2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Enhancing relevance re-ranking using nature-inspired meta-heuristic optimization algorithms

Amel Ksibi, Anis Ben Ammar and Chokri Ben Amar
REGIM-Lab.: REsearch Groups in Intelligent Machines,
University of Sfax, ENIS,
BP 1173, Sfax, 3038, Tunisia
Email: [amel.ksibi, anis.benammar, chokri.benamar]@ieee.org

***Abstract*—Over the last years, relevance re-ranking has been an attractive research, aiming to re-order the initial image search result list by which relevant ones should be at the top ranking list and irrelevant ones should be pruned. In this paper, we propose to explore two population-based meta-heuristic algorithms, which are Particle Swarm optimization(PSO), and Cuckoo search(CS), in order to solve the relevance re-ranking problem as a constrained regularisation framework. By doing so, we define two reranking processes, refereed as APSO-Rank and CS-Rank that converge to the optimal ranked list. Results are further provided to demonstrate the effectiveness and performance of these two reranking processes.**

I. Introduction

Empowered by the rapid advances in technology and the proliferation of social Web, an ever-increasing amount of social images has been emerged[1][2]. Such explosive visual data has led to a surge of research activity in visual information retrieval topic. The key problem is to find images that are relevant to a given query over a large-scale collection. Although the popularity of existent commercial visual search engines such as Google, Bing and Flickr, the top-k obtained results are, usually, not relevant, yet, far from user satisfactory. To cope with this problem, relevance re-ranking has attracted increasing attention in recent years. Discovering knowledge or visual patterns from such a noisy ranked list to guide the re-ranking process is difficult[3].

Recently, studies have focused on investigating in meta-heuristic optimization to enhance relevance re-ranking[4]. Indeed, meta-heuristic algorithms based on biological behaviour and physical systems in nature have been proposed to solve intractable optimization problems and have achieved fascinating results[5]. Specifically, population-based stochastic optimization techniques have received increasing attention since these techniques provide nearly-optimal solutions in highly non-linear, multidimensional solution spaces, with lower complexity and faster convergence than traditional algorithms. The purpose of population-based optimization is to predicate on the collective search of the patterns for the solutions of the related problem. Indeed, a population-based optimization technique begins its search with a population of random solutions. Thereafter, in each iteration the population is updated by using a population-update algorithm to survive the fittest solutions and remove the worst ones from the population.

Motivated by these observations, we propose to explore two population-based meta-heuristic mechanisms, which are Particle Swarm optimization(PSO[6]) as the most popular one, and Cuckoo search(CS)[7] as the newest one, in order to solve the relevance re-ranking problem. The major advantages using PSO and CS for re-ranking are twofold: first, the robustness and the effectiveness of the "group" behaviour in avoiding the potential noise propagation from the initial ranking results; and second, simplicity and the low computation cost[8].

To the best of our knowledge, this is the first work that introduces the Cuckoo Search optimization into the re-ranking process. In contrast, it is worth mentioning that PSO optimization has been used in content-based image retrieval systems, specifically for improving the re-ranked results. For instance, Okayama[9] was the first person to introduce the PSO into the re-ranking process. In this work, parameters were optimized according to the users' evaluation of the retrieved image ranking using Particle Swarm Optimization. Experiments had shown an improvement in the retrieval ranking suiting the user's preference when feature space mapping and parameter optimization were used. Broilo et al. [10] proposed a Particle Swarm Optimization algorithm, appropriately designed to exploit the user feedback in Content-based image retrieval. Obtained results demonstrated that their method outperforms traditional relevance feedback approaches, showing a much higher precision/recall.

All the above methods have used the PSO for explicit relevance feedback that relies on the intervention of the user. In such case, the objective is to find the most similar photos to the selected photos examples in the user feedback. In contrast, our objective is to reorder initial search results without any added information so that the new ranked list contains more relevant photos to the given query. In this context, Zhang et al. [11] proposed an Adaptive PSO framework (APSO) for visual re-ranking based on visual consistency and ranking distance regularization. Experiments showed that APSO reranking outperformed several existing reranking approaches. However,the major drawback in this method consists in the fact that the authors have ignored the semantic consistency which can improve the pair-wise similarity among images.

The rest of this paper is organized as follows: in section 2, we discuss related work on relevance reranking. In section 3, we propose a new regularization framework to model the relevance re-ranking problem. In section 4, we describe the proposed APSO-Rank algorithm. In section 5, we detail the proposed CS-Rank algorithm. In section 6, we report the experiments and obtained results.

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

II. State of the art: Relevance reranking

Relevance-based re-ranking[12] aims to re-order the initial image search result list by which relevant results should be at the top ranking list and irrelevant results should be pruned. Over the last years, relevance re-ranking has been an attractive research area where various approaches have been proposed and can be roughly classified into two categories[13]: supervised re-ranking and unsupervised reranking.

A. Supervised reranking

Supervised reranking aims to train a classifier using sample data from the initial search results and then sort all results by the relevance degree estimated from the classifier. The key problem resides in the selection of training data. Actually, three main solutions can be distinguished. The first one consists in choosing these data using relevance feedback mechanisms (manually[14] or active learning[15]). The second one is to adopt the process of blind relevance feedback[16], in which top ranked documents are selected as training data, assuming that these documents tend to be relevant. The third one consists in using external information resources such as concept-based semantic representation and social data[17][18].

B. Unsupervised re-ranking

Generally, unsupervised re-ranking is based on the smoothness assumption which considers that similar documents should have closer relevance scores. As such, unsupervised re-ranking aims to discover and mine patterns by exploiting inter-documents similarities. Obviously, there are two main effective ways. The first way is to use clustering techniques due to the effectiveness of detecting patterns. For instance, in [19], authors proposed a bag-based re-ranking framework. First, relevant images are clustered using both textual and visual features. By treating each cluster as a "bag" and the images in the bag as "instances", the reranking problem is formulated as a multi-instance (MI) learning problem. Pedronette and Torres [20] proposed an iterative clustering approach based on distances correlation and on the similarity of ranked lists, assuming that if two images are similar, their distances to other images and therefore their ranked lists should be similar as well. The second way is to adopt graph-based representation where a graph is constructed using photos as nodes and their similarities as edges. Recently, graph based methods have shown their effectiveness in results re-ranking since this kind of methods, usually integrate stochastic optimization algorithms. For example, in [21], authors formulated the reranking task as a random walk process over the graph and the initial relevance scores are smoothed by the propagation through edges. Zhang et al.[11] proposed particle swarm optimization over the graph to recover the "genuine" ranking list from the helpful but noisy one generated by initial textual search.

III. New regularization framework for relevance re-ranking formulation

In this section, we propose to model the relevance reranking problem via a new regularization framework inspired by the graph-based semi-supervised methods[12].

The following notations will be used throughout the rest of this paper:

Suppose we have a photo set D with N photos to be reranked where$D = x_1, ..., x_N$, a ranking score list $s = [s_1, ..., s_N]^T$ as a vector of the ranking scores corresponding to a photo list $x = [x_1, ..., x_N]^T$ where s_i denotes the relevance score of photo x_i and $\bar{s}$ denotes the initial ranking score list. S denotes the set of all possible s, then $S = \{\mathbf{s_j}\}_j^M (1 < M \ll N!)$ where $s_j = [s_{j1}, ..., s_{jN}]$ denotes the j-th ranking list containing the N photos.

The re-ranking process aims to re-order initially retrieved photos so as to improve precision at the top-ranked final results. Here, we define re-ranking as looking for the "genuine" objective score list s^* by minimizing an energy function f defined as follows:

$$s^* = \arg\min_{s \in S} f(s, \bar{s}) \tag{1}$$

In this section, we will opt for solving the re-ranking problem from the perspective of regularization constraints at two levels: the smoothness constraint and the fitting constraint. The first means that a good energy function should not change too much between nearby points; while the second means that a good energy function should not change too much from the initial state. By doing so, we will derive an optimal re-ranking function based on score consistency that integrates visual consistency and semantic consistency, and ranking distance from the initial ranked list.

A. Smoothness constraint: Score consistency f_c

Score consistency consists of visual and semantic consistencies. In fact, photos that are visually and semantically similar should have close ranking scores in the "genuine" ranking list. Specifically, this assumption is modelled by estimating the pair-wise visual consistency and pair-wise semantic consistency, which can be obtained from the visual features and the semantic features by:

$$f_c(s, x) = \sum_{i=1}^{N} \sum_{j=1}^{N} \varphi_{ij}(s, x) \tag{2}$$

where $\varphi_{ij}(s, x)$ is the energy function based on the hybrid similarity between photos. Indeed, we use contextual correlation inter-photos to derive the energy function, as follows:

$$\varphi_{ij}(s, x) = \frac{1}{2} w_{ij} (s_i - s_j)^2 \tag{3}$$

where

$$w_{ij} = S_h(x_i, x_j) \tag{4}$$

We note that the "sum" operation is the most used operation for measuring the smoothness over graph[].

B. Fitting constraint: Ranked list distance f_d

Despite the noise within the initial ranking score list $\bar{s}$, the latter still reflects, valuable information, even minimal, about the "genuine" ranked list. Thus, the expected re-ranked list s should not be changed too much from the initial list $\bar{s}$. As such, this assumption can be modelled as a fitting constraint in the regularization re-ranking formulation.

$$f_d(s, \bar{s}) = d(s, \bar{s}) \tag{5}$$

where $d(s, \bar{s})$ is the distance reflecting the disagreement between the initial ranking list $\bar{s}$and the re-ranked list s. One intuitive idea to evaluate the disagreement between two lists is to apply a "point-wise" approach which calculates at each point the score difference, and then sums the results up. Despite its simplicity, this approach considers only the local difference at each position, and fails to capture the global disagreement between two scores lists in term of ranking list order. As an alternative, the "pair-wise" approach is introduced to estimate the disagreement by summing the difference of all possible pairs between two lists. However, it still fails to capture effectively the disagreement. Differently, Zhang et al.[11] define the disagreement by counting the amount of pairs that disagree on the ordinal relations in the two lists. Thus, this assumption is formulated as follows:

$$d(s, \bar{s}) = \sum_{(i,j) \in P_{\bar{s}}} \xi(s_i, s_j) \tag{6}$$

where $P_{\bar{s}} \equiv (i, j) : \bar{s_i} > \bar{s_j}$ and $\xi(s_i, s_j) = 1 \quad if \quad s_i < s_j$.

Although this technique enhances the disagreement estimation, it still requires a highly computational cost. Therefore, we define a new ranking distance that computes the disagreement between two ranked lists by estimating the intersection between two sub-lists of them at different levels of k, where $k \leq K \leq N$, and K is a computational constant defined empirically.

$$d(s, \bar{s}) = 1 - \frac{1}{K} * \sum_{k}^{K} \frac{1}{k} * |g(s, k) \cap g(\bar{s}, k)| \tag{7}$$

where $g(s, k)$extracts the k first photos from s having the highest scores, such that $|g(s, k)| = k$. We can note that if two ranked lists having the same photos at the first positions, the size of the intersection set is greater and the value of d is smaller as well.

C. Optimization function

From the above discussions, our reranking problem can be formulated by integrating the two above components: scores consistency and ranked list distance in the equation 1. As such, we aim to minimize the following energy function:

$$s^* = \arg\min_{s \in S} f(s, \bar{s}) = \arg\min_{s \in S} \{f_c(s, x) + c * f_d(s, \bar{s})\} \tag{8}$$

where $c \in [0, 1]$ is a trade-off parameter that balances the above constraints. Indeed, a smaller value of c means that the consistency plays the major role in re-ranking process; while a larger c indicates that the initial list has the major influence in the re-ranking process.

Since this function is analytically intractable, efficient optimization techniques are highly desired to solve it.

IV. Enhancing search results reranking via Particle-Swarm Optimization

A. Particle Swarm Optimization principles

Particle Swarm Optimization (PSO) is a population based stochastic optimization technique developed by Russell C. Eberhart and James Kennedy in 1995[6], inspired by social behaviour of bird flocking or fish schooling.

Fundamentally, individuals in PSO are named particles. Each particle has a position in the search space, a velocity with which it moves in the space, and a memory of the best solution found by the particle pi. The PSO algorithm works by simultaneously maintaining several candidate solutions in the search space. During each iteration of the algorithm, each candidate solution is evaluated by the objective function being optimized, determining the fitness of that solution. Each candidate solution can be as a particle that flies in the search space with a velocity which is dynamically adjusted according to its own flying experiences and those of its companions, towards the swarm's global best and the particle's personal best positions. These positions define, respectively, the social and the cognitive factor of the swarm and are weighted by two parameters that influence the swarm behaviour in the search process.

It should be noted that the PSO algorithm has no knowledge of the underlying objective function, and thus has no way of knowing if any of the candidate solutions are near to or far away from a local or global maximum. The PSO algorithm simply uses the objective function to evaluate its candidate solutions, and operates upon the resultant fitness values.

Mathematically, given a particle i having in d dimension search space:

- the position vector $X_i = (x_{i1}, ..., x_{id})$
- the velocity vector $V_i = (v_{i1}, ..., v_{id})$
- the pbest vector $P_i = (p_{i1}, ..., p_{id})$
- the gbest vector $P_g = (p_{g1}, ..., p_{gd})$

Updating new positions and velocities of the next generation can be determined as follows:

$$V_i(t) = \omega * V_i(t-1) + C_1 * randn_1 * (P_i - X_i(t-1)) + C_2 * randn_2 * (P_g - X_i \tag{9}$$

$$X_i(t) = X_i(t-1) + V_i(t) \tag{10}$$

where $randn_1$ and $randn_2$ are random numbers which are uniformly distributed. C_1 and C_2 are the acceleration constants whose values are usually set to 2.0. These constants control, respectively, the cognitive and social behaviours of the particle. ω is the inertia weight that controls the influence of particle's direction in the next move.

B. Relevance re-ranking optimization using adaptive PSO algorithm: APSO-Rank

In this section, we aim to solve the reranking problem by minimizing the objective function in Eq 8 using PSO optimization. For this purpose, we consider the set of M score lists S as the swarm where each score list s_j is a particle. Each initial particle $\mathbf{s}_j^0$ has an initial position given by the scores in the list, an initial velocity $\mathbf{v}_j^0$. At iteration t, the particle $\mathbf{s}_j^t$ updates its velocity and position using eq 9 and eq. 10, where $X_i(t)$ corresponds to $\mathbf{s}_j^t$. It is worth noting that $\|\mathbf{v}_j^t\| \leq V^{max}$ to confine the velocity within a predefined reasonable range.

At the end of t-th iteration, the fitness value of each particle is adjusted with the optimization function $f(s, \bar{s})$ using eq.8. Then, the pbest $\mathbf{s}_j^*$ and gbest s^* are both updated as follows:

$$\mathbf{s}_j^* = \begin{cases} \mathbf{s}_j^{t+1} & \text{if} \quad f(\mathbf{s}_j^{t+1}, \bar{s}) < f(\mathbf{s}_j^*, \bar{s}) \\ \mathbf{s}^* & \text{else} \end{cases}$$

$$s^* = \arg\min_{\mathbf{s}_j^*} f(\mathbf{s}_j^*, \bar{s}) \tag{11}$$

At each iteration, particles search for the optima until reaching convergence or reaching the maximum number of iterations. At convergence, the genuine re-ranked list is obtained by the s^* at the last iteration.

From the aforementioned equations, it can be seen that the PSO algorithm has several parameters to be tuned which are: the inertia constant, the acceleration constants, the maximum velocity v^{max}, the maximum number of iterations (generations) and the initial states of particles(initial positions). As such, these parameters should be determined empirically which may yield to the risk of swarm explosion and divergence especially in a high dimensional state space.

To address this problem, we propose to update our PSO algorithm by an advanced variant of PSO, termed as "Adaptive Particle swarm optimization"[11] which has the advantage to self-tune its parameters automatically.

Firstly, the acceleration parameters C_1 and C_2 are adjusted at each iteration as follows:

$$C_1 = \frac{2.0 * f(\mathbf{s}_j^*, \bar{s})}{f(\mathbf{s}_j^*, \bar{s}) + f(s^*, \bar{s})} \tag{12}$$

$$C_2 = \frac{2.0 * f(s^*, \bar{s})}{f(\mathbf{s}_j^*, \bar{s}) + f(s^*, \bar{s})} \tag{13}$$

By doing so, the preference for the "cognitive" factor and the "social" factor is determined by the fitness value in each iteration.

Secondly, The APSO algorithm generates an initial population with a uniform distribution of solutions in order to ensure better searching capability.

Finally, the inertia constant is updated, for each particle, according to the distance of the particles of a particular generation(iteration) from the global best. The value of ω for each particle is given by:

$$\omega_i = \omega_0 * (1 - \frac{dist_i}{dist_{max}}) \tag{14}$$

where ω_0 is a random constant in $[0.5, 1]$, $dist_i$ is the cosine distance of i-th particle from the $pbest$, and $dist_{max}$ is the maximum distance of a particle from the $gbest$ in that generation. By doing so, we ensure that in case of particles that have moved away from the $gbest$, the direction of particles will be attracted toward the $gbest$.

V. Relevance re-ranking optimization using Cuckoo Search via Levy flight

A. Cuckoo Search Principles

1) Cuckoo breeding behaviour: Cuckoo search, one of the latest nature-inspired meta-heuristic algorithms, is based on the brood parasitism of some cuckoo species[7]. This parasite behaviour is extremely fascinating. In fact, some cuckoo species such as the "ani" and "Guira" cuckoos lay their eggs in host nests, and mimic external characteristics of host eggs such as color and spots, resulting in reducing the probability of their eggs being abandoned and thus increasing their breeding. Moreover, the choice of nest is also amazing. In fact, cuckoos often select a nest where the host bird just laid its own eggs. As such, the cuckoo eggs hatch slightly earlier than their host eggs. Therefore, the first hatched cuckoo chick will, instinctively, evict the host eggs by pushing the eggs out of the nest, in order to increase the feeding opportunity provided by its host bird. Furthermore, some species of cuckoo chick can also mimic the call of host chicks to gain access to more feeding chance. In case this strategy is unsuccessful, the host can throw the cuckoo's egg away, or simply abandon its nest, making a new one in another place.

2) Levy flight: In nature, animals search for food in a random or quasi-random manner. In general, the forage path of an animal is effectively a random walk because the next move is based on the current location/state and the transition probability to the next location. Which direction it chooses depends implicitly on a probability which can be modelled mathematically. There's another type of random walk that has a little intelligence behind it, termed as Levy flight by the French mathematician, Paul Pierre Levy. This model is commonly represented by small random steps followed in the long term by large jumps. In fact, when foraging for food, animals might use a Levy flight strategy. Indeed, if an animal can't find any food in a particular area, it's probably best to go somewhere else. Subsequently, such behaviour has been applied to optimization and optimal search, and preliminary results show its promising capability [22].

3) Cuckoo search via Levy flight algorithm: The behaviour of cuckoos is combined in cuckoo search algorithm with Levy flights in order to effectively search a new nest[23]. As such, cuckoo Search (CS) algorithm can be summarized using three idealized rules, as follows:

- Each cuckoo selects randomly a nest to lay one egg at a time.
- The number of available host nests is fixed and the best nests with high-quality eggs will be carried over to the next generations.
- When the egg laid by a cuckoo is discovered by the host bird with a probability $pa \in [0, 1]$, the latter can either throw the egg away, or simply abandon the nest and build a new nest in another place. This implies that the fraction pa of n nests is replaced by new nests (with new random solutions).

For simplicity, we can use the following simple representations that each nest represents a solution and a cuckoo egg represents a new solution. The aim is to use the new and potentially better

solutions (cuckoo eggs). An initial population of host nest is generated randomly. The algorithm runs till the convergence is reached. At each iteration a cuckoo is selected at random using levy flight as given [23]. Obviously, this algorithm can be extended to the more complicated case where each nest has multiple eggs representing a set of solutions. For this present work, we will use the simplest approach where each nest has only a single egg. In this case, there is no distinction between egg, nest or cuckoo, as each nest corresponds to one egg which also represents one cuckoo.

When generating new solutions x^{t+1} for, say cuckoo i, a Levy flight is performed as follows:

$$x_i{}^{t+1} = x_i{}^t + \alpha \oplus Levy(s, \lambda) \tag{15}$$

where $\alpha > 0$ is the step size that follows the Levy distribution. The product $\oplus$ means entry-wise multiplications. Levy flights essentially provide a random walk, while their random steps are drawn from a Levy distribution for large steps.

$$Levy(step, \lambda) \sim step^{-\lambda}, (1 < \lambda <= 3) \tag{16}$$

where step is step size drawn from a Levy distribution.

Levy flight is classified as a stochastic equation for a random walk. In general, a random walk is a Markov chain whose next status/location only depends on the current location (the first term in the above equation) and the transition probability (the second term). However, a substantial fraction of the new solutions should be generated by far field randomization and their locations should be far enough from the current best solution; this will make sure that the system will not be trapped in a local optimum [7]. Statistically, this is done through the stochastic process with both stationary and independent increments. In order to generate random numbers with symmetric Levy distribution, the most efficient and yet straightforward ways is to use the so-called Mantegna algorithm for a symmetric Levy stable distribution.

In Mantegna's algorithm, the step length $step$ can be calculated by:

$$step = \frac{\sigma(\beta).randn}{randn}^{\frac{1}{\beta}} \tag{17}$$

where β is a constant $(1 \leq \beta \leq 3)$ and randn is random value $(randn \in [0, 1])$. The standard deviation $\sigma(\beta)$ is calculated as Eq.18 where Γ is the gamma function.

$$\sigma(\beta) = \frac{\Gamma(1+\beta).sin(\pi.\frac{\beta}{2})}{\Gamma((\frac{1+\beta}{2}).\beta.2^{(\frac{\beta-1}{2})})}^{\frac{1}{\beta}} \tag{18}$$

B. relevance re-ranking optimization using Cuckoo Search: CS-Rank

In this section, our interest is to investigate the cuckoo search optimization in order to find the "genuine" ranked list by minimizing the objective function f in Eq 8. For this purpose, we consider the set of M score lists as the initial nests where each score list s_j has a position given by the scores in the list. The algorithm starts with taking one by one solution from the initial population (M nests) and then replacing it by a new one generated as follows:

$$s_i{}^{t+1} = W * s_i{}^t + stepsize * randn_i \tag{19}$$

where W is the inertia weight parameter,that is responsible to control the acceleration of the cuckoo in its original direction. W is defined as follows:

$$W = W_{max} - ((W_{max} - W_{min})/iter_{max}) * iter \tag{20}$$

The stepsize parameter is computed by the following equation:

$$stepsize = 0.01 * step * (s_i - s_{best}) \tag{21}$$

where $(s_i - s_{best})$ means that the solution remains unchanged when the solution corresponds to the best solution.

The update process of the s_{best} pattern in the CS algorithm is defined by:

$$s_{best} = s_i \qquad if \qquad f(s_{best}) < f(s_i) \tag{22}$$

The process of abandoning a fraction(pa) of worst solutions (nests) and building new ones is defined by:

$$s_i = \begin{cases} s_i + stepsize * randn_i & \text{if} \quad randn_i > pa \\ s_i & \text{else} \end{cases} \tag{23}$$

VI. Experiments and Results

In this section, we evaluate our enhanced relevance re-ranking framework using APSO and CS optimization over NusWide dataset[24] and compare them with the base-line re-ranking approach "Random Walk with restart" [25]. Also,we discuss the influence of different settings of parameters in our two proposed methods.

A. Experimental Setup

Our experiments are conducted on the challenging real-word NUS-WIDE dataset. It is one of the largest social media datasets which contains 269,648 Flickr images accompanied by their associated tags and their visual features (500 Bag of visual words). Each image is also indexed by 81 concepts. All the images are employed as the database images for retrieval, and all the 81 concepts are used for query analysis. In addition, we select a set of 5 common ambiguous short queries, including Apple, Jaguar, Dove, Pear, Jordan, Eagle[1]. In order to disambiguate these queries, the user expands the query by selecting the most representative concepts to his need. Then, results are reranked according to the introduced concepts in the query, followed by an optimization step ensured by our base-line Random Walk algorithm [25], or the proposed algorithms APSO-Rank and CS-Rank. For the performance metric, we adopt the Average precision for each query for the top-20 returned photos since our aim in this paper is to enhance the precision at top-ranked photos.

B. Experimental Results

In his section, we study the impact of varying the values of different parameters of both algorithms, APSO-Rank and CS-Rank, in order to define the most sensitive parameters to the re-ranking process.

1) PSO Parameters tuning:

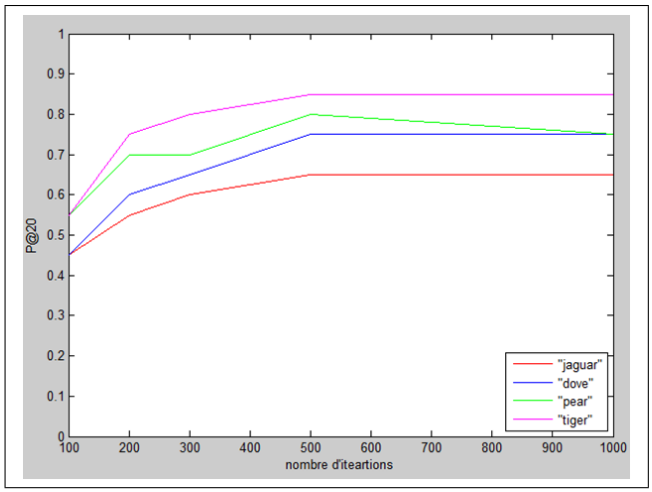

Fig. 1. Iteration round tuning in APSO-Rank

a) Iteration round: As mentioned before, the APSO-Rank stops when the maximum iteration number is encountered. As such, the genuine ranked list is identified by the global best at the last iteration. In order to avoid too many useless iterations, it will be interesting to find an optimal number of iterations. For this purpose, we vary the number of iterations(i.e.,100,200,300,500,1000) and then we evaluate the performance of the obtained APSO-Rank with those five values over four queries. As shown in fig.1, the performance keeps raising as the iteration number increases. However, after 200 iterations, the precision increases slowly. Subsequently, we can select 200 as the maximum number of iterations to avoid unnecessary computational costs.

b) Population size: We also investigate the performance of APSO-Rank with a different population size. For this purpose, we vary the size of the swarm from 10 to 60 with step equal to 5. Fig.2 illustrates the obtained values of P@20 for different population sizes. It can be seen that using 20 particles provides the best performances for all queries. Thus, we set the optimal population size as equal to 20, to reduce the computational cost.

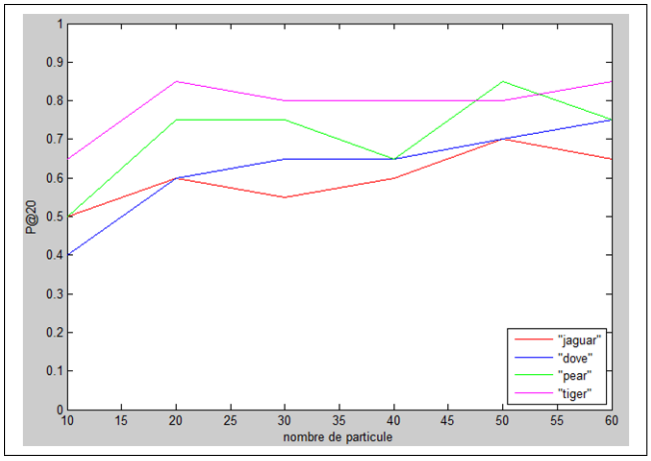

Fig. 2. Population size tuning in APSO-Rank

2) CS Parameters tuning:

a) Nests number: We conduct the experiment with varying the nests number from 5 to 50 nest with step as 5. We estimate the AP@20 and we obtain the following figure Fig.3. We can note that when using 10 nests, we obtain the best

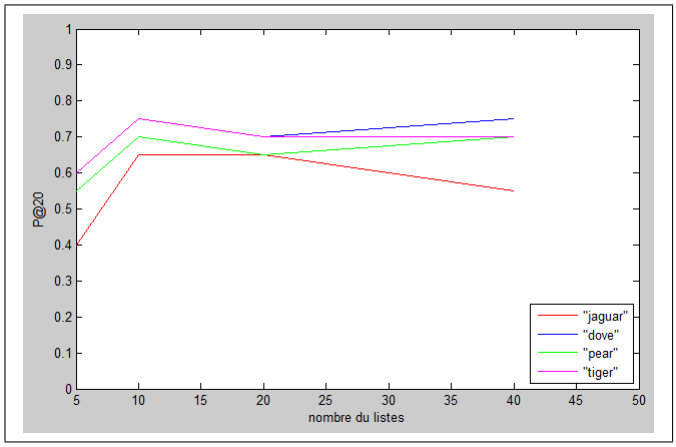

Fig. 3. Nests number tuning in CS-Rank

precision values. By augmenting this parameter, the precision increases too slowly and for some queries decreases. As Such, we define 10 as the optimal nests number.

b) Probability pa: The parameter pa introduced in the CS helps the algorithm to find globally improved solutions. Thus, pa is a very important parameter in fine-tuning of solution vectors. Fig.4 shows the the impact of varying the values of pa in the retrieval precision. It can be seen that a small value of pa provides better results , while if pa increases,the precision decreases since the system is unable to find the best solutions. It is worth noting that a small value of pa yields to increasing the number of iterations which decreases the efficiency of the algorithm. Thus, we should tune the parameter pa at each iteration.

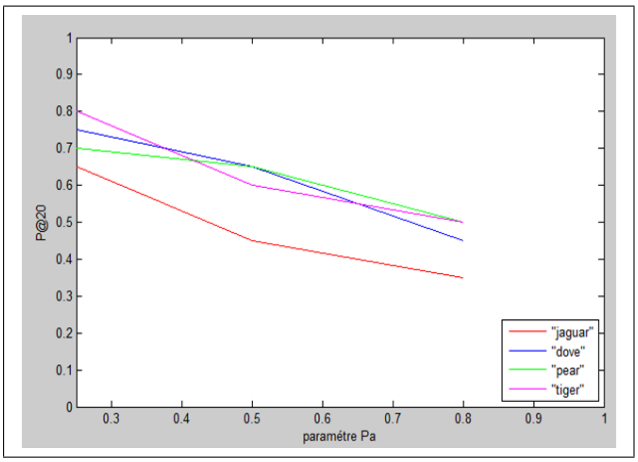

Fig. 4. Probability pa tuning in CS-Rank

3) Performance comparison:

a) Precision analysis: In this experiment, we will evaluate our two methods (CS-rank and APSO-Rank) by comparing them with tag-baseline and random walk(RWR-rank). For this purpose, we set the APSO-Rank parameters

as follows: 20 particles, 200 iterations. Also, we set CS-Rank parameters as follows: 20 nests, pa=0.25, 200 iterations. Fig.5 illustrates the top results of query "jaguar", from which we can see that the results of APSO-Rank and CS-Rank are more relevant than the results of Random Walk method and base-line system. Fig.6 shows the MAP for the above

Fig. 5. Top results of query "jaguar" using different methods

mentioned methods over five queries and top-20 returned results. It can be seen that CS-rank and APSO-Rank achieve more consistent performance improvements than the others. To be detailed, APSO-Rank achieves 0.315 as MAP, while CS-rank outperforms all methods with MAP value equal to 0.326. Base-line method obtains no-so-good MAP value equal to 0.281; also RWR-rank with 0.287.

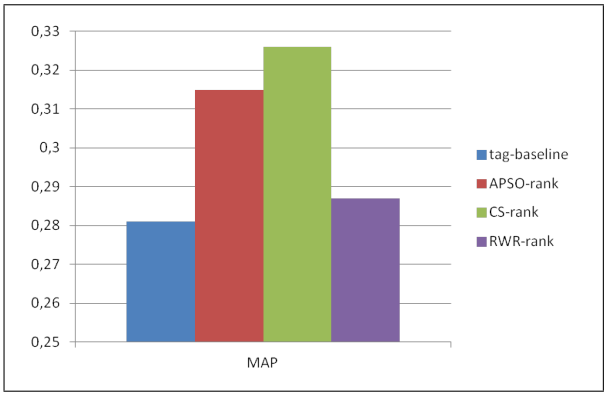

Fig. 6. MAP comparison of different methods

b) Time complexity analysis: The time complexity of APSO-Rank and CS-Rank algorithms can be derived from their pseudo-codes. The time complexity analysis reflects the calculation's complexity, involved in the algorithm, in order to estimate the necessary time to produce the output. The time complexity of the APSO-Rank algorithm is given by: $O(Ngen * (Nparticle * d + NParticle + Nparticle * d)) \simeq O(Ngen * Nparticle * d)$. Meanwhile, the time complexity of the CS-Rank algorithm is given by: $O(Ngen * (NNest * d + NNest + NNest * d)) \simeq O(Ngen * Nparticle * d)$, where d is the space dimension, Ngen is the generation number or the iteration round.

The algorithm is run till the stopping condition is met which in this case is till the solutions converge to a point with a tolerance of 0.001.

The algorithms are run on a system with core $i-7$ processor, $8GB$ memory on Matlab version $7.12.0.635$. The execution time in seconds taken by these algorithms is given in Figure7 Looking into the time complexity figure, we can see

Algorithms	« jaguar »	« tiger »	« dove »
PSO	180.45	286.21	139.58
CS	45.26	73.09	37.59

Fig. 7. Time taken by the algorithms for different queries(in seconds)

clearly that CS-Rank performs more efficiently than APSO-Rank. The main reason is the fact that CS algorithm uses levy flight. Thus we can deduce that levy flights helps to converge to the solution fast thereby increasing the efficiency. To investigate in deep the efficiency criteria, we study the

Fig. 8. Fitness convergence of CS-Rank and APSO-Rank

convergence of the fitness functions over the iterations for APSO-Rank and CS-Rank. Figure8 shows both the convergence of fitness of CS-Rank and APSO-Rank according to the number of iterations. The results show that APSO-Rank may converge prematurely to a local optimum, while cuckoo search can usually converge to the global optimality.

VII. Conclusion

Two new stochastic relevance re-ranking methods are presented in this paper. The relevance re-ranking problem is, then, formulated as an optimization problem, and is solved by using disjointly the particle swarm optimization and the cuckoo search. The score consistency regularization and ranking distance are integrated into the proposed APSO-Rank and CS-rank to derive the objective function. Experiments were conducted over the well known Nus-Wide photo collection and the obtained results have demonstrated that these two methods outperform the most used re-ranking approach "Random Walk with restart". In addition, experiments have shown that CS-rank achieved the best performance by comparing it to APSO-Rank since the Cuckoo search is more efficient and effective. In fact, experiments have demonstrated that particle swarm optimization can converge quickly to the best solution, whatever

the global best or the local best. So, there is no guarantees that the APSO-Rank converges to the global optimum. Meanwhile, Cuckoo search satisfies the global convergence conditions and thus it can usually converge to the global optimality. Indeed, thanks to cuckoo search capabilities (local search and global search), the global optimum can be reached.

ACKNOWLEDGMENT

The authors would like to acknowledge the financial support of this work by grants from General Direction of Scientific Research (DGRST), Tunisia, under the ARUB program.

REFERENCES

[1] A. Ksibi, A. B. Ammar, and C. B. Amar, "Adaptive diversification for tag-based social image retrieval," *IJMIR*, vol. 3, no. 1, pp. 29–39, 2014.

[2] A. Ksibi, G. Feki, A. B. Ammar, and C. B. Amar, "Effective diversification for ambiguous queries in social image retrieval," in *CAIP (2)*, 2013, pp. 571–578.

[3] G. Feki, A. Ksibi, A. Ben Ammar, and C. Ben Amar, "Improving image search effectiveness by integrating contextual information," in *Content-Based Multimedia Indexing (CBMI), 2013 11th International Workshop on*. IEEE, 2013, pp. 149–154.

[4] R. Sousa, I. Yevseyeva, J. F. P. da Costa, and J. S. Cardoso, "Multicriteria models for learning ordinal data: A literature review," in *Artificial Intelligence, Evolutionary Computing and Metaheuristics*. Springer, 2013, pp. 109–138.

[5] X.-S. Yang, *Nature-inspired metaheuristic algorithms*. Luniver Press, 2010.

[6] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*. IEEE, 1995, pp. 39–43.

[7] X.-S. Yang and S. Deb, "Engineering optimisation by cuckoo search," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 1, no. 4, pp. 330–343, 2010.

[8] P. Civicioglu and E. Besdok, "A conceptual comparison of the cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms," *Artificial Intelligence Review*, pp. 1–32, 2013.

[9] M. Okayama, N. Oka, and k. Kameyama, "Relevance optimization in image database using feature space preference mapping and particle swarm optimization," in *Neural Information Processing*. Springer, 2008, pp. 608–617.

[10] M. Broilo and F. G. De Natale, "A stochastic approach to image retrieval using relevance feedback and particle swarm optimization," *Multimedia, IEEE Transactions on*, vol. 12, no. 4, pp. 267–277, 2010.

[11] L. Zhang, T. Mei, Y. Liu, D. Tao, and H.-Q. Zhou, "Visual search reranking via adaptive particle swarm optimization," *Pattern Recognition*, vol. 44, no. 8, pp. 1811–1820, 2011.

[12] X. Tian, Y. Yang, J. Wang, X. Wu, and X.-S. Hua, "Bayesian visual reranking," *Multimedia, IEEE Transactions on*, vol. 13, no. 4, pp. 639–652, 2011.

[13] Z. Ji, P. Jing, Y. Su, and Y. Pang, "Rank canonical correlation analysis and its application in visual search reranking," *Signal Processing*, vol. 93, no. 8, pp. 2352–2360, 2013.

[14] M. Villegas, L. A. Leiva, and R. Paredes, "Interactive image retrieval based on relevance feedback," in *Multimodal Interaction in Image and Video Applications*. Springer, 2013, pp. 83–109.

[15] M. Wang and X.-S. Hua, "Active learning in multimedia annotation and retrieval: A survey," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 2, p. 10, 2011.

[16] T. Yao, T. Mei, and C.-W. Ngo, "Co-reranking by mutual reinforcement for image search," in *Proceedings of the ACM International Conference on Image and Video Retrieval*. ACM, 2010, pp. 34–41.

[17] L. Nie, S. Yan, M. Wang, R. Hong, and T.-S. Chua, "Harvesting visual concepts for image search with complex queries," in *Proceedings of the 20th ACM international conference on Multimedia*, ser. MM '12. New York, NY, USA: ACM, 2012, pp. 59–68.

[18] M. Naaman, "Social multimedia: highlighting opportunities for search and mining of multimedia data in social media applications," *Multimedia Tools and Applications*, vol. 56, no. 1, pp. 9–34, 2012.

[19] L. Duan, W. Li, I.-H. Tsang, and D. Xu, "Improving web image search by bag-based reranking," *Image Processing, IEEE Transactions on*, vol. 20, no. 11, pp. 3280–3290, 2011.

[20] D. C. G. Pedronette and R. d. S. Torres, "Exploiting clustering approaches for image re-ranking," *Journal of Visual Languages and Computing*, vol. 22, no. 6, pp. 453–466, 2011.

[21] Y. Jing and S. Baluja, "Visualrank: Applying pagerank to large-scale image search," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, no. 11, pp. 1877–1890, 2008.

[22] A. F. Kamaruzaman, A. M. Zain, S. M. Yusuf, and A. Udin, "Levy flight algorithm for optimization problems-a literature review," *Applied Mechanics and Materials*, vol. 421, pp. 496–501, 2013.

[23] X.-S. Yang and S. Deb, "Cuckoo search via levy flights," in *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*. IEEE, 2009, pp. 210–214.

[24] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y.-T. Zheng, "Nus-wide: A real-world web image database from national university of singapore," in *Proc. of ACM Conf. on Image and Video Retrieval (CIVR'09)*, Santorini, Greece., July 8-10, 2009.

[25] A. Ksibi, A. Ben Ammar, and C. Ben Amar, "Enhanced context-based query-to-concept mapping in social image retrieval," in *Content-Based Multimedia Indexing (CBMI), 2013 11th International Workshop on*. IEEE, 2013, pp. 85–89.