for every collection by the TREC Confusion Track participants are shown as reference.

Collection	*Method*	*MRR*
Original (5)	PSO	0.8560
	LS	0.7665
	Best TREC	0.7353
	Worst TREC	0.3039
Degraded 5% (8)	PSO	0.6674
	LS	0.5013
	Best TREC	0.5737
	Worst TREC	0.1900
Degraded 20% (7)	PSO	0.5623
	LS	0.2354
	Best TREC	0.4978
	Worst TREC	0.1174

Both approaches achieved parameter settings that allowed the linear combination model to outperform the best of the individual runs in each collection. In this task the optimal parameter settings computed with PSO produced statistical significant better results in terms of MRR than the ones estimated with LS. The main explanation for this behaviour is that the optimal parameter setting for this task is achieved when some of the runs are ignored ($\alpha_i = 0$). LS fails when this is required, because it implies a very large sampling interval in such parameter with respect to the others. Nevertheless, PSO has no such problem, as it performs a global search of the parameter space. Furthermore LS is very sensitive to the initial position because it searches in a restricted area around such

initial point, area defined by the initial interval of the sampling, being difficult to reach the points close to the limits of the parameter space. We have to notice that normalisation of the rankings' scores, prior to the linear combination was tested (in every experiment), but differences in performance with and without normalisation are negligible.

4.3 TREC Web Track, Metasearch Task

In order to test our proposal when the number of parameters is larger, we decided to use the TREC 2001 Web Track Homepage finding task results [6] to reproduce the metasearch task. In this case, the participants provided 43 different runs that we used to produce the overall metasearch ranking with a linear combination model. So, in terms of parameter setting, the optimisation techniques have to adjust the weights of the 43 parameters $\alpha_1 \dots \alpha_{43}$. The results of this additional experiment are reported in Table 3.

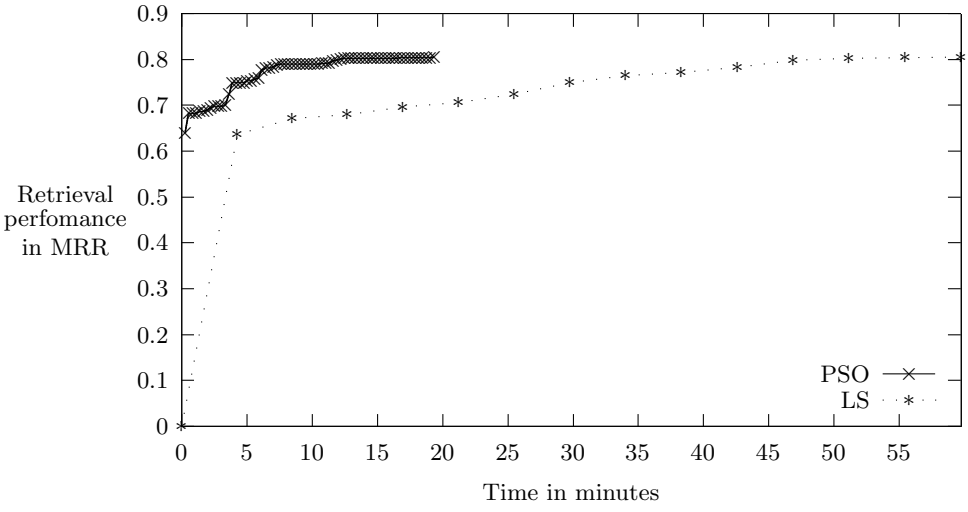
Table 3. Mean Reciprocal Rank (MRR) for the metasearch task over the runs of the Home-page finding task of the TREC 2001 Web Track [6], when tuning the parameters of the linear combination with Particle Swarm Optimisation (PSO) and with Line Search (LS). Between brackets is the number of parameters to optimise. Best and worst values for every collection by the TREC Web Track participants are shown as reference.

Collection	*Method*	*MRR*
WT10g (43)	PSO	0.8080
	LS	0.8035
	Best TREC	0.7740
	Worst TREC	0.0540

Again, as expected, the parameter settings computed with the optimisation techniques allowed to the metasearch model to outperform the best individual run (MRR 0.774). In this case, the results in terms of effectiveness for both optimisation methods are quite similar. However, we must analyse other factors such as efficiency and convergence speed. In order to do so we tracked, for each generation in PSO and each iteration in LS, the necessary time[2] for computing their best parameter settings and the fitness values in terms of MRR. Those data are presented in Figure 1. In this experiment no degree of parallelisation was applied to either of the two methods in order to fairly compare their behaviour. To obtain the best parameters in terms of MRR performance, it is clear that the time needed with both approaches is significantly different, varying from 20 minutes for PSO to 1 hour for the LS method. The most important point is the convergence speed to the best parameter setting: while PSO achieves the best

[2] Reported data are the average of five different computations on an Intel platform with 2 Quad Core E5504 processors and 16GB of RAM

Fig. 1. Evolution over time of the best parameter setting performance, achieved by Particle Swarm Optimisation (PSO) and Line Search(LS) in the TREC 2001 Web Track metasearch task.

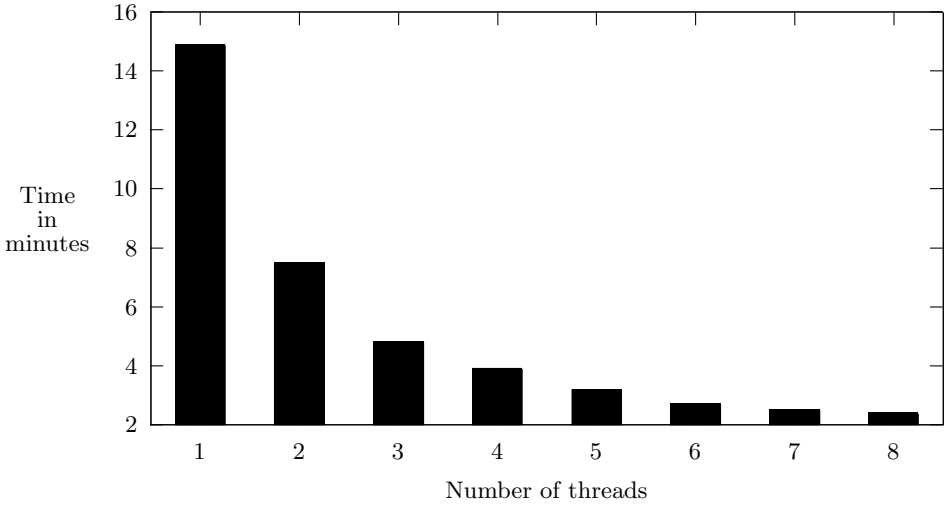


parameter setting around minute 10, for LS it is not until the very end when that combination is achieved. This is a clear and a crucial point in order to choose between LS and the PSO proposal.

We present and additional experiment designed to show the goodnesses of the PSO proposal. PSO allows a straightforward parallelisation of the evolution computation at particle level. If we check the pseudo-code presented in Algorithm 2 (Section 3.2), the parallelisation corresponds with the un-folding of the *for all* loop which starts in line 5. So, under the same experimental settings presented above, we tracked the times expended in computing 50 generations of the PSO method when increasing the parallelisation degree, i.e. the number of threads used for the computation[3]. Results are reported in Figure 2, showing the benefits of the algorithm parallelisation. The experiment was carried out in a dual Quad-Core processor, so the number of threads was varied from 1 to 8. When we used eight threads we achieved the best parameter setting for the 43 participant runs in less than two and a half minutes. If we have a quick look to the graph we can observe that the speed-up when increasing the number of threads almost

[3] The parallelisation process is a simple shared memory approach where a pool of threads is created and each of the threads calculates the fitness values of different particles stored in memory, and in which the synchronisation requirements are minimum.

Fig. 2. Evolution of the time expended in the evolution of 50 generation in the PSO algorithm when increasing the parallelisation degree.



fit a linear form, being the small different due to the necessary synchronising requirements.

5 Conclusions and Future Work

In this paper we introduced Particle Swarm Optimisation as a method for optimising parameter settings in Information Retrieval problems. We compared the results of our proposal against the standardly used Line Search algorithm. We performed three different experiments in traditional retrieval problems. From the results of those experiments we can conclude that PSO performs better in terms of effectiveness and efficiency. PSO obtains parameter values that are better in performance than the ones obtained with Line Search. Furthermore, we exposed some problems of Line Search when the optimal parameter values are in the endings of the search space, problems that PSO does not suffer thanks to its global search approach. In terms of efficiency, PSO is more efficient than LS, since the convergence to the optimal parameter setting in the PSO evolution is much faster than in the LS method. Moreover, the parallelisation of PSO is straightforward, allowing to obtain the best parameter settings with almost linear speed-ups, depending on the available number or processing cores.

As future work we would like to compare the performance of the studied methods with other metrics and tasks where the number of parameters to optimise is even larger, together with the hybrid combination of both search methods.

The design of not-blind PSO methods, i.e., methods which take advantage of the analytical form of the retrieval functions, is also a promising field of research.

Acknowledgements: This paper has been funded by the Ministry of Science and Innovation of the Kingdom of Spain under research project ref. TIN2011-27294.

References

1. Javed A. Aslam and Mark Montague. Models for metasearch. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '01, pages 276–284, New York, NY, USA, 2001. ACM.
2. Chris Buckley and Gerard Salton. Optimization of relevance feedback weights. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '95, pages 351–357, New York, NY, USA, 1995. ACM.
3. Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pages 89–96, New York, NY, USA, 2005. ACM.
4. R. C. Eberhart and Y. Shi. Comparing inertia weights and constriction factors in particle swarm optimization. In *Proceedings of the 2000 Congress on Evolutionary Computation*, volume 1, pages 84–88, 2000.
5. David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.
6. David Hawking and Nick Craswell. Overview of the TREC-2001 web track. *NIST Special Publication*, (500-250):61–67, 2001.
7. Paul B. Kantor and Ellen M. Voorhees. The TREC-5 confusion track: Comparing retrieval methods for scanned text. *Inf. Retr.*, 2(2/3):165–176, 2000.
8. James Kennedy and Russell C. Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, pages 1942–1948. Piscataway, NJ, 1995.
9. James Kennedy and Russell C. Eberhart. *Swarm intelligence*. Morgan Kaufmann Publishers Inc., CA, USA, 2001.
10. Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Program.*, 45:503–528, December 1989.
11. David G. Luengerber. *Linear and nonlinear programming*. Addison Wesley, 1984.
12. Javier Parapar, Ana Freire, and Álvaro Barreiro. Revisiting n-gram based models for retrieval in degraded large collections. In *Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval*, ECIR '09, pages 680–684, Berlin, Heidelberg, 2009. Springer-Verlag.
13. Michael Taylor, Hugo Zaragoza, Nick Craswell, Stephen Robertson, and Chris Burges. Optimisation methods for ranking functions with multiple parameters. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, CIKM '06, pages 585–593, New York, NY, USA, 2006. ACM.
14. Ellen M. Voorhees and Donna K. Harman. *TREC: Experiment and Evaluation in Information Retrieval*. The MIT Press, 2005.
15. Hugo Zaragoza, Nick Craswell, Michael Taylor, Suchi Saria, and Stephen Robertson. Microsoft Cambridge at TREC-13: Web and hard tracks. *NIST Special Publication*, (500-261), 2004.