Electronic Letters on Science & Engineering 11(2) (2015)
Available online at www.e-lse.org

Optimization of attendee lists by using metaheuristics: A case study on symposium planning in a college

Hasan Makas[1,*], Funda Makas[2],Nejat Yumusak[3]
[1]Ereğli Iron & Steel Works Co., 67330, Karadeniz Ereğli, Zonguldak, Turkey
[2]TED Kdz. Eregli College, 67300, Kdz. Eregli, Zonguldak, Turkey
[3]Sakarya University, Faculty of Computer and Information Sciences, Department of Computer Engineering, 54187, Serdivan, Sakarya, Turkey

Abstract: Combinatorial optimization problems are drawing more attention and rapidly developing research field ranging from industrial world to educational applications. Discrete forms of some metaheuristics can generate optimal solutions to such kind of problems. Case study reported in this paper uses discrete versions of the migrating birds optimization algorithm, the artificial bee colony algorithm and the genetic algorithm to solve given symposium session optimization problem. Optimal combinations of the attendee lists for symposium sessions were created by using these three algorithms. Optimized attendee lists provided balanced distributions of attendee to the sessions. Thus, large saloon requirements of symposium planners were minimized. Then, scope of the problem was extended hypothetically, and proposed methods were employed one more time in order to measure their performances on the extended problem. The results show that metaheuristic algorithms used in this paper can achieve good combinatorial optimizations on symposium session optimization problem even if big increases occur in problem dimension.

Keywords: Metaheuristic planning; symposium session optimization; migrating birds optimization algorithm; artificial bee colony algorithm; genetic algorithm

Özet: Kombinasyonel optimizasyon problemleri fazlaca dikkat çeken ve endüstriyel dünyadan eğitim uygulamalarına kadar uzanan hızla gelişen bir araştırma sahasını oluşturmaktadır. Bazı meta-sezgisellerin ayrık formları bu türden problemler için optimal çözümler üretebilmektedirler. Bu makalede verilen vaka çalışmasında, sempozyum oturumlarının optimize edilmesi için göçmen kuşlar optimizasyon algoritması, yapay arı koloni algoritması ve genetik algoritmanın ayrık versiyonlarını kullanılmıştır. Her üç algoritma kullanılarak sempozyum oturumları için optimal katılımcı listesi kombinasyonları oluşturulmuştur. Katılımcı listelerinin optimize edilmesi katılımcıların oturumlara dengeli dağılımlarını sağlamıştır. Böylece sempozyum organizatörlerinin büyük salon gereksinimleri minimize edilmiştir. Daha sonra, problem kapsamı farazi olarak genişletilmiş ve önerilen yöntemler genişletilmiş problemler üzerinde denenerek yöntemlerin performansları bir kes daha test edilmiştir. Sonuçlar, problem boyutunda büyük artışlar olsa bile, bu makalede verilen meta-sezgisel algoritmaların sempozyum oturum optimizasyonu problemi için iyi performans sergileyeceğini ortaya koymuştur.

Anahtar Kelimeler
Meta-sezgisel planlama; sempozyum oturum optimizasyonu; göçmen kuşlar optimizasyon algoritması; yapay arı koloni algoritması; genetik algoritma

1. Introduction

Research area of combinatorial optimization is at the intersection of applied mathematics, computer science and operations research. An integer, a subset, a permutation or a graph structure is thought to be an object in combinatorial optimization problems, and this object is searched from a finite set[1]. Combinatorial optimization problems generally represent some relations between a set of variables and a decision satisfying the constraints. Some of well-known combinatorial optimization problems are travelling salesman problem, quadratic

* *Corresponding author;* e-mail : hasanmakas@gmail.com

ISSN 1305-8614

assignment problems, time tabling problems, scheduling problems, flow network problems and vehicle routing problems.

Although problem specific complete methods give optimal solutions, they need exponential computation time for the worst problem case. So, the metaheuristics, which are approximate methods, have been getting more and more attention for last 20 years in order to solve combinatorial optimization problems[2]. They can be defined as iterative methods which mimic exploitation and exploration behaviours of the agents. Most of them are inspired by the nature and they are not problem specific. They are generally neighbourhood search methods, and constitute a large class among improvement algorithms. They explore problem space globally and search neighbourhoods of the existing solutions locally to get new better solutions.

Some of the most popular and recent metaheuristic algorithms that have been introduced by researchers so far are the Genetic Algorithm (GA) [3], the Simulated Annealing (SA) algorithm[4] the Tabu Search (TS) algorithm[5], the Ant Colony Optimisation (ACO) algorithm[6], the Particle Swarm Optimization (PSO) algorithm[7], the Differential Evolution (DE) algorithm[8], the Harmony Search (HS) algorithm[9], the Artificial Bee Colony (ABC) algorithm[10], the Monkey Search (MS) algorithm[11], the Firefly Algorithm (FA)[12], the Intelligent Water Drops (IWD) algorithm[13], the Cuckoo Search (CS) algorithm[14-15], the Bat Algorithm (BA)[16-17] and the Migrating Birds Optimization (MBO) algorithm[18].

The metaheuristic algorithms are designed to solve a wide range of optimization problems. They are generally applied to the problems for which there is no specific method to solve. In our case study, considering that preparation of attendee lists for the sessions of a symposium is an optimisation problem, we thought that combinatorial metaheuristics can be employed to get well balanced lists. Description of the case study is as follows.

Counselling and guidance department members of TED Kdz Eregli College in Turkey organize a traditional symposium on professions every year. They invite prominent and successful people representing several professions. Invited speakers make presentations on their professions and give detailed information to the students about opportunities of their job. Progress of the symposium is realized in parallel sessions. Number of parallel sessions is equal to the number of invited speakers and each profession presentation is repeated three times. That is, an invited speaker makes his/her presentation in the first session; and then, he/she repeats the same presentation in second and third sessions. Therefore, among the different profession presentations, each student has a chance to attend to three of them.

At the beginning, organizers make a survey to the students to find out their choices. Students select three of the professions which are listed in a questionnaire form. Then, the organizers create attendee lists for each session of the presentations. They aim to obtain balance between sessions of a presentation in order to reduce the number of large saloon requirements. Obviously, if total number of student choices to a profession is achieved to be partitioned into three equal parts, then it is assumed that the partitioning is performed with maximum success. In this way, large saloon requirements can be minimized. Here is the problem presents itself at this point.

Organizers were trying to achieve balanced student distributions into the sessions by writing down the choices of each student to the session attendee lists one by one and manually. Therefore, they were trying a great many combinations by making many revisions on the lists. This process was progressing in a laborious and frustrating manner, and it was taking too long time. After organizers finished the preparation of session attendee lists, it was being seen that the student distributions in lists were not balanced well as desired. For instance, there were many students in the first session of law profession presentation, but there was a few in its second and third sessions. Because of excessive students in the first session, a large saloon must be reserved for the presentations of this profession. Because the number of large saloons was limited, revisions on session lists were repeated again and again to prevent from such cases. It should be

noted that revisions were done a little bit by randomly and by intuitions. On the other hand, some of invited speakers declared that they could attend to only first two sessions because of their busyness. Taking such kind of restrictions into account, students who selected these professions were being directed to presentations of these professions in the first two sessions, and this operation was causing unbalanced distributions in session attendee lists. At the end, unfortunately, imbalances in lists were taking a lot of criticisms from attendees despite all these efforts.

In this study, we proposed discrete versions of the MBO algorithm, ABC algorithm and GA for symposium session optimization problem (SSOP). The paper is organized as follows. Section 2 explains SSOP by deriving mathematical definitions, gives theoretical backgrounds of conventional MBO algorithm, ABC algorithm and GA and explains the differences in proposed versions of them. Section 3 shows the results of proposed planning method and discusses them. Section 4 concludes the whole study and recommends a possible future work.

2. Theoretical Backgrounds

2.1. Definition of SSOP

The problem explained in previous section can be described as follows. Considering that identification (ID) numbers from 1 to *np* are given to presentations, total number of attendees for the sessions can be represented by a matrix in the form of

$$S = \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1n_p} \\ s_{21} & s_{22} & \cdots & s_{2n_p} \\ s_{31} & s_{32} & \cdots & s_{3n_p} \end{bmatrix} \tag{1}$$

where S is the session distribution matrix; s_{ij} is the total number of attendees in i^{th} session of the j^{th} presentation and n_p is the total number of presentations. Each element in S matrix is calculated after completing the partitioning of students according to their presentation choices. Sequences of presentations assigned to students can be represented by a matrix in the form of

$$C = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n_s} \\ c_{21} & c_{22} & \cdots & c_{2n_s} \\ c_{31} & c_{32} & \cdots & c_{3n_s} \end{bmatrix} \tag{2}$$

where C is the presentation sequence matrix; c_{ij} is ID number of the presentation representing that j^{th} student will attend to i^{th} session of this presentation and n_s is the total number of students who will attend to symposium. That is, each column in C matrix represents presentation IDs of a student in a sequence, and the corresponding student attends to presentations according to this sequence. Initial presentation sequence of j^{th} student is determined randomly considering student's choices and session availability constraints given by

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n_p} \\ a_{21} & a_{22} & \cdots & a_{2n_p} \\ a_{31} & a_{32} & \cdots & a_{3n_p} \end{bmatrix} \tag{3}$$

where A is session availability constraint matrix; a_{ij} is availability of the i^{th} session of j^{th} presentation and n_p is the total number of presentations. Matrix A consists of zeros and ones. If

an element in matrix A is 1, it means that the corresponding session exists. Otherwise, the corresponding session does not exist. Desired mean attendee value for the sessions of each presentation can be represented as

$$Mean = \begin{bmatrix} mean_1 & mean_2 & \cdots & mean_{n_p} \end{bmatrix} \tag{4}$$

where $Mean$ is mean attendee matrix; $mean_i$ represents the mean attendee value for the sessions of i^{th} presentation and n_p is the total number of presentations.

As mentioned in previous section, the aim in SSOP is to get a good balanced S matrix. That is, by trying different combinations of C in accordance with constraints in A, it is aimed to get s_{ij} values of S as close as to $mean_j$ in $Mean$. A new session distribution matrix S is obtained for each new combination of presentation sequence matrix C. Mean squared error (MSE) of distribution matrix S can be calculated by

$$MSE(S) = \frac{1}{n_p} \sum_{j=1}^{n_p} \sum_{i=1}^{3} \left(s_{ij} - mean_j \right)^2 \tag{5}$$

where s_{ij} is the total number of attendees in i^{th} session of j^{th} presentation; $mean_j$ represents desired mean attendee value of j^{th} presentation and n_p is the total number of presentations. This calculated MSE value can be used as the cost value of the corresponding C combination. To sum up, the problem is to minimize the MSE to get an optimal distribution in S depending on combinations in C and constraints in A.

2.2. Conventional ABC algorithm and its proposed form for SSOP

The ABC algorithm is a swarm intelligence based metaheuristic algorithm. It is inspired by foraging behaviours of bees. In the algorithm, position of a food source represents a possible solution of optimization problem and nectar amount of a food source corresponds to the quality or fitness of that solution. Both total number of employed bees and that of onlooker bees are equal to that of food sources[19]. Therefore, total number of food sources is equal to half of the colony size.

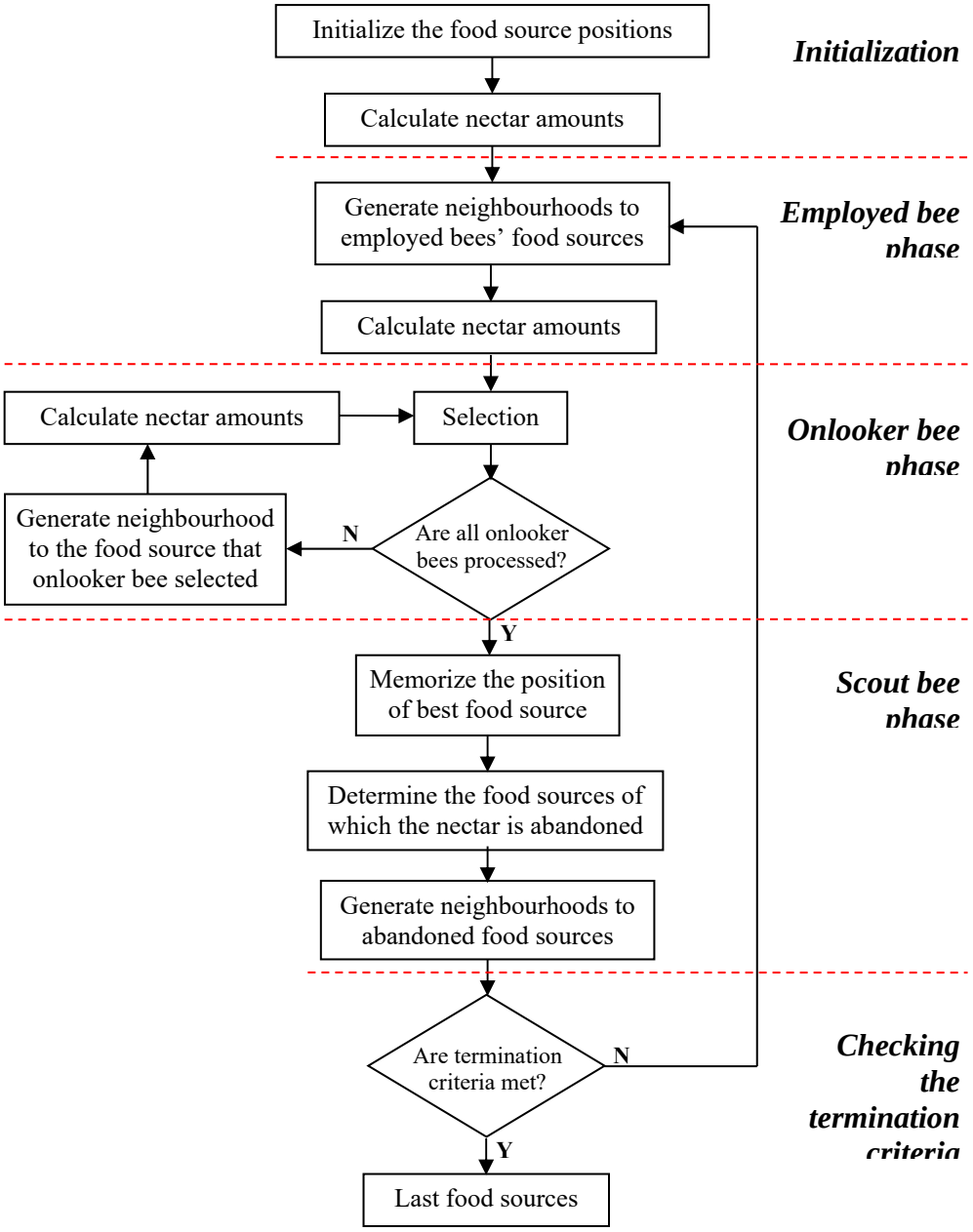

Figure 1. Flow diagram of conventional ABC algorithm

There are three phases in calculations of each iteration as seen in Figure 1[23]. These are employed bee, onlooker bee and scout bee phases. Considering real valued numerical problems, algorithm generates randomly distributed initial solutions to the food sources at the beginning by using

$$x_{ij} = x_j^{min} + \text{rand} \cdot \left(x_j^{max} - x_j^{min} \right) \tag{6}$$

where x_{ij} is the position of i^{th} solution in j^{th} dimension; x_j^{min} and x_j^{max} are the minimum and maximum limit values for j^{th} dimension and rand is a uniform random number ranging from 0 to 1. After completing the food source initialization, algorithm phases start to run until termination criteria are met. The first phase is the employed bee phase in which employed bees try to enhance their solutions by generating new neighbours via

$$v_{ij} = x_{ij} + \phi \cdot \left(x_{ij} - x_{kj} \right) \tag{7}$$

where x_{ij} is the position of i^{th} solution in j^{th} dimension; k is randomly selected index value different from i; x_{kj} is the position of k^{th} solution in j^{th} dimension; ϕ is a random number ranging from -1 to 1 and v_{ij} is generated new neighbourhood position in j^{th} dimension for i^{th} solution. Then, a greedy selection mechanism which is only based on the fitness values of food sources is used to make a selection between existing food sources and new generated neighbours. Fitness value of a source is

$$Fitness_i = \begin{cases} \dfrac{1}{\left(1 + f_i\right)} & , \quad f_i \geq 0 \\ 1 + \text{abs}\left(f_i\right) & , \quad f_i < 0 \end{cases} \tag{8}$$

where f_i and $Fitness_i$ are cost and fitness values of i^{th} source respectively. After performing the greedy selection operations, selection probabilities of updated sources by onlooker bees are calculated via

$$p_i = \frac{Fitness_i}{\sum_{j=1}^{SN} Fitness_j} \tag{9}$$

where p_i is the selection probability of i^{th} source by onlooker bees and SN is the total number of food sources.

The second phase is onlooker bee phase in which onlooker bees make their selections according to the corresponding p_i probabilities. That is, the higher the selection probability is, the more chance the corresponding source can be selected and enhanced by onlooker bees. Each onlooker bee generates a neighbourhood to its selection via Equation (7). Then, it makes a greedy selection between selected source and new generated source in order to enhance the source selected.

The third phase is the scout bee phase in which new food sources are generated by scouts via Equation (6) for the sources whose nectar is abandoned. In implementations, a failure counter is assigned to each source. After completing a neighbourhood creation for the corresponding food source in employed bee or onlooker bee phases, this counter is increased by one if no improvement is achieved. Otherwise, it is set to zero. Whenever failure count of a source exceeds a predefined limit value, employed bee of the corresponding food source is transformed into a scout bee to find a new food source by making global search via Equation (6). Scout bees are re-transformed into employed bees after they generate new food sources.

Combinatorial problems need discrete algorithms or discrete versions of numerical algorithms. We proposed a discrete ABC algorithm for SSOP solution. The main differences in our discrete ABC algorithm are as follows,

- Discrete ABC algorithm initializes presentation sequence matrix C using random sequence permutations of students' choices instead of Equation (6). By taking matrix A into account during initialization, each solution set (matrix C) is prevented from conflicting with session availability (matric A).

- Instead of Equation (7), discrete ABC algorithm creates neighbourhoods by selecting a student randomly (a randomly selected column in C) and by randomly changing

presentation sequence of the student selected. Similar to previous item, constraints in matrix *A* is taken into account during neighbourhood creation.

- Discrete ABC algorithm uses Equation (5) to calculate cost of each solution. Substituting these cost values in Equation (8), discrete ABC algorithm calculates the fitness values.

2.3. Conventional MBO algorithm and its proposed form for SSOP

The MBO algorithm which is a nature inspired metaheuristic neighbourhood search approach was introduced recently. It simulates V flight formation of migrating birds. With induced drag reduction, V shaped flight style in Figure 2 is an effective formation for birds to save the energy[18]. For instance, in a V shaped flight formation consisting of 25 members, each bird can achieve a reduction in induced drag as large as 65%. This results in a flight range increase about 70%[20].

Figure 2. A typical V flight formation of the birds

Benefit mechanism of this formation can be explained briefly as follows. A pair of vortices, which is seen in Figure 3, is created owing to the wing movements. Looking in the direction of flight, vortices from left and right wing tips rotate in clockwise and counter clockwise directions respectively[18-20]Vortices create downwash and upwash forces for the birds flying behind. Downwash is undesirable because it increases the induced drag on a wing in flight. On the other hand, upwash is beneficial because it decreases the induced drag on a wing in flight. So, all the birds in V formation except for the leader bird locate mostly in upwash regions of vortices. Thus, they get benefit of this upwashes and reduce their energy consumption. To sum up, the leader bird in that formation is the one spending the most energy and the birds in other positions get benefit from the birds in front[18].

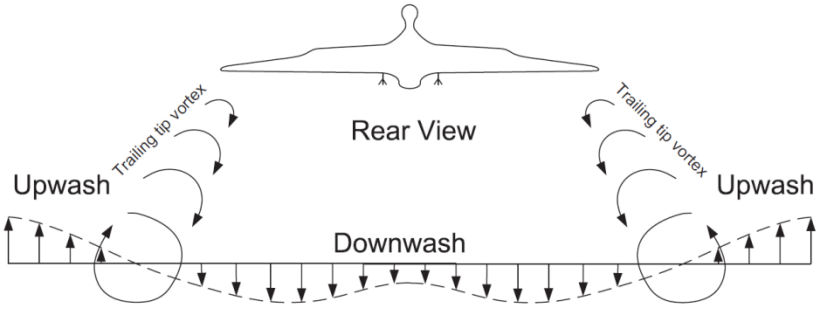

Figure 3. Regions of upwash and downwash created by trailing vortices

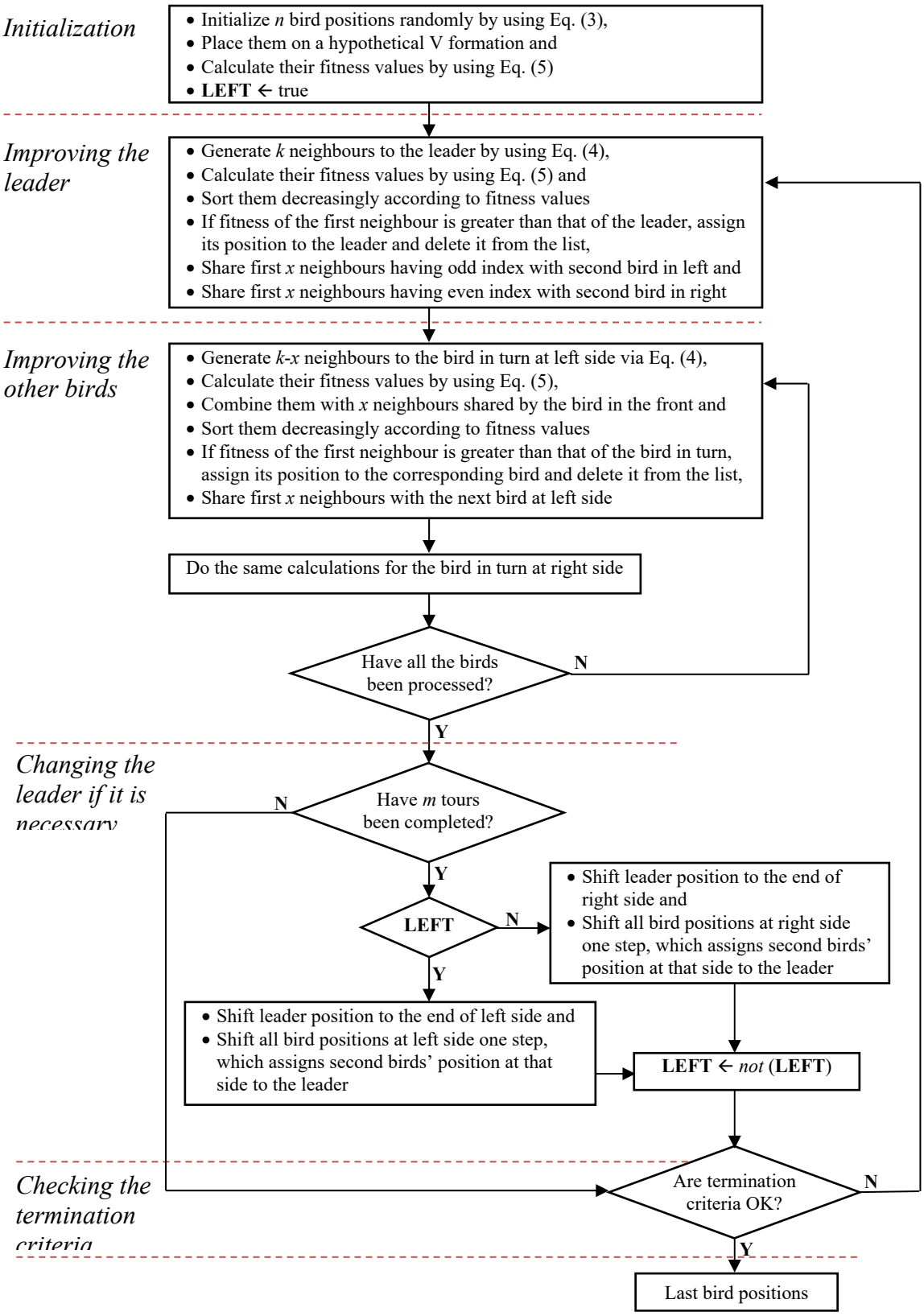

Figure 4. Flow diagram of conventional MBO algorithm

A flow diagram for MBO algorithm is shown in Figure 4[21] and the parameters used in MBO algorithm are given in Table 1[18]

Table 1. Parameters of the MBO algorithm

Parameter	Description
n	Number of solutions (flock/population size)
k	Total number of neighbour solutions to be considered (flight speed^{-1})
x	Number of neighbour solutions to be shared with the next solution (upwash benefit)
m	Number of tours to change the leader
K	Maximum iteration or tour number

Positions of the birds in MBO algorithm represent possible solutions. The algorithm runs as follows. In the first step, the flock is initialized. Equation (6) can be used to initialize positions of the birds. After initialization, one of the solutions is chosen as leader bird and all of the solutions are placed on a hypothetical V formation arbitrarily. In the next steps, algorithm starts with the first solution which corresponds to the leader bird, and progresses on the lines from leader to tails so that the algorithm could improve each solution by using its neighbour solutions[18].

In the second step, the leader bird is tried to be improved. Hence, k neighbours are generated and their fitness values are calculated. If the best neighbour solution shows an improvement on leader, the position of that neighbour solution is assigned to leader solution. Then, $2x$ unused best solutions are shared with 2 birds in second row. The neighbourhood creation method given in Equation (7) can be used to create neighbours.

In the third step, other birds are tried to be improved. For each bird in turn, (k-x) neighbours are generated and their fitness values are calculated. These neighbours are combined with x unused neighbours coming from the birds in front. So, the total number of neighbour solutions to be considered for the corresponding bird is k like in the leader bird. If the best neighbour solution among them shows an improvement on the corresponding bird, position of this neighbour solution is assigned to the corresponding bird. Then, x unused best solutions are shared with the next bird. This neighbourhood sharing mechanism simulates the benefit of upwash caused by trailing tip vortices in V flight formation. One iteration ends after completing improvement trials for all birds.

In the fourth step, the leader bird is thought to be tired after performing a predefined number of iterations (m). So, if predefined number of iterations is reached, then the leader bird is changed. Changing the leader is performed by shifting it to the end of one side on hypothetical V formation and by assigning the second solution at that side to the leader position. Steps from 2 to 4 are repeated until predefined termination criteria are satisfied by means of generated solutions. Then, the algorithm stops and gives the solution having best fitness value as overall solution.

Since combinatorial problems need discrete working disciplines, we used a discrete version of MBO algorithm for the solution of SSOP. For discrete MBO algorithm, we used the same initialization, neighbourhood creation and fitness calculation methods described for discrete ABC algorithm in previous section.

2.4. Conventional GA and its proposed form for SSOP

In GA, the actual solutions which are called phenotypes are represented by chromosomes which are called genotypes. It is essential in GA to design a proper chromosome representation and to determine a proper fitness calculation method. The phenotypes are converted to genotypes before the application of genetic operators and the genotypes are converted to phenotypes before the fitness calculations.

Figure 5 shows a basic GA process. Firstly, initial population is created randomly. Then, fitness of each solution is calculated. If the solution having best fitness value is not met termination criteria, then genetic operators are applied to the chromosomes.

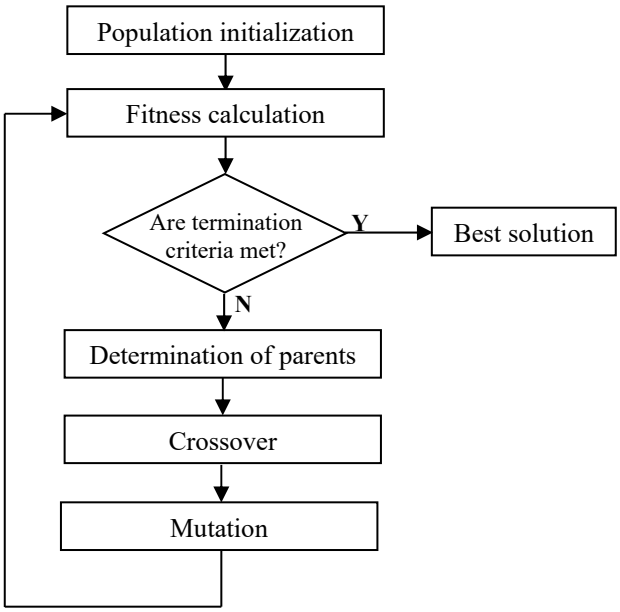

Figure 5. Flow diagram of the a basic GA process

In the application of genetic operators, first step is determination of the parents. There are several GA implementations and they use different methods in application. In our implementation, number of parent pairs is equal to half of the population size and we used roulette wheel method for[22] determination of the parents.

The second genetic operator is the crossover. Most common crossover method for binary coded chromosomes is shown in Figure 6 and called one point crossover. The same crossover point is selected in each parent. The parts delimited by crossover points are interchanged by the parents. Two new offsprings are generated from two parents by this way.

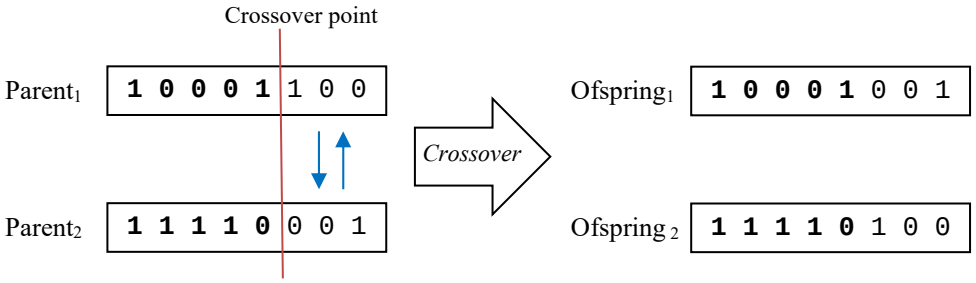

Figure 6. One point crossover for binary coded chromosomes

The third genetic operator is the mutation. For binary coded GA, a predefined percentage of bits in chromosomes are toggled to implement mutation operation.

In the proposed GA, we used the same initialization and fitness calculation methods described for discrete ABC and discrete MBO algorithms in previous two sections. As a combinatorial approach, we used the following crossover method which closely mimics binary coded GA in accordance with SSOP. In this method, a chromosome is thought to be in the form of Equation (2). Let the parents be

$$Parent_1 = \begin{bmatrix} c_{11}^{mom} & c_{12}^{mom} & \cdots & c_{1\alpha}^{mom} & \cdots & c_{1n_s}^{mom} \\ c_{21}^{mom} & c_{22}^{mom} & \cdots & c_{2\alpha}^{mom} & \cdots & c_{2n_s}^{mom} \\ c_{31}^{mom} & c_{32}^{mom} & \cdots & c_{3\alpha}^{mom} & \cdots & c_{3n_s}^{mom} \end{bmatrix} \tag{10}$$

$$Parent_2 = \begin{bmatrix} c_{11}^{dad} & c_{12}^{dad} & \cdots & c_{1\alpha}^{dad} & \cdots & c_{1n_s}^{dad} \\ c_{21}^{dad} & c_{22}^{dad} & \cdots & c_{2\alpha}^{dad} & \cdots & c_{2n_s}^{dad} \\ c_{31}^{dad} & c_{32}^{dad} & \cdots & c_{3\alpha}^{dad} & \cdots & c_{3n_s}^{dad} \end{bmatrix} \tag{11}$$

where α is a randomly produced integer number ranging from 1 to n_s. The new variables which will appear in offsprings are calculated via

$$P_{new\_mom} = \begin{bmatrix} p_{new\_mom1} \\ p_{new\_mom2} \\ p_{new\_mom3} \end{bmatrix} = \text{randc}\left(\begin{bmatrix} c_{1\alpha}^{mom} \\ c_{2\alpha}^{mom} \\ c_{3\alpha}^{mom} \end{bmatrix} \right) \tag{12}$$

$$P_{new\_dad} = \begin{bmatrix} p_{new\_dad1} \\ p_{new\_dad2} \\ p_{new\_dad3} \end{bmatrix} = \text{randc}\left(\begin{bmatrix} c_{1\alpha}^{dad} \\ c_{2\alpha}^{dad} \\ c_{3\alpha}^{dad} \end{bmatrix} \right) \tag{13}$$

where randc is a function that produces a new random combination of 3×1 matrix taking session availability constraint matrix A into account. Then, the offsprings including new variables are generated by replacing $c_{*\alpha}^{mom}$ and $c_{*\alpha}^{dad}$ with P_{new_mom} and P_{new_dad} respectively and by swapping the right side of selected variables in parents as shown in

$$Offspring_1 = \begin{bmatrix} c_{11}^{mom} & c_{12}^{mom} & \cdots & p_{new\_mom1} & \cdots & c_{1n_s}^{dad} \\ c_{21}^{mom} & c_{22}^{mom} & \cdots & p_{new\_mom2} & \cdots & c_{2n_s}^{dad} \\ c_{31}^{mom} & c_{32}^{mom} & \cdots & p_{new\_mom3} & \cdots & c_{3n_s}^{dad} \end{bmatrix} \tag{14}$$

$$Offspring_2 = \begin{bmatrix} c_{11}^{dad} & c_{12}^{dad} & \cdots & p_{new\_dad1} & \cdots & c_{1n_s}^{mom} \\ c_{21}^{dad} & c_{22}^{dad} & \cdots & p_{new\_dad2} & \cdots & c_{2n_s}^{mom} \\ c_{31}^{dad} & c_{32}^{dad} & \cdots & p_{new\_dad3} & \cdots & c_{3n_s}^{mom} \end{bmatrix} \tag{15}$$

We used a combinatorial mutation in mutation step. In this method, a predefined percentage of columns from offsprings are selected randomly, and then new random combinations are produced instead of these selected columns in accordance with the session availability constraint matrix A.

2.5. Applications of the proposed methods to SSOP

We applied the procedure given in Figure 7 to solve the SSOP. This procedure runs as follows. Firstly, the population in which each member represents a possible presentation sequence matrix C is initialized randomly. Then, the main loop starts with the calculation of fitness values for population members. After fitness calculations, one of the metaheuristics

mentioned in this paper is applied to get members of new population, and the member having best fitness value is tried instead of previous best presentation sequence matrix *C*. If the MSE calculated decreases, then the best population member of iteration is assigned to the best presentation sequence matrix *C*. The main loop runs until the termination criteria are met.

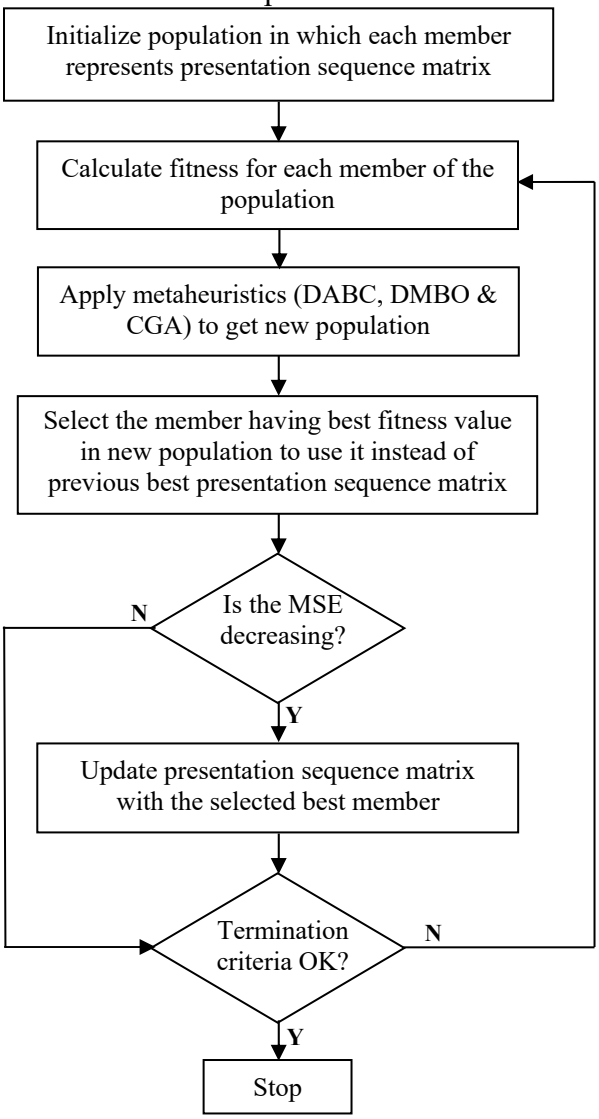

Figure 7. Application of proposed methods for SSOP

3. Equivalent Experimental Setup

In May 2015, 20 speakers were invited to Traditional Professions Symposium of TED Kdz Eregli College by organization committee. Following professions were introduced to the students by speakers.

1. Medicine,
2. Dentistry,
3. Medical Documentation,
4. Biomedical Engineering,
5. Pharmacy,
11. Environmental Engineering,
12. Industrial Engineering,
13. Economy,
14. Gastronomy,
15. Tourism and Hotel Management,

6. Electronics Engineering,
7. Computer Engineering,
8. Metallurgical Engineering,
9. Mechanical Engineering,
10. Civil Engineering,
16. Logistics,
17. Archaeology,
18. Law,
19. Accountancy and
20. Psychology

Firstly, representing these professions by symbols from P1 to P20, we created representative student choices arbitrarily for test purpose. Then, we employed the MBO, ABC and GA to get optimal attendee lists from representative data. This was an equivalent experiment which mimics the SSOP in our case study. Firstly, we did this experiment without any session restriction. That is, all of the presentations had three session; in other words, all members of matrix A were 1. Then, we did the experiment two more time by adding session restrictions.

After completing equivalent experiments of the case study, we expanded the problem dimensions by increasing the number of students and the number of professions in order to check the performances of proposed methods. The results for these cases are given and discussed in next section.

4. Results

3.1. Results for the Equivalent Experiments of Case Study

We set the parameters given in Table 2 for the algorithms. Main motivation of the settings was to get totally equal calculations. In the MBO algorithm, k calculations are performed for the leader bird and (k-x) calculations are performed for the others in one cycle. In the ABC algorithm and GA, n calculations are performed totally in one cycle. Considering this reality, the total number of calculations for all algorithms, corresponding to the parameters in Table 2, is 5500. The MSE value was reached as 0.3 for all algorithms. We performed the calculations 10 times for each algorithm and got the same MSE value for each execution.

Table 2. Parameters of the algorithms and their calculation costs

		MBO	ABC	GA
Parameters	Iteration (N)	500	550	550
	Population size (n)	5	10	10
	Generated neighbourhood (k)	3	-	-
	Shared neighbourhood (x)	1	-	-
	Iteration to change the leader (m)	10	-	-
	Mutation rate (%)	-	-	0.1
Calculation costs	Number of total calculations	**5500**	**5500**	**5500**
	MSE	**0.3000**	**0.3000**	**0.3000**

After a successful execution of each algorithm in no restriction case, calculated total numbers of attendees for each session of each profession presentation are given in Table 3. It is

clearly seen that the numbers of attendees in each session of a presentation are nearly equal. So, the algorithms achieve to get well balanced session distributions. MSE trends of the algorithms are shown by the graphs in Figure 8. They show that predefined maximum number of iterations are enough for the case study.

Table 3. Session distributions for no restriction case (P: Presentation, S: Sessions)

	S	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20
MBO	**1**	28	20	20	22	19	12	19	14	20	19	17	14	18	14	16	15	12	15	17	16
	2	27	20	20	22	19	11	20	15	20	19	17	14	19	13	16	15	12	15	17	16
	3	27	21	20	22	19	11	20	14	20	19	17	14	18	14	15	15	12	15	18	16
ABC	**1**	27	20	20	22	19	11	20	14	20	19	17	14	18	14	16	15	12	15	18	16
	2	28	20	20	22	19	12	20	15	20	19	17	14	18	13	15	15	12	15	17	16
	3	27	21	20	22	19	11	19	14	20	19	17	14	19	14	16	15	12	15	17	16
GA	**1**	27	21	20	22	19	12	20	15	20	19	17	14	18	13	15	15	12	15	17	16
	2	28	20	20	22	19	11	19	14	20	19	17	14	18	14	16	15	12	15	18	16
	3	27	20	20	22	19	11	20	14	20	19	17	14	19	14	16	15	12	15	17	16

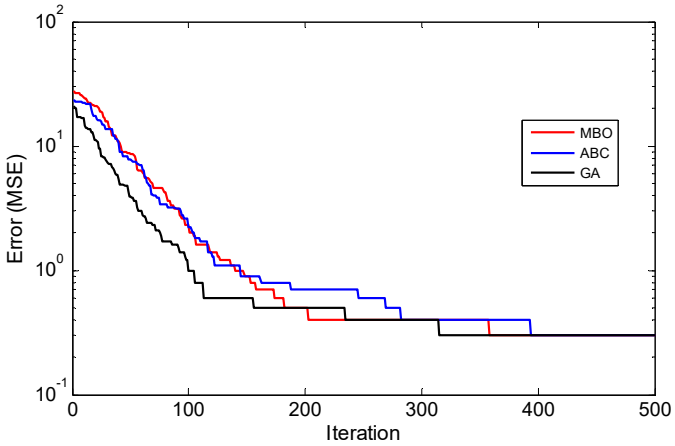

Figure 8. MSE Trends of the algorithms for no restriction case

The same calculations were performed for one restriction and two restrictions cases. In one restriction case, it was assumed that presentation of profession P1 had two sessions. In two restrictions case, it was assumed that presentations of profession P1 and P2 had two sessions. Parameters given in Table 3 were used again for these cases. Calculations were achieved with MSE value of 3.4667 for one restriction case and that of 9.9583 for two restriction case. Calculated total numbers of attendees for each session of each presentation are given in Table 4 and 5 for one restriction and two restriction cases respectively. It is seen that the numbers of attendees in each session of restricted presentations are nearly equal. On the other hand, while the numbers of attendees in first two sessions of unrestricted presentations are nearly equal, those in the last session are a little bit more. This is the result of that the students are canalized to the restricted presentations in first and second sessions. So, it can be said that the algorithms achieve to get balanced session distributions as optimal as possible. MSE progresses of the algorithms are shown by the graphs in Figure 9 and Figure 10. They show that predefined maximum number of iterations are more than the iteration needed for this case study.

Table 4. Session distributions for one restriction case (P: Presentation, S: Sessions)

	S	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20
MBO	**1**	41	20	20	21	18	11	19	14	19	18	15	13	18	13	15	15	11	14	17	15
	2	41	20	19	21	18	11	19	13	19	18	17	13	17	13	15	14	12	15	16	16
	3	0	21	21	24	21	12	21	16	22	21	19	16	20	15	17	16	13	16	19	17
ABC	**1**	41	20	19	21	19	10	19	14	20	18	16	13	18	13	15	14	11	14	17	15
	2	41	19	19	21	18	11	19	14	19	19	16	14	17	13	15	15	11	15	16	15
	3	0	22	22	24	20	13	21	15	21	20	19	15	20	15	17	16	14	16	19	18
GA	**1**	41	20	19	21	19	11	19	13	19	18	16	13	18	13	15	14	12	14	17	15
	2	41	19	19	21	18	10	19	14	19	18	17	13	18	13	15	15	11	15	17	15
	3	0	22	22	24	20	13	21	16	22	21	18	16	19	15	17	16	13	16	18	18

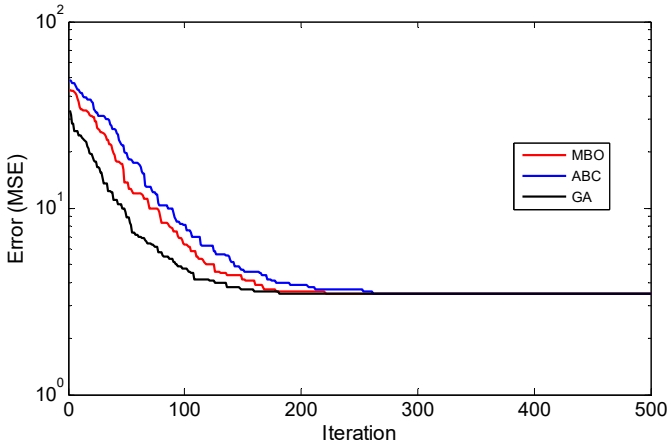

Figure 9. MSE Trends of the algorithms for one restriction case

Table 5. Session distributions for two restrictions case (P: Presentation, S: Sessions)

	S	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20
MBO	**1**	41	31	19	20	18	10	19	13	18	17	15	13	17	12	14	14	11	14	16	15
	2	41	30	18	21	17	10	18	13	19	18	16	13	17	13	14	14	11	14	16	14
	3	0	0	23	25	22	14	22	17	23	22	20	16	21	16	19	17	14	17	20	19
ABC	**1**	41	31	19	21	17	10	18	13	19	18	16	12	17	13	14	13	11	13	16	15
	2	41	30	18	21	18	10	18	13	18	18	16	13	17	12	14	14	11	14	16	15
	3	0	0	23	24	22	14	23	17	23	21	19	17	21	16	19	18	14	18	20	18
GA	**1**	41	30	18	20	18	10	19	13	19	17	16	13	17	12	14	14	11	14	16	15
	2	41	31	19	21	17	10	18	13	19	18	15	13	17	12	15	13	10	14	16	15
	3	0	0	23	25	22	14	22	17	22	22	20	16	21	17	18	18	15	17	20	18

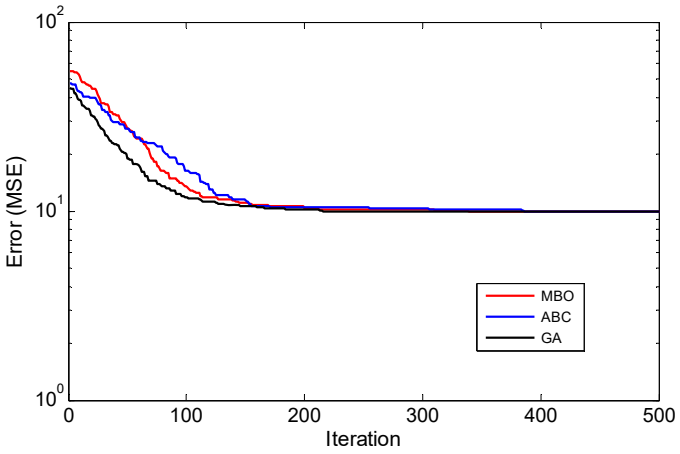

Figure 10. MSE Trends of the algorithms for two restrictions case

3.2. Results for the expanded problem

After completing equivalent experiments of the case study, we expanded the problem dimensions by increasing number of students to 5000 and number of presentations to 40. We used this expanded problem to check the performances of proposed methods. Again, we made 3 profession choices arbitrarily for each one of 5000 students. Aiming a fair competition among the algorithms, we used the settings given in Table 6 to get nearly equivalent total calculations for each algorithm. It was assumed that there was no session restriction. That is, all of the presentations had three sessions.

Table 6. Parameters of the algorithms and their calculation costs for expanded problem

		MBO	ABC	GA
Parameters	Iteration (*N*)	4902	5000	5000
	Population size (*n*)	25	50	50
	Generated neighbourhood (*k*)	3	-	-
	Shared neighbourhood (*x*)	1	-	-
	Iteration to change the leader (*m*)	10	-	-
	Mutation rate (%)	-	-	0.1
Calculation costs	Number of total calculations	**250002**	**250000**	**250000**
	MSE	**0.5167**	**0.4667**	**0.5167**

Session distribution results of the algorithms are given in Table 7. From the results, it is clearly seen that the total numbers of students in sessions for each presentation are well balanced. The best performance is obtained by the ABC algorithm. Distributions of the students to all of the sessions are globally optimal. The MBO algorithm and the GA show the same performances and their performances are very close to the performance of ABC algorithm. Both GA and MBO algorithm have only one suboptimal solution for the presentations P11 and P32 respectively.

Figure 11 gives optimization characteristics of the algorithms by showing MSE vs. iteration trends. MBO and ABC algorithms find reasonable solutions before reaching to 2000'th iteration. On the other hand, GA approaches to optimal solution slowly. ABC algorithm achieves

to reach global optimal solution, before iteration count reaches to 5000. It makes good global exploration to escape from suboptimal solutions owing to its random exploration capability in scout bee phase. The MBO and the GA could not escape from suboptimal solution before the iterations end. However, thinking the big problem dimension, although their solutions are suboptimal, these solutions are reasonable and very close to global optimal solution.

Table 7. Session distributions for expanded problem without any restriction (P: Presentation, S: Sessions)

MBO	**S**	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**	**P10**	**P11**	**P12**	**P13**	**P14**	**P15**	**P16**	**P17**	**P18**	**P19**	**P20**
	1	66	8	77	14	74	93	110	83	136	133	121	166	107	170	75	204	199	211	148	227
	2	67	8	77	13	74	93	109	82	136	132	121	165	106	170	75	205	199	211	147	228
	3	67	8	77	13	74	94	110	83	137	133	122	166	106	169	76	204	200	211	148	227
	S	**P21**	**P22**	**P23**	**P24**	**P25**	**P26**	**P27**	**P28**	**P29**	**P30**	**P31**	**P32**	**P33**	**P34**	**P35**	**P36**	**P37**	**P38**	**P39**	**P40**
	1	137	182	274	189	200	217	86	110	172	62	133	195	72	125	63	39	80	111	63	68
	2	138	183	275	189	201	217	86	110	171	61	133	197	72	125	63	38	80	111	63	69
	3	137	182	274	190	200	217	85	110	171	61	133	196	72	124	63	39	80	111	62	68
ABC	**S**	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**	**P10**	**P11**	**P12**	**P13**	**P14**	**P15**	**P16**	**P17**	**P18**	**P19**	**P20**
	1	66	8	77	13	74	94	110	83	136	133	121	166	107	170	75	205	200	211	148	227
	2	67	8	77	13	74	93	109	82	136	132	122	165	106	169	76	204	199	211	148	228
	3	67	8	77	14	74	93	110	83	137	133	121	166	106	170	75	204	199	211	147	227
	S	**P21**	**P22**	**P23**	**P24**	**P25**	**P26**	**P27**	**P28**	**P29**	**P30**	**P31**	**P32**	**P33**	**P34**	**P35**	**P36**	**P37**	**P38**	**P39**	**P40**
	1	138	182	274	189	200	217	85	110	171	61	133	196	72	125	63	38	80	111	63	68
	2	137	183	275	190	201	217	86	110	172	62	133	196	72	124	63	39	80	111	62	68
	3	137	182	274	189	200	217	86	110	171	61	133	196	72	125	63	39	80	111	63	69
GA	**S**	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**	**P10**	**P11**	**P12**	**P13**	**P14**	**P15**	**P16**	**P17**	**P18**	**P19**	**P20**
	1	67	8	77	13	74	93	110	83	136	133	120	166	107	169	75	204	200	211	147	228
	2	66	8	77	14	74	93	109	82	137	132	122	165	106	170	75	204	199	211	148	227
	3	67	8	77	13	74	94	110	83	136	133	122	166	106	170	76	205	199	211	148	227
	S	**P21**	**P22**	**P23**	**P24**	**P25**	**P26**	**P27**	**P28**	**P29**	**P30**	**P31**	**P32**	**P33**	**P34**	**P35**	**P36**	**P37**	**P38**	**P39**	**P40**
	1	137	182	275	189	201	217	86	110	171	61	133	196	72	125	63	39	80	111	63	68
	2	138	183	274	190	200	217	86	110	172	62	133	196	72	125	63	38	80	111	62	69
	3	137	182	274	189	200	217	85	110	171	61	133	196	72	124	63	39	80	111	63	68

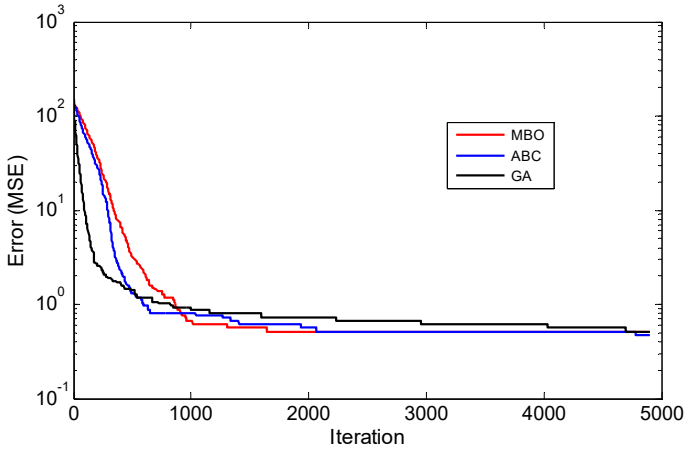

Figure 11. MSE Trends of the algorithms for expanded problem without restriction

5. Conclusions and Future Work

Combinatorial optimization methods make enumerations, permutations and combinations on solution sets to get more optimal solutions. In this paper, we proposed discrete or combinatorial forms of three well-known metaheuristics to solve a combinatorial optimization problem on symposium session optimization. Our methods make enumeration on students' choices, create different permutations and try new combinations in a discipline through the optimization process. It is clearly seen from the results that they achieve to get optimal solutions for the case study problem. They also give optimal or reasonably suboptimal solutions for an over expanded version of the case study problem.

Exploration and exploitation are two important issues in an optimization algorithm and should be balanced well. A good exploration characteristic enables the algorithm to escape from local optimum traps (suboptimal solutions). On the other hand, good exploitation characteristic enables algorithm to perform a good approach to global optimum solution. Considering this optimization facts, good exploration characteristic of the ABC algorithm may be adapted to the MBO algorithm to get more stable and robust behaviours in the MBO algorithm as a future work. Thus, a hybrid algorithm having quick and stable characteristics may be obtained to solve combinatorial optimization problems such as SSOP.

References

[1]Papadimitriou, C.H., and K. Steiglitz. 1982. *Combinatorial Optimization – Algorithms and Complexity*. New York, Dover Publications Inc.

[2]Blum, C., and A. Roli. 2003. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys* 35: 268–308.

[3]Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems*. Ann Arbor, University of Michigan Press.

[4]Kirkpatrick, S., C.D. Gelatt, and M.P. Vecchi. 1983. Optimization by simulated annealing. *Science* 220, no. 4598: 671–680.

[5]Glover, F. 1986. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research* 13, no. 5: 533–549.

[6]Dorigo, M. 1992. Optimization, learning and natural algorithms. *Ph.D. Thesis*, Politecnico di Milano.

[7]Eberhart, R.C., and J. Kennedy. 1995. A new optimizer using particle swarm theory. Paper read at proceedings of the sixth international symposium on micromachine and human science, in Nagoya, Japan.

[8]Storn, R., and K. Price. 1997. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization* 11: 341–359.
[9]Geem, Z.W., J.H. Kim, and G.V. Loganathan. 2001. A new heuristic optimization algorithm: harmony search. Simulation 76: 60–68.
[10]Karaboga, D., and B. Akay. 2009. A comparative study of Artificial Bee Colony algorithm. *Applied Mathematics and Computation* 214: 108–132.
[11]Mucherino, A., and O. Seref. 2007. A novel meta-heuristic approach for global optimization. Paper read at proceedings of the conference on data mining, system analysis and optimization in biomedicine, Gainesville, Florida.
[12]Yang, X.S. 2008. Firefly algorithm. *Nature-Inspired Metaheuristic Algorithms*, (p. 79–90). Frome, UK, Luniver Press.
[13]Shah-Hosseini, H. 2009. The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm. *International Journal of Bio-inspired Computation (IJBIC)* 1: 71–79.
[14]Yang, X.S., and S. Deb. 2009. Cuckoo search via Lévy flights. Paper read at proceedings of the World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore, India
[15]Yang, X.S., and S. Deb. 2010. Engineering optimisation by cuckoo search. *Int. J. Math. Modelling & Num. Optimisation* 1: 330-343.
[16]Yang, X.S. 2010. A New Metaheuristic Bat-Inspired Algorithm. J.R. Gonzalez et al., eds. *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)* 284: 65-74.
[17]Yang, X.S. 2011. Bat algorithm for multi-objective optimisation. *Int. J. Bio-Inspired Computation* 3, no. 5: 267-274.
[18]Duman, E., M. Uysal, and A.F. Alkaya. 2012. Migrating Birds Optimization: A new metaheuristic approach and its performance on quadratic assignment problem. *Information Sciences* 217: 65–77.
[19]Karaboğa, D. 2005. An idea based on honey bee swarm for numerical optimization. *Technical Report TR06*. Erciyes University, Engineering Faculty, Computer Engineering Department.
[20]Lissaman P.B.S., and C.A. Schollenberger. 1970. Formation Flight of Bird. *Science* 168: 1003 – 1005.
[21]Makas, H., and N. Yumusak. 2013. New Cooperative and Modified Variants of the Migrating Birds Optimization Algorithm. Paper read at proceedings of the 10th International Conference on Electronics, Computer and Computation (ICECCO'13), Ankara, Turkey.
[22]Moreno, J., D.A. Ovalle, and R.M. Vicari. 2012. A genetic algorithm approach for group formation in collaborative learning considering multiple student characteristics. *Computers & Education* 58: 560–569.
[23]Akay, B. 2009. Performance analysis of artificial bee colony algorithm on numerical optimization problems. *Ph.D. Thesis*, Erciyes University, Kayseri, Turkey.

.