ISSN 2007-9737

A New Method For Personnel Selection Based On Ranking Aggregation Using A Reinforcement Learning Approach

Yaima Filiberto[1], Rafael Bello[2], Ann Nowe[3]

[1] University of Camaguey, Camaguey, Cuba

[2] University of Las Villas, Villa Clara, Cuba

[3] Vrije Universiteit Brussel, Brussel, Belgium

yaima.filiberto@reduc.edu.cu, rbellop@uclv.edu.cu, ann.nowe@como.vub.ac.be

Abstract. In this paper propose a new approach to the problem of aggregating rankings for obtaining an overall ranking. This is also referred to as the aggregation ranking in the personnel selection problem. Our approach is based on a distance measure between the individual and the overall ranking, and looks for the solution that minimizes the disagreement between the input rankings and the resulting aggregation. The method uses a reinforcement learning approach to build the aggregation and its performance and comparison with other approaches shows promising results.

Keywords. Aggregating rankings, personnel selection, reinforcement learning.

1 Introduction

Selection of qualified human resources is a key success factor for an organization. The adequate personnel training has a huge effect on improving the employees' performance, which has a direct impact on the growth and competence of the whole organization, especially in large-size and multinational companies and organizations. The personal selection is one of the fundamental activities of the human resource management; it has as objective to select the most appropriate candidate for the organization. An important activity of organizations is to seek more powerful ways of ranking a set of employees or personnel who have been evaluated in terms of different competencies [1]. Personnel selection is the process of choosing among the candidates applying for a particular job in the company, those who have the qualifications required to perform the job in the best way [2]. The personnel selection problem involves many conflicting objectives; and therefore it is a complicated problem.

This process is defined as a comparison and decision making process. In this process the human experts have an active participation. Recently, authors in [3], proposed to consider this problem a multi-criteria decision making problem under uncertainty. Many conflicting criteria should be considered when comparing alternatives to choose from, therefore a Multi-Criteria Decision Making (MCDM), approach is used [4, 5, 6, 7, 8, 9, 10]. The application of ranking and choice processes to decision making is crucial in different human activities (engineering, economics, etc). The main goal of managers is to obtain a ranking of the set of candidates who have been evaluated according to different competence; therefore, the development of efficient and flexible information aggregation methods has become a main issue in personnel selection [11].

A ranking is an ordering of a set of elements (objects, alternatives, actions, or candidates in the personal selection), indicating some sort of preference relationship among them, from the best

to the worst, while these objects are evaluated from multiple points of view considered relevant for the problem. Every ranking can be viewed as being produced by applying an overall ordering criterion to a given set of objects. As different people tend might judge the criteria differently, they usually end up with different orderings.

Dealing with permutations/rankings is a research area, which has gained a great interest. Ranking is among the most frequent real-world decision problems, the reason is that ranking data is ubiquitous nowadays and we can find applications in many fields like preference lists, voting in elections, information retrieval, collaborative filtering, combinatorial optimization, computational biology, etc, [12]. Magazines regularly publish rankings of universities, colleges, study programs, hospitals, pension funds, or cities [13].

More formally the problem can be described as follows: Let n items labeled $1, 2, ..., n$ to be ranked; then, any permutation p of these items represents a ranking. Given n items, several rankings can be generated from the preferences of several decision makers (different perspectives), and frequently those rankings of alternatives must be aggregated in order to create an overall ranking or a consensus ranking; this is referred to as the aggregation ranking problem or rank aggregation problem. Rank aggregation is the problem of combining multiple rankings into a single, aggregate ranking; that is, given a set of M permutations of n elements, identify the permutation which best represents this set of rankings.

For instances, in politics, rankings focus on the comparison of economic, social, environmental, and governance performance of countries; in [14], authors address the issue of how to construct suitable aggregates of individual journal rankings, using an optimization-based consensus ranking approach. The personal selection problem is another area in which the rankings are relevant; different rankings of the candidate workers can be established taking into account different criteria and an interesting problem is to aggregate these rankings to support the decision maker.

In the aggregation ranking problem a way of measuring how different two rankings are is required, and distances are the conventional tool to do that. A common approach to this problem is to find a permutation that minimizes the sum of the distances to the voters rankings, where in principle any distance (-like), function on permutations can be used [15]. The most frequently used distance measures among rankings are the Spearman footrule distance and the Kendall τ distance [16]; the Spearman footrule distance between two given rankings is defined as the sum, over all the candidates i of the absolute differences between the ranks of i with respect to the two rankings; the Kendall distance between two rankings is given by the minimum number of pairwise adjacent transpositions needed to transform one ranking into another.

Computing the consensus ranking is equivalent to the rank aggregation problem. The problem of computing the consensus ranking is nowadays an active field of research. An aggregation for a set of rankings $R_1, ..., R_k$ is a ranking τ such that the distance $D(\sigma, R_1, ..., R_k)$, is minimal, where D is a distance function for rankings and σ the consensus ranking, see equation 5) below. The known Kemeny ranking problem [17], consists of finding the ranking that minimizes the total number of disagreements with the set of rankings. Finding the Kemeny ranking is an NP-hard problem for $N > 4$ (N is the number of rankings) [18]. Several proposals for its exact computation have been presented as well as algorithms of heuristic nature [12]. According to Ali and Meila [19], there are formulations of the problem that lead to exact algorithms (without polynomial running time guarantees), and there are also a large number of heuristic and approximate algorithms; in their study, they compare a wide variety of algorithms including exact, branch and bound, specific heuristic, and voting algorithms, as well as approximate algorithms. The selection of the most appropriate aggregation method for a new application in decision-making is a difficult task [20].

A new method to rank aggregation is proposed in this paper; it is based on a Reinforcement Learning (RL) approach.

RL is a powerful technique to learn to take optimal decisions by trying out actions and evaluating the effect. Gradually the performance is

increasing based of the feedback that is received, see [21], for an introduction to the field and [22], or an overview of recent advances. In the setting considered here, the feedback is the quality of the resulting aggregated ranking.

An important aspect of the learning process is the balancing of the exploration versus exploitation. Exploration refers to trying out new actions or decisions, exploitation refers to using the knowledge already acquired so far. In this paper , we will show that RL can lead to an efficient exploration of the search space. To the best of our knowledge our approach is novel to the ranking problem. [1]. In the next section some related work to the rank aggregation problem is discussed.

The method proposed in this paper is presented in section 3. After that, an experimental study about the performance of this method is reported in section 4.

2 Related Work

Among the most commonly applied methods for this purpose are those based on distance measures between individual and collective preferences, and which look for the solution that minimizes the disagreement across decision makers [24]. The Kemeny ranking problem is the problem of finding the ranking defined by equation (1); it is the ranking that minimizes the total number of disagreements with the input rankings:

$$\pi_0 = \arg\min \frac{1}{N} \sum_{i=1}^{N} d(\pi_i, \pi_0). \tag{1}$$

The distance d represents the Kendall distance between two permutations defined as the minimum number of adjacent transpositions needed to turn one into other.

Below we summarize some algorithms that have been proposed to solve this NP-hard minimization problem given in 1.

Bargagliotti [25], presents a study about the aggregation of ranked data; she analyzes some characteristics and difficulties of this problem and the relation between the overall ranking and the input rankings. There are different methods for extracting overall rankings into specific applications but also, there are some more general methods to solve this problem.

In [12], authors propose to use genetic algorithm to tackle the rank aggregation problem. According to the authors these algorithms perform especially well when they face complex instances (those with large dimension and small degree of consensus). The study shows that the GA always obtains the best result (especially when the number of rankings increases), but however with respect to computational time, the GA algorithms are slower than other approaches.

In [15], a new approach to the problem of aggregating preferences of multiple agents based on the notion of popular ranking is introduced: a ranking of a set of elements is popular if there is no other permutation of the elements that a majority of the voters prefer. They analyzed the computational complexity and proved it is NP-hard to find a ranking with a majority of preferences.

The problem of aggregating preference rankings is analyzed in [26], where the authors propose a method based on Ordered Weighted Averaging (OWA) [27], operators. Each candidate may receive some votes in different ranking places; the total score of each candidate is the weighted sum of the votes she/he receives in different ranking places. Usually, the quantity the votes depends on the places (for instance in the Borda-Kendall method [28]); a key issue of the preference aggregation is how to determine the weights associated with different ranking places. In [26], OWA is used to determine the weights associated with different ranking places; OWA operators are also used in [11], moreover, the authors introduce a parametric aggregation model based on the fuzzy weighted.

The problem of preference ranking in the case of partial and/or incomplete preference data at multiple times is studied in [29]; an algorithm is developed to determine the maximum consensus sequences from the users' partial ranking data.

An other alternative is presented in [30], a semi-supervised ranking aggregation method is proposed, whose preference constraints of several

[1] In [23], RL is used to formulate web pages ranking algorithms, but this problem is not related with the rank aggregation problem studied here

Table 1. Equations (2) and (3)

$$p_m(t+1) = p_m(t) + \alpha_{reward}(1-\beta(t))(1-p_m(t)) - \alpha_{penalty}\beta(t)\,p_m(t), \quad (2)$$

$$if\ a_m\ is\ the\ action\ taken\ at\ time\ t.$$

$$p_j = p_j(t+1) - \alpha_{reward}(1-\beta(t))\,p_j(t) + \alpha_{penalty}\beta(t)\left[(r-1)^{-1} - p_j(t)\right], \quad (3)$$

$$if\ a_m \neq a_j.$$

item pairs are given and the aggregation function is learned based on the ordering agreement of different rankers; in these methods a weight vector is used.

In the next section a new method is proposed, the purpose is to introduce a general approach, designed for a broad variety of applications, and taking into account a minimum number of parameters (usually the estimation/learning of the model parameters can be done), which simplifies the process.

3 Learning Automata

Learning Automata (LA) [31, 32] are simple reinforcement learning components for adaptive decision making in unknown environments. An LA operates in a feedback loop with its environment and receives feedback (reward or punishment), for the actions taken. A single learning automaton maintains a probability vector p over its actions, which it updates according to a reinforcement scheme. Examples of linear reinforcement schemes are linear reward-penalty, linear reward-inaction and linear reward-penalty. The general update scheme is given in Table 1.

In this table, $p_i(t)$ is the probability of selecting action i at time step t. The constant α_{reward} and $\alpha_{penalty}$ are the reward and penalty parameters. When $\alpha_{reward} = \alpha_{penalty}$, the algorithm is referred to as linear reward-penalty (L_{R-P}), when $\alpha_{penalty}$ = 0, it is referred to as linear reward-inaction (L_{R-I}) and when $\alpha_{penalty}$ is small compared to α_{reward}, it is called linear $reward - \varepsilon - penalty$ ($L_{R\varepsilon-P}$). $\beta(t)$ is the reward received by the reinforcement signal for an action taken at time step t. r is the number of actions [33].

4 Our Approach to Rank Aggregation for the Personnel Selection Problem

In our setting we consider a set of n candidates $C = c_1, c_2, ..., c_n$, which have been ranked by m experts resulting in m rankings $R = R_1, R_2, ..., R_m$.

The decision maker needs to aggregate the rankings in R to make an overall decision. The purpose is to determine the consensus of the rankings in R, which minimizes the total dissimilarity between itself and the original rankings, a distance between a pair of rankings is defined, and finally, the ranking, that minimizes the sum of the distances from this ranking to all rankings in R, is returned.

Two questions need to be addressed in the approach we presented above: How to define the distance between rankings? What algorithm should be used to compute efficiently the aggregation? In this paper the Spearman footrule distance is used, given by expression 4. The Spearman footrule distance between two given rankings σ and τ defined over a finite set of candidate U is defined by expression 4:

$$F(\sigma,\tau) = \sum_{i\in U} |\sigma(i) - \tau(i)|, \quad (4)$$

where $\sigma(i)$ represents the position (or rank) of i in σ. The time complexity to compute the distance between two rankings using the Spearman footrule distance is linear [16].

This distance is extended to measure distances between one ranking and a set of rankings. Given a ranking σ and the set $R = R_1, ..., R_m$ one can

define the Spearman footrule distance of σ to *R* as equation (5):

$$D(\sigma, R) = \sum_{\tau \in R} F(\sigma, \tau). \quad (5)$$

According to [16], a strong connection between the Kemeni optimal aggregation and the aggregation based on the Spearman footrule (called footrule aggregation exists; this result is interesting if we note that the footrule aggregation can be computed in polynomial time, consequently, this can avoid the difficult task of computing the Kemeni optimal aggregation, by approximating this aggregation with the footrule aggregation. This proves the usefulness of the Spearman footrule distance in the case of the full rankings aggregation problem. This distance was used in [20] to compare the results of different methods.

We propose an algorithm based on the reinforcement learning to minimise expression (5) efficiently. The proposed method uses learning automata for learning the overall ranking. This idea is inspired by the method proposed in [33] for permutation learning. The advantages of this approach are that it does not use any problem specific information, does not rely on domain knowledge and only very few parameters are involved.

Our approach assumes a stochastic matrix M with n (number of candidates) rows and n columns is used; where M_{ij} expresses the probability for the candidate i is to be on the position j, which corresponds to the probability of taking an action in the ranking, each column and row should therefore sum up to 1. Using the information represented in M a ranking is built (see generating a permutation), the quality of this ranking is calculated using expression 5, after that, the matrix M is updated according to linear update scheme, (see expressions 2 and 3 in Table 1). The process is repeated until some stopping condition is met.

Generating a permutation from M: Uniformly select a row i and then use a roulette wheel selection on this row for determining on which position, i.e. j, we have to put element i in the permutation. After this, we reduce the matrix by removing row i and column j. Then we re-normalize the remaining rows, and repeat the process until all rows (and also all columns) have been selected once. Which results in a complete permutation.

Retrieve a reward: Using 5 as a reward function.

Update M using reward r: The probability matrix M is updated with an LA update scheme for each row i, and the selected action is determined by j; equations 2 and 3 are used. After updating all rows the matrix M remains doubly stochastic. To update the matrix M, we use the linear reward-inaction update scheme, because it has nice theoretical convergence results [33].

5 Experimental results

First, the performance of the method is reported; following, we show how the proposed consensus ranking methodology performs against other methods of aggregating individual rankings is studied, based on an evolutionary method. In Table 1 are the criteria and parameters used in the experimentation.

In Table 2 and Fig. 1 we show the results of the average minimum distance found in 30 runs, of the proposed algorithm for100, 1000, 1000 and 50000 iterations and the number of candidates ranging from 3 to 10 candidates. The results show that our approach always finds the optimal result for small instances, i.e. NC $<=$ 6 in at most 1000 iterations. In case of larger instances, we find good solutions in 1000 iterations, but increasing the number of iterations allows to improve the quality of the overall ranking that is found by our algorithm.

Table 3 shows the number of iterations needed for the algorithm to find the minimum distance value averaged over 30 runs, of the proposed algorithm for 100, 1000, 1000 and 50000 iterations and from 3 to10 candidates for a job; as it can be observed that for small instances up to 6 candidates the algorithm is able to find the optimal solution (error is zero in Table 2) in less than 500 iterations, for larger instances the algorithm also quickly finds quite good solutions. For example in the case of 10 candidates, the best solution found after max. 1000 iterations has an error 0.11.

PersSelecRL
1. Generate a uniform doubly stochastic matrix M with: $\forall\, i = 1...n, \forall\, j = 1...n,\, M_{ij} = 1/n$
2. Generate a permutation π using M
3. Retrieve a reward $r = D(\pi)$ for the selected permutation
4. Update M using reward $r = \frac{D(\sigma,T)}{n^2-(D(\sigma,T)-1)}$
5. Repeat from step 2 until stopping condition.

Algorithm 1: PersSelecRL

Table 2. Parameters used in the experiments, including problem size

$Criteria$	$Parameters$
Number of candidates for a one job (NC)	3, 4, 5, 6, 7, 8, 9, 10
Number of iteration (NI)	1000, 5000, 10000, 50000
Number of run (NR)	10
Number of expert (NE)	3
Stop condition (SC)	reach NI or 5 iteration without perform
α_{reward}	0.75

Table 3. Average of the minimum distance (Dpi) found by the algorithm for the number of iterations NI ranging from 1000 up to 50000 and 3 to 10 candidates for one job

NC/NI	1000	5000	10000	50000
3	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00
7	0.04	0.02	0.01	0.00
8	0.08	0.04	0.04	0.01
9	0.11	0.06	0.06	0.04
10	0.11	0.08	0.07	0.05

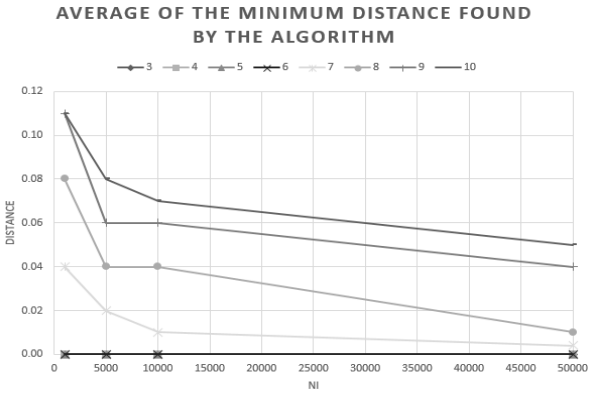

Fig. 1. Average of the minimum distance found by the algorithm for the number of iterations ranging from 1000 up to 50000 and 3 to 10 candidates for one job

This error quickly drops to 0.08 after max. 5000 iterations.

In Figure 2 the number of permutations are displayed grouped in distance intervals for all possible permutations and the permutations generated by the algorithm respectively, for 10 job candidates. Figure 2 gives an idea of the hardness of the problem, only 41 permutations belong to the first interval corresponding to a distance between 0 and 0.9. Figure 3, shows that the proposed algorithm tends to generate permutations nearby optimal value zero.

Figure 2 illustrates the hardness of the problem as there are only very few permutations in the best bin and Figure 3 gives some insight in how algorithm explores the search space.

Below we compare our algorithm to the well-known Borda-Kendall (BK) method proves to be a typical application of OWA operator weights in preference aggregation, that is, the BK method corresponds to a special case of the OWA operator weight method [26]. In the study presented

Table 4. Shows the average iteration number where the algorithm finds the minimum distance value, for experiments with the number of iteration ranging from 1000 to 50000 and number of candidates for one job from 3 to 10

NC/NI	1000	5000	10000	50000
3	2.60	2.10	2.50	2.50
4	14.90	8.80	9.60	7.90
5	51.60	53.50	49.20	74.20
6	254.00	374.20	356.50	210.30
7	376.70	803.40	1210.00	2790.40
8	365.10	1819.00	2592.70	10376.00
9	367.60	1954.00	2490.20	19687.4
10	368.90	2172.90	3567.00	18625.90

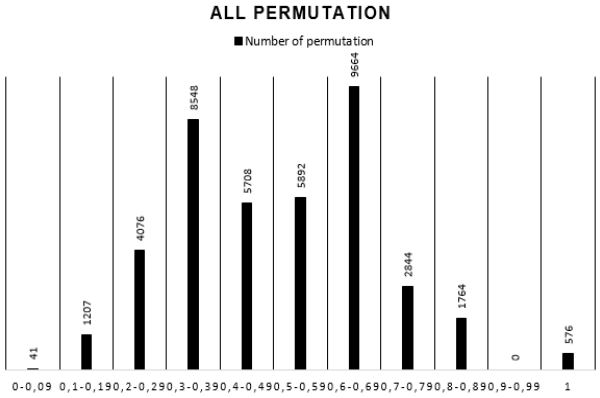

Fig. 2. Shows all possible permutations for the case of 8 candidates and their corresponding evaluations, grouped in bins with size of 0.1

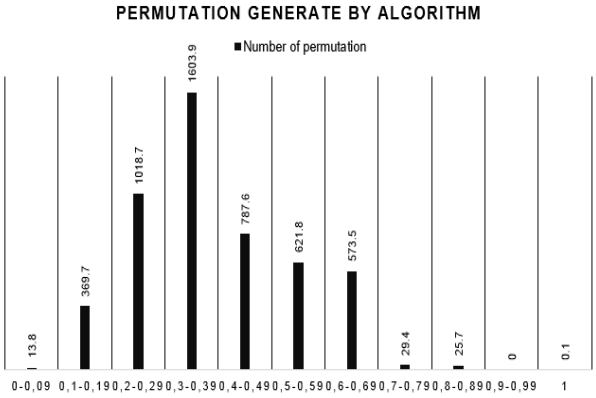

Fig. 3. Shows all generated permutations by our algorithm for the case of 8 candidates and their corresponding evaluations, grouped in bins with size of 0.1

in [20] the BK methods obtained similar results as other more complex methods, and according to the expert criteria this method approved as second-best due to its simplicity. With its simple calculations.

5.1 Borda-Kendall Method

The Borda-Kendall method is the most widely used technique for rank aggregation. For m candidates, the BK method assigns a weight m to the first ranking place, m-1 to the second place until the weight of one is assigned to the last ranking place.

The final rankings are determined by a weighted sum, where the alternative with the highest sum is most preferred followed by the other alternatives in descending sum order. Since this method determines weights to be used in a weighted sum, it is called a weight-determining method. Several weight-determining methods have been developed from this one.

In this case are m voters who vote on n candidates. Each candidate will receive some votes at different ranking places. The BK method assigns the first ranking place a mark of one, the second ranking place a mark of two, and so on. The total score (Zi) each candidate receives can be computed by aggregating the results from the simple equation 6:

$$\sum_{j=1}^{n} j v_{ij}. \tag{6}$$

Table 5. Probability of finding the best permutation by eight candidates

NI	Probability permutation random	Probability permutation GA	Probability permutation proposed-algorithm
1000	0.63	0.67	0.90
700	0.50	0.58	0.81
600	0.45	0.55	0.75
500	0.39	0.46	0.72
400	0.32	0.41	0.60
300	0.26	0.33	0.48
200	0.18	0.26	0.39
100	0.10	0.20	0.31

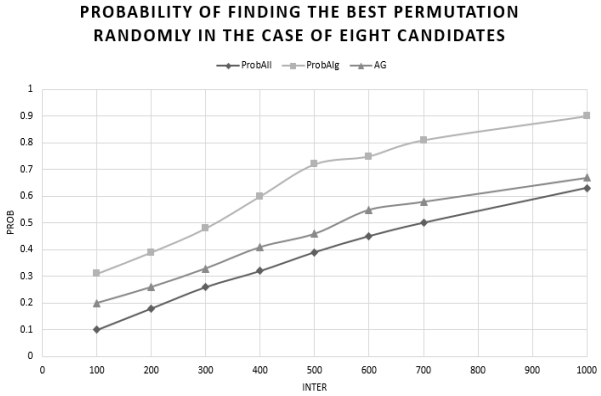

Fig. 4. Probability of finding the best permutation randomly in the case of eight candidates

The votes that each candidate receives j-th ranking place. The best candidate will be the one with the least total score.

Table 4 and figure 4 show the probability of finding permutations in the range from 0 to 0.09 for a D number of iterations between 1000 and 100 as seen in column 1 of the table; the results of the proposed algorithm are compared (column 3) to all possible permutations probability (column 2) and the method based on the Genetic Algorithm (GA) proposed in [12] (column 4), which was selected due to the good results shown in the paper, compared to the other existing methods.

Table 4 shows the probability of finding the best permutation by eight candidates.

Fig. 4 presents probability of finding the best permutation randomly in the case of eight candidates. The table shows that our algorithm is doing significantly better than both random search and a genetic algorithm approach [12]. Actually the approach proposed by [12] is similar to random search in terms of results.

6 Conclusion

The personnel selection problem is a typical problem in decision making. The problem concerns a set of candidates that are evaluated from the perspective of several criteria, to be ordered in to a single ranking. A common problem in personnel selection is to add these rankings to get a definite order to assist the decision maker in the selection.

The method proposed in this paper for the aggregation is based on the reinforcement learning approach. The goal is to find a ranking which is as much as possible in line to the individual rankings.

Our study shows that we can efficiently find the best solution, or near best solution for large instances, compared to other methods. Another advantage of our approach is that it requires only few parameters, so it is easy to use by the decision maker, moreover the implementation is simple.

References

1. **Gungor, Z., Serhadhoglu, G., & Kesen, S. E.(2009)**. A fuzzy ahp approach to personnel selection problem. *Applied Soft Computing*, Vol. 9, pp. 641–646.

2. **Akhlaghi, E. (2011)**. A rough-set based approach to design an expert system for personnel selection. *World Academy of Science, Engineering and Technology*, Vol. 78, pp. 245–248.
3. **Kelemenis, A. & Askounis, D. (2010)**. A new topsis-based multi-criteria approach to personnel selection. *Expert Systems with Applications*, Vol. 37, pp. 4999–5008.
4. **Afshari, A., Mojahed, M., Yusuff, R., Hong, T., & Ismail, M. (2010)**. Personnel selection using electre. *Journal of Applied Sciences*, Vol. 10, No. 23, pp. 3068–3075.
5. **Balezentis, A., Balezentis, T., & Brauers, W.** Personnel selection based on computing with words and fuzzy multimoora. *Expert Systems with Applications*, Vol. 39, No. 9, pp. 7961–7967.
6. **Dursun, M. & Karsak, E. E. (2010)**. A fuzzy mcdm approach for personnel selection. *Expert Systems with Applications*, Vol. 37, pp. 4324–4330.
7. **Kabak, M., Burmaoglu, S. & Kazancoglu, Y. (2012)**. A fuzzy hybrid mcdm approach for professional selection. *Expert Systems with Applications*, Vol. 39, No.3, pp. 3516–3525.
8. **Perez, L., Martinez, E., & Martinez, J. (2012)**. A new fuzzy topsis approach to personnel selection with veto threshold and majority voting rule. *11th Mexican International Conference on Artificial Intelligence, Advances in Artificial Intelligence and Applications*, (MICAI), No. 6389593, pp. 105–110.
9. **El-Santawy, M. & Zean, R. (2012)**. El-Dean, On using vikor for ranking personnel problem. *Life Science Journal*, Vol. 9, No. 4, pp. 1534–1536.
10. **El-Santawy, M. & Ahmed, A. (2012)**. Personnel training selection problem based on sdv-moora. *Life Science Journal*, Vol. 9, SUPPL. 2, pp. 152–154.
11. **Canos, L. & Liern, V. (2008)**. Soft computing-based aggregation methods for human resource management. *European Journal of Operational Research*, Vol. 189, pp. 669–681.
12. **Aledo, J. A., Gamez, J. A., & Molina, D. (2013)**. Tackling the rank aggregation problem with evolutionary algorithms. *Applied Mathematics and Computation* No. 222, pp. 632–644.
13. **Kadzinski, M., Greco, S., & Slowinski, R. (2012)**. Extreme ranking analysis in robust ordinal regression. *Omega*, No. 40, pp. 488–501.
14. **Theussl, S., Reutterer, T. & Hornik, K. (2014)**. How to derive consensus among various marketing journal rankings?. *Journal of Business Research*, Vol. 67, pp. 998–1006.
15. **Zuylen, A. van., Schalekamp, F., & Williamson, D. P. (2014)**. Popular ranking. *Discrete Applied Mathematics*, Vol. 165, pp. 312–316.
16. **Dinu, L. P. & Manea, F. (2006)**. An efficient approach for the rank aggregation problem. *Theoretical Computer Science*, Vol. 359, pp. 455–461.
17. **Kemeny, J. & Snell, J. (1962)**. Mathematical models in the social sciences. *Blaisdell*.
18. **Dwork, C., Kumar, R., Naor, M., & Sivakumar, D. (2001)**. Rank aggregation methods for the web. *Proceedings of the 10th international conference on World Wide Web*, ACM, pp. 613–622.
19. **Ali, A. & Meila, M. (2012)**. Experiments with Kemeny ranking: what works when? *Mathematical Social Sciences*, Vol. 64, No. 1, pp. 28–40.
20. **Fields, E. B., Okudan, G. E., & Rank, A. O. M. (2013)**. Aggregation methods comparison: A case for triage prioritization. *Expert Systems with Applications*, Vol. 40, pp. 1305–1311.
21. **Sutton, R. & Barto, A. (1998)**. *Reinforcement Learning: An Introduction*. MIT Press.
22. **Wiering, M. & van Otterlo, M. (2012)**. *Reinforcement Learning: State of the Art*. Springer Verlag.
23. **Derhami, V., Khodadadian, E., Ghasemzadeh, M., Mohammad, A., & Bidoki, Z. (2013)**. Applying reinforcement learning for web pages

ranking algorithms. *Applied Soft Computing*, Vol. 13, pp. 1686–1692.

24. **Contreras, I. (2011)**. Emphasizing the rank positions in a distance-based aggregation procedure. *Decision Support Systems*, Vol. 51, pp. 240–245.
25. **Bargagliotti, A. E. (2009)**. Aggregation and decision making using ranked data. *Mathematical Social Sciences*, Vol. 58, pp. 354–366.
26. **Wang, Y. M., Luo, Y., & Hua, Z. (2007)**. Aggregating preference rankings using owa operator weights. *Information Sciences*, Vol. 177, pp. 3356–3363.
27. **Yager, R. (1988)**. On ordered weighted averaging aggregation operators in multicriteria decision making. *IEEE Transactions on Systems Man and Cybernetics*, Vol. 18, pp. 183–190.
28. **Cook, W. & Kress, M. (1990)**. A data envelopment model for aggregating preference rankings. *Management Science*, Vol. 36, pp. 1302–1310.
29. **Chen, Y. L. & Cheng, L. C. (2010)**. An approach to group ranking decisions in a dynamic environment. *Decision Support Systems*, Vol. 48, pp. 622–634.
30. **Chen, S., Wang, F., Song, Y. & Zhang, C. (2011)**. Semi-supervised ranking aggregation. *Information Processing and Management*, Vol. 47, pp. 415–425.
31. **Narendra, K. & Thathachar, M. (1989)**. *Learning Automata: An Introduction*. Prentice-Hall International, Inc.
32. **Thathachar, M. & Sastry, P. (2004)**. Networks of Learning Automata: Techniques for Online Stochastic Optimization. *Kluwer Academic Publishers*.
33. **Wauters, T., Verbeeck, P., De Causmaecker, K., & Vanden Berghe, G. (2012)**. Fast permutation learning. *Lectures Notes on Computer Science*, Vol. 7219, pp. 292–306.

Article received on 23/02/2016; accepted on 07/02/2017.
Corresponding author is Yaima Filiberto.