Applied Soft Computing 13 (2013) 158–164

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

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

A hybrid meta-heuristic algorithm for optimization of crew scheduling

A. Azadeh [a,*], M. Hosseinabadi Farahani [a], H. Eivazy [b], S. Nazari-Shirkouhi [a], G. Asadipour [a]

[a] Department of Industrial Engineering, Center of Excellence for Intelligent Based Experimental Mechanics, Department of Engineering Optimization Research, College of Engineering, University of Tehran, Iran
[b] Department of Civil Engineering, University of Alberta, Canada

ARTICLE INFO

Article history:
Received 20 June 2010
Received in revised form 28 June 2012
Accepted 4 August 2012
Available online 16 August 2012

Keywords:
Crew scheduling
Combinatorial optimization
Particle swarm optimization
Ant colony optimization
Genetic algorithm
Meta-heuristics

ABSTRACT

Crew scheduling problem is the problem of assigning crew members to the flights so that total cost is minimized while regulatory and legal restrictions are satisfied. The crew scheduling is an NP-hard constrained combinatorial optimization problem and hence, it cannot be exactly solved in a reasonable computational time. This paper presents a particle swarm optimization (PSO) algorithm synchronized with a local search heuristic for solving the crew scheduling problem. Recent studies use genetic algorithm (GA) or ant colony optimization (ACO) to solve large scale crew scheduling problems. Furthermore, two other hybrid algorithms based on GA and ACO algorithms have been developed to solve the problem. Computational results show the effectiveness and superiority of the proposed hybrid PSO algorithm over other algorithms.

1. Introduction

Crew planning and scheduling problem is the determination of the number of crew with special skills and assignment of crew to meet demand. The objectives are minimizing costs, meeting customers' demands, and distributing work equally. The scheduling of crew members, which is the assignment of crew members to a flight for some period of time, is generally divided into crew scheduling and crew rostering [1]. These two problems have some differences in various airlines. Andersson et al. [2] have examined the major differences in terms of crew categories, fleet types, network structures, rules and regulations, regularity of the flight timetables, and cost structures. In fact, the objective of crew scheduling is to build a set of feasible pairings for minimizing the total crew costs and satisfying the given flight schedule, labor union and government regulations, the fleet routes, and the company's own policy. Whereas, in crew rostering problems, the objective is to find pairings assigned to crew members, satisfying crew member's skills, vacations, and other requirements.

Crew scheduling problem can be defined as assignment of a crew to a set of works. The crew scheduling problem can be formulated as a set covering problem (SCP) or set partitioning problem (SPP), which is the problem of covering some flights by crew members so that all the flights are covered, while the cost is minimized [1]. Crew scheduling is an NP-hard problem. Thus, it cannot be exactly solved in a reasonable computation time. It is more than three decades that the crew scheduling problem has taken numerous attentions in operations research community. The main challenge in this problem is that there is not any general method to work well with all kinds of nonlinear cost functions and constraints [3]. Meanwhile, this problem goes to a complicated problem when the number of inputs increases.

Most Researchers examined the crew scheduling problem as a SCP or SPP. For example, Rubin [4] modeled a large-scale crew scheduling problem as a SCP and suggested a solution algorithm based on decomposition and complete enumeration techniques. Gerbracht [5] formulated the crew scheduling problem as a SPP and proposed a solution method on the basis of Lagrangian relaxation and branch-and-bound (B&B) techniques. Lavoie et al. [6] modeled the crew scheduling problem as a SCP, where the column generation approach was used to solve this problem. Anbil et al. [7] modeled the crew scheduling as a SPP and developed a SPIRINT approach that would efficiently generate columns for the solution of large-scale problems. Chu et al. [8] formulated the crew pairing problem as a SPP and developed a graph-based branching heuristic for solving the restricted SPP representing a collection of the best pairings. Barnhart and Shenoi [9] presented an approximate model for the long-haul crew pairing problem. Stojkovic et al. [10] modeled a day-to-day operational airline crew scheduling problem as a SPP and to solve it, developed a column generation method embedded in a branch-and-bound search tree. Wang [11] considered the problem as a SCP and solved it by linear programming. Yan

* Corresponding author.
E-mail address: aazadeh@ut.ac.ir (A. Azadeh).

1568-4946/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.asoc.2012.08.012

and Chang [12] modeled this problem as SPP and a column generation approach was suggested to solve the problem efficiently with a case study in Taiwan airline. Other methods applied in solving this problem are such as the multi-commodity network flow approach [13], the matching model [14], dynamic programming [15], and the network model [16].

Some researchers solved crew scheduling problem in large scale with meta-heuristics algorithms. For example, Ozdemir and Mohan [17] solved this problem using a genetic algorithm (GA) applied to a flight graph representation that represents several problem-specific constraints. Crawford et al. [3] formulated the problem as a SCP and solved it by ant colony optimization (ACO). They solved some test examples of Airline Flight Crew Scheduling by ACO algorithms and some hybridizations of ACO with constraint programming techniques like forward checking. Lo and Deng [18] modeled the problem as a traveling salesman problem (TSP) with flight graph representation and used the ACO algorithm to search near-optimal solutions for airline crew schedules.

PSO algorithm is a stochastic optimization technique developed by Eberhart and Kennedy in 1995 [19] and inspired by social behavior of bird flocking or fish schooling. In comparison to GA, PSO can be performed more easily and have fewer parameters to adjust. These advantages make PSO not only suitable for scientific research but also for engineering applications. The PSO has attracted broad attention in the fields of evolutionary computing and optimization. Although the PSO has been initially developed for continuous optimization problems, it has been successfully applied for solving discrete problems as well [20]. PSO has been used for solving TSP [20,21], vehicle routing problem (VRP) [22,23], cell formation problem [24], feature selection [25], and shortest path problem [26]. PSO has also been implemented for solving different scheduling problems such as single machine scheduling [27], parallel machine scheduling [28], flow shop scheduling [29–31], job shop scheduling [32,33], and open shop scheduling [34,35].

In this paper, a hybrid algorithm particle swarm optimization (PSO) algorithm is proposed for solving the crew scheduling problem. PSO is incorporated with a local optimization heuristic based on the problem-specific knowledge of the search space. Recent studies use GA or ACO to solve large scale crew scheduling problems. Furthermore, two other hybrid algorithms based on GA and ACO algorithms have been developed to solve the problem. Computational results show the effectiveness and superiority of the proposed hybrid PSO algorithm.

The organization of this paper is as follows. In Section 2, the problem of crew scheduling is described. In Section 3, different steps of the PSO algorithm is mentioned in detail. The proposed PSO algorithm is explained in Section 4. The results of the comparison of the proposed PSO algorithm with the well-known algorithms are discussed in Section 5. In Section 6, the conclusion of the paper is presented.

2. Problem description

The aim of crew scheduling is to assign a set of flights to a set of crew members with the objective of minimizing total crew assigning costs and satisfying various functional and legal restrictions including labor union and government regulations as well as the company's own policy [19]. The following regulatory restrictions and security rules have to be considered:

1. In a duty period, the city of arrival of a flight is the same city of departure of its immediately succeeding flight.
2. The sit time, i.e., the amount of time between two consecutive flights must be restricted within lower and upper bounds. If the sit time exceeds the maximum allowed time, the crew members must be sent to rest in hotel.
3. Number of flights in a pairing cannot exceed a prescribed maximum number of flights.
4. Total flying time in a pairing cannot exceed a maximum allowed flying time.
5. Duration of a pairing cannot exceed a prescribed amount of time.
6. Each pairing should start and end at the same base city, i.e., the city where the crew members are stationed. Otherwise, the crew members are taken to their respective bases as ordinary passengers at the end of the pairing by a deadhead flight.

The crew scheduling problem is solved as a daily problem in which it is assumed that every flight leg is operated every day. Although it is possible that some flight legs are not flown in specific days, it has been shown that the solution to the daily problem is a good approximation to the weekly and monthly problems [36].

The cost includes the basic payment for each pairing, the basic minimum payment for each flight, the bonus for overtime flying, the additional pay for the sit time, hotel expenses, and crew flying as passengers (deadheads).

3. Particle swarm optimization

PSO introduced by Kennedy and Eberhart [19] as a method for nonlinear optimization, has been inspired by the flight of birds in a flock or motion of particles. PSO is a population-based optimization algorithm in which each particle is an individual and the set of particles forms the swarm. In order to solve the optimization problem, any potential solution in search space is considered as a potential position for particles. Swarm of particles move through a multi-dimensional search space under a defined dynamics of flight and communicate about the historical solutions in order to find better ones. The position of ith particle in an n-dimensional search space is represented by an n-dimensional vector.

$$x_i = (x_{i1}, x_{i2}, \ldots, x_{in}) \tag{1}$$

Also, every particle possesses a velocity vector as follow:

$$v_i = (v_{i1}, v_{i2}, \ldots, v_{in}) \tag{2}$$

In any iteration, positions and velocities are updated by means of the following equations:

$$v_{ij}(t+1) = I_{ij}v_{ij}(t) + R_{1ij}c_1(P_{ji} - x_{ij}(t)) + R_{2ij}c_2(P_{gj} - x_{ij}(t)) \tag{3}$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \tag{4}$$

where P_{ji} and P_{gj} are, respectively, the jth coordinate of the best previous position of ith particle and the global best previous position, i.e., the best position found so far. R_{1ij} and R_{2ij} are two random variables in [0,1], c_1 and c_2 are acceleration constants, and I_{ij} is the parameter used to model the inertia effect of particle's previous velocity. I_{ij} is used to control exploitation vs. exploration. A lager I_{ij} yields to maintaining high velocities by particles and therefore, prevents particles from becoming trapped in local optima. On the other hand, if a smaller value is assigned to I_{ij}, the particles would possess smaller velocities and would be encouraged to exploit the same search space area. The process of PSO is shown in Fig. 1.

4. The proposed PSO algorithm

The proposed PSO algorithm for solving the crew scheduling problem is described in this section. The overall structure of the proposed algorithm is shown in Fig. 2.

Initialize a population of particles with random positions and velocities

repeat

for each particle *i* ***do***

Update the velocity of particle *i*

Update the position of particle *i*

Evaluate the fitness value of particle *i* by mapping the position of particle *i* into the solution space

Update the particle best position and the global best position

end for

until a termination criterion, e.g. a predefined number of iterations, is met

Fig. 1. The process of particle swarm optimization.

4.1. Position representation

In the proposed algorithm, the particle *k* position, i.e., a solution to the crew scheduling problem is represented by a priority list. The priority list is a vector of *n* elements (*n* is the number of flights) in which each element shows the priority of the corresponding flight. By ordering the elements of the priority list, it can be easily transformed to a permutation list which is a permutation from 1 to total number of flight legs. Representing a solution by means of a priority list makes it easy intelligible and comprehensible and has the advantage of reducing computational complexity [34]. Furthermore, it provides the all information such as departure and arrival cities and times required for checking the legality of the pairings and hence, the feasibility of the solution.

4.2. Generation of initial particles

At first, the feasible initial solutions are created. In this regard, the following steps are applied:

(1) Find inseparable flights
(2) Specify Starter and Terminator flights. Every created pairing should start with one of the nodes (flight) in Starter list and end with a node in Terminator list.
(3) Generate a feasible solution.

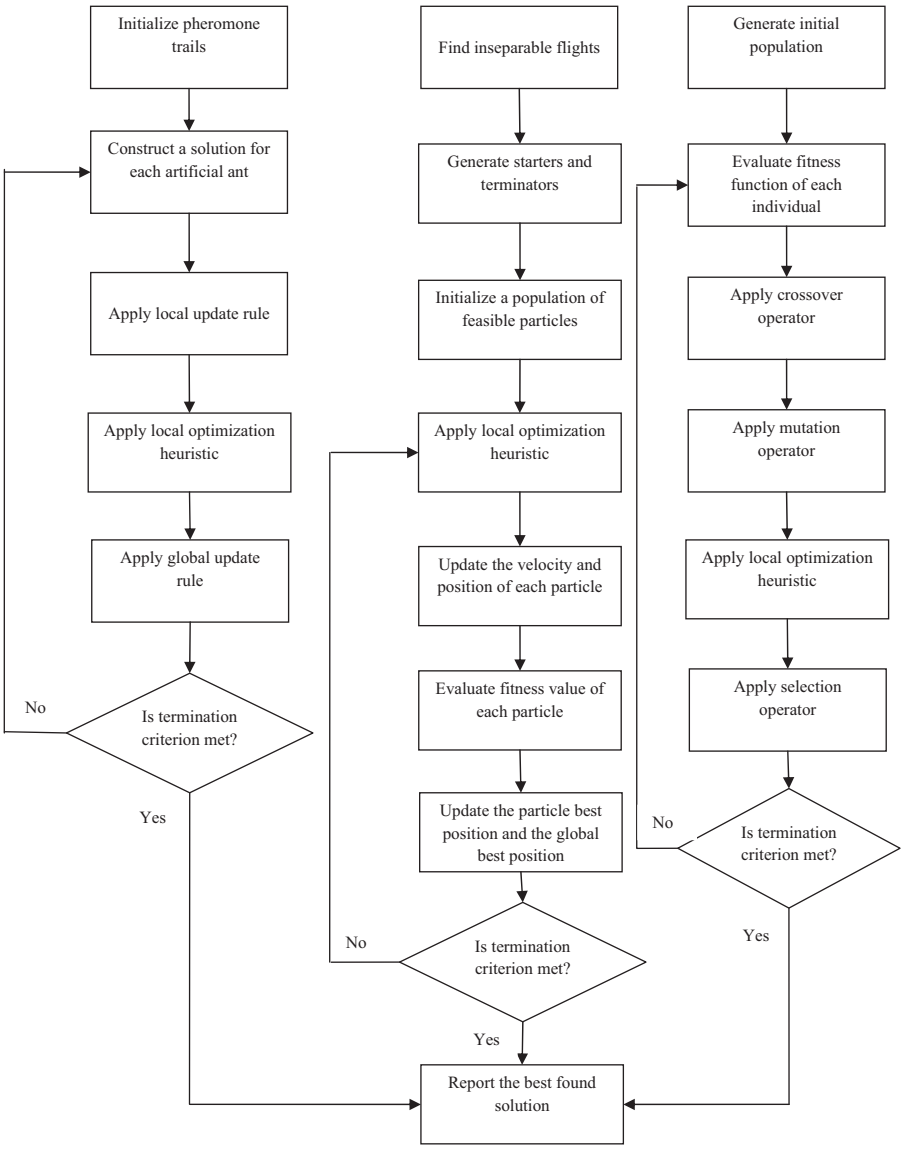

Fig. 2. The flow chart of the proposed hybrid metaheuristic algorithm.

Table 1
Sample initial schedule.

	Flight 0	Flight 1	Flight 2	Flight 3	Flight 4	Flight 5	Flight 6	Flight 7	Flight 8
Flight 0				1		1		1	1
Flight 1			1				1		
Flight 2				1		1		1	1
Flight 3					1				
Flight 4						1		1	1
Flight 5							1		
Flight 6								1	1
Flight 7									
Flight 8									

Table 2
Solutions obtained by applying the stated algorithm in 4.1.1.

	Flight 0	Flight 1	Flight 2	Flight 3	Flight 4	Flight 5	Flight 6	Flight 7	Flight 8
Flight 0				1		1		1	1
Flight 1			1				1		
Flight 2				1		1		1	1
Flight 3					1				
Flight 4						1		1	1
Flight 5							1		
Flight 6								1	1
Flight 7									
Flight 8									

Table 3
The produced random numbers for each crew allocated to every flight.

	Flight 0	Flight 1	Flight 2	Flight 3	Flight 4	Flight 5	Flight 6	Flight 7	Flight 8
Crew 1	0. 69	0. 07	0. 07					0. 55	0. 15
Crew 2	0. 24	0. 54	0. 54					0. 04	0. 39

4.2.1. Inseparable flights

In order to find inseparable flights, the following steps are implemented:

(a) Find a flight which is the destination of just one source. If there exists more than one such flight, then choose one of them randomly. If those flights which are not covered contain cities that are not repeated in other covered flights, there is no feasible solution to the problem.
(b) Change all other 1 values of that source.
(c) If there are any other flight which is the destination of just one source go to step a.

Table 1 shows the sample schedule. Also, Table 2 shows the feasible solutions obtained by applying the stated algorithm in 4.1.1. According to Table 2, a solution would not be feasible if {(1-2), (2-3), (5-6)} are separated.

4.2.2. Starter(s) and Terminator(s)

According to Table 1 a flight is starter if the source is 0 (as a base city) and is terminator if the destination is 0. Thus, we have Starters = {0, 1}, and Terminators = {7, 8}, but as mentioned before, 1 and 2 are inseparable, therefore, Starters = {0, 1-2}.

4.2.3. Feasible solution

Any pairing in a feasible solution should start with a member of Starters and end with a member of Terminators. To allocate starters and terminators to crews, random numbers are used. Table 3 shows random numbers produced for crews for every flight. For example, consider the allocation of flight 0 where the generated numbers for the first and second crews are 0.69 and 0.24, respectively. Thus, flight 0 will be allocated to Crew 1 as it has the greater value. The same procedure should be repeated for flight 1 (actually flight 1 and flight 2 as they are inseparable flights) as starter and flight 7 and flight 8 as terminators. It is mentioned that terminators should be allocated to the crews that have starters too, i.e., if a crew does not have any starter, it cannot have any terminator either and the selection should be done among remaining crews which have starter. Table 3 shows the allocation of crews to each flight based on the generated random numbers shown in Table 3.

Intermediate flights are those which are neither starter nor terminator. In this example, they are 3, 4, 5, and 6 (Table 4). According to Table 2, flight 3 is destination of both flights 0 and 2, hence, it should be allocated to the crew that covers flight 0 or flight 2. Therefore, one of them is selected randomly, for example flight 0, and then flight 3 is allocated to crew 1. What we should remember is that, as flights 3 and 4 are inseparable, they should be allocated to the same crew, i.e., crew 1 covers both flights 3 and 4. This procedure repeats for flights 5 and 6 too.

Table 4
The allocation of crews to flights.

	Flight 0	Flight 1	Flight 2	Flight 3	Flight 4	Flight 5	Flight 6	Flight 7	Flight 8
Crew 1	1							1	
Crew 2		1	1						1

Table 5
The obtained initial solution.

	Flight 0	Flight 1	Flight 2	Flight 3	Flight 4	Flight 5	Flight 6	Flight 7	Flight 8
Crew 1	1			1	1			1	
Crew 2		1	1			1	1		1

According to Table 5, one initial solution is as follows:

Crew 1: {0, 3, 4, 7}
Crew 2: {1, 2, 5, 6, 8}

4.3. Cost evaluation

The aim of crew scheduling is to find a feasible solution considering legal and regulatory rules which minimizes the total crew costs including the flight payments, the rest expenses and deadhead costs. The flight payments is identified by summing up three terms, including the basic payment for each pairing, the basic minimum payment for each flight, and the bonus for each extra minute of flight if the flight time exceeds the predefined minimum flight time. The rest expenses include the payment for each minute of rest time and the hotel expenses. If the sit time is less than or equal to the maximum allowed sit time, the rest expenses are determined by multiplying the sit time and the payment for each minute of rest time. Otherwise, the crew members must be sent to rest in hotel and therefore, the hotel expenses must also be included. As mentioned before, if the city of arrival of the last flight of a pairing is not the base city of that pairing, the crew members should be taken to the base city as ordinary passengers by a deadhead flight and hence, a deadhead cost must be paid.

4.4. Local search heuristic

A local optimization heuristic is proposed to search for improved solutions in some areas of the search space that are most likely to have better solutions. The proposed procedure is repeatedly applied to each solution generated by PSO and helps the algorithm find solutions with high quality in earlier steps. This local search checks if any subtle change in solution improves the total crew costs and when such improvement occurs the change is recorded. The cycle of the procedure is started all over again until no further improvement is recorded. The stepwise procedure is as follows:

1. Starting from the first flight in the priority list, set $i \leftarrow 1$
2. ***For*** each flight j ($j \neq i$) that its departure city is same as the arrival city of flight i ***do***
   Insert the flight j immediately after the flight i in the priority list
   Evaluate the total crew cost
   If the total crew cost is improved, record the new priority list
3. ***If*** i is less than the total number of flights, set $i \leftarrow i+1$ and got Step 2.
4. ***For*** each flight i ***do***
   For each flight j ($j \neq i$) ***do***
   Exchange the priorities of flights i and j
   Evaluate the total crew cost
   If the total crew cost is improved, record the new priority list
5. ***If*** any improvement has been recorded, go to Step 1, ***Else*** stop.

5. Computational results

The proposed algorithm has been coded in MATLAB and run on a 2.00 GHz PC with 2.00 GB of RAM. After tuning the parameters of the algorithm, the computational experiments are carried out in two main parts. First, the contribution of the proposed local search is investigated and then, the proposed PSO is compared to other well-known metaheuristic algorithms. In order to evaluate the efficiency of the proposed algorithm, it has been run on 20 randomly generated problems consisting 25, 50, 100 and 150 flight legs. A random generation procedure has been implemented to generate the problem instances. The problems have been considered as daily crew pairing problems and hence, the 150 flight legs problem is a reasonable size. Moreover, it is noted that in an airline that uses various types of aircrafts, the crew scheduling problem should be solved for each aircraft type independently.

5.1. Parameters setting

There are some parameters in the algorithm that need to be set including swarm size, number of iterations, c_1, and c_2. In order to set the parameters, some preliminary experiments have been conducted on some randomly generated problems of different sizes. Different values for the parameters have been tested in the experiments. Swarm size has been tested between 10 and 70 in increments of 20. Number of iterations has been tested between 10 and 30 in increments of 5. c_1 and c_2 have been tested between 0.1 and 0.9 in increments of 0.2. Several combinations of the parameter values have been tested on randomly generated problems of different sizes and the following parameter values have been found superior: swarm size 30, number of iterations 20, $c_1 = 0.7$ and $c_2 = 0.3$.

5.2. Contribution of the local search heuristic

In order to evaluate the effectiveness of the proposed local optimization heuristic, an experimental test has been conducted. Hence, the algorithm with (HPSO) and without (PSO) local search heuristic has been run on the large sized problem instances. HPSO and PSO have been run on the test problems for same amount of time and the best and the average of the crew costs obtained by each algorithm over 10 runs on each instance have been recorded. Table 6 shows the computational results. As can be seen, the results obtained by HPSO are remarkably superior to PSO. Therefore, the proposed local search heuristic has an important effect on the quality of the solutions.

Table 6
Computational results with (HPSO) and without (PSO) local search heuristic.

Problem	PSO		HPSO	
	Average	Best	Average	Best
15-100-1	63,652.3	63,584	63,591.2	63,492
15-100-2	62,556.4	62,441	61,949.7	61,833
15-100-3	64,689.0	64,612	64,023.6	63,896
15-100-4	60,793.1	60,422	60,472.5	60,261
15-100-5	56,697.8	56,680	56,760.9	56,530
20-150-1	87,119.7	86,977	86,998.2	86,775
20-150-2	88,172.7	88,009	88,016.5	87,887
20-150-3	88,913.4	86,215	86,461.6	86,112
20-150-4	89,136.5	89,043	89,126.4	88,908
20-150-5	87,426.3	87,283	87,373.8	87,145

Table 7
Comparison of HPSO and other metaheuristic algorithms.

Problem	HGA			ACO			HPSO		
	Best	Average	t (s)	Best	Average	t (s)	Best	Average	t (s)
5_25_1	12,864	12,952.0	4.01	12,864	12,864.0	4.48	12,864	12,864.0	4.35
5_25_2	14,771	14,856.0	4.32	14,771	14,816.0	4.83	14,771	14,771.0	4.78
5_25_3	14,863	14,897.5	4.19	14,863	14,863.6	4.89	14,863	14,863.6	4.29
5_25_4	15,394	15,474.8	4.14	15,394	15,443.3	4.33	15,394	15,436.1	4.47
5_25_5	15,877	15,893.6	4.32	15,877	15,878.5	4.73	15,877	15,881.9	4.74
Av. gap (%)	0.000	0.367		0.000	0.070		0.000	0.004	
10_50_1	32,317	32,342.0	26.71	32,317	32,357.0	26.24	32,317	32,346.3	23.88
10_50_2	31,025	31,033.5	20.98	31,025	31,038.9	25.26	31,025	31,040.2	18.19
10_50_3	29,186	29,315.6	20.39	29,186	29,204.6	23.47	29,186	29,201.1	24.79
10_50_4	32322	32,322.0	21.20	32,322	32,322.0	25.56	32,322	32,322.0	24.43
10_50_5	28,802	28,962.0	20.94	28,802	28,853.2	25.61	28,802	28,811.2	24.87
Av. gap (%)	0.000	0.183		0.000	0.044		0.000	0.007	
15_100_1	63,477	63,802.8	118.34	63,492	63,710.3	125.03	63,492	63,591.2	115.63
15_100_2	61,806	62,172.3	129.89	61,908	62,097.6	124.89	61,833	61,949.7	119.95
15_100_3	63,806	64,226.9	133.69	63,937	64,162.9	131.91	63,896	64,023.6	126.01
15_100_4	60,447	60,975.7	147.93	60,491	60,835.3	129.32	60,261	60,472.5	117.76
15_100_5	56,536	57,080.0	127.57	56,576	56,990.4	103.13	56,530	56,760.9	103.35
Av. gap (%)	0.064	0.481		0.171	0.330		0.042	0.000	
20_150_1	86,838	87,933.9	164.67	87,009	88,078.4	137.73	86,775	86,998.2	125.71
20_150_2	87,994	88,885.7	159.98	88,084	88,576.7	153.18	87,887	88,016.5	140.88
20_150_3	86,361	88,930.7	146.37	86,615	89,914.6	141.69	86,112	86,461.6	132.50
20_150_4	89024	89,679.1	159.62	89,052	89,502.2	142.74	88,908	89,126.4	134.70
20_150_5	87,215	87,754.5	148.16	87,241	87,614.0	135.19	87,145	87,373.8	128.90
Av. gap (%)	0.139	1.195		0.270	1.314		0.000	0.000	

5.3. Comparison with other algorithms

To evaluate the performance of the proposed HPSO algorithm, it has been compared with other metaheuristic algorithms hybridized with the proposed local search heuristic: hybrid genetic algorithm (HGA) and ACO algorithm. The experiments have been conducted on 20 randomly generated problems of different sizes. Each problem has been solved by each of the algorithms 10 times and then, the best, the average crew costs and the computation time have been reported. Furthermore, for each problem set, the average gaps have been determined by averaging solution gaps measured by the mean percentage difference from the best solution found by the three algorithms. Data related to the problem instances and the obtained results are shown in Table 7.

According to Table 7, the means of average gaps obtained by HPSO are remarkably smaller than those of the other algorithms. The average gaps of the best solutions found by the three algorithms are zero for the small-sized test problems (i.e. 5_25_* and 10_50_*). As the size of the problems increases the superiority of PSO over HGA and ACO is more distinguishable. PSO is also superior to other algorithms in terms of the average gaps of the average solutions which show the robustness of PSO relative to HGA and ACO. To assess the significance of the differences between the average solutions obtained by the three algorithms, an analysis of variance (ANOVA) is conducted. It has been tested whether the null hypothesis H_0: $\mu_1 = \mu_2 = \mu_3$ is not rejected. It has been concluded that the three treatments differ at significance level $\alpha = 0.05$. In order to compare the pairs of treatment means, least significance difference (LSD) method has been applied. Results of LSD show that at $\alpha = 0.05$, there is no difference between treatments 1 and 2 (i.e. HGA and ACO) but $\mu_3 < \mu_1$ and $\mu_3 < \mu_2$ and hence, the results obtained by HPSO are significantly better than those of HGA and ACO in terms of average solutions.

6. Conclusion

In this paper, a PSO-based algorithm hybridized with a local search heuristic has been presented for solving the crew scheduling problem. The objective of the algorithm is to minimize total crew costs while regulatory and legal restrictions are satisfied. Computational results show the effectiveness of the proposed local search heuristic. Recent studies use GA or ACO to solve large scale crew scheduling problems. Furthermore, two other hybrid algorithms based on GA and ACO algorithms have been developed to solve the problem. Results on problem instances of different sizes show the effectiveness and robustness of the proposed hybrid PSO algorithm.

Acknowledgement

The authors are grateful for the valuable comments and suggestion from the respected reviewers. Their valuable comments and suggestions have enhanced the strength and significance of our paper. This study was supported by a grant from University of Tehran (Grant No. 8106013/1/04). The authors are grateful for the support provided by the College of Engineering, University of Tehran, Iran.

References

[1] S. Yan, Y.P. Tu, A network model for airline cabin crew scheduling, European Journal of Operational Research 140 (2002) 531–540.
[2] E. Andersson, E. Housos, N. Kohl, D. Wedelin, Crew pairing optimization, International Series in Operations Research and Management Science (1998) 228–258.
[3] B. Crawford, C. Castro, E. Monfroy, A hybrid ant algorithm for the airline crew pairing problem, Lecture Notes in Computer Science 4293 (2006) 381–391.
[4] J. Rubin, A technique for the solution of massive set covering problems, with application to airline crew scheduling, Transplantation Science 7 (1973) 34–48.
[5] R. Gerbracht, A new algorithm for very large crew pairing problems, in: 18th AGIFORS Symposium Proceedings, Vancouver, Canada, 1978, pp. 315–341.
[6] S. Lavoie, M. Minoux, E. Odier, A new approach for crew pairing problems by column generation with an application to air transportation, European Journal of Operational Research 35 (1988) 45–58.
[7] R. Anbil, others, Recent advances in crew-pairing optimization at American Airlines, Interfaces 21 (1991) 62–74.
[8] H.D. Chu, E. Gelman, E.L. Johnson, Solving large scale crew scheduling problems, European Journal of Operational Research 97 (1997) 260–268.
[9] C. Barnhart, R.G. Shenoi, An approximate model and solution approach for the long-haul crew pairing problem, Transplantation Science 32 (1998) 221.
[10] M. Stojkovic, F. Soumis, J. Desrosiers, The operational airline crew scheduling problem, Transplantation Science 32 (1998) 232–245.
[11] K.C. Wang, Applications of constraint programming and mathematical programming techniques to solve large-scale cabin crew scheduling problems, Chiao Tung University, M.S. Thesis, 2002.

[12] S. Yan, J.C. Chang, Airline cockpit crew scheduling, European Journal of Operational Research 136 (2002) 501–511.
[13] L. Bodin, B. Golden, A. Assad, M. Ball, Routing and scheduling of vehicles and crews: the state of the art, Computers & Operations Research 10 (1983) 63–211.
[14] M. Ball, L. Bodin, R. Dial, A matching based heuristic for scheduling mass transit crews and vehicles, Transplantation Science 17 (1983) 4–31.
[15] J.E. Beasley, B. Cao, A dynamic programming based algorithm for the crew scheduling problem, Computers & Operations Research 25 (1998) 567–582.
[16] N. Balakrishnan, R.T. Wong, A network model for the rotating workforce scheduling problem, Networks 20 (1990) 25–42.
[17] H.T. Ozdemir, C.K. Mohan, Flight graph based genetic algorithm for crew scheduling in airlines, Information Science 133 (2001) 165–173.
[18] C.C. Lo, G.F. Deng, Using ant colony optimization algorithm to solve airline crew scheduling problems, in: Third International Conference on Natural Computation, 2007 (ICNC 2007), 2007.
[19] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.
[20] X.H. Shi, Y.C. Liang, H.P. Lee, C. Lu, Q.X. Wang, Particle swarm optimization-based algorithms for TSP and generalized TSP, Information Processing Letters 103 (2007) 169–176.
[21] Y. Marinakis, M. Marinaki, A hybrid multi-swarm particle swarm optimization algorithm for the probabilistic traveling salesman problem, Computers & Operations Research 37 (2010) 432–442.
[22] T.J. Ai, V. Kachitvichyanukul, A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery, Computers & Operations Research 36 (2009) 1693–1702.
[23] F.P. Goksal, I. Karaoglan, F. Altiparmak, A hybrid discrete particle swarm optimization for vehicle routing problem with simultaneous pickup and delivery, Computers & Industrial Engineering (2012), http://dx.doi.org/10.1016/j.cie.2012.01.005.
[24] H. Rezazadeh, R. Mahini, M. Zarei, Solving a dynamic virtual cell formation problem by linear programming embedded particle swarm optimization algorithm, Applied Soft Computing 11 (2011) 3160–3169.
[25] L.-Y. Chuang, C.-H. Yang, J.-C. Li, Chaotic maps based on binary particle swarm optimization for feature selection, Applied Soft Computing 11 (2011) 239–248.
[26] A.W.N. Mohemmed, C.T. Sahoo, K. Geok, Solving shortest path problem using particle swarm optimization, Applied Soft Computing 8 (2008) 1643–1653.
[27] D. Anghinolfi, M. Paolucci, A new discrete particle swarm optimization approach for the single-machine total weighted tardiness scheduling problem with sequence-dependent setup times, European Journal of Operational Research 193 (2009) 73–85.
[28] A. Husseinzadeh Kashan, B. Karimi, A discrete particle swarm optimization algorithm for scheduling parallel machines, Computers & Industrial Engineering 56 (2009) 216–223.
[29] C.J. Liao, C.T. Tseng, P. Luarn, A discrete version of particle swarm optimization for flow shop scheduling problems, Computers & Operations Research 34 (2007) 3099–3111.
[30] X. Wang, L. Tang, A discrete particle swarm optimization algorithm with self-adaptive diversity control for the permutation flowshop problem with blocking, Applied Soft Computing 12 (2012) 652–662.
[31] C.-J. Liao, E. Tjandradjaja, T.-P. Chung, An approach using particle swarm optimization and bottleneck heuristic to solve hybrid flow shop scheduling problem, Applied Soft Computing (2012), http://dx.doi.org/10.1016/j.asoc.2012.01.011.
[32] D.Y. Sha, C.-Y. Hsu, A hybrid particle swarm optimization for job shop scheduling problem, Computers & Industrial Engineering 51 (2006) 791–808.
[33] G. Moslehi, M. Mahnam, A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search, International Journal of Production Economics 129 (2011) 14–22.
[34] D.Y. Sha, C.-Y. Hsu, A new particle swarm optimization for the open shop scheduling problem, Computers & Operations Research 35 (2008) 3243–3261.
[35] S. Noori-Darvish, I. Mahdavi, N. Mahdavi-Amiri, A bi-objective possibilistic programming model for open shop scheduling problems with sequence-dependent setup times, fuzzy processing times, and fuzzy due dates, Applied Soft Computing 12 (2012) 1399–1416.
[36] D. Klabjan, E.L. Johnson, G.L. Nemhauser, E. Gelman, S. Ramaswamy, Solving large airline crew scheduling problems: random pairing generation and strong branching, Computational Optimization and Applications 20 (2001) 73–91.