Approaching the Rank Aggregation Problem by Local Search-based Metaheuristics

Juan A. Aledo[*], José A. Gámez[†], David Molina [‡]

This is the accepted version of:

Juan A. Aledo, Jose A. Gámez. David Molina
Approaching the Rank Aggregation Problem by Local Search-based Metaheuristics.
Journal of Computational and Applied Mathematics, 354:445-456 (2019)
https://doi.org/10.1016/j.cam.2018.03.014

Please, visit the provided url to obtain the published version.

[*]Department of Mathematics, University of Castilla-La Mancha [juan.angel@aledo.uclm.es]

[†]Department of Computer Systems, University of Castilla-La Mancha [jose.gamez@uclm.es]

[‡]Department of Mathematics, University of Castilla-La Mancha [david.molina@uclm.es]

Abstract

Encouraged by the success of applying metaheuristics algorithms to other ranking-based problems (Kemeny ranking problem and parameter estimation for Mallows distributions), in this paper we deal with the rank aggregation problem (RAP), which can be viewed as a generalization of the Kemeny problem to arbitrary rankings. While in the Kemeny problem the input is a set of permutations, the RAP consists in obtaining the consensus permutation for a sample of arbitrary rankings.

This is an NP-hard problem which can be approached by using greedy heuristic algorithms (e.g. Borda). Such algorithms are fast but the solutions so obtained are far to be optimal. In this paper we propose the use of more complex search processes to deal with the RAP. In particular, we perform a comparative study among some local-based search metaheuristics: hill climbing (HC), iterated local search (ILS), variable neighborhood search (VNS) and greedy randomized adaptive search procedure (GRASP).

We provide a complete analysis of the experimental study regarding accuracy and number of iterations required to reach the best solution. From the results we can conclude that the selection of a suitable neighborhood plays a key role, and that depending on the available resources (cpu time) a different algorithm (VNS, ILS or GRASP) could be the proper choice.
Keywords: Ranking, Rank aggregation problem, Metaheuristic algorithm, Kendall distance, Permutation, Partial ranking.

1 Introduction

In the last decades, the *Rank Aggregation Problem* (RAP) [22, 23, 45] has gained popularity in several fields as statistics and machine learning because of its significant applications in many real-world problems [12, 15, 22, 25, 35], including information retrieval and recommender systems.

Rankings represent preferences in a natural way. Given a set of items $[n] = \{1, 2, \ldots, n\}$, a ranking π is an ordering of (some of) these items. The case when the rankings are permutations (complete rankings without ties) has received a great attention in the literature [28, 34]. However, many real world problems usually deal with incomplete rankings, that is, those where only p items are ranked, $2 \leq p < n$, and/or with rankings where some items are equally preferred or tied (see Section 2).

Given a sample of arbitrary rankings, the solution to the RAP is the *consensus* permutation, that is, the permutation which best summarizes the rankings in the sample. When all the rankings in the sample are permutations, this problem is known as the *Kemeny ranking problem* [6, 37]. Both are NP-hard problems when the number of rankings to aggregate is greater than 3 [9, 10].

The RAP has been widely studied and several proposals have arisen in the last years. For instance, Fagin et al [26, 27] proposed four new metrics for comparing complete rankings with ties and developed an algorithm to aggregate multiple rankings of this nature. On the other hand, Dwork et al [22, 23] focused on the framework of incomplete rankings without ties and their aggregation. Another interesting familiy of rank aggregation problems are those which produce a weak order as output by considering different distances to define the objective function [5, 7, 18, 17, 31, 49]. Recently, metaheuristics algorithms have been used to approach several ranking-based problems [2, 4, 18, 40, 41].

In this paper we deal with problem instances of the RAP containing any kind of rankings (complete, incomplete, with and without ties). In a recent paper [3], the authors have developed a version of the greedy Borda algorithm tailored to this problem, which clearly outperforms the standard Borda method. Now, we go one step further, and with the goal of obtaining better solutions (closer to the global optimum) we propose the use of more complex search engines. Since local search-based metaheuristics have shown a good trade-off between efficiency and accuracy in related problems (e.g. the traveling salesman problem [42], the routing-packing problem [14], the vertex separation problem [20], the linear ordering problem [13], etc.), in this study we focus on this family of metaheuristics. Specifically, we perform a comparative study among the following algorithms:

- *Hill climbing (HC)* algorithm [46]. It is a local search algorithm which iteratively tries to improve a given solution by moving at each iteration to the neighbor representing the biggest increase (decrease) in the evaluation or objective function with respect to the current solution.

- *Iterated local search (ILS)* method [39]. It is a multi-start local search algorithm based on the HC algorithm, which tries to escape from the local optimum by perturbing it, and use the resulting configuration to seed a new HC iteration.

- *Variable neighborhood search (VNS)* [32]. It is also a multi-start local search algorithm which, instead of modifying the starting point at each iteration, changes to a different neighborhood.

- *Greedy randomized adaptive search procedure (GRASP)* [43]. It is a multi-start local search algorithm which, at each iteration, constructs a randomized informed solution and locally improves it.

Thus, the main contribution of this paper is to provide a comparative study among the selected MHs. We consider different neighborhoods and allowed resources (number of fitness evaluations). The results show the influence of the starting point (see Section 4.3) and the selected neighborhood in the performance of the (local) search algorithms (see Section 4.4). Regarding the trade-off between goodness of the solution and number of evaluations, we get that the VNS is the best choice under limited resources, while the GRASP performs better when more evaluations are allowed (Section 3.1).

2 Preliminaries

2.1 Rankings

Given a set $[n] = \{1, 2, \ldots, n\}$ of items, a *ranking* π represents an order of preference of (some of) these items.

Rankings can be classified as *complete* (the n items are ranked) or *incomplete* (only p items are ranked, $2 \leq p < n$). Rankings can also be classified as *with-ties*[1] or *without-ties*, depending on if they present lack of preference information among some ranked items. The set of all the (complete or incomplete, with or without ties) rankings of the elements $1, 2, \ldots, n$ will be denoted as $\widetilde{\mathbb{S}}_n$.

Rankings will be written as lists of items, from most to least preferred, separated by vertical bars. Items between two consecutive vertical bars constitute a *bucket*. Items in the same bucket (separated by commas) are *tied* regarding the preference criterion.

Thus, by

$$\sigma = (x_{11}, x_{12} \ldots, x_{1j_1} | x_{21}, x_{22} \ldots, x_{2j_2} | \ldots | x_{k1}, x_{k2} \ldots, x_{kj_k})$$

[1] In the literature, rankings with ties are also known as partial or weak rankings [27].

with $1 \leq j_i$, $1 \leq k \leq n$ and $2 \leq \sum_{i=1}^{k} j_i \leq n$, we will denote a ranking where the items $x_{11}, x_{12} \ldots, x_{1j_1}$ are in the first bucket, $x_{21}, x_{22} \ldots, x_{2j_2}$ are tied in the second bucket and so on. In particular, we will denote by $B_i(\sigma) = \{x_{i1}, x_{i2} \ldots, x_{ij_i}\}$ the i-th bucket of σ.

Table 1 shows examples of the different types of rankings for $n = 4$.

Table 1: Example of different types of rankings ($n = 4$).

Rankings	Complete	Incomplete
Without-ties	$(3\|2\|4\|1)$	$(2\|4\|1)$
With-ties	$(3,2\|4\|1)$	$(3\|4,1)$

In particular, complete rankings without ties are permutations of the items $1, 2, \ldots, n$. As usual, we will denote by $\mathbb{S}_n$ the set of permutations of n items, $\mathbb{S}_n \subseteq \widetilde{\mathbb{S}}_n$. Given $\pi \in \mathbb{S}_n$, we will denote by $\pi(i)$ the i-th ranked element in π. Besides, for $i, j \in [n]$, we will write $i \prec_\sigma j$ to indicate that i precedes j in a ranking $\sigma \in \widetilde{\mathbb{S}}_n$. We will also use $\prec_\sigma$, when necessary, to indicate preference order among buckets. Thus, according to our notation, we have $B_1(\sigma) \prec_\sigma B_2(\sigma) \prec_\sigma \cdots \prec_\sigma B_k(\sigma)$.

2.2 The rank aggregation problem

Given a sample with N rankings $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_N\} \subseteq \widetilde{\mathbb{S}}_n$, the *Rank Aggregation Problem* (RAP) [22, 23, 45] consists in obtaining the consensus permutation in $\mathbb{S}_n$ (see Definition 3). Roughly speaking, the consensus permutation is the one minimizing the differences regarding all the rankings in the sample. The difference between two rankings is computed by using a proper distance. The *Kendall tau* distance is usually considered for the RAP [1, 22, 23]:

Definition 1 (Kendall distance [38])**.** The *Kendall distance* $d(\pi, \overline{\pi})$ between two permutations $\pi, \overline{\pi} \in \mathbb{S}_n$ is defined as the total number of pairwise disagreements between π and $\overline{\pi}$. There is disagreement over an item pair (i, j), $i, j \in [n]$, if the relative order of i and j is different in π and $\overline{\pi}$.

Although Kendall distance was originally defined to compare two permutations, it has been adapted to cope with other types of rankings [1, 26, 22, 23].

Based on the classical idea of the Kendall distance d regarding to counting pairwise disagreements, in [3] the following extension of d was introduced for managing whichever pair of rankings:

Definition 2 (Extended Kendall distance)**.** The *extended Kendall distance* $d'(\sigma, \overline{\sigma})$ between two rankings $\sigma, \overline{\sigma} \in \widetilde{\mathbb{S}}_n$ is defined as the total number of item pairs over which they disagree, ignoring item pairs (i, j) non-ranked in both σ and $\overline{\sigma}$. More precisely, let $B^i(\sigma)$, $B^j(\sigma)$, $B^i(\overline{\sigma})$ and $B^j(\overline{\sigma})$ be the buckets[2] containing i and j in σ and $\overline{\sigma}$ respectively. Then, there is disagreement over an item pair (i, j) if:

i) $B^i(\sigma) \neq B^j(\sigma)$ and $B^i(\overline{\sigma}) \neq B^j(\overline{\sigma})$, and

ii) $B^i(\sigma) \prec_\sigma B^j(\sigma) \wedge B^j(\overline{\sigma}) \prec_{\overline{\sigma}} B^i(\overline{\sigma})$ or
$B^j(\sigma) \prec_\sigma B^i(\sigma) \wedge B^i(\overline{\sigma}) \prec_{\overline{\sigma}} B^j(\overline{\sigma})$.

In particular, when applied to permutations, d' agrees with the Kendall distance d. As commented in [3], although d' is not even a pseudometric, it can be used as a similarity measure to deal with rankings in $\widetilde{\mathbb{S}}_n$. In fact, d' is non-negative and symmetric, but the triangle inequality does not hold. For instance, $d'((1|2|3), (1, 2\,|3)) = 0$, $d'((2|1|3), (1, 2\,|3)) = 0$ and $d'((1|2|3), (2|1|3)) = 1$.

Definition 3 (Rank aggregation problem [23])**.** Given a sample with N rankings $\Sigma = \{\sigma_1, \sigma_2, \cdots, \sigma_N\} \in \widetilde{\mathbb{S}}_n$, the *rank aggregation problem* consists in finding the permutation π_0 that satisfies

$$\pi_0 = argmin_{\pi \in \mathbb{S}_n} \frac{1}{N} \sum_{i=1}^{N} d'(\sigma_i, \pi) \tag{1}$$

where $d'(\sigma_i, \pi)$ stands for the extended Kendall distance between σ_i and π. π_0 in (1) is the permutation that minimizes the sum of the total number of disagreements with respect to the rankings in the sample, and is called the *aggregation* or *consensus ranking* of the sample.

If the N rankings of the sample are permutations, the rank aggregation problem becomes the well-known *Kemeny ranking problem* [6]. Since permutations can be viewed as without-ties rankings of n buckets, each one formed by a unique item, the RAP can be thought as a generalization of the Kemeny ranking problem [3].

[2]Recall that a ranking without ties can be viewed as a ranking with ties where the size of all the buckets is 1.

3 Local search-based Metaheuristic algorithms

Since solving the RAP is an NP-hard problem ($N > 3$) [45], heuristic algorithms are usually considered to deal with it. In order to improve the quality of the solutions obtained by standard greedy algorithms (e.g. Borda), we propose to use metaheuristics (MH) algorithms [30]. Undoubtedly, there is room for their application, because due to the their exploitation/exploration trade-off when carrying out the search, they are expected to find good solutions in complex problem instances.

The two main components of local search algorithms are the *objective function* and the *neighborhood* used. As in this work we look for the permutation that minimizes the total number of disagreements with the rankings contained in the sample, given $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_N\} \subseteq \widetilde{\mathbb{S}}_n$ we define the objective function as

$$f(\pi, \Sigma) = \frac{1}{N} \sum_{i=1}^{N} d'(\sigma_i, \pi), \qquad \pi \in \mathbb{S}_n. \tag{2}$$

Note that the function f so defined also makes sense when π is an arbitrary ranking (not necessarily a permutation).

Regarding the neighborhood, we have selected the two most used ones in the literature [16, 11]:

- *The interchange neighborhood ($\mathcal{N}_X$).* A permutation $\overline{\pi}$ is in the neighborhood $\mathcal{N}_X(\pi)$ of π if $\overline{\pi}$ can be obtained from π by interchanging two elements. For instance, (1 2 3 4 5) is a neighbor of (1 2 5 4 3), while (1 2 3 4 5) is not a neighbor of (1 2 5 3 4). Formally:

$$\mathcal{N}_X(\pi) = \left\{ \begin{array}{l} \overline{\pi} \ : \ \exists i, j \in \{1, \ldots, n\} \text{ with } i \neq j \text{ s.t.} \\ \\ \qquad \overline{\pi}(j) = \pi(i) \\ \qquad \overline{\pi}(i) = \pi(j) \\ \qquad \forall k \neq i, j \quad \overline{\pi}(k) = \pi(k) \end{array} \right\}$$

  Notice that $\mathcal{N}_X(\pi)$ has $\frac{n^2-n}{2}$ members.

- *The insert neighborhood ($\mathcal{N}_I$).* A permutation $\overline{\pi}$ is in the neighborhood $\mathcal{N}_I(\pi)$ of π if $\overline{\pi}$ can be obtained from π by removing an element and

reinserting it in a different position. For instance, (1 2 3 4 5) is a neighbor of (1 2 5 3 4), while (1 2 3 4 5) is not a neighbor of (1 2 5 4 3). Formally:

$$\mathcal{N}_I(\pi) = \left\{ \begin{array}{l} \overline{\pi} \ : \ \exists i, j \in \{1, \ldots, n\} \text{ with } i \neq j \text{ s.t.} \\ \\ \quad \text{if } i < j \\ \qquad\qquad \overline{\pi}(k) = \pi(k), k < i \wedge k > j \\ \qquad\qquad \overline{\pi}(k) = \pi(k+1), i \leq k < j \\ \qquad\qquad \overline{\pi}(j) = \pi(i) \\ \quad \text{if } i > j \\ \qquad\qquad \overline{\pi}(k) = \pi(k), k < j \wedge k > i \\ \qquad\qquad \overline{\pi}(k) = \pi(k-1), j < k \leq i \\ \qquad\qquad \overline{\pi}(j) = \pi(i) \end{array} \right\}$$

Notice that $\mathcal{N}_I(\pi)$ has $(n-1)^2$ members.

Finally, another important issue in local search is the solution selected to seed the search process. It motivates our first experiment (Section 4.3).

In the next subsections we describe the selected approaches.

3.1 Hill Climbing

The simplest local-search method is the Hill Climbing (HC) algorithm [30, 46], which iteratively tries to improve the current solution by evaluating its neighborhood and selecting the best neighbor regarding the objective function. This type of algorithm guarantees finding a local optimum in the search space defined by the neighborhood used. Figure 1 shows the pseudo-code of the HC algorithm, particularized to the RAP:

HC is a fast algorithm that returns a local optima regarding the used neighborhood. Because of this, several MHs are designed by using it as building block, coupled with procedures to (try to) escape from local optima.

3.2 Iterated local search (ILS) [39]

ILS is a multi-start local search procedure which runs a HC algorithm at each iteration. It takes as starting point for iteration $t+1$ a perturbation of

	HC (π, Σ, $\mathcal{N}$)
1	**input**: π, the starting point or initial candidate solution.
2	**input**: Σ, the sample.
3	**input**: $\mathcal{N}$, the neighborhood to be used.
4	$f_\pi \leftarrow f(\pi, \Sigma)$ // evaluate π
5	**Repeat**
6	Compute the neighborhood $\mathcal{N}(\pi)$
7	$\pi^* \leftarrow \arg\min_{\pi' \in \mathcal{N}(\pi)} f(\pi', \Sigma)$
8	**if** $f(\pi^*, \Sigma) < f_\pi$ **then** $(\pi, f_\pi) \leftarrow (\pi^*, f(\pi^*, \Sigma))$
9	**until** π does not change
10	**return** (π, f_π)

Figure 1: HC pseudocode.

the local optimum returned by the HC at iteration t. The process is repeated until the stopping criterion is satisfied.

Using a perturbed local optimum instead of a non-informed (randomly generated) individual lets re-use previous knowledge in order to (re-)start the search from promising points. The strength of the perturbation must be carefully managed to avoid worsening the current *good* solution too much, but to allow the algorithm to escape from the current local optimum basin of attraction. The perturbation is carried out by means of a *Shaking(π,k)* function which obtains the new individual by sequentially applying k random interchanges to the input permutation π.

Figure 2 shows the pseudocode of the ILS algorithm.

ILS has been profusely applied to solve problems in the space of permutations [4, 36, 44, 47].

3.3 Variable neighborhood search (VNS) [32]

In contrast to ILS, VNS does not change the starting point after being trapped in a local optimum, but the neighborhood instead. This allows VNS to explore a different (usually wider) area in which better solutions may be found. A set of neighborhoods of increasing complexity is usually considered; then, once a better solution is tracked down in a neighborhood, the VNS returns to the simplest one.

In the literature related to permutation problems, a particularly successful instance of VNS is one that alternates $\mathcal{N}_X$ and $\mathcal{N}_I$ during the search process

	ILS $(\pi, \Sigma, k, \mathcal{N})$
1	**input**: π, the starting point or initial candidate solution.
2	**input**: Σ, the sample.
3	**input**: k, the number of interchanges for the *Shaking* function.
4	**input**: $\mathcal{N}$, the neighborhood to be used.
5	$f_\pi \leftarrow f(\pi, \Sigma)$ // evaluate π
6	$\pi^* \leftarrow \pi$
7	**While** not stopping criterion **do**
8	$(\pi^*, f_{\pi^*}) \leftarrow HC(\pi^*, \Sigma, \mathcal{N})$
9	**if** $f_{\pi^*} < f_\pi$ **then** $(\pi, f_\pi) \leftarrow (\pi^*, f_{\pi^*})$
10	$\pi^* \leftarrow$ *Shaking*(π^*,k)
11	**end**
12	**return** (π, f_π)

Figure 2: ILS pseudocode.

[11, 16]. In our study we apply this approach, which is outlined in Figure 3.

	VNS $(\pi, \Sigma, \mathcal{N} = \{\mathcal{N}_1, \mathcal{N}_2\})$
1	**input**: π, the starting point or initial candidate solution.
2	**input**: Σ, the sample.
3	**input**: $\mathcal{N}$, the set of neighborhoods to be used.
4	$f_\pi \leftarrow f(\pi, \Sigma)$ // evaluate π
5	**Repeat**
6	$(\pi, f_\pi) \leftarrow \mathrm{HC}(\pi, \Sigma, \mathcal{N}_1)$
7	$(\pi, f_\pi) \leftarrow \mathrm{HC}(\pi, \Sigma, \mathcal{N}_2)$
8	**until** π does not change
9	**return** (π, f_π)

Figure 3: VNS pseudocode.

In our experiments $\mathcal{N}_1 = \mathcal{N}_X$ and $\mathcal{N}_2 = \mathcal{N}_I$.

3.4 Greedy randomized adaptive search procedure (GRASP)

GRASP is a multi-start MH algorithm whose popularity has increased in the last years [21, 43]. Each GRASP iteration can be divided into two phases: *construction* and *improving*. The construction phase consists in obtaining a randomized informed solution, while the second phase is dedicated to the

improvement of the constructed solution, usually by means of a local search algorithm.

The construction phase must be suitably designed to assure good but diverse starting points for the second phase. This procedure is usually carried out by using a *greedy randomized construction function* which involves the step-wise construction of a candidate solution using a randomized greedy process. In practice, it works by creating at each step a candidate list (RCL) containing only the most promising elements according to the objective function. Then, one of the elements in the RCL is randomly selected and incorporated to the current partial solution. This procedure is repeated until the solution is completed.

This solution is then taken as the initial starting point for the improving process, whose aim is to attain a local optimum.

Now, we particularize the GRASP procedure to our problem:

- *Construction phase.* Given an incomplete without ties ranking π with p elements, $0 \leq p < n$, and an element j not ranked in π, we define the *adding* function $A(\pi, j)$ which returns the ranking with $p+1$ elements that ranks j after π.

  In this phase we start with the *empty* ranking (.) and iteratively use the adding function to construct the solution.

  At each step, the cost of adding each remaining element to the current ranking is evaluated by using the objective function (2) and the best $r\%$ elements are introduced in the RCL. Then, one element is randomly selected from the RCL and added to the ranking. The process is repeated until all the n elements are ranked.

- *Improving phase.* We use the HC algorithm described in Section 3.1.

Figure 4 shows the pseudocode of the greedy randomized construction function (GRCF), while Figure 5 shows the full pseudocode of the GRASP algorithm:

Remark: Notice that in line 5 of algorithm GRCF we have abused of the notation in the use of $f(A(\pi, j), \Sigma)$ because, except in the last iteration, $A(\pi, j)$ is not a permutation but an incomplete ranking without ties. However, this is not a problem because, as pointed out after equation 2, $f(.,.)$ uses the *extendend Kendall distance* $d'(.,.)$, which is defined for any type of rankings.

```
    GRCF (r, Σ)
1   input: r, the percentage of elements for the RCL.
2   input: Σ, the sample.
3   π ← (.)
4   For k = 1 to n do
5       RCL = {j : j is not ranked in π
                   and f(A(π, j), Σ) is in the top r%}
6       Select an element j from RCL randomly.
7       π ← A(π, j)
8   end
9   return π.
```

Figure 4: GRASP construction phase pseudocode.

```
    GRASP (Σ, 𝒩, r, maxIt)
1   input: Σ, the sample.
2   input: 𝒩, the neighborhood to be used.
3   input: r, the percentage of elements for the RCL.
4   input: maxIt, the number of iterations.
5   (π, f_π) ← ((.), +∞)
5   For k = 1 to maxIt do
6       π ← GRCF(r, Σ)
7       (π*, f_π*) ← HC(π, Σ, 𝒩)
8       if f_π* < f_π then (π, f_π) ← (π*, f_π*)
9   end
10  return (π, f_π).
```

Figure 5: GRASP pseudocode.

4 Experimental Evaluation

This section describes the experiments carried out in our study. We describe here the datasets of rankings generated, the methodology employed and the two experiments designed.

4.1 Datasets

We carried out the comparison among the described algorithms over a benchmark of 22 datasets previously used in [3]. As in [48], these datasets can be classified in *explicit*, if rankings were available directly (Sushi, $n = 100$ and $N = 5000$); *derived*, if rankings were obtained from different preference relations as item ratings (MovieLens, $n = 207$ and $N = 3255$ [48]); and *transformed*, if rankings were changed from complete to incomplete with ties ones (F1(p_d, p_b), $n = 25$ and $N = 20$; Tour(p_d, p_b), $n = 153$ and $N = 20$; ATP50(p_d, p_b), $n = 50$ and $N = 52$; ATP100(p_d, p_b), $n = 100$ and $N = 52$; and ATP200(p_d, p_b), $n = 200$ and $N = 52$).

The transformed datasets were constructed following the next procedure: for every single complete ranking in the dataset, from most to least preferred, each item u was deleted with probability p_d. If the item was kept, it was included in the current bucket with probability p_b and placed in the first place of a new bucket otherwise. The following combinations for (p_d, p_b) were used: F1$(1/3, 1/2)$, F1$(1/3, 3/4)$, F1$(1/2, 1/2)$, F1$(1/2, 3/4)$, Tour$(1/3, 3/4)$, Tour$(1/3, 5/6)$, Tour$(2/3, 3/4)$, Tour$(2/3, 5/6)$, ATP50$(1/3, 3/4)$, ATP50$(1/3, 5/6)$, ATP50$(2/3, 3/4)$, ATP50$(2/3, 5/6)$, ATP100$(1/3, 3/4)$, ATP100$(1/3, 5/6)$, ATP100$(2/3, 3/4)$, ATP100$(2/3, 5/6)$, ATP200$(1/3, 3/4)$, ATP200$(1/3, 5/6)$, ATP200$(2/3, 3/4)$ and ATP200$(2/3, 5/6)$. In order to avoid the sampling bias, 20 different datasets were generated for each different transformed dataset combination. The generated benchmark contains 402 datasets[3].

[3]The Sushi datasets is available in http://www.preflib.org/data/election/sushi/ED-00014-00000003.toi.
The MovieLens1M base datase is available in http://grouplens.org/datasets/movielens/1m/.
The base datasets for F1, Tour, ATP50, ATP100 and ATP200 are available in http://simd.albacete.org/wp-content/media-ftp/sites/1/02_Datasets-GMM.rar

4.2 Methodology.

The following algorithms and parameter settings were considered in the experiments:

- *HC and ILS*. They take as neighborhoods $\mathcal{N}_X$ (algorithms HC(X) and ILS(X)) or $\mathcal{N}_I$ (algorithms HC(I) and ILS(I)) (see Section 3.1).

- *VNS*. The algorithm alternates between the neighborhoods $\mathcal{N}_X$ and $\mathcal{N}_I$, starting by $\mathcal{N}_X$ since it is more efficient in terms of number of evaluations[4] ($\frac{n(n-1)}{2}$ versus $(n-1)^2$ neighbors). Once a good solution (local optimum) is identified by using $\mathcal{N}_X$, the search engine changes to $\mathcal{N}_I$, which has a better performance but at a higher cost. Then, the algorithm iterates between $\mathcal{N}_X$ and $\mathcal{N}_I$ until no improvement is obtained.

- *GRASP*. Again both neighborhoods are used leading to GRASP(X) and GRASP(I).

MH algorithms explore many potential solutions in order to learn from the search space landscape. Because of this, we designed two different experimental configurations by varying the maximum number of allowed evaluations:

- *Configuration 1*. Algorithms could carry out a large number of evaluations: $500\,n^2$. The aim is to study whether the algorithm is able to obtain good solutions provided it has enough resources.

- *Configuration 2*. The maximum number of allowed evaluations is considerably smaller: $100\,n^2$. The aim is to test which algorithms are able to arrive faster to good solutions. Faster algorithms are preferred in anytime conditions, that is, when a fast response is needed. Furthermore, these algorithms scale better to problem instances with a large dimension (n).

As local search methods strongly depend on the solution used to seed the search, we started with a preliminary experimental study (Experiment 1) to compare three possible ways of computing the starting point (see Section 4.3). Once we selected the best way to seed the algorithms, we developed a broad

[4]In preliminary experiments we also tested the algorithm starting with $\mathcal{N}_I$, obtaining similar results in accuracy but worst in terms of number of evaluations.

experimental comparison (Experiment 2) involving all the MH algorithms (see Section 4.4). In both experiments Configurations 1 and 2 were tested.

As most of the metaheuristics used in our study are intrinsically stochastic or get this behaviour because of the use of shaking, we carried out 10 independent runs of these algorithms for each dataset. Then, to avoid the bias introduced by the randomness, the average of the results obtained in the 10 runs was used in the comparative analysis. We report accuracy (mean extended Kendall distance, Definition 2) and efficiency (number of evaluations until the best solution of that run is found).

The code of all the algorithms was written in Java. Experiments were carried out in computers running under Linux operating system and a maximum of 20 GB of RAM memory.

4.3 Experiment 1

As local search methods strongly depend on the solution used to seed the search, in this experiment we considered three different initialization processes of establishing the starting point for the local search approaches:

- *Random (RN).* A permutation is selected at random.
- *Borda (BO).* A generalized version [3] of modified Borda count algorithm [24] is used.
- *Borda (EB).* A Borda count-based algorithm which uses *extension sets* to deal with the uncertainty introduced by ties and missing items. This algorithm was introduced in [3].

In this experiment, to avoid the effect of randomness, we only considered algorithms with a deterministic behaviour, i.e. HC and VNS.

4.3.1 Results

The algorithms HC(I), HC(X) and VNS were ran for all the datasets considering the three methods described above to generate the initial solution. In the case of RN, the same randomly generated initial solution was considered for the three algorithms.

The three possible initializations were studied for each algorithm independently. First, we compared the accuracy of the obtained solution (mean

extended Kendall distance (2)) and, when necessary, the number of fitness evaluations. The two configurations previously described were studied.

The statistical analysis was based on the Friedman Test + Holm's Procedure [8, 19]. First, a Friedman ranks test ($\alpha = 0.05$) [29] was carried out to decide if all the tested algorithms were equivalent or not. Then, a Holm's post-hoc procedure ($\alpha = 0.05$) [33] was performed by selecting as control the algorithm having the best mean rank.

In our study, we first applied this statistical analysis to the accuracy results and, for those algorithms which were not significantly different, the same procedure was applied regarding the number of fitness evaluations.

Configuration 1 Table 2 shows the outcome of the statistical analysis when $500\,n^2$ fitness evaluations were allowed. For the RN case the average of 10 independent runs was considered. The tables on the left part correspond to the accuracy analysis, while the ones on the right part represent the evaluations-based analysis for those algorithms having an equivalent performance regarding accuracy. In each inner table, from left to right the columns show the adjusted p-value using Holm's procedure, the ranking computed for the Friedman test, and the comparison of each algorithm (win/tie/loss) with respect to the control. Boldfaced results correspond to non-rejected hypotheses.

From the results we can observe that EB is the best choice for the three algorithms. In the cases when other algorithms are equivalent regarding accuracy, EB is the outstanding initializing method regarding number of evaluations.

Configuration 2 Table 3 shows the outcome of the statistical analysis when $100\,n^2$ fitness evaluations were allowed. As in Configuration 1, again EB is the outstanding initializing method.

4.4 Experiment 2

In this section we compare the MH algorithms HC, ILS, VNS and GRASP. For HC, ILS and GRASP we consider both neighborhoods $\mathcal{N}_X$ and $\mathcal{N}_I$ leading to six different algorithms: HC(X), HC(I), ILS(X), ILS(I), GRASP(X) and GRASP(I).

Table 2: Statistical analysis for Experiment 1 - Configuration 1. Tables on the left part correspond to the accuracy analysis. Tables on the right part correspond to the evaluations-based analysis for those algorithms having an equivalent performance regarding accuracy. Boldfaced results correspond to non-rejected hypotheses.

method	pvalue	rank	win	tie	loss
HC(I)-EB	-	1.57	-	-	-
HC(I)-BO	**2.9129e-01**	1.89	11	3	8
HC(I)-RN	2.3802e-03	2.55	19	0	3

method	pvalue	rank	win	tie	loss
HC(I)-EB	-	1.00	-	-	-
HC(I)-BO	3.3841e-07	2.00	22	0	0

method	pvalue	rank	win	tie	loss
HC(X)-EB	-	1.27	-	-	-
HC(X)-RN	5.2556e-04	2.32	19	0	3
HC(X)-BO	3.2795e-04	2.41	19	0	3

-

method	pvalue	rank	win	tie	loss
VNS-EB	-	1.70	-	-	-
VNS-RN	**2.6334e-01**	2.14	14	0	8
VNS-BO	**2.6334e-01**	2.16	14	1	7

method	pvalue	rank	win	tie	loss
VNS-EB	-	1.00	-	-	-
VNS-BO	5.2556e-04	2.05	22	0	0
VNS-RN	1.8044e-10	2.95	22	0	0

Table 3: Statistical analysis for Experiment 1 - Configuration 2. Tables on the left part correspond to the accuracy analysis. The table on the right part shows the evaluations-based analysis for those algorithms having an equivalent performance regarding accuracy. Boldfaced results correspond to non-rejected hypotheses.

method	pvalue	rank	win	tie	loss
HC(I)-EB	-	1.25	-	-	-
HC(I)-BO	3.4808e-02	1.89	16	1	5
HC(I)-RN	1.7414e-07	2.86	22	0	0

-

method	pvalue	rank	win	tie	loss
HC(X)-EB	-	1.23	-	-	-
HC(X)-BO	5.2556e-04	2.27	19	0	3
HC(X)-RN	4.8610e-05	2.50	20	0	2

-

method	pvalue	rank	win	tie	loss
VNS-EB	-	1.43	-	-	-
VNS-BO	**9.7254e-02**	1.93	16	1	5
VNS-RN	1.2937e-04	2.64	18	0	4

method	pvalue	rank	win	tie	loss
VNS-EB	-	1.02	-	-	-
VNS-BO	4.7683e-07	1.98	21	1	0

In the light of Experiment 1, we chose EB to seed the search for trajectory-based metaheuristics HC, ILS and VNS. As baseline we also considered the results obtained by EB (Tables 4 and 5).

As stopping criteria we used the two configurations described in Section 4.2. As ILS and GRASP are iterative algorithms, they were ran until the maximum number of fitness evaluations was achieved. HC algorithm may stop before that maximum number of evaluations (if trapped in a local optimum). To make a fair comparison between VNS and GRASP/ILS we allowed VNS to achieve the maximum number of evaluations by using the shaking function when it was trapped in a local optimum. Notice that, in essence, this approach mimics the one of ILS with respect to HC.

Apart from the number of maximum iterations given by the selected configuration, the following parameters need to be specified:

- The k parameter of the *Shaking* function is set to $k = 0.15\,n$.
- The r parameter of the GRCF function is set to $r = 20\%$.

Tables 4 and 5 show the accuracy (mean extended Kendall distance) for all the algorithms and datasets. The results are averaged over 10 independent runs for the non-deterministic algorithms: GRASP, ILS and VNS. The best result for each dataset is boldfaced.

By inspection, we can observe that GRASP(I) and ILS(I) are the algorithms which obtained the best results. However, to be in a position of extracting significant conclusions we carried out the same statistical study described in Experiment 1. Table 6 shows the outcome of the statistical analysis for both configurations.

According to the statistical analysis we can conclude that:

- The use of $\mathcal{N}_I$ is more suitable than $\mathcal{N}_X$ to deal with the RAP.
- The algorithms using randomness (ILS, GRASP and VNS) to escape from local optima outperform the greedy HC.
- In Configuration 1, that is, when the algorithms can use a large number of fitness evaluations, ILS(I) is ranked as the best one, although GRASP(I) and VNS(I) are not different in terms of accuracy when considering statistical significance.

Table 4: Accuracy (mean extended Kendall distance) for Configuration 1, averaged over 10 independent runs. The best result for each dataset is boldfaced.

problem	EB	GRASP(I)	GRASP(X)	HC(I)	HC(X)	ILS(I)	ILS(X)	VNS
Sushi100	50437.0	47796.8	47873.8	47797.0	47921.0	**47795.6**	47829.2	47796.0
MovieLens	1640502.0	1612296.5	1612730.2	1612298.0	1612798.0	**1612288.5**	1612586.1	1612365.5
F1(1/3, 1/2)	521.4	**492.2**	492.5	492.5	494.9	**492.2**	492.5	492.3
F1(1/3, 3/4)	376.3	**356.1**	356.3	356.5	358.9	**356.1**	356.3	356.3
F1(1/2, 1/2)	267.4	**241.3**	241.7	241.9	245.2	**241.3**	241.6	241.4
F1(1/2, 3/4)	182.2	**163.8**	163.9	164.1	166.5	**163.8**	163.8	163.8
Tour(1/3, 3/4)	27741.4	26904.2	26988.5	26911.6	27010.2	**26893.3**	26947.9	26896.3
Tour(1/3, 5/6)	26152.0	25373.0	25455.3	25380.5	25477.2	**25365.3**	25417.7	25368.7
Tour(2/3, 3/4)	6141.7	5554.2	5621.8	5560.5	5637.8	**5544.4**	5586.6	5546.2
Tour(2/3, 5/6)	5496.0	4934.5	4999.1	4941.3	5013.8	**4924.0**	4962.8	4926.4
ATP50(1/3, 3/4)	1903.5	**1735.8**	1736.7	1736.2	1742.3	1735.8	1736.6	1735.8
ATP50(1/3, 5/6)	1486.8	**1360.3**	1361.1	1360.8	1364.5	1360.4	1361.0	1360.5
ATP50(2/3, 3/4)	408.6	310.6	314.2	311.6	320.5	**310.5**	312.0	311.1
ATP50(2/3, 5/6)	302.2	218.0	221.4	219.2	225.9	**217.9**	219.1	218.2
ATP100(1/3, 3/4)	9804.1	9034.8	9056.7	9037.9	9073.0	**9034.5**	9048.2	9035.6
ATP100(1/3, 5/6)	8338.5	7699.8	7716.5	7702.0	7730.4	**7699.3**	7710.4	7701.0
ATP100(2/3, 3/4)	2184.4	1747.8	1769.3	1751.8	1777.4	**1746.4**	1756.5	1747.5
ATP100(2/3, 5/6)	1736.8	1376.0	1393.7	1380.1	1399.8	**1374.3**	1383.5	1375.3
ATP200(1/3, 3/4)	37481.9	33906.0	34014.8	33912.3	34047.2	**33901.4**	33983.5	33904.7
ATP200(1/3, 5/6)	33425.1	30230.0	30318.2	30233.6	30333.7	**30226.1**	30290.5	30230.8
ATP200(2/3, 3/4)	8890.3	6968.4	7044.8	6973.4	7055.2	**6962.0**	7010.5	6964.5
ATP200(2/3, 5/6)	7460.8	5761.0	5827.5	5763.6	5842.9	**5754.3**	5798.4	5757.4

Table 5: Accuracy (mean extended Kendall distance) for Configuration 2, averaged over 10 independent runs. The best result for each dataset is boldfaced.

problem	EB	GRASP(I)	GRASP(X)	HC(I)	HC(X)	ILS(I)	ILS(X)	VNS
Sushi100	50437.0	**47797.0**	47907.6	**47797.0**	47921.0	**47797.0**	47887.9	47835.0
MovieLens	1640502.0	1613767.0	1612798.0	1613767.0	1612798.0	1613767.0	1612798.0	**1612718.0**
F1(1/3, 1/2)	521.4	**492.3**	493.1	492.5	494.9	492.3	493.2	492.6
F1(1/3, 3/4)	376.3	**356.2**	356.9	356.5	358.9	356.2	356.8	356.4
F1(1/2, 1/2)	267.4	**241.4**	242.7	241.9	245.2	241.4	242.4	241.7
F1(1/2, 3/4)	182.2	**163.8**	164.6	164.1	166.5	163.8	164.3	164.0
Tour(1/3, 3/4)	27741.4	26923.8	27010.2	26923.8	27010.2	26923.8	26992.0	**26906.4**
Tour(1/3, 5/6)	26152.0	25394.0	25477.2	25394.0	25477.2	25394.0	25461.8	**25380.1**
Tour(2/3, 3/4)	6141.7	5580.6	5637.6	5580.6	5637.8	5580.6	5621.8	**5561.2**
Tour(2/3, 5/6)	5496.0	4958.9	5013.6	4958.9	5013.8	4958.9	4998.5	**4940.7**
ATP50(1/3, 3/4)	1903.5	**1735.9**	1738.6	1736.2	1742.3	1735.9	1739.2	1736.0
ATP50(1/3, 5/6)	1486.8	**1360.4**	1362.5	1360.8	1364.5	1360.5	1362.5	1360.6
ATP50(2/3, 3/4)	408.6	311.1	316.7	311.6	320.5	**310.8**	315.5	311.9
ATP50(2/3, 5/6)	302.2	218.7	223.0	219.2	225.9	**218.4**	221.9	218.8
ATP100(1/3, 3/4)	9804.1	9037.9	9066.9	9037.9	9073.0	**9037.2**	9064.6	9037.2
ATP100(1/3, 5/6)	8338.5	7702.0	7724.6	7702.0	7730.4	**7701.4**	7723.7	7702.6
ATP100(2/3, 3/4)	2184.4	1751.8	1774.4	1751.8	1777.4	1750.5	1770.0	**1750.1**
ATP100(2/3, 5/6)	1736.8	1380.1	1397.8	1380.1	1399.8	1378.7	1393.8	**1377.8**
ATP200(1/3, 3/4)	37481.9	34006.5	34047.2	34006.5	34047.2	34006.5	34042.7	**33928.6**
ATP200(1/3, 5/6)	33425.1	30315.1	30333.7	30315.1	30333.7	30315.1	30330.9	**30242.0**
ATP200(2/3, 3/4)	8890.3	7062.2	7055.2	7062.2	7055.2	7062.2	7050.6	**6985.3**
ATP200(2/3, 5/6)	7460.8	5842.8	5842.9	5842.8	5842.9	5842.8	5837.1	**5775.4**

Table 6: Statistical test (accuracy) for Configuration 1 (left) and Configuration 2 (right). Boldfaced results correspond to non-rejected hypotheses.

method	pvalue	rank	win	tie	loss
ILS(I)	-	1.18	-	-	-
GRASP(I)	**1.3965e-01**	2.27	16	4	2
VNS	**1.1280e-01**	2.59	22	0	0
HC(I)	6.5095e-05	4.32	22	0	0
ILS(X)	1.7975e-06	4.91	22	0	0
GRASP(X)	3.7640e-09	5.73	22	0	0
HC(X)	1.9983e-14	7.00	22	0	0
EB	1.8618e-19	8.00	22	0	0

method	pvalue	rank	win	tie	loss
VNS	-	2.14	-	-	-
ILS(I)	**9.2036e-01**	2.55	11	0	11
GRASP(I)	**9.2036e-01**	2.68	12	0	10
HC(I)	**1.4670e-01**	3.59	18	0	4
ILS(X)	1.4298e-03	4.77	22	0	0
GRASP(X)	5.8069e-06	5.73	22	0	0
HC(X)	1.4241e-08	6.55	22	0	0
EB	1.4221e-14	8.00	22	0	0

- Regarding Configuration 2, that is, when less evaluations ($100\,n^2$) are available, VNS becomes the best option in terms of accuracy, although ILS(I), GRASP(I) and HC(I) are not different in terms of statistical significance.

For both configurations and those algorithms non significantly different regarding accuracy, we studied the number of evaluations needed to find its best solution (Table 7). The statistical analysis is shown in Table 8. As can be seen, for Configuration 1 ($500\,n^2$ evaluations) among the equivalent accurate algorithms (ILS(I), GRASP(I) and VNS), GRASP(I) was the more efficient one. On the other hand, the outstanding algorithm for Configuration 2 ($100\,n^2$ evaluations) was VNS. Regarding Configuration 2, we can observe that GRASP(I), ILS(I) and HC(I) achieved the maximum number of allowed evaluations without reaching a local optimum. However, in the case of VNS the alternation between $\mathcal{N}_X$ and $\mathcal{N}_I$ allowed the algorithm to progress faster.

5 Conclusions

In this paper we carried out a comparative study among different local search based metaheuristics to deal with the Rank Aggregation Problem.

From the experiments and the posterior statistical analysis we can conclude that: (1) the selected neighborhood plays a key role, being the one based on *interchanging* items ($\mathcal{N}_I$) the more suitable to deal with the RAP; (2) the iterated version (by using shaking) of VNS is the best choice when some sort of anytime behaviour is required, that is, when the number of available fitness evaluations is restricted; and (3) the GRASP(I) shows the best tradeoff between accuracy and efficiency when the algorithms are allowed to perform a large number of fitness evaluations.

Table 7: Number of evaluations until the best solution was found by the algorithm for Configuration 1 (left) and Configuration 2 (right). Best results are boldfaced.

problem	GRASP(I)	ILS(I)	VNS
Sushi100	**1249381.2**	2795766.8	3748653.5
MovieLens	**9278900.0**	15611240.0	11822500.0
F1(1/3, 1/2)	**9358.5**	11522.7	30970.5
F1(1/3, 3/4)	16001.2	**15928.1**	30032.9
F1(1/2, 1/2)	**12718.8**	15123.6	28344.0
F1(1/2, 3/4)	**7816.3**	10178.4	25794.8
Tour(1/3, 3/4)	**5949467.0**	9336942.0	8383229.5
Tour(1/3, 5/6)	**5978073.0**	9052289.0	8466278.0
Tour(2/3, 3/4)	**5389928.5**	9591481.0	9096342.0
Tour(2/3, 5/6)	**5935165.5**	9829655.0	9479393.0
ATP50(1/3, 3/4)	**77041.6**	81660.4	120583.5
ATP50(1/3, 5/6)	**80459.4**	92636.8	82998.2
ATP50(2/3, 3/4)	282500.8	**234569.0**	390315.3
ATP50(2/3, 5/6)	**291186.2**	296421.2	298685.6
ATP100(1/3, 3/4)	**1725869.4**	1776262.8	1874875.2
ATP100(1/3, 5/6)	1752451.0	1746166.8	**1428282.6**
ATP100(2/3, 3/4)	**2283339.0**	2972731.0	2352479.2
ATP100(2/3, 5/6)	**2223691.2**	2971345.0	2658391.5
ATP200(1/3, 3/4)	**10391383.0**	13789709.0	11192464.0
ATP200(1/3, 5/6)	**9261262.0**	13608123.0	11409278.0
ATP200(2/3, 3/4)	**10145022.0**	16245669.0	14916155.0
ATP200(2/3, 5/6)	**8906247.0**	14644214.0	15030182.0

problem	GRASP(I)	HC(I)	ILS(I)	VNS
Sushi100	910801.0	910801.0	910801.0	**746461.4**
MovieLens	**4284900.0**	**4284900.0**	**4284900.0**	**4284900.0**
F1(1/3, 1/2)	7699.3	**5101.0**	8350.8	8647.8
F1(1/3, 3/4)	10612.6	**4351.0**	9260.3	6336.9
F1(1/2, 1/2)	11224.6	**5251.0**	11018.5	8032.2
F1(1/2, 3/4)	7762.3	**4981.0**	8224.9	5964.6
Tour(1/3, 3/4)	2327803.5	2327803.5	2327803.5	**2050307.6**
Tour(1/3, 5/6)	2329364.0	2329364.0	2329364.0	**2023523.4**
Tour(2/3, 3/4)	2340900.0	2340900.0	2340900.0	**2100519.0**
Tour(2/3, 5/6)	2337442.2	2337442.2	2337442.2	**2096351.2**
ATP50(1/3, 3/4)	58090.6	46551.0	53472.4	**39354.3**
ATP50(1/3, 5/6)	68295.0	43978.5	59328.2	**37149.5**
ATP50(2/3, 3/4)	85175.6	**56718.5**	97989.9	77245.6
ATP50(2/3, 5/6)	94828.7	**54391.0**	111502.1	73099.0
ATP100(1/3, 3/4)	531631.0	531631.0	609841.2	**470696.8**
ATP100(1/3, 5/6)	523216.0	523216.0	607712.7	**427458.5**
ATP100(2/3, 3/4)	599941.0	599941.0	742798.4	**546679.8**
ATP100(2/3, 5/6)	584596.0	584596.0	731066.9	**554550.6**
ATP200(1/3, 3/4)	4000000.0	4000000.0	4000000.0	**3992970.5**
ATP200(1/3, 5/6)	4000000.0	4000000.0	4000000.0	**3962115.5**
ATP200(2/3, 3/4)	4000000.0	4000000.0	4000000.0	**3993945.2**
ATP200(2/3, 5/6)	4000000.0	4000000.0	4000000.0	**3972045.2**

Table 8: Statistical test (efficiency) for Configuration 1 (left) and Configuration 2 (right). Boldfaced results correspond to non-rejected hypotheses.

method	pvalue	rank	win	tie	loss
GRASP(I)	-	1.18	-	-	-
VNS	9.3864e-05	2.41	21	0	1
ILS(I)	9.3864e-05	2.41	19	0	3

method	pvalue	rank	win	tie	loss
VNS	-	1.43	-	-	-
HC(I)	3.5558e-02	2.25	15	1	6
GRASP(I)	8.7352e-05	3.02	20	1	1
ILS(I)	5.0594e-06	3.30	20	1	1

Acknowledgements

This work has been partially funded by the Spanish Government (MINECO) and FEDER funds through project TIN2016-77902-C3-1-P.

References

[1] N. Ailon, Aggregation of partial rankings, p-ratings and top-m lists, Algorithmica 57 (2) (2010) 284–300.

[2] J. A. Aledo, J. A. Gámez, D. Molina, Tackling the rank aggregation problem with evolutionary algorithms, Applied Mathematics and Computation 222 (2013) 632–644.

[3] J. A. Aledo, J. A. Gámez, D. Molina, Using extension sets to aggregate partial rankings in a flexible setting, Applied Mathematics and Computation 290 (2016) 208–223.

[4] J. A. Aledo, J. A. Gámez, D. Molina, Using metaheuristic algorithms for parameter estimation in generalized Mallows models, Applied Soft Computing 38 (2016) 308–320.

[5] J. A. Aledo, J. A. Gámez, A. Rosete, Utopia in the solution of the bucket order problem, Decision Support Systems 97 (2017) 69–80.

[6] A. Ali, M. Meila, Experiments with Kemey ranking: What works when?, Mathematical Social Sciences 64 (1) (2012) 28–40.

[7] S. Amodio, A. D'Ambrosio, R. Siciliano, Accurate algorithms for identifying the median ranking when dealing with weak and partial rankings under the Kemeny axiomatic approach, European Journal of Operational Research 249 (2016) 667–676.

[8] J. Arias, J. Cózar, ExReport: Fast, reliable and elegant reproducible research (2015).
URL `http://exreport.jarias.es/`

[9] J. Bartholdi, C. A. Tovey, M. A. Trick, Voting schemes for which it can be difficult to tell who won the election, Social Choice and Welfare 6 (2) (1989) 157–165.

[10] N. Betzler, R. Bredereck, R. Niedermeier, Theoretical and empirical evaluation of data reduction for exact Kemey rank aggregation, Autonomous Agents and Multi-Agent Systems 28 (5) (2014) 721–748.

[11] J. Ceberio, Solving Permutation-based Combinatorial Optimization Problems with Estimation of Distribution Algorithms and Extensions Thereof, PhD Thesis, University of the Basque Country (EHU), 2014.

[12] J. Ceberio, E. Irurozki, A. Mendiburu, J. A. Lozano, A distance-based ranking model estimation of distribution algorithm for the flowshop scheduling problem, Evolutionary Computation, IEEE Transactions on 18 (2) (2013) 286–300.

[13] J. Ceberio, A. Mendiburu, J. A. Lozano, The linear ordering problem revisited, European Journal of Operational Research 241 (3) (2015) 686–696.

[14] S. Ceschia, A. Schaerf, T. Stützle, Local search techniques for a routing-packing problem, Computers & Industrial Engineering 66 (4) (2013) 1138–1149.

[15] W. Cheng, J. Hühn, E. Hüllermeier, Decision tree and instance-based learning for label ranking, in: Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09, ACM, 2009.

[16] W. E. Costa, M. C. Goldbarg, E. G. Goldbarg, New VNS heuristic for total flowtime flowshop scheduling problem, Expert Systems with Applications 39 (9) (2012) 8149 – 8161.

[17] A. D'Ambrosio, W. J. Heiser, A recursive partitioning method for the prediction of preference rankings based upon Kemeny distances, Psychometrika 81 (3) (2016) 774–794.

[18] A. D'Ambrosio, G. Mazzeo, C. Iorio, R. Siciliano, A differential evolution algorithm for finding the median ranking under the Kemeny axiomatic approach, Computers & Operations Research 82 (2017) 126–138.

[19] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.

[20] A. Duarte, L. F. Escudero, R. Martí, N. Mladenovic, J. J. Pantrigo, J. Sánchez-Oro, Variable neighborhood search for the vertex separation problem, Computers & OR 39 (12) (2012) 3247–3255.

[21] A. Duarte, R. Martí, Tabu search and GRASP for the maximum diversity problem, European Journal of Operational Research 178 (1) (2007) 71–84.

[22] C. Dwork, R. Kumar, M. Naor, D. Sivakumar, Rank aggregation methods for the web, in: WWW, 2001.

[23] C. Dwork, R. Kumar, M. Naor, D. Sivakumar, Rank aggregation revisited (2001).

[24] P. Emerson, The original Borda count and partial voting, Social Choice and Welfare 40 (2) (2013) 353–358.

[25] R. Fagin, R. Kuma, D. Sivakumar, Efficient similarity search and classification via rank aggregation, in: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, SIGMOD'03, ACM, 2003.

[26] R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, E. Vee, Comparing and aggregating rankings with ties, in: PODS, 2004.

[27] R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, E. Vee, Comparing partial rankings, SIAM J. Discrete Math 20 (3) (2006) 628–648.

[28] M. A. Fligner, J. S. Verducci, Distance based ranking models, Journal of the Royal Statistical Society 48 (3) (1986) 359–369.

[29] M. Friedman, A Comparison of Alternative Tests of Significance for the Problem of m Rankings, The Annals of Mathematical Statistics 11 (1) (1940) 86–92.

[30] M. Gendreau, J.-Y. Potvin, Handbook of Metaheuristics, 2nd ed., Springer, 2010.

[31] A. Gionis, H. Mannila, K. Puolamaki, A. Ukkonen, Algorithms for discovering bucket orders from data, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '06, 2006, pp. 561–566.

[32] P. Hansen, N. Mladenovi?, J. A. Moreno, Variable neighbourhood search: methods and applications, Annals of Operations Research 175 (1) (2010) 367–407.

[33] S. Holm, A simple sequentially rejective multiple test procedure, Scandinavian Journal of Statistics 6 (1979) 65–70.

[34] J. Huang, Probabilistic reasoning and learning on permutations: Exploiting structural decompositions of the symmetric group, Ph.D. thesis, Carnegie Mellon University (2011).

[35] B. N. Jackson, P. S. Schnable, S. Aluru, Consensus genetic maps as median orders from inconsistent sources, IEEE/ACM Trans. Comput. Biol. Bioinformatics 5 (2) (2008) 161–171.

[36] A. A. Juan, H. R. Lourenço, M. Mateo, R. Luo, Q. Castella, Using iterated local search for solving the flow-shop problem: Parallelization, parametrization, and randomization issues, International Transactions in Operational Research 21 (1) (2014) 103–126.

[37] J. L. Kemeny, J. G. Snell, Mathematical models in the social sciences, Blaisdell, New York.

[38] M. G. Kendall, A new measure of rank correlation, Biometrika 30 (1/2) (1938) 81–93.

[39] H. R. Lourenço, O. C. Martin, T. Stützle, Iterated local search, in: Handbook of Metaheuristics, volume 57 of International Series in Operations Research and Management Science, Kluwer Academic Publishers, 2002.

[40] M. Mandal, S. Maity, A. Mukhopadhyay, Partial rank aggregation using multiobjective genetic algorithm: Application in ranking genes, in: Eighth International Conference on Advances in Pattern Recognition, ICAPR 2015, Kolkata, India, January 4-7, 2015, 2015.

[41] G. Nápoles, R. Falcón, Z. Dikopoulou, E. Papageorgiou, R. Bello, K. Vanhoof. Weighted aggregation of partial rankings using Ant Colony Optimization, Neurocomputing 250 (2017) 109–120.

[42] L. Paquete, T. Stützle, Design and analysis of stochastic local search for the multiobjective traveling salesman problem, Computers & OR 36 (9) (2009) 2619–2631.

[43] M. G. C. Resende, Greedy randomized adaptive search procedures, in: Encyclopedia of Optimization, Second Edition, Springer, 2009, pp. 1460–1469.

[44] R. Ruiz, T. Stützle, A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem, European Journal of Operational Research 177 (3) (2007) 2033–2049.

[45] F. Schalekamp, A. van Zuylen, Rank aggregation: Together we're strong, in: ALENEX, 2009.

[46] B. Selman, C. P. Gomes, Hill-climbing search, in: Encyclopedia of Cognitive Science, John Wiley & Sons, Ltd, 2006, pp. 333–336.

[47] T. Stützle, Iterated local search for the quadratic assignment problem, European Journal of Operational Research 174 (3) (2006) 1519–1539.

[48] A. Ukkonen, Clustering algorithms for chains, Journal of Machine Learning Research 12 (2011) 1389–1423.

[49] A. Ukkonen, K. Puolamaki, A. Gionis, H. Mannila, A randomized approximation algorithm for computing bucket orders, Information Processing Letters 109 (7) (2009) 356 – 359.