Analysing the performance of migrating birds optimisation approaches for large scale continuous problems

Eduardo Lalla-Ruiz[1], Eduardo Segredo[2], Stefan Voß[1], Emma Hart[2], and Ben Paechter[2]

[1] Institute of Information Systems
University of Hamburg
Hamburg, Germany
`{eduardo.lalla-ruiz,stefan.voss}@uni-hamburg.de`
[2] School of Computing
Edinburgh Napier University
Edinburgh, Scotland, UK
`{e.segredo,e.hart,b.paechter}@napier.ac.uk`

Abstract. We present novel algorithmic schemes for dealing with large scale continuous problems. They are based on the recently proposed population-based meta-heuristics *Migrating Birds Optimisation* (MBO) and *Multi-leader Migrating Birds Optimisation* (MMBO), that have shown to be effective for solving combinatorial problems. The main objective of the current paper is twofold. First, we introduce a novel neighbour generating operator based on Differential Evolution (DE) that allows to produce new individuals in the continuous decision space starting from those belonging to the current population. Second, we evaluate the performance of MBO and MMBO by incorporating our novel operator to them. Hence, MBO and MMBO are enabled for solving continuous problems. Comparisons are carried out by applying both aforementioned schemes to a set of well-known large scale functions.

Keywords: continuous neighbourhood search; migrating birds optimisation; large scale continuous problems; global optimisation

1 Introduction

Nature-inspired computing counts with an extensive variety of algorithms mimicking natural processes and events from the universe that are frequently used for tackling real-world optimisation problems. Along these algorithms, those inspired by the collective living and travelling of animals have attracted a considerable interest from the related research community [12]. In this regard, the collective behaviour and swarm intelligence of migratory birds and its algorithmic translation have been recently studied by Duman et al. [2], and Lalla-Ruiz et al. [4]. Authors exploit, by means of their corresponding proposed algorithmic approaches, the advantage of sharing information and cooperating among a group of individuals. While *Migrating Birds Optimisation* (MBO), which is inspired by the V-flight formation of migratory birds with one leader, was proposed in [2], in [4], based on field studies, *Multi-leader Migrating Birds Optimisation* (MMBO) was introduced, which allows different types of flight formation shapes, as well as several leading individuals, to be managed.

Recently, MBO has shown its good performance for combinatorial problems, such as the *Quadratic Assignment Problem* (QAP) [2], the *Dynamic Berth Allocation Problem* (DBAP) [5], and *Hybrid Flow-shop Scheduling* [8], among others. In regard to continuous optimisation, an initial adaptation to low-dimensional continuous problems was developed in [1]. Results provided by said scheme, however, showed a poor performance with respect to other approaches [11]. Regarding MMBO, it showed to provide better quality results than those achieved by MBO for the QAP [4]. Concerning its performance for continuous optimisation, as far as we know, this is the first time that MMBO is enabled for dealing with these types of problems, as well as the first time that MBO is assessed when solving large scale continuous problems.

The main goal of this work is to propose suitable adaptations of MBO and MMBO for tackling continuous optimisation problems. In addition to these adaptations, we propose a novel neighbourhood structure based on the well-known *Differential Evolution* (DE) [10] with the aim of enabling them for operating and, consequently, generating solutions in a continuous decision space. The computational experimentation provided in this work, which involves the use of a set of well-known large scale continuous problems [7], indicates that our proposals are able to improve, for some cases, the results obtained by one of the best-performing variants of DE considering that set of large scale functions [3].

2 Schemes based on migrating birds optimisation for continuous problems

This section focuses on describing our algorithmic proposals. Section 2.1 is devoted to describe the scheme MBO, while the approach MMBO is depicted in Section 2.2. Finally, in Section 2.3 we introduce our novel neighbour generating operator based on DE.

2.1 Migrating birds optimisation

Migrating Birds Optimisation (MBO) is a population-based algorithm based on the V-formation flight of migrating birds. It considers a population or flock, of individuals or birds, that are aligned in a V-flight formation. Following that formation, the first individual corresponds to the leader of the flock and the other ones define the rest of the flock. The birds maintain a cooperative relationship among them by means of sharing information. The way the flow of information is shared is unidirectional. Namely, one individual sends information and the other receives it. The direction of the information shared starts from the leader bird and goes to the rest of the flock by following the V-shape flight formation.

Algorithm 1: Migrating Birds Optimisation pseudocode ([2])

```
Require: $n$, $K$, $m$, $k$, and $x$
 1: Generate $n$ initial solutions in a random manner and place them on an hypothetical
    V-formation arbitrarily
 2: $g = 0$
 3: while ($g < K$) do
 4:   for ($j = 1 : m$) do
 5:     Improve the solution of the leader bird by generating $k$ neighbours
 6:     $g = g + k$
 7:     for all (non-leader bird $s$ in the flock) do
 8:       Improve the solution of the non-leader bird by using $k - x$ generated neighbours
          and $x$ unused best neighbours from those birds in the front of it
 9:       $g = g + (k - x)$
10:     end for
11:   end for
12:   Move the leader bird to the end and forward one of the birds following it to the leader
      position
13: end while
14: Return best solution in the flock
```

Algorithm 1 depicts the pseudocode of MBO. The input parameters are: (i) the number of birds in the flock (n), (ii) the maximum number of neighbour solutions generated by the birds (K), (iii) the number of iterations performed before changing the leader bird (m), (iv) the number of neighbours generated by each bird (k), and (v) the number of best discarded solutions to be shared among birds (x). The first step consists of generating n individuals or birds (line 1). The number of neighbour solutions generated by the population of birds, i.e. g, is initially set to zero (line 2). During the search process, firstly, the leader bird generates k neighbour solutions. In case the best solution generated leads to an improvement in terms of the objective function value, the leader bird replaces its solution by that neighbour solution (line 5). Secondly, each follower bird generates a number of $k - x$ neighbour solutions (lines 7–10) and receives the best x neighbour solutions from the bird in front of it. If one of the generated or received solutions leads to an improvement, then that follower bird is replaced by that bird that allows the maximum improvement (line 8). This V-formation is maintained until a prefixed number of iterations, $m > 0$, is reached. Once that, the leader bird becomes the last bird in the V-formation and one of its immediate followers becomes the new leader (line 12). Then, the search process is restarted until m iterations are reached again. MBO is executed until a maximum number of neighbour solutions, i.e. K, are generated (line 3). Finally, we should mention that, in our case, neighbours are created (lines 5 and 8) by using the operator described in Section 2.3.

2.2 Multi-leader migrating birds optimisation

Multi-leader Migrating Birds Optimisation (MMBO) is a novel population-based meta-heuristic inspired by the flight formation of migratory birds which tries to improve its predecessor MBO. In MMBO, birds are distributed in a line formation mimicking the flight formation of migratory birds, which is determined according to given *relationship criteria*, e.g. by means of the objective function

Algorithm 2: Multi-leader Migrating Birds Optimisation pseudocode ([4])

```
Require: $n$, $K$, $k$, and $x$
 1: Create the initial population $P$ by randomly generating $n$ individuals
 2: while ($K$ neighbours have not been generated) do
 3:     Determine the interaction among individuals of $P$ and establish the formation
 4:     while (stopping formation criterion is not met) do
 5:         Generate $k$ neighbour solutions for each individual included into $P_L \cup P_I$
 6:         Replace each individual included into $P_L$ for its best neighbour solution if
            it leads to an improvement
 7:         Replace each individual included into $P_I$ for its best neighbour solution
 8:         for all (individual $\in P_F$) do
 9:             Generate $k - x$ neighbour solutions
10:             Get best $x$ unused best solutions from the previous individual in the group
11:             Replace individual for its best found solution if leads to an improvement
12:         end for
13:     end while
14: end while
15: Return best solution found by some individual from $P$
```

of the problem at hand. Depending on those criteria, we can have birds located at positions that are closer than others regarding the front of the migratory formation during the flight. Each individual generates a given number of feasible solutions through a predefined neighbourhood structure. The obtained solutions reflect the particular point of view about the solution space of each individual. As mentioned above, during the search of MMBO, and depending on the relationship criteria and the share of information among individuals, different roles can be defined:

- *Leader.* It is that individual which has provided the best objective value when compared to the adjacent ones. Therefore, it does not receive information from any individual, but shares x solutions with each adjacent individual. The leader generates k neighbour solutions. Since the objective function value determines the position within the formation, the leader is the best-behaved individual, and consequently, the most advanced one in its group within the formation. The set of leader individuals is denoted as P_L.
- *Follower.* It is that individual which explores the search space considering its own information and the information received from the solutions in front of it within the formation. Since it receives information, it generates $k - x$ neighbour solutions and receives x solutions from its adjacent individual. The set of follower individuals is denoted as P_F.
- *Independent.* It is that individual which is not currently included into any group of the set of individuals. Thus, it does not exchange information with any other individual. It generates k neighbour solutions. The set of independent individuals is denoted as P_I.

The pseudocode of MMBO is depicted in Algorithm 2. The first step is to generate the set of individuals, i.e. P, which consists of n randomly generated individuals (line 1). As long as the stopping criterion is not met, the MMBO iterates (line 2). In this work, we consider a stopping criterion based upon a maximum number of neighbour solutions to be generated (K). The relationship criteria among individuals are based on the objective function value. This allows to recognise the groups, as well as the formation (line 3). Then, the search process starts (lines 4–13) and it is executed until a stopping formation criterion is met. In case that said criterion is satisfied, the search process is stopped in order to establish a new formation. During the search process, firstly, each individual $i \in P_L \cup P_I$ generates a population of k neighbour solutions (line 5). In case the best generated neighbour solution leads to an improvement in terms of the objective function value, the individual moves to that solution (lines 6–7). Secondly, each follower individual $i \in P_F$, generates $k - x$ neighbour solutions (line 9) and receives x solutions from its adjacent individuals according to the formation (line 10). Then, if some solution, either received or generated, leads the follower individual to an improvement, then it will be replaced for that solution (line 11). The remainder of the best discarded solutions will be shared with its adjacent individual. As in the case of MBO, in MMBO neighbours are generated (lines 5 and 9) by applying the operator introduced in Section 2.3.

2.3 Neighbour generating operator based on differential evolution

In this work, we introduce a novel neighbour generating operator to be used with MBO and MMBO in order to enable their operation with continuous optimisation problems. This operator is based on the well-known *Differential Evolution* (DE), a search algorithm which was specifically proposed for global optimisation [10].

For encoding individuals, a vector of D real-valued decision variables or dimensions x_i is used, i.e. $\boldsymbol{X} = [x_1, x_2, \ldots, x_i, \ldots, x_D]$. The objective function $f(\boldsymbol{X})(f : \Omega \subseteq \mathbb{R}^D \to \mathbb{R})$ determines the quality of every vector $\boldsymbol{X}$. Hence, finding a vector $\boldsymbol{X}* \in \Omega$, where $f(\boldsymbol{X}*) \leq f(\boldsymbol{X})$ is satisfied for all $\boldsymbol{X} \in \Omega$, is the goal in a global optimisation problem. Considering box-constrained problems, the feasible region Ω is defined by $\Omega = \prod_{i=1}^{D}[a_i, b_i]$, where a_i and b_i represents the lower and upper bounds of variable i.

Regarding the most widely used nomenclature for DE [10], i.e. DE*/x/y/z*, where x is the vector to be mutated, y defines the number of difference vectors used, and z indicates the crossover approach, our neighbour generating operator is inspired by the scheme DE/rand/1/bin. We selected this variant due to its simplicity and popularity and because it was able to provide the best performance in previous work with the set of large scale problems we consider herein [3].

Given a particular individual $\boldsymbol{X}_{j=1 \ldots NP}$ (*target vector*) from a population of either MBO or MMBO with size NP, a neighbour individual is obtained as follows. First, the *mutant generation strategy* rand/1 is applied for obtaining a *mutant vector* ($\boldsymbol{V}_j$). Any vector in the population different from the target vector is randomly selected as the *base vector*. Thus, the mutant vector is generated as Eq. 1 shows. It is worth mentioning that r_1, r_2, and r_3 are mutually exclusive integers randomly selected from the range $[1, NP]$, all of them different from the index j as well. Finally, F denotes the *mutation scale factor*.

$$\boldsymbol{V}_j = \boldsymbol{X}_{r_3} + F \times (\boldsymbol{X}_{r_1} - \boldsymbol{X}_{r_2}) \tag{1}$$

After obtaining the mutant vector, it is combined with the target vector to produce the *trial vector* ($\boldsymbol{U}_j$) through a crossover operator. The combination of the mutant vector generation strategy and the crossover operator is usually referred to as the *trial vector generation strategy*. One of the most commonly applied crossover operators, which is considered in this work, is the *binomial crossover* (*bin*). The crossover is controlled by means of the crossover rate CR, and uses Eq. 2 for producing a trial vector, where $x_{j,i}$ represents decision variable i belonging to individual $\boldsymbol{X}_j$. A random number uniformly distributed in the range $[0, 1]$ is given by $rand_{j,i}$, and $i_{rand} \in [1, 2, ..., D]$ is an index selected at random that ensures that at least one variable belonging to the mutant vector is inherited by the trial one. Variables are thus inherited from the mutant vector with probability CR. Otherwise, variables are inherited from the target vector.

$$u_{j,i} = \begin{cases} v_{j,i} \; if \; (rand_{j,i} \;\leq\; CR \; or \; i \;=\; i_{rand}) \\ x_{j,i} \; otherwise \end{cases} \tag{2}$$

It can be observed that the trial vector generation strategy may generate vectors outside the feasible region Ω. In this work, unfeasible values are reinitialised at random in their corresponding feasible ranges. Finally, we should note that the trial vector becomes the newly generated neighbour.

3 Experimental evaluation

In this section we describe the experiments carried out with both algorithms depicted in Section 2. In addition to those schemes, we also considered the variant DE/rand/1/bin as an independent approach for comparison purposes.

Experimental method MBO and MMBO, as well as DE/rand/1/bin, were implemented by using the *Meta-heuristic-based Extensible Tool for Cooperative Optimisation* (METCO) [6]. Experiments were run on a Debian GNU/Linux computer with four AMD® Opteron™ processors (model number 6164 HE) at 1.7 GHz and 64 GB RAM. Every execution was repeated 30 times, since all experiments used stochastic algorithms. Bearing the above in mind, comparisons were carried out by applying the following statistical analysis [9]. First, a *Shapiro-Wilk test* was performed to check whether the values of the results followed a normal (Gaussian) distribution or not. If so, the *Levene test* checked for the homogeneity of the variances. If the samples had equal variance, an ANOVA *test* was done. Otherwise, a *Welch test* was performed. For non-Gaussian distributions, the non-parametric *Kruskal-Wallis* test was used. For all tests, a significance level $\alpha = 0.05$ was considered.

Problem set A set of scalable continuous optimisation functions proposed in the *2013* IEEE *Congress on Evolutionary Computation* (CEC'13) for its LSGO competition [7] was considered as the problem set. We should note that this suite is the latest proposed for large scale global optimisation in the field of the CEC, and therefore, it was also used for the LSGO competitions organised in CEC'14 and

Table 1: Parameterisation of the approaches MBO, MMBO, and DE/rand/1/bin

Parameter values for MBO and MMBO			
Parameter	Value	Parameter	Value
Stopping criterion (K)	$3 \cdot 10^6$ evals.	Number of neighbors (k)	4
Flock size (n)	150	Number of flights (x)	1
Number of flights (m)	10		
Parameter values for DE/rand/1/bin			
Stopping criterion	$3 \cdot 10^6$ evals.	Mutation scale factor (F)	0.5
Population Size (NP)	150	Crossover rate (CR)	0.9

CEC'15. The suite consists of 15 different problems (f_1–f_{15}) with different features: fully-separable functions (f_1–f_3), partially additively separable functions (f_4–f_{11}), overlapping functions (f_{12}–f_{14}), and non-separable functions (f_{15}). By following the suggestions given for different editions of the LSGO competition, we fixed the number of decision variables D to 1000 for all the above functions, with the exception of functions f_{13} and f_{14}, where 905 decision variables were considered due to overlapping subcomponents.

Parameters Table 1 shows parameter values considered in this work for MBO and MMBO. They were selected by carrying out a previous parameter setting study. As it can be observed in Section 2, parameter m is only considered by MBO. In past research, a configuration of the scheme DE/rand/1/bin, from among a candidate pool with more than 80 different parameterisations of said approach, was able to provide the best overall results for problems f_1–f_{15} with 1000 decision variables [3]. This is the main reason why our neighbour generating operator is based on DE/rand/1/bin. Moreover, that best-performing configuration, whose parameter values (NP, F, and CR) are also shown in Table 1, is considered herein as an independent method for measuring the performance attained by MBO and MMBO. Our operator also makes use of those parameter values. Finally, the stopping criterion was fixed to a maximum amount of 3×10^6 evaluations, following the recommendations provided by the LSGO competition.

Figure 1 shows box-plots reflecting the results obtained by the considered schemes. It can be observed that, for some problems (f_2, f_3, f_5, and f_9) MBO or MMBO were able to clearly obtain better solutions than those provided by the best-performing variant of DE/rand/1/bin found for the large scale problems we consider in this work, thus showing the benefits that can be obtained from our hybridisation between MBO/MMBO and our novel neighbour generating operator based on DE. Since our neighbour generating operator is based on DE/rand/1/bin, it was expected that results obtained by MBO and MMBO were very similar to those provided by the former scheme executed independently. However, the features of MBO and MMBO for sharing information among individuals, as well as for establishing a structure among them, combined with the the exploration and exploitation abilities of our neighbour generating operator based on DE/rand/1/bin, were able to obtain even better results in 4 out of 15 problems. Taking into account the remaining functions, we should note that MBO

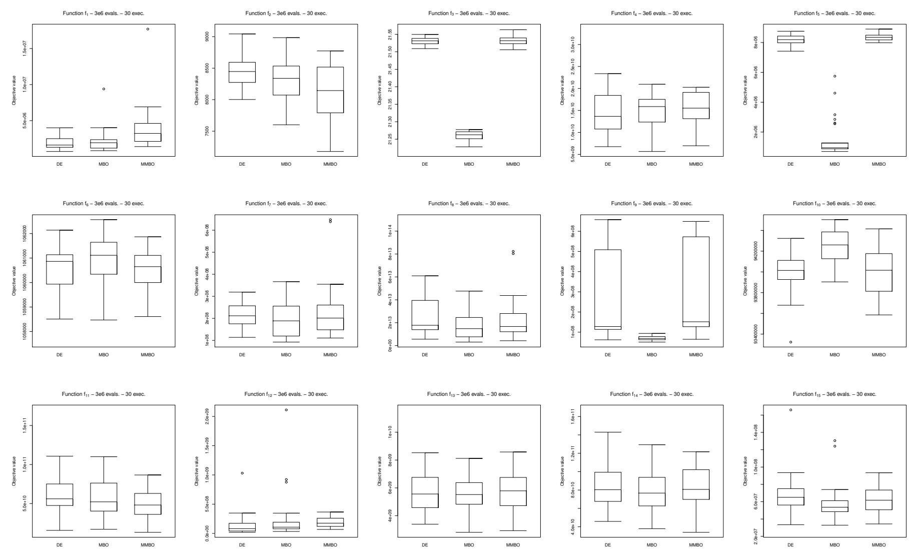

Fig. 1: Box-plots showing the results obtained by different schemes for functions f_1–f_{15}

and/or MMBO were able to achieve similar solutions than those attained by the best-behaved variant of DE.

In order to give the conclusions exposed above with statistical confidence, Table 2 shows, for each problem, the p-values obtained from the statistical comparison between the approach MBO and the rest of approaches, by following the statistical procedure explained at the beginning of the current section. It also shows cases for which MBO was able to statistically outperform other strategy (↑), cases where other strategy outperformed MBO (↓), and cases where statistically significant differences between MBO and the corresponding method did not arise (↔). Scheme A statistically outperforms method B if there exist statistically significant differences between them, i.e. if the p-value is lower than $\alpha = 0.05$, and if at the same time, A provides a lower mean and median of the objective value than B, since we are dealing with minimisation problems. In addition, for those cases where statistically significant differences between MBO and the corresponding scheme appeared, but one approach obtained the lowest mean, while the other one provided the lowest median, an '*' is shown. Finally, Table 3 shows the same information, but regarding MMBO.

With respect to MBO, it is worth mentioning that it was able to outperform DE in 4 out of 15 problems (f_3, f_5, f_9, and f_{15}). It was outperformed by DE, however, only in the case of function f_{11}. For remaining problems, MBO and DE did not present statistically significant differences. Concerning MMBO, we should note that it was able to beat DE in problems f_2 and f_{11}, it was beaten by DE considering functions f_1 and f_{12}, and both approaches did not show statistically significant differences when dealing with remaining test cases. Bearing the above

Table 2: Statistical comparison between MBO and remaining schemes considering problems f_1–f_{15}

f	Alg.	p-value	Dif.	f	Alg.	p-value	Dif.	f	Alg.	p-value	Dif.
f_1	DE	6.152e-01	$\leftrightarrow$	f_2	DE	1.172e-01	$\leftrightarrow$	f_3	DE	2.868e-11	$\uparrow$
	MMBO	3.093e-04	$\uparrow$		MMBO	8.525e-02	$\leftrightarrow$		MMBO	2.872e-11	$\uparrow$
f_4	DE	5.946e-01	$\leftrightarrow$	f_5	DE	2.872e-11	$\uparrow$	f_6	DE	8.366e-02	$\leftrightarrow$
	MMBO	8.592e-01	$\leftrightarrow$		MMBO	2.872e-11	$\uparrow$		MMBO	1.287e-01	$\leftrightarrow$
f_7	DE	2.612e-01	$\leftrightarrow$	f_8	DE	7.852e-02	$\leftrightarrow$	f_9	DE	5.225e-09	$\uparrow$
	MMBO	3.671e-01	$\leftrightarrow$		MMBO	2.254e-01	$\leftrightarrow$		MMBO	1.348e-09	$\uparrow$
f_{10}	DE	5.434e-05	$\downarrow$	f_{11}	DE	5.444e-01	$\leftrightarrow$	f_{12}	DE	7.852e-02	$\leftrightarrow$
	MMBO	5.715e-04	$\downarrow$		MMBO	1.515e-01	$\leftrightarrow$		MMBO	3.450e-02	*
f_{13}	DE	9.764e-01	$\leftrightarrow$	f_{14}	DE	2.871e-01	$\leftrightarrow$	f_{15}	DE	1.355e-02	$\uparrow$
	MMBO	9.307e-01	$\leftrightarrow$		MMBO	2.049e-01	$\leftrightarrow$		MMBO	1.039e-01	$\leftrightarrow$

Table 3: Statistical comparison between MMBO and remaining schemes considering problems f_1–f_{15}

f	Alg.	p-value	Dif.	f	Alg.	p-value	Dif.	f	Alg.	p-value	Dif.
f_1	DE	8.401e-05	$\downarrow$	f_2	DE	1.402e-03	$\uparrow$	f_3	DE	8.786e-01	$\leftrightarrow$
	MBO	3.093e-04	$\downarrow$		MBO	8.525e-02	$\leftrightarrow$		MBO	2.872e-11	$\downarrow$
f_4	DE	4.965e-01	$\leftrightarrow$	f_5	DE	6.671e-02	$\leftrightarrow$	f_6	DE	8.130e-01	$\leftrightarrow$
	MBO	8.592e-01	$\leftrightarrow$		MBO	2.872e-11	$\downarrow$		MBO	1.287e-01	$\leftrightarrow$
f_7	DE	5.742e-01	$\leftrightarrow$	f_8	DE	5.543e-01	$\leftrightarrow$	f_9	DE	2.739e-01	$\leftrightarrow$
	MBO	3.671e-01	$\leftrightarrow$		MBO	2.254e-01	$\leftrightarrow$		MBO	1.348e-09	$\downarrow$
f_{10}	DE	9.176e-01	$\leftrightarrow$	f_{11}	DE	2.713e-02	$\uparrow$	f_{12}	DE	2.962e-03	$\downarrow$
	MBO	5.715e-04	$\uparrow$		MBO	1.515e-01	$\leftrightarrow$		MBO	3.450e-02	*
f_{13}	DE	7.901e-01	$\leftrightarrow$	f_{14}	DE	9.176e-01	$\leftrightarrow$	f_{15}	DE	3.912e-01	$\leftrightarrow$
	MBO	9.307e-01	$\leftrightarrow$		MBO	2.049e-01	$\leftrightarrow$		MBO	1.039e-01	$\leftrightarrow$

in mind, it is important to remark that MBO/MMBO were able to provide better solutions than those achieved by DE in 6 out of 15 problems, thus improving the conclusions extracted by observing box-plots. However, DE was able to outperform MBO/MMBO in 3 out of 15 functions. This means that MBO/MMBO were able to attain similar or even better solutions than DE in 12 out of 15 problems.

If we compare MBO with respect to MMBO we can make the following observations. MBO provided statistically better results than MMBO in 4 problems ($f1$, $f3$, $f5$, and $f9$), while the latter was statistically better than the former only in the case of function f_{10}. Taking into account the remaining problems, statistically significant differences did not appear between both schemes.

4 Conclusions and future work

Algorithms inspired by the nature comprise an important type of solution approaches used for solving practical problems. Some of these approaches have been successfully applied to combinatorial optimisation, such as MBO and MMBO. Nevertheless, to our best knowledge, they have not been used for tackling large scale continuous problems. Hence, in this work we propose novel adaptations of both population-based meta-heuristics for solving relevant problems in this research area. For doing that, we developed a novel neighbour generating operator based on DE that allows new individuals to be generated in the continuous decision space, thus enabling MBO and MMBO for dealing with continuous problems.

The experimental evaluation carried out indicates that our proposals are suitable and competitive for performing the optimisation of large scale continuous problems. In this regard, the obtained results demonstrate that MBO and MMBO are able to obtain similar solutions, and even better for some cases, than those provided by one of the best-performing variants of DE considering the set of large scale continuous problems at hand.

Bearing in mind the contributions of this work, our research agenda will be focused on the assessment of the influence that the different parameters of MBO and MMBO have over their performance when solving continuous problems. Additionally, an analysis about the impact that different neighbourhood structures have over the behaviour of MBO and MMBO might also be of great interest.

References

1. Alkaya, A.F., Algin, R., Sahin, Y., Agaoglu, M., Aksakalli, V.: Performance of migrating birds optimization algorithm on continuous functions. In: Advances in Swarm Intelligence, pp. 452–459. Springer (2014)
2. Duman, E., Uysal, M., Alkaya, A.: Migrating birds optimization: A new metaheuristic approach and its performance on quadratic assignment problem. Information Sciences 217, 65–77 (2012)
3. Kazimipour, B., Li, X., Qin, A.: Effects of population initialization on differential evolution for large scale optimization. In: 2014 IEEE Congress on Evolutionary Computation (CEC). pp. 2404–2411 (July 2014)
4. Lalla-Ruiz, E., de Armas, J., Expósito-Izquierdo, C., Melián-Batista, B., Moreno-Vega, J.M.: Multi-leader migrating birds optimization: A novel nature-inspired metaheuristic for combinatorial problems. International Journal of Bio-Inspired Computation In press (2015)
5. Lalla-Ruiz, E., Expósito-Izquierdo, C., de Armas, J., Melián-Batista, B., Moreno-Vega, J.M., et al.: Migrating birds optimization for the seaside problems at maritime container terminals. Journal of Applied Mathematics 2015 (2015)
6. León, C., Miranda, G., Segura, C.: METCO: A Parallel Plugin-Based Framework for Multi-Objective Optimization. International Journal on Artificial Intelligence Tools 18(4), 569–588 (2009)
7. Li, X., Tang, K., Omidvar, M., Yang, Z., Qin, K.: Benchmark Functions for the CEC'2013 Special Session and Competition on Large Scale Global Optimization. Tech. rep., Evolutionary Computation and Machine Learning Group, RMIT University, Australia (2013)
8. Pan, Q.K., Dong, Y.: An improved migrating birds optimisation for a hybrid flowshop scheduling with total flowtime minimisation. Information Sciences 277, 643–655 (2014)
9. Segura, C., Coello, C.A.C., Segredo, E., Aguirre, A.H.: A Novel Diversity-Based Replacement Strategy for Evolutionary Algorithms. IEEE Transactions on Cybernetics In press, 1–14 (2015)
10. Storn, R., Price, K.: Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. J. of Global Optimization 11(4), 341–359 (Dec 1997)
11. Tan, Y., Li, J., Zheng, Z.: ICSI 2014 Competition on Single Objective Optimization (ICSI-2014-BS). arXiv preprint arXiv:1501.02128 (2015)
12. Yang, X.S.: Nature-inspired metaheuristic algorithms. Luniver press (2010)