Information Sciences 217 (2012) 65–77

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Migrating Birds Optimization: A new metaheuristic approach and its performance on quadratic assignment problem

Ekrem Duman [a,*], Mitat Uysal [b], Ali Fuat Alkaya [c]

[a] Ozyegin University, Department of Industrial Engineering, Alemdag, Cekmekoy, Istanbul, Turkey
[b] Dogus University, Department of Computer Engineering, Acibadem, Kadikoy, Istanbul, Turkey
[c] Marmara University, Department of Computer Engineering, Goztepe, Kadikoy, Istanbul, Turkey

ARTICLE INFO

Article history:
Received 23 December 2010
Received in revised form 9 June 2012
Accepted 23 June 2012
Available online 2 July 2012

Keywords:
Metaheuristics
Optimization
Birds' migration
V-shape topology
Benefit mechanism

ABSTRACT

We propose a new nature inspired metaheuristic approach based on the V flight formation of the migrating birds which is proven to be an effective formation in energy saving. Its performance is tested on quadratic assignment problem instances arising from a real life problem and very good results are obtained. The quality of the solutions we report are better than simulated annealing, tabu search, genetic algorithm, scatter search, particle swarm optimization, differential evolution and guided evolutionary simulated annealing approaches. The proposed method is also tested on a number of benchmark problems obtained from the QAPLIB and in most cases it was able to obtain the best known solutions. These results indicate that our new metaheuristic approach could be an important player in metaheuristic based optimization.

1. Introduction

Solving large scale combinatorial optimization problems optimally is often intractable and one usually has to be content with near optimal solutions. Near optimal solutions are found by heuristic algorithms which can broadly be classified as constructive and improvement algorithms. Constructive algorithms start from scratch and build a solution gradually whereas improvement algorithms start with a complete solution and try to improve it. Heuristic algorithms are usually developed to solve a specific problem in hand. There is also a class of heuristic algorithms that can be used to solve a large class of problems, either directly or with minor modifications, called metaheuristics [20].

Most metaheuristic algorithms can also be named as neighborhood (or, local) search procedures. These are a wide class of improvement algorithms where at each iteration an improving solution is found by searching the "neighborhood" of the current solution. A critical issue in the design of a neighborhood search algorithm is the choice of the neighborhood structure, that is, the manner in which the neighborhood is defined [2].

So far many metaheuristics have been proposed by researchers. Among these the genetic algorithm proposed by Holland [24], the simulated annealing proposed by Kirkpatrick et al. [28], the tabu search proposed by Glover [19], the ant colony optimization proposed by Dorigo [11] and the particle swarm optimization proposed by Eberhart and Kennedy [15] are the most popular ones. The harmony search algorithm [18], the artificial bee colony algorithm [27], the monkey search algorithm [36], the differential evolution [46] and the firefly algorithm [52] are examples of other competitive metaheuristics proposed recently. Most of these metaheuristics are inspired by nature. This is an indication that although we the mankind are the most intelligent creature in the world, we have lessons to learn from the perfectness of nature.

* Corresponding author. Tel.: +90 216 564 90 00; fax: +90 216 564 90 50.
E-mail address: ekrem.duman@ozyegin.edu.tr (E. Duman).

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2012.06.032

Metaheuristics have been successfully applied to many different areas and problems from manufacturing [26] to services [33], from scheduling [30] to transportation [6], from health [42] to sports [23], from justice [17] to entertainment [53], from data mining [34] to curve fitting [50], from robotics [8] to timetabling [1] and from geology [4] to astronomy [9]. It is possible to find thousands of similar studies in the literature. Here we can only name just a few of them.

The application domain we focus on is the quadratic assignment problem (QAP) due to our prior work and familiarity. The QAP is best described as the plant layout problem, where a number of departments are to be located on a number of locations so that the transportation cost among the departments is minimized. Typically, the number of departments and locations are taken as equal to each other [32]. The QAP is an NP-Hard problem and it is very difficult to solve it optimally once the number of departments exceeds 15 [14]. Parallel to its complexity the research work on the QAP is still diverging and many researchers still report their new research in leading journals. Some of the recent studies regarding QAP can be listed as follows.

Duman and Or [14] addressed the QAP in the context of specific assembly machines. Ramkumar et al. [41] proposed a new iterated fast local search heuristic for solving QAP in facility layout design. Drezner [12] and Misevicius and Rubliauskas [35] tried to solve QAP with hybrid genetic algorithms wheras, Ravindra et al. [43] proposed a greedy genetic algorithm for the QAP. There is also a recent survey of the metaheuristics applied to QAP [37]. A recent study considered effective formulation reductions for the QAP [54]. Another recent example is self controlling tabu search algorithm where the application is held on QAP instances [16]. Robust tabu search is another important contribution for solving the quadratic assignment problem with less complexity and fewer parameters and it is still being improved [40,47,48].

We name our new nature inspired metaheuristic algorithm as the Migrating Birds Optimization (MBO) algorithm since it is inspired from the V formation flight of the migrating birds which is a very effective formation in energy minimization [31]. To test the performance of the MBO algorithm the studies of Duman and Or [14] and Kiyicigi et al. [29] are taken as the benchmarks. In [14] a number of heuristic algorithms including 2-opt, 3-opt and their combinations and the metaheuristics tabu search, simulated annealing and guided evolutionary simulated annealing are implemented and compared for the solution of the quadratic assignment problem. Then in [29] this comparison is extended to cover genetic algorithms and scatter search also. In this study, the MBO algorithm is compared with the best performing procedure in those studies and two additional metaheuristics which are implemented now and better solutions are obtained in all test problems. Similar to other metaheuristics the MBO is also a parametric procedure and its performance may depend on how effectively its parameters are settled. In addition to this, the MBO is also applied to standard benchmark problems in QAPLIB and very successful results are obtained.

The outline of the study is as follows. In the next section we give some information on how birds fly and what benefits they can obtain in using the V flight formation. Based on the birds' story, the MBO algorithm is detailed in Section 3. The results obtained with the initial set of parameters are given in Section 4. Detailed parameter fine tuning experiments are discussed in Section 5 where the results obtained by the best set of parameters are also given. Section 6 discusses the results obtained on a number of standard QAP instances available in QAPLIB. Section 7 concludes by providing a summary of the study and directions for further study.

2. Migration of birds

The shape of a bird wing is called an airfoil. As the airfoil moves through the air, air goes above and below. The air flow over the upper surface has to move farther than the lower part of the wing. In order for the two air flows to make it to the edge of the wing at the same time, the top air must go faster. Consequently, the air on the upper part has a lower pressure than the air moving over the lower part (Fig. 1). This pressure difference makes the lifting possible by the wing.

For a lone bird, speed is the most important factor in achieving lift. Lift can be increased by increasing the forward speed of the wing through the air [31]. This is because as the speed is higher the pressure difference should be removed in a shorter time and thus a higher uplift pressure is achieved. The power needed to generate this lifting momentum is called induced power, which is distinguished from the profile power – the power needed to move the bird through the air against skin friction [31].

"The high pressure air under the wing flows around the tip and inward across the dorsal wing surface. This latter flow forms a sheet of turbulent air shedding from the trailing edge into the bird's wake. This planar vortex sheet rolls up into two concentrated tubular vortices, one issuing from each wing tip. The vortices, which are slightly inboard of the wing tips, produce large regions of upwash outboard of the wing and a region of downwash more centrally" (Fig. 2). The regions of

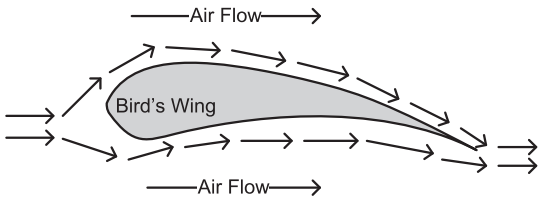

Fig. 1. Wing of a bird.

upwash may contribute to the lift of a following bird, thus reducing its requirement for induced power [10]. This explains why birds, especially the migrating birds which have to fly long distances fly together in specific formations.

The V formation is the most famous formation that the migrating birds use to fly long distances. It gets this name because of the similarity of the shape the birds make to the letter "V" (Fig. 3). Here there is a bird leading the flock and two lines of other birds following it. As will be detailed below, it is believed to be a very efficient formation for migrating birds. However, it is not the only formation that bird flocks use. Other typical formations are column formation [21], bow-shaped formation [3] or the J and echelon formations which are the variants of the V formation where one leg of the formation is shorter or missing entirely [45].

Two alternative hypotheses have been proposed to explain the use of V formation by birds. First, this way it is possible to save energy during flight [5,31]. Secondly, the V may reflect a mechanism by which birds avoid collisions with each other and stay in visual contact [10,21]. Although, according to some researchers who worked empirically (especially based on the photographs of the bird flocks taken) it was not possible to reject the second hypothesis [10] which can also be explained by the positive correlation of wing-tip spacing (WTS) and depth or by the constant angle of the V (Fig. 3), the main drive of the formation is proved to be saving energy.

The pioneering study which brings a mathematical explanation to the energy saving in V formation is that of Lissaman and Schollenberger [31]. In that study, it was stated that as birds approach each other (a smaller WTS) and as the number of birds increase more energy will be saved. So that, a group of 25 birds for example would have approximately 71% more flight range than a single bird. These results were obtained from aerodynamics theory where birds at the sizes of a plane were assumed. Also, for the WTS only positive values were assumed (i.e. the case of overlapping were not considered).

The study of Lissaman and Schollenberger [31] was followed by some experimental studies. For Canadian geese with a wing span of 1.5 m, the maximum saving is shown to be obtained when there is an overlap of 16 cm [5]. Later the optimum WTS is obtained to be [25]:

$$\mathrm{WTS}_{\mathrm{opt}} = -0.05b \tag{1}$$

where b is the wing span.

In addition to WTS, energy saving may also be affected by the depth (the distance a bird flying behind the bird in leader position). The vortex sheet behind a fixed wing in steady level flight rolls up to form two concentrated vortices within two chord lengths (maximum wing width) of the wing [44]. Thus the optimum depth can be formulated as:

$$D_{\mathrm{opt}} = 2w \tag{2}$$

where w is the maximum width of the wing. Actually, the depth is determined by the wing span, WTS and the angle alpha which provides a comfortable visual contact between birds. If there is a fixed span to width ratio on birds then, once the span is known, the depth can be calculated. In this regard it does not seem to be an independent flight parameter. Perhaps because of this, the effect of depth was not seen as important as WTS and it was ignored by most researchers.

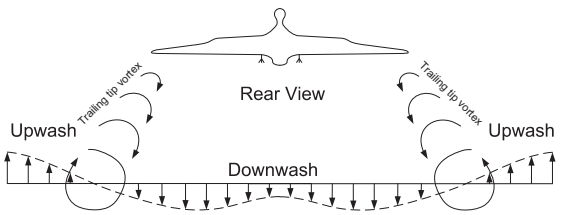

Fig. 2. Regions of upwash and downwash created by trailing vortices.

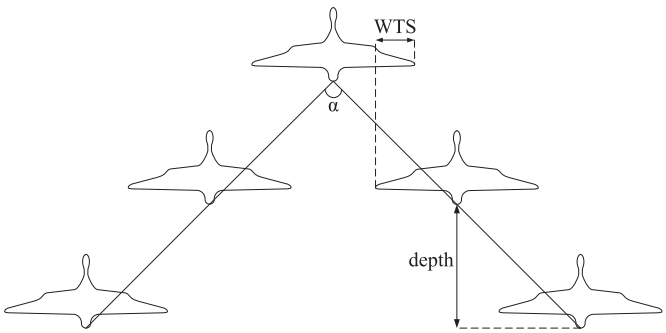

Fig. 3. The V formation.

In the V formation the leader bird is the one spending most energy. The birds in the other positions get benefit from the birds in front. It sounds reasonable that energy saving is higher as we go back in the line but we could not find a study in the literature to support this idea. However it was stated that, the savings of the birds other than the leader bird are either the same [3] or the saving is a bit more for the birds in the middle part [22].

Typically it has been assumed that when the leader bird gets tired after flying for some time it goes to the end of the line and one of the birds following it takes the leader position. However, although no empirical evidence is provided, according to the study of Andersson and Wallander [3] typically the bird flocks consist of the members of the same family where the strongest bird (e.g. father) takes the lead or a combination of several families where the stronger members take turns to lead.

Although theoretically it was shown that the more the number of birds the more the saving [31], in practice the number of birds in the flocks observed are limited [10,22]. As an explanation to this, it was stated that as the size of the flock increases it gets more difficult to get in line for the birds at the rear and the coordination is lost. Consequently the birds at the rear do not benefit from the formation flight and then they may form another smaller flock of their own [45]. On the other hand, the family structure of Andersson and Wallander [3] can be an alternative explanation for some of the smaller size flocks.

A solution algorithm which is inspired from the V flight formation is presented in the next section.

3. The Migrating Birds Optimization algorithm

The MBO algorithm is a neighborhood search technique. It starts with a number of initial solutions corresponding to birds in a V formation. Starting with the first solution (corresponding to the leader bird), and progressing on the lines towards the tails, each solution is tried to be improved by its neighbor solutions (for the implementation of QAP, a neighbor solution is obtained by pairwise exchange of any two locations). If the best neighbor solution brings an improvement, the current solution is replaced by that one. There is also a benefit mechanism for the solutions (birds) from the solutions in front of them. Here we define this benefit mechanism as sharing the best unused neighbors with the solutions that follow (here "unused" means a neighbor solution which is not used to replace the existing solution). In other words, a solution evaluates a number of its own neighbors and a number of best neighbors of the previous solution and considered to be replaced by the best of them. Once all solutions are improved (or tried to be improved) by neighbor solutions, this procedure is repeated a number of times (tours) after which the first solution becomes the last, and one of the second solutions becomes first and another loop starts. The algorithm is stopped after a number of iterations.

Below, first the notation used and then the pseudocode of the MBO algorithm are given. Let,

n = the number of initial solutions (birds)
k = the number of neighbor solutions to be considered
x = the number of neighbor solutions to be shared with the next solution
m = number of tours
K = iteration limit

Pseudocode of MBO:

```
1.  Generate n initial solutions in a random manner and place them on an hypothetical V formation arbitrarily
2.  i = 0
3.  while(i < K)
4.    for (j = 0; j < m; j++)
5.      Try to improve the leading solution by generating and evaluating k neighbors of it
6.      i = i + k
7.      for each solution s_r in the flock (except leader)
8.        Try to improve s_r by evaluating (k–x) neighbors of it and x unused best neighbors from the solution in the
    front
9.          i = i + (k − x)
10.     endfor
11.   endfor
12.   Move the leader solution to the end and forward one of the solutions following it to the leader position
13. endwhile
14. return the best solution in the flock
```

As should already be noticed, the MBO algorithm has great similarities with the migrating birds' story. First it treats the solutions as birds aligned on a V formation. The number of neighbors generated (k) can be interpreted as the induced power required which is inversely proportional to the speed (recall the discussion in the previous section). With a larger k we would assume that birds are flying at a low speed where we can also make the analogy that while traveling at a low speed, one can

explore the surrounding in more detail. The benefit mechanism between the birds is respected and by generating fewer neighbors for the solutions at the back, it was made possible that they get tired less and save energy by using the neighbors of the solutions in the front. The parameter x is seen as the WTS where an optimum value can be sought for. Its optimum value could be interpreted as the optimum overlap amount of the wingtips. In line 4, the parameter m can be regarded as the number of wing flaps or the profile power needed where we can assume that, as each bird travels the same distance, they all spend the same profiling energy. In line 12, similar to the real birds' story, the bird who spent the most energy and thus got tired moves back to get some rest and another bird fills its position.

For the MBO to perform better, it is necessary to determine the best values of some parameters. These are the number of birds to have in the flock (n), the speed of the flight (k), the WTS (x) and the number of wing flaps before a change in the order of the birds or the profiling energy spent (m). Similar to birds' story, one could expect some certain values of these parameters and their combinations might increase the performance of the algorithm. For example we might expect smaller values of x could be expected to perform better since the optimum WTS (overlap) was shown to be a very small figure. Also modest values of k and m might be better which can be regarded as a compromise between profile and induced powers. Another parameter that needs to be decided on is the iteration limit (K) where obviously we could expect better solutions with higher values of K with the cost of higher run times.

The philosophy of the MBO is that, by starting with a number of parallel solutions, it is aimed to explore more areas of the feasible solution space. The exploration is made possible by looking at the neighbor solutions. Each time one of the solutions (the one in the leader position) is explored in more detail. When one of the solutions fails to improve itself by its own neighbors and if the solution in the front is more promising, it is replaced by one of the neighbors of the solution in the front. This way the neighborhood around the more promising solution will be explored in a greater detail (by the combined forces of two birds or solutions). Still after a few iterations these solutions may go to different directions as long as they find improvements along their ways. However, after some time we might expect most of the solutions converge to one or several neighborhoods where local optima or even the global optimum are contained. The convergence can be faster with larger values of x but in that case the termination could take place before the feasible region is thoroughly explored and thus the results obtained might not be good.

The properties of the MBO which distinguishes it from the other metaheuristic approaches are a number of solutions running in parallel and the benefit mechanism between the solutions. Parallel processing can somehow be regarded as inherited to genetic algorithms and scatter search. On the other hand, although the MBO seems to have some similarities to swarm intelligence algorithms and in particular to the ABC where better solutions are explored more, the benefit mechanism introduced here is totally unique to the MBO.

4. Application

The implementation and comparison of MBO with other metaheuristic algorithms is made primarily on some real life QAP instances. In the following we first describe these benchmark data and give some short information on the algorithms compared. Then, in the last subsection, we provide some initial experimental results obtained.

4.1. Description of benchmark data

To test the performance of the MBO algorithm the QAP as described in [14] and the test problems solved therein are used. That QAP was related to the printed circuit board (PCB) assembly shops where the decision of where to locate the different electronic components in the cells of a linear feeder mechanism was leading to arise of a QAP. It is a special type of QAP in that, the departments are located on a line (as opposed to the classical example of two dimensional layout optimization problem given for the QAP) and the flow matrix is very sparse (being most entries equal to zero).

The study [14] was based on [13] where the coding was made on Turbo Pascal 7. For the study [29] and here, since it was not possible to rerun those codes and extract the best performing algorithm recorded therein (it was coded as H14) as a benchmark heuristic to the MBO algorithm, we had to re-code it using Java. Fortunately we were able to obtain (with no or very small differences) the data of eight out of 11 problems used in [14]. Then, we tested our new H14 on these eight problems to see if we would get similar results and indeed we got (note that it is not possible to obtain exactly the same results since random number seeds used now and then are different). Then we continued with the coding of the MBO algorithm. The codes for H14, the MBO algorithm (together with detailed textual descriptions of its procedures) and the data of the eight problems can be obtained through the *url* http://www.mbo.dogus.edu.tr.

4.2. Description of benchmark algorithms

In [14], to find a good performing solution to the QAP a number of heuristic algorithms including 2-opt, 3-opt and the metaheuristics tabu search, simulated annealing and guided evolutionary simulated annealing (a hybrid of genetic algorithms and simulated annealing) were implemented. Including the different settings of the parameters 44 alternative heuristics were obtained whose performances are then compared on 11 different problems obtained from real PCB assembly shops. At the end, the heuristic named as H14 was found to be the best one which is a simulated annealing implementation

with initial temperature (T) equaling 1000 and number of exchange trials (R) at the initial temperature equaling 20. After R exchange trials, the temperature is divided by 1.1 and R is multiplied by 1.5. Similar to the analogy to metal annealing, when the temperature is high we easily accepted worse solutions and satisfied with a little exploration of the neighborhood however, after the temperature is decreased, worse solutions are hardly accepted but a larger neighborhood is explored. H14 was one of the 10 SA heuristic implementations having different parameter values.

Later the comparison of metaheuristic approaches when applied to the QAP was extended to include genetic algorithms (GA) and scatter search (SS) in [29]. The performances of 12 different GA implementations and 14 different SS applications are compared with that of H14 and it was observed that H14 was still the winner.

Later, by the suggestions of an anonymous referee, during the course of this study we have included the PSO and DE metaheuristics in the benchmark algorithms lists. Both algorithms were originally developed for the optimization of continuous functions. One of the few studies who made an application of PSO to a discrete problem was due to Pan et al. [39] who solved a flow shop scheduling problem. In our study we picked up their approach and used insert mutation and one-cut crossover operator for mutation and crossover operators. As for DE, Pan et al. [38] proposed a simple and novel discrete DE (DDE). It basically relies on perturbations of the previous generation best solution, called the mutant population, to be recombined by each of the target population individual to generate a trial population. Ultimately a selection operator is applied to both competing population individuals, namely trial and target population individuals, to decide on which one should be the member of the target population for the next generation. We used the insert operator defined in Tasgetiren et al. [49] and two-cut PTL crossover operator defined in Pan et al. [39].

Both PSO and DE need their parameters to be fine tuned. We performed this task on the eight PCB data described above where the runtime was equated with the other metaheuristics. For PSO the best parameter values appeared to be $c_1 = 0.8$, $c_2 = 0.8$ and $w = 0.5$ where c_1, and c_2 are the crossover probabilities and w corresponds to mutation probability. For DE, we concluded that $cxp = 0.8$, $mp = 0.2$ and $p = 1$ is a good parameter set where cxp, mp and p stand for crossover probabiliy, mutation probability and mutation strength, respectively.

4.3. Initial results obtained

The results obtained where the above mentioned metaheuristics applied to the eight PCB data are given in Table 1.

Each problem is solved 10 times using different random number seeds and the average performance on these 10 runs are tabulated in Table 1. However, for H14 we provide more detail so that the *avg*, *min* and *max* columns display the average, minimum and maximum values of the objective function among those 10 runs. For the other algorithms we only displayed the average performance on the 10 runs (deviation from the average performance of H14). The figure N shows the number of component types (departments), *density* shows the percentage of nonzero entries in the flow matrix and the number of pairwise exchange trials is limited with N^3 as in H14. The board names are the same with the ones in [14]. We can see that the performance of simulated annealing (H14) is considerably better than the other metehauristics.

After the implementation of the MBO algorithm, to get the first results we made a quick analysis to determine the first set of parameter values (a more detailed parameter analysis is made in the next section). This analysis which is made on two problems can be seen in Table 2 where the minimum figure of each column is given in bold. The *min* column provides a measure for best performance, the *max* column for the worst performance and the *avg* for the average performance. Obviously if one has sufficient time to wait for a better solution, the results given under *min* are more prominent. However, if run time allowance is limited then the *avg* and *max* measures are also important. As this decision can change by the application domain we preferred to look at all measures. *cost-B1* and *cost-B7* columns give the average of these three columns for boards B1 and B7, respectively. The total cost on two problems (sum of *cost-B1* and *cost-B7*) is given in the *cost* column. According to this column which can be regarded as the overall performance, the parameter values being $n = 51$, $k = 7$, $x = 3$ and $m = 20$ are selected to continue with. This row is highlighted in grey in the table.

The above determined parameter values are applied to other six problems also and the results given in Table 3 are obtained. The results indicate that the MBO algorithm performs considerably better than the simulated annealing algorithm (H14). As for the average performance (*avg* column), it performed more than two per cent better based on the average of

Table 1
The results obtained by simulated annealing (H14).

Board	Name	N	Density (%)	H14			GA (%)	SS (%)	TS (%)	GESA (%)	PSO (%)	DE (%)
				Avg	Min	Max						
B1	PS11AK17N3	58	4.55	1165	1076	1206	5.58	28.24	9	6	17.7	58.6
B2	PS11AK12-7	54	4.59	842	800	912	8.91	22.21	8	2	18.5	64.8
B3	PS11AK08-9	52	4.80	820	740	882	4.39	21.10	4	1	22.4	59.6
B5	PS11AK16-5	50	8.12	1543	1474	1680	3.31	16.85	4	1	14.9	43.2
B6	PS11AK1011	48	5.69	807	756	896	5.20	21.07	2	−1	16.3	55.6
B7	PS11AK16-4	49	7.62	1461	1392	1536	2.05	9.45	3	0	13.6	37.4
B8	PS11AK16-3	47	8.60	1396	1370	1460	0.86	15.69	5	1	11.3	40.0
B9	PS11AK15-4	40	7.88	752	718	768	1.46	11.17	7	1	15.8	36.0
	Average						3.97	18.22	5.25	1.38	16.32	49.39

Table 2
Initial analysis to determine the parameter values.

n	k	x	m	B1			B7			*Cost-B1*	*Cost-B7*	Cost
				Avg	Min	Max	Avg	Min	Max			
11	7	3	1	1145	1068	1192	1489	1420	1594	1135	1501	2636
25	7	3	1	1151	1072	1232	1462	1394	1506	1152	1454	2606
25	11	5	1	1162	1102	1230	1459	1418	1518	1165	1465	2630
25	21	10	1	1155	1114	1232	1461	1404	1504	1167	1456	2624
51	7	3	1	1134	1068	1212	1429	**1390**	1466	1138	1428	2567
51	11	5	1	1117	1064	1206	**1422**	1392	1466	1129	**1427**	2555
51	21	10	1	1127	**1062**	1174	1451	1428	1486	1121	1455	2576
51	41	15	1	1142	1092	1192	1460	1406	1548	1142	1471	2613
101	7	3	1	1128	1090	1174	1437	1402	1484	1131	1441	2571
101	11	5	1	1141	1116	1164	1439	1408	1470	1140	1439	2579
101	21	10	1	1136	1098	1182	1454	1406	1516	1139	1459	2598
101	41	15	1	1145	1106	1170	1481	1430	1510	1140	1474	2614
11	7	3	20	1148	1102	1200	1478	1420	1560	1150	1486	2636
25	7	3	20	**1109**	1068	1150	1426	1394	1468	**1109**	1429	2538
25	11	5	20	1139	1068	1206	1443	1394	1536	1138	1458	2596
25	21	10	20	1152	1106	1212	1451	1412	1486	1157	1450	2606
51	7	3	20	1117	1064	1146	1425	1398	**1462**	**1109**	1428	**2537**
51	11	5	20	1145	1090	1178	1432	**1390**	1486	1138	1436	2574
51	21	10	20	1120	1096	1144	1444	1408	1482	1120	1445	2565
51	41	15	20	1128	1078	**1160**	1458	1406	1518	1122	1461	2583
101	7	3	20	1119	1082	**1160**	1445	1408	1494	1120	1449	2569
101	11	5	20	1140	1118	1176	1444	1402	1482	1145	1443	2587
101	21	10	20	1136	1108	1164	1437	1406	1464	1136	1436	2572
101	41	15	20	1168	1148	1202	1458	1422	1490	1173	1457	2629

Table 3
The full results obtained with the initially set parameter values.

Board	N	H14				MBO				Improvement			
		Avg	Min	Max	Cost	Avg	Min	Max	Cost	Avg (%)	Min (%)	Max (%)	Cost (%)
B1	58	1165	1076	1206	1149	1117	1064	1146	1109	4.30	1.13	5.24	3.55
B2	54	842	800	912	851	820	780	860	820	2.68	2.56	6.05	3.76
B3	52	820	740	882	814	784	746	826	785	4.59	−0.80	6.78	3.52
B5	50	1543	1474	1680	1566	1536	1464	1594	1531	0.46	0.68	5.40	2.18
B6	48	807	756	896	820	798	776	816	797	1.13	−2.58	9.80	2.78
B7	49	1461	1392	1536	1463	1425	1398	1462	1428	2.53	−0.43	5.06	2.39
B8	47	1396	1370	1460	1409	1387	1370	1412	1390	0.65	0.00	3.40	1.35
B9	40	752	718	768	746	736	720	758	738	2.17	−0.28	1.32	1.07
Average										2.31	0.04	5.38	2.58

eight problems. In terms of the best result on ten runs (*min* column), while H14 outperformed the MBO on some problems, their performances are almost the same. However, as we look at the worst performances (*max* column), MBO is much better. This indicates that the MBO is a more stable algorithm than H14. The superiority of the MBO in terms of the average and worst case performances are also validated by a one-sided t test with $\alpha = 0.05$ where the test did not find a significant difference in terms of the best performances. Overall, MBO seems to be 2.58% better than H14 (according to the *cost* column which is equal to the average of *avg*, *min* and *max* columns). We will come back to this analysis after the detailed parameter fine tuning experiments given in the next section.

5. Parameter fine tuning

In this section we try to find the best values of the four parameters of the MBO which may affect its performance. For this we have determined a number of possible and reasonable values for the parameters as listed in Table 4. There is some rationale behind these values. For example the value of n is taken as an odd number so that we could have a V with equal leg lengths. Also, by considering the solution in the front the value of k should be greater than or equal to $(2x + 1)$ so that after it uses the best neighbor it can have enough number of neighbors to share with the following solutions. In Table 4, the number of $k * x$ combinations are 81 and of these 29 are infeasible. So we had a total number of 3640 $(10 * 7 * (81 - 29))$ different combinations of parameters. Actually for the reasons explained in Section 3, we did not expect some of these combinations to perform good but just to have a broad experimentation we run them all. As we stated before, another parameter that needs to be decided on is the iteration limit (K) where better solutions could be expected with higher values of K. To stay comparable with H14, in our experiments we kept the value of K constant at N^3.

Table 4
Values of parameters used in computational experiments.

Parameter	Values
number of birds in the flock (n)	3, 5, 7, 9, 13, 19, 25, 51, 101, 201
speed of the flight (k)	3, 5, 7, 9, 13, 19, 25, 51, 101
number of wing flaps (m)	1, 2, 5, 10, 20, 40, 80
WTS (x)	1, 2, 3, 4, 5, 6, 7, 8, 9

The MBO algorithm is run 10 times for each combination of the parameters and thus 36,400 runs are performed for each of the eight QAP problem instances. The run time per instance was about 10 hours on the average on a PC having Intel Core2 Duo processor at 2.83 GHz running Windows Vista 32-bit OS with 3 GB RAM. Based on the results obtained, some detailed analyses are made as discussed below. However, as some of these combinations do not seem to be meaningful they were excluded from the analysis. For example, if we consider the combination $n = 101$, $k = 25$, $m = 80$, $x = 8$, the iteration limit K will be exceeded in the first pass of while loop (line 3 in the pseudocode) for $N = 50$ ($101 \times (25 - 8) \times 80 > 50^3$), meaning that there is no time to change the leader bird even once. Since the basic philosophy of the MBO requires all birds take turn to take the lead, the combinations which do not allow all birds to take the lead at least once are disregarded (this was also because their performances turned out to be quite poor).

In the following four subsections, we investigate the performance of MBO when these four parameters are changed.

5.1. Effect of the number of birds (n)

In the first analysis, the number of birds (n) is fixed and all results taken with various combinations of k, x and m are averaged to find average results for that value of n. These average results (rounded to the nearest integer) are given in Table 5 where the minimum values of each column are indicated in bold (this applies also to Tables 6–9 that follow). The entry of $n = 201$ for B9 is blank because it failed the *at least one leadership for all birds requirement* as explained above. We see that the case of 51 birds results in the best performance which is followed by 101 birds. However, as we increase the number of birds further, the performance gets worse similar to the case of fewer number of birds. In other words, there is a U shape behavior in the performance of MBO with respect to the number of birds. Actually with larger number of birds (parallel solutions) we might expect a more thorough search of the feasible space but with a limited total number of iterations the neighborhood of those solutions may not be explored in enough detail and this may be the explanation of worse behavior in large n. This observation seems to be in parallel to the birds' story given in Section 2 where it is believed that there is a reasonable number of birds in flocks. As the problem size of our eight boards are around 51 one might think that there is a relation between the problem size and the optimum value of n but this should be verified by additional and independent set of experiments.

As compared to the analyses of the other parameters (see the next subsections) the variation in the cost values is the highest when n is varied. Thus, we conclude that n is the most important parameter of MBO.

5.2. Effect of the speed of the flight (k)

As we stated before, the number of neighbors generated (which equals to k for the leading bird) can be interpreted as the induced power required for the flight which is known to be inversely proportional with the forward speed. The performance of the MBO algorithm with respect to speed of the flight is presented in Table 6.

From Table 6 we can observe that as the value of k increases, the performance of the MBO algorithm degrades with $k = 3$ giving the best performance. Since at low k values birds are assumed to be flying at higher speeds, this corresponds to saying that higher speeds are better for energy minimization. This could be interpreted as although birds will get more tired at higher speeds they will need less energy for lifting themselves and will be able to fly more distances as they will spend their energy for moving forward mostly.

Table 5
Average cost eight QAP problems with respect to number of birds.

n	B1	B2	B3	B5	B6	B7	B8	B9
3	1179	857	829	1604	830	1486	1421	758
5	1171	851	823	1595	822	1479	1415	754
7	1165	844	820	1589	819	1473	1411	750
9	1161	839	816	1582	816	1468	1408	747
13	1153	832	810	1571	810	1460	1403	743
19	1145	826	803	1562	805	1452	1397	739
25	1138	821	800	1554	802	1450	1394	737
51	**1128**	**813**	**791**	1538	**795**	**1435**	**1389**	**731**
101	1129	818	795	**1529**	803	1436	1394	739
201	1163	846	829	1552	839	1476	1437	

Table 6
Average cost of eight QAP problems with respect to speed of the flight.

k	B1	B2	B3	B5	B6	B7	B8	B9
3	**1148**	**826**	**803**	**1558**	**806**	**1453**	**1395**	**738**
5	1152	828	807	1566	809	1459	1399	744
7	1156	832	809	1571	812	1462	1402	746
9	1157	834	812	1574	813	1464	1405	746
13	1158	837	814	1580	815	1467	1407	748
19	1160	841	816	1583	816	1469	1408	749
25	1160	840	816	1583	817	1469	1410	749
51	1161	843	818	1585	818	1470	1412	750
101	1162	846	821	1590	822	1475	1417	754

From this analysis the natural question of what will happen for $k = 1$ arises. The case $k = 1$ requires that $x = 0$ where, the MBO turns out to be improving n solutions with straightforward pairwise exchanges running independently and in parallel. This special case will be investigated below in Section 5.4.

5.3. Effect of the number of wing flaps (m)

Once the order of birds in the V formation is settled, m can be interpreted as the length of time this order is maintained which mainly determines how much the leading bird will be tired due to the forward move. A small m value, such as one, means that each successor solution benefits from its predecessor only once and the birds are relocated immediately before the leading bird gets tired much. On the other hand, a large m value, such as 80, means that the same order of birds is maintained for a long time and the leading bird gets too tired. We may expect that moderate values of m may result in better performance. The results obtained with different values of m are given in Table 7.

We see that, as expected, there is a U shape behavior in the performance when m is varied. We get lower costs for $m = 10$ and $m = 20$ (with $m = 10$ being slightly better) and for smaller or larger m values the costs are higher. An interpretation of this may be that for small m values, the order of birds changes quickly leading to a lack of convergence. On the other hand for larger m values, the worse performance can be explained by converging and getting stuck in local optima quickly. However, the variation in performance is not as much as the cases where n or k are varied and thus m is a less critical parameter.

5.4. Effect of WTS (x)

WTS is an important parameter of the MBO algorithm which is one of its distinguishing properties from the other metaheuristics. Its value determines the degree of benefit birds can get from their predecessor birds. Since here we defined the benefit mechanism as the number of good neighbor solutions obtained from the predecessor solution, a very high value of x means that all solutions will be similar to each other and thus, there may be a premature convergence. The results obtained with different values of x are given in Table 8 where parallel to our expectations $x = 1$ produced better results (remember in the real birds' story the optimum overlap amount was shown to be a small percentage of the wing span).

Next, as we have also indicated in the analysis of k, we wanted to analyze the special case of $k = 1$ and $x = 0$ (i.e. the case of parallel straightforward pairwise exchanges). For this purpose, we fixed the value of most effective parameter n to 51 (its best value) and obtained the results for different values of m, k and x. The average results are displayed in Table 9. We see that, although there is a gradual decrease in the cost as we decrease x down to one, there is a sudden increase if it is further decreased to zero. So we may conclude that some collaboration between the solutions (birds) is useful.

Here we would like to comment more on the special value of $x = 1$. When $x = 1$, the best unused neighbor of a solution is propagated to its immediate predecessor and it is used by that one or not but, it is depleted at that level. However, when x is greater than 1, some of them can also be propagated to the solutions further back which may in return result in a quicker convergence and worse performance.

Table 7
Average cost of eight QAP problems with respect to number of wing flaps.

m	B1	B2	B3	B5	B6	B7	B8	B9
1	1160	841	817	1585	819	1471	1412	749
2	1160	841	816	1582	817	1469	1409	748
5	**1157**	838	815	1581	815	1467	1408	**747**
10	**1157**	**837**	**813**	1579	**814**	1465	**1406**	**747**
20	**1157**	838	814	**1576**	**814**	**1464**	**1406**	748
40	1159	838	**813**	1578	815	1467	1407	750
80	1162	840	**813**	1579	817	1469	1408	749

Table 8
Average cost of the eight QAP problems with respect to WTS.

x	B1	B2	B3	B5	B6	B7	B8	B9
1	**1153**	**833**	**810**	**1571**	**811**	**1460**	**1403**	**744**
2	1157	836	813	1577	814	1465	1406	747
3	1158	839	814	1580	815	1468	1408	748
4	1160	840	816	1583	816	1470	1410	750
5	1162	840	818	1584	817	1471	1410	750
6	1162	842	819	1585	819	1470	1411	750
7	1161	844	818	1586	819	1473	1411	752
8	1162	843	819	1588	819	1473	1411	751
9	1162	843	819	1588	820	1472	1412	751

Table 9
Average cost of the eight QAP problems with respect to WTS ($n = 51$).

x	B1	B2	B3	B5	B6	B7	B8	B9
0	1130	815	790	1522	798	1428	1396	742
1	**1118**	**805**	**785**	**1517**	**790**	**1422**	**1382**	**727**
2	1123	810	788	1532	792	1433	1388	729
3	1128	813	790	1545	793	1432	1389	731
4	1132	816	793	1541	795	1439	1388	731
5	1139	814	798	1544	796	1442	1392	735
6	1134	817	798	1553	799	1447	1397	742
7	1129	823	793	1555	800	1445	1395	737
8	1137	820	799	1554	806	1447	1395	738
9	1134	817	795	1555	804	1443	1393	733

Table 10
The results obtained after the parameters are fine tuned.

Board	H14				MBO (best-set)				Improvement			
	Avg	Min	Max	Cost	Avg	Min	Max	Cost	Avg (%)	Min (%)	Max (%)	Cost (%)
B1	1165	1076	1206	1149	1124	1074	1174	1124	3.67	0.19	2.73	2.19
B2	842	800	912	851	803	764	824	797	4.83	4.71	10.68	6.74
B3	820	740	882	814	784	762	840	795	4.62	−2.89	5.00	2.24
B5	1543	1474	1680	1566	1496	1462	1546	1501	3.17	0.82	8.67	4.22
B6	807	756	896	820	786	758	816	787	2.65	−0.26	9.80	4.06
B7	1461	1392	1536	1463	1416	1398	1456	1423	3.18	−0.43	5.49	2.75
B8	1396	1370	1460	1409	1378	1358	1402	1379	1.34	0.88	4.14	2.12
B9	752	718	768	746	729	722	736	729	3.16	−0.55	4.35	2.32
Average									3.32	0.31	6.36	3.33

5.5. Results obtained by the parameter fine tuned MBO

According to the above discussed experiments the best set of parameter values can be identified as $n = 51$, $m = 10$, $k = 3$ and $x = 1$. The results obtained with this best-set are given below in Table 10.

According to these results, the performance of the MBO has improved almost one per cent as compared to the results obtained by the initial set of parameters. The new results are better than the SA in every category and this is validated by the t tests on *min*, *max*, *avg* and *cost* columns with $\alpha = 0.05$. As a summary comparison we can say that, for the PCB oriented QAP instances, the MBO algorithm outperforms the SA by 3.33%.

6. Application of MBO to benchmark problems in QAPLIB

After observing that MBO gets very successful results for the QAP instances arising from the real PCB assembly data, we wanted to see its ability in obtaining optimum solutions. For this purpose we applied to QAPLIB web page where more than one hundred QAP instances are available together with their optimum or best known solutions (BKS) [7]. It could be assumed that, although these BKS are not named as optimum they should be very close to it since huge amount of computational effort have been given to obtain them.

To have an idea on the performance of MBO on these instances, we limited the run time with N^3 iterations. This corresponded to 0.2 s for a problem of size 32 and to 25 s for a problem of size 80 on the specified machine. Note that this might

Table 11
Results on benchmark problems with standard parameter values.

Problem	Size	Density (%)	BKS	MBO (%)	H14 (%)	GA (%)	SS (%)	PSO (%)	DE (%)
Sparse									
esc32e	32	1.17	2	0.00	0.00	0.00	0.00	0.00	0.00
esc32f	32	1.17	2	0.00	0.00	0.00	0.00	0.00	0.00
esc32g	32	1.76	6	0.00	0.00	0.00	0.00	0.00	0.00
esc32h	32	27.54	438	0.00	5.94	0.91	4.57	5.02	15.07
esc64a	64	3.17	116	0.00	31.03	5.17	12.07	8.62	24.14
tai64c	64	4.13	1,855,928	0.00	5.23	1.78	4.89	0.32	0.19
Dense									
lipa40b	40	94.81	476,581	5.29	21.15	21.74	21.32	22.46	24.73
sko49	49	97.96	23,386	1.27	8.41	5.94	8.04	7.92	12.50
wil50	50	98	48,816	0.57	4.15	3.48	3.42	4.17	7.58
tai60b	60	98.33	608,215,054	0.17	17.58	6.42	15.30	11.13	26.45
lipa70a	70	97.16	169,755	0.81	1.44	1.38	1.49	1.41	1.76
lipa80a	80	97.52	253,195	0.74	1.32	1.32	1.29	1.28	1.59
Average				0.74	8.02	4.01	6.03	5.19	9.50

Table 12
Results on benchmark problems with improved parameter values.

Problem	n	k	m	x	MBO	Deviation (%)
Sparse						
esc32e	All	All	All	All	2	0.00
esc32f	All	All	All	All	2	0.00
esc32g	All	All	All	All	6	0.00
esc32h	Many	Many	Many	Many	438	0.00
esc64a	All	All	All	All	116	0.00
tai64c	Many	Many	Many	Many	1855928	0.00
Dense						
lipa40b	Many	Many	Many	Many	476581	0.00
sko49	5	9	80	1	23420	0.15
wil50	13	13	5	4	48834	0.04
tai60b	3	19	2	9	608231607	0.00
lipa70a	19	19	5	2	170817	0.63
lipa80a	19	101	1	9	254729	0.61
Average						0.12

be far less than the time spent for obtaining the BKS. As we would not be able to solve all 137 problems of QAPLIB we made a selection from them. In the selection we first wanted that the problem sizes (number of departments) are close to our PCB problems. Second, we wanted to test the effect of sparsity, if it has any. Consequently, we took six sparse problems similar to PCB problems, and another set of six dense problems to solve with the MBO. The names, sizes and densities of these 12 problems are given in Table 11.

The results obtained when these problems are solved with the parameter values settled in the previous section ($n = 51$, $k = 3$, $m = 10$, $x = 1$) are also tabulated in Table 11. In contrast with the analysis of the previous section, only the minimum costs obtained in 10 runs are considered since the aim is to check the ability of MBO in finding BKS. We observe that MBO was able to find BKS for all sparse problems and it was very close to BKS for dense problems except lipa40b. For the algorithms a change in the performance on sparse and dense problems can be expected as it was also observed in [14]. This is due to the fact that, in dense problems there can be many different solutions with a similar objective value and any pairwise exchange can result in a (small) change on it. In such a case finding the best solution in the neighborhood of a solution would require a lot of effort (more iterations). On the other hand, in sparse problems, most exchange trials would result in no change in objective value (since most flow values are zero) and the better solutions of the problem should be fewer in number and are situated in some certain areas of the feasible space. In such a case focusing on these certain areas and exploring them in more detail could help in finding a better solution. As MBO has the property of converging on some neighborhoods, this can be the explanation of why it performs better on sparse problems. Consequently, it can be expected that its performance on dense problems be increased by increasing the number of iterations or weakening its fast convergence property by smaller x values. In general terms one should note that no set of different problems or instances of the same problem with different characteristics as it is the case here can be expected perform good with the same set of parameter values [51].

These problems are also solved by the GA and SS with their parameters as determined in [29] and by PSO and DE with their parameters determined on PCB problems (see Section 3). The deviations of the minimum of the 10 runs from BKS are displayed in Table 11. As it could be observed, MBO seems to be a good solver of the QAPLIB problems also.

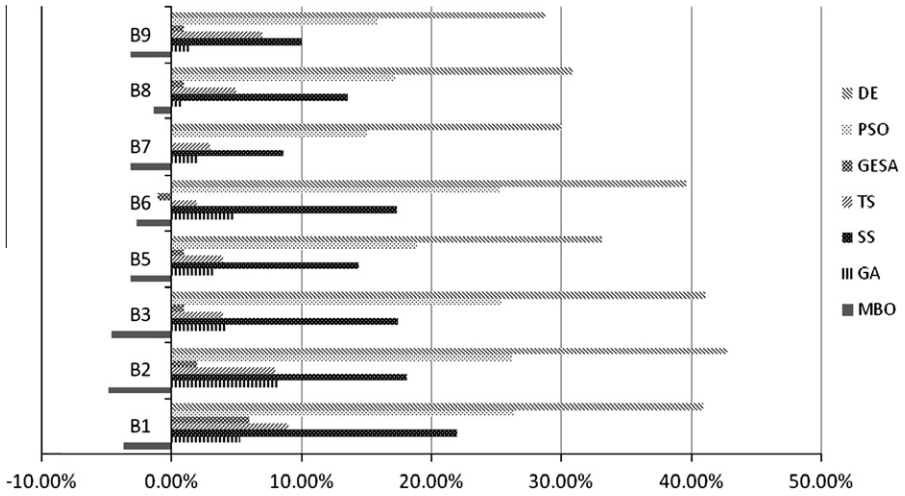

Fig. 4. Performances of MBO and other metaheuristics as compared to SA.

At this point we wanted to know if MBO could perform better if the parameter values were varied. For this purpose, each problem is solved again with a number of different parameter sets as determined in Table 4 similar to the previous section. The new best solutions, the parameter values giving these best solutions are given in Table 12. If all or most of the parameter sets gave the best solution, this is indicated by the words "all" or "many".

This time we observe that, the performance of MBO has improved further with only 0.12% average deviation from BKS. Note that, if one is very keen for a better result, he might obtain even better solutions by running MBO for a longer time.

7. Summary, conclusions and future work

In this study, inspired from the V flight formation of the migrating birds, we proposed a new metaheuristic approach which we name as the Migrating Birds Optimization (MBO) algorithm. In order to explain the logic behind the algorithm we first gave the necessary and sufficiently detailed information on bird flying.

The performance of the algorithm is tested on solving quadratic assignment problems arising from printed circuit board assembly workshops. Two previous studies on this problem where five different metaheuristic approaches are implemented and compared are taken as the benchmarks. The MBO algorithm outperformed the best performing heuristic reported therein (the simulated annealing) by about three per cent on the average. The relative performances of all these metaheuristics as compared to SA (including the PSO and the DE implemented in this study) are displayed in Fig. 4 (based on the average performances on 10 runs with different random number seeds).

The MBO algorithm is also applied to a set of benchmark problems obtained from QAPLIB where it was able to find the best known solution in most cases. These results indicate that MBO has a potential for becoming one of the most competitive metaheuristic algorithms. The benefit mechanism makes it possible to explore the more promising areas of the search space in more detail. This mechanism which is unique to MBO can be seen as its strength.

As immediate areas of further research, the MBO algorithm can be applied to other problem domains (e.g. continuous function optimization) and can be compared with additional metaheuristics. More analyses can be made on parameter fine tuning on sparse and dense QAP problems separately. Also, on contrary to propagating best unused neighbors to proceeding solutions, other forms of benefit mechanism that best fit to application domain under concern can be devised and implemented.

References

[1] S. Abdullah, H. Turabieh, On the use of multi neighbourhood structures within a tabu-based memetic approach to university timetabling problems, Information Sciences 191 (1) (2012) 146–168.
[2] R.K. Ahuja, O. Ergun, J.B. Orlin, A.P. Punnen, A survey of very large scale neighborhood search techniques, Discrete Applied Mathematics 123 (2002) 75–102.
[3] M. Andersson, J. Wallander, Kin selection and reciprocity in flight formation, Behavioral Ecology 15 (1) (2004) 158–162.
[4] M.T. Ayvaz, Application of harmony search algorithm to the solution of groundwater management models, Advances in Water Resources 32 (6) (2009) 916–924.
[5] J.P. Badgerow, F.R. Hainsworth, Energy savings through formation flight? A re-examination of the vee formation, Journal of Theoretical Biology 93 (1981) 41–52.
[6] L. Barcos, V. Rodríguez, M.J. Álvarez, F. Robusté, Routing design for less-than-truckload motor carriers using ant colony optimization, Transportation Research Part E: Logistics and Transportation Review 46 (3) (2010) 367–383.
[7] R.E. Burkard, E. Çela, S.E. Karisch, F. Rendl, QAPLIB – A quadratic assignment problem library, 2010. <http://www.opt.math.tu-graz.ac.at/qaplib/>. (Accessed 11.07.10).

[8] O. Castillo, R. Martinez-Marroquin, P, Melin, F. Valdez, J. Soria, Comparative study of bio-inspired algorithms applied to the optimization of type-1 and type-2 fuzzy controllers for an autonomous mobile robot, Information Sciences 192 (1) (2012) 19–38.
[9] P. Charbonneau, Genetic algorithms in astronomy and astrophysics, Astrophysical Journal Supplement Series 101 (1995) 309–334.
[10] C.J. Cutts, J.R. Speakman, Energy savings in formation flight of pink-footed geese, Journal of Experimental Biology 189 (1994) 251–261.
[11] M. Dorigo, Optimization, learning and natural algorithms, Ph.D. Thesis, Politecnico di Milano, Italy, 1992.
[12] Z. Drezner, Extensive experiments with hybrid genetic algorithms for the solution of the quadratic assignment problem, Computers & Operations Research 35 (2008) 717–736.
[13] E. Duman, Optimization issues in automated assembly of printed circuit boards, PhD Thesis, Bogazici University, 1998.
[14] E. Duman, I. Or, The quadratic assignment problem in the context of the printed circuit board assembly process, Computers & Operations Research 34 (2007) 163–179.
[15] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan, 1995, pp. 39–43.
[16] N. Fescioglu-Unver, M.M. Kokar, Self controlling tabu search algorithm for the quadratic assignment problem, Computers & Industrial Engineering 60 (2011) 310–319.
[17] V. Furtado, A. Melo, A.L.V. Coelho, R. Menezes, R. Perrone, A bio-inspired crime simulation model, Decision Support Systems 48 (1) (2009) 282–292.
[18] Z.W. Geem, J.H. Kim, G.V. Loganathan, A new heuristic optimization algorithm: harmony search, Simulation 76 (2001) 60–68.
[19] F. Glover, Future paths for integer programming and links to artificial intelligence, Computers & Operations Research 13 (5) (1986) 533–549.
[20] F. Glover, G.A. Kochenberger, Handbook of Metaheuristics, Kluwer Academic Publishers, 2003.
[21] L.L. Gould, F. Heppner, The vee formation of Canada geese, Auk 91 (1974) 494–506.
[22] F.R. Hainsworth, Precision and dynamics of positioning by Canada geese flying in formation, Journal of Experimental Biology 128 (1987) 445–462.
[23] J.P. Hamiez, J.K. Hao, Using solution properties within an enumerative search to solve a sports league scheduling problem, Discrete Applied Mathematics 156 (10) (2008) 1683–1693.
[24] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, 1975.
[25] D. Hummel, M. Beukenberg, Aerodynamsiche Interferenseffekte beim formationsflug von vögeln, Journal of Ornithology 130 (1989) 15–24.
[26] M. Kapanoglu, W.A. Miller, An evolutionary algorithm-based decision support system for managing flexible manufacturing, Robotics and Computer-Integrated Manufacturing 20 (6) (2004) 529–539.
[27] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: Artificial Bee Colony (ABC) Algorithm, Journal of Global Optimization 39 (3) (2007) 459–471.
[28] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.
[29] B. M. Kıyıcığı, E. Duman, A.F. Alkaya, Finding best performing solution algorithm for the QAP, in: Proceedings of IMS2010, Sarajevo, Bosnia Herzegovina, 2010, pp. 510–522.
[30] Z. Lian, X. Gu, B. Jiao, A similar particle swarm optimization algorithm for permutation flowshop scheduling to minimize makespan, Applied Mathematics and Computation 175 (1) (2006) 773–785.
[31] P.B.S. Lissaman, C.A. Shollenberger, Formation flight of birds, Science 168 (1970) 1003–1005.
[32] E.M. Loiola, N.M. Abreu, P.O.B. Netto, P. Hahn, T. Querido, A survey of the quadratic assignment problem, European Journal of Operational Research 176 (2007) 657–690.
[33] N. Mansour, H. Tabbara, T. Dana, A genetic algorithm approach for regrouping service sites, Computers & Operations Research 31 (8) (2004) 1317–1333.
[34] M. Marinaki, Y. Marinakis, C. Zopounidis, Honey bees mating optimization algorithm for financial classification problems, Applied Soft Computing 10 (3) (2010) 806–812.
[35] A. Misevicius, D. Rubliauskas, Testing of hybrid genetic algorithms for structured quadratic assignment problems, Informatica 20 (2009) 255–272.
[36] A. Mucherino, O. Seref, A novel meta-heuristic approach for global optimization, in: Proceedings of the Conference on Data Mining, System Analysis and Optimization in Biomedicine, Gainesville, Florida, 2007, pp. 162–173.
[37] M. Nehi, S. Gelareh, A survey of meta-heuristic solution methods for the quadratic assignment problem, Applied Mathematical Sciences 1 (2007) 2293–2312.
[38] Q.-K. Pan, M.F. Tasgetiren, Y.-C. Liang, A discrete differential evolution algorithm for the permutation flowshop scheduling problem, Computers & Industrial Engineering 55 (2008) 795–816.
[39] Q.-K. Pan, M.F. Tasgetiren, Y.-C. Liang, A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem, Computers & Operations Research 35 (9) (2008) 2807–2839.
[40] G. Paul, An efficient implementation of the robust tabu search heuristic for sparse quadratic assignment problems, European Journal of Operational Research 209 (2011) 215–218.
[41] A.S. Ramkumar, S.G. Ponnambalam, N. Jawahar, A new iterated fast local search heuristic for solving QAP formulation in facility layout design, Robotics and Computer-Integrated Manufacturing 25 (2009) 620–629.
[42] G.N. Ramos, Y. Hatakeyama, F. Dong, K. Hirota, Hyperbox clustering with Ant Colony Optimization (HACO) method and its application to medical risk profile recognition, Applied Soft Computing 9 (2) (2009) 632–640.
[43] K.A. Ravindra, B.O. James, T. Ashish, A greedy genetic algorithm for the quadratic assignment problem, Computers & Operations Research 27 (2007) 917–934.
[44] J.M.V. Rayner, A new approach to animal flight mechanics, Journal of Experimental Biology 80 (1979) 17–54.
[45] P. Seiler, A. Pant, J.K. Hedrick, A systems interpretation for observations of bird V-formations, Journal of Theoretical Biology 221 (2003) 279–287.
[46] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (1997) 341–359.
[47] E.D. Taillard, Robust tabu search for the quadratic assignment problem, Parallel Computing 17 (1991) 443–455.
[48] E.D. Taillard, Comparison of iterative searches for the quadratic assignment problem, Location Science 3 (1995) 87–105.
[49] M.F. Tasgetiren, Q.-K. Pan, P.N. Suganthan, Y.-C. Liang, A discrete differential evolution algorithm for the no-wait flowshop scheduling problem with total flowtime criterion, in: Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Scheduling, Hawaii, USA, 2007, pp. 251–258.
[50] M. Uysal, Using heuristic search algorithms for predicting the effort of software projects, Applied and Computational Mathematics 8 (2) (2009) 251–262.
[51] D. Wolpert, W. Macready, No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 67–82.
[52] X.S. Yang, Firefly algorithm, in: Nature-Inspired Metaheuristic Algorithms, Luniver Press, Frome, UK, 2008, pp. 79–90.
[53] C.C. Yang, J. Yen, H. Chen, Intelligent internet searching agent based on hybrid simulated annealing, Decision Support Systems 28 (3) (2000) 269–277.
[54] H. Zhang, C. Beltran-Royo, M. Constantino, Effective formulation reductions for the quadratic assignment problem, Computers & Operations Research 37 (2010) 2007–2016.