Bee Colony Algorithm for Proctors Assignment

Nashat Mansour and Mohamad Kassem Taha
Department of Computer Science and Mathematics
Lebanese American University
Lebanon
nmansour@lau.edu.lb, mohamad.taha@lau.edu

***Abstract*— Proctor assignment refers to assigning proctors to examinations with the objective of having the appropriate number of proctors assigned to examinations, subject to conditions such as minimizing the load of proctoring and preventing any conflicting assignments. This problem is intractable, and hence, heuristics algorithms are needed to find good solutions. In this paper, we propose a new solution for the proctor assignment problem based on the Bee Colony meta-heuristic algorithm. The Bee Colony algorithm is a recent population-based search algorithm that mimics the natural behavior of swarms of honey bees during the process of collecting food. The algorithm performs a neighborhood search combined with a random search to balance exploration and exploitation. The food source identified by a honey bee is associated with a candidate solution to the proctors assignment problem. The search accomplished by three types of bees over a number of iterations aiming to find the source with the highest nectar value (fitness value of a candidate solution). We apply the Bee Colony algorithm to previously published data. Experimental results show good solutions that maximize the preferences of proctors while preserving the fairness of the workload given to proctors. The results also show that the Bee Colony algorithm outperforms other methods on most subject problems.**

Keywords—bee colony algorithm; constraint-based assignment; intelligent computing; meta-heuristics; proctor assignment

I. INTRODUCTION

Large-size schools and universities struggle when scheduling examinations and assigning an appropriate number of proctors without conflicts. This challenge gives rise to the problem of assigning proctors to examinations. Proctors can be faculty members or teaching assistants who have time constraints. Some proctors who are graduate students themselves may have to attend their own examinations, which imposes another constraint.

The Proctor Assignment to Examinations (PAE) problem is considered to be an extension of the Generalized Assignment Problem [1] which aims to assign tasks to agents while minimizing the cost of such assignment taking into consideration that an agent has limited or restricted resources. The PAE is an intractable problem. If we want to assign a number of proctors p for examinations that should take place over a number of time slots t, and for each of these slots there are a different assignments, then there will be $(t \times p!)/(p-a)!$ different combinations of assignments [2]. When this is performed manually, the team responsible for assigning proctors to examinations could spend an enormous amount of time, with unavoidable error rate. Hence, automating this process would not only save time but would also achieve many of the desired goals, mainly optimizing the proctors' preferences. Since the PAE problem is intractable, and enumerations and deterministic algorithms are not feasible, heuristics or meta-heuristics are required to produce good sub-optimal solutions.

Not much work has been reported on the PAE problem. Two meta-heuristics, the scatter search algorithms and genetic algorithms, have been developed to solve this problem. The scatter search solution [1] produced acceptable good results compared to other methods like integer programming. The Genetic algorithm [2] were able to produce the minimally acceptable solution that meets the client's need. However, research have been reported on related problems such as assigning judges to competitions using tabu search [3], assigning examinations to timeslots and rooms using scatter search [4], assigning teachers to courses using simulated annealing or tabu search [5], and assigning responsibilities to object oriented classes using multi-objective genetic algorithms [6].

In this paper, we present the application of a recent population-based search algorithm, the Bee Colony meta-heuristic algorithm (BC), to the PAE problem. The Bee Colony algorithm mimics the behavior of swarms of honey bees during the process of collecting food [7, 8]. It has been applied to other NP-hard problems with promising results. These problems include the fragment assembly problem [9], timetabling problems [10], vehicle routing optimization [11], job shop scheduling [12,13], and assigning terminals to concentrators [14]. We also modify the Scatter Search algorithm (SS) to a Multi-Objective Scatter Search (MOSS) version applied to PAE. The two proposed meta-heuristics, BC and MOSS, are applied to previously published data and are compared to the SS meta-heuristic and CPlex integer programming (IP). Our experimental results show that the BC algorithm could produce better solutions.

This paper is organized as follows. Section 2 presents the proctor assignment problem. Section 3 describes the design of the BC meta-heuristic and MOSS. Section 4 presents and discusses the experimental results. Section 5 concludes the paper.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

II. Proctor Assignment to Examinations Problem Description

PAE problem refers to the assignment of proctors to examinations subject to a number of constraints:

- A specific number of proctors are required for each examination. Due to variations in class size, each examination requires a different number of proctors in order to meet the proctoring requirement.
- A proctor has a limited number of hours that can be devoted to proctoring examinations. Not all proctors have the same amount of free time. Proctors might be part-time students, full-time students, teaching assistants, and academic staff. The assignment process should take into account the time availability of each proctor.
- A proctor can only be assigned to one examination at any one time. Proctoring assignments should not overlap for the same proctor in order to avoid conflicts.
- A proctor can only be assigned to an examination that does not conflict with his own examination. Some proctors have to sit for their own examinations; therefore it is expected that they are inclined to proctor on particular days while trying to avoid others. Accordingly, one of the main objectives of the PAE problem is to assign proctors to examinations in such a manner as to maximize these preferences.
- Load balance between proctors. Another consideration is to balance the workload among all proctors. In order to measure the fairness of the workload, we could minimize the difference between proctors with a heavy workload and those with a light workload. To implement this idea, we formulate a model so that we maximize the minimum workload to be assigned to each proctor. This workload can be calculated by dividing the number of hours assigned to each proctor by the number of available hours to him.

We assume that an examination day is split into two periods and that an examination can take place over one or two periods. We denote by J the set of m examinations (j = 1, ..., m) and I the set of n proctors (i = 1, ..., n). Also, the parameters for this problem are as follows: a_i = maximum number of available hours for proctor i; b_j = number of hours associated with examination j; t_j = number of proctors required for examination j; c_{ij} = preference of proctor i for examination j, which is an integer between 0 and 2; P_i = the set of examinations that overlap with any proctor i's examinations; T_k = the set of examinations scheduled in period k; d = number of examination days; y_i = the number of proctoring hours assigned to proctor i. The variable x_{ij} is a flag, where x_{ij} = 1 if proctor i is assigned to examination j, otherwise x_{ij} = 0. The objectives of the (PAE) problem are [1]:

$$Max \;\; f(x) = \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij} x_{ij} \tag{1}$$

$$Min \;\; g(x) = \sum_{i=1}^{n} (y_i / a_i) \tag{2}$$

Subject to:

$$\sum_{j=1}^{m} b_j x_{ij} \le a_i \qquad i = 1, \ldots, n \tag{3}$$

$$\sum_{j=1}^{m} b_j x_{ij} - a_i y \ge 0 \qquad i = 1, \ldots, n \tag{4}$$

$$\sum_{i=1}^{n} x_{ij} = t_j \qquad j = 1, \ldots, m \tag{5}$$

$$\sum_{j \in T_k} x_{ij} \le 1 \qquad i = 1, \ldots, n; k = 1, \ldots, 2d \tag{6}$$

$$x_{ij} = \{0,1\} \qquad i = 1, \ldots, n; j = 1, \ldots, m \tag{7}$$

$$x_{ij} = 0 \qquad j \in P_i \tag{8}$$

Equation (1) represents the sum of the preferences of the proctors' assignments while (2) aims to minimize the proctors utilization which is defined as the ratio of the hours that a proctor is assigned to examinations over his/her available hours. The purpose of the PAE problem is to satisfy these objectives concurrently. Equations (3) to (8) represent the constraints we have. Equation (3) ensures that the number of the assigned hours for each proctor does not exceed his/her total available hours. Equation (4) determines the minimum proctors' utilization. Equation (5) ensures that all examinations are assigned the required number of proctors. Equation (6) averts a proctor to be assigned more than one examination during a specific period. Equation (7) implements the conditional flag on what is considered to be a decision variable. Equation (8) ensures that proctors are not assigned to examinations that overlap with their own examinations. The weighted combined function, as described in [1], for the PAE problem is given as:

$$h(x) = f'(x) + \alpha g(x) \tag{9}$$

where $$f'(x) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} \frac{c_{ij}}{c_{\max}(j)} x_{ij}}{\sum_{j=1}^{m} t_j}$$

and $$c_{\max}(j) = \max_i (1, c_{ij})$$

The parameters of the weighted function $g(x)$ and $f'(x)$ are both restricted between 0 and 1 since the minimum preference value is zero. We use α = 0.5 to indicate equal weight for satisfying proctors' preferences and proctoring load reduction. We evaluate our solutions based on the weighted function, $h(x)$, which combines both objectives.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

III. Bee Colony Algorithm

A. Background

The BC Algorithm is an algorithm that mimics the natural foraging behaviour of honey bees in searching for food sources in order to maximize the nectar amount, i.e., to find optimal solutions [7]. It is a population-based search algorithm in which food sources/sites are modified/explored by artificial bees, where the objective is to discover sites of food sources with the highest nectar amount. The hive houses three types of bees and the algorithm requires setting a number of parameters, namely: *e* which is the number of employed bees that search for food sources/sites; *n* which is the number of onlooker bees that select good food sources from those identified by employed bees; *s* which is the number of scout bees that randomly search for food sources; and n_{gh} which is the size of the patch that includes the site and its neighbourhood. The BC algorithm combines local and global search methods with the aim of balancing the exploration and exploitation processes.

The algorithm initially starts by placing *s* scout bees in the search space in a random fashion. Next, the fitness values (nectar amount) of the visited sites are evaluated. The employed bees visit the sites and carry out a neighborhood search, aiming to find food sources with more nectar. Then, the employed bees share the information of their findings with the onlooker bees through waggle dances.

Each onlooker bee evaluates all the nectar information and probabilistically choose identified food sources. For each source, a neighborhood search is conducted. If the onlooker bee does not find an improved food source over L cycles/attempts, it abandons this source/site. For abandoned sites, scout bees will randomly discover replacement food sites/sources.

B. Bee Colony Algorithm Design for PAE

The BC algorithm aims to find a good sub-optimal PAE solution from a pool of evolving candidate solutions. The steps of the BC algorithm are shown in Fig. 1.

1) Solution Representation, Initial Population, and Fitness Function

The actual PAE solution is represented as a linked list where *I* denotes the set of *n* proctors (i = 1, ..., n) linked to their list of assigned examinations *J* where *J* denotes the set of *m* examinations (j = 1, ..., m). The element Proctor(i) represents the proctor that will be allocated to a chosen examination and period object denoted by (E,P) where P is the selected period allocated to an examination, while E corresponds to the examination list that ranges from {1,…,m}, noting that the preference of proctors and the examination periods are also acquired as input data. Another data structure used to represent a solution is an array representing each proctor's assignment to examinations as a percentage value of preference.

The algorithm starts by randomly initializing *s* PAE candidate solutions, which correspond to *s* scout bees randomly searching *s* food sites. The random initialization ensures that the number of proctors per examination is respected.

```
1.  Initialize the set of s candidate PAE
    solutions/scout-bees/food-sites randomly;
2.  Evaluate the fitness (nectar amount) of each
    site/candidate solution;
3.  Repeat
            //Forming new population
4.  (Employed bees) conduct a neighborhood search
    for improving sites/candidate solutions;
5.  Share information obtained by employed bees
    with onlooker bees through waggle dances;
6.  (Onlooker bees) probabilistically select
    candidate solutions of employed bees;
7.  (Onlooker bees) conduct a neighborhood search
    for improving selected candidate solutions and
    identify abandoned candidate solutions, after L
    attempts;
8.  (Scout bees) randomly generate candidate
    solutions to replace the abandoned ones;
9.  Combine the candidate solutions of both bees'
    search, of onlooker and scout bees, to form a
    new set of candidate solutions/food-sites;
10. Save the best fitness value of candidate
    solutions achieved so far;
11. Until (no improvement in the best fitness value
    over 3 successive iterations OR reached maximum
    number of iterations)
```

Fig. 1. BC algorithm outline.

The fitness of each candidate solution/site is computed using equation 9, which corresponds to the nectar amount of the food sites visited by the bees.

2) Generating Improved Candidate Solutions

In our implementation, the number of candidate solutions identified by employed bees (*e*) is equal to those initially identified by scout bees (*s*). In Step 4 of Fig. 1, each candidate solution is subjected to a neighborhood search by randomly selecting 25% of the proctors and randomly reassigning one of their examination assignments, provided that the required number of proctors per examination is still respected, otherwise the reassignment is retracted. Then, the fitness function is recomputed. The modified solution is accepted only if the resulting fitness function is improved.

Next, the *e* candidate PAE solutions that are produced by the employed bees' neighborhood search are evaluated in terms of the fitness function. Then, the roulette wheel probabilistic mechanism is applied to each candidate solution for the onlooker bees to identify which of these solutions are acceptable to remain in the population. According to the roulette wheel technique, the probability of acceptance of a candidate solution is based on the fraction of the fitness value of this solution with respect to the total sum of the fitness values of all the solutions found by the employed bees. The successful candidate solutions are then subjected to the same neighborhood search procedure as that with the employed bees. But, this time the search is attempted up to L = 3 times for a solution that does not show improvement in the fitness value in the initial attempt(s). Candidate solutions that fail to improve after the L attempts are considered to be abandoned food sources and are removed from the population.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

The abandoned candidate solutions are then replaced (by the scout bees) with others obtained by global random search, similar to the procedure of the initial population. The candidate solutions of both the onlooker and scout bees are combined to form the next generation of PAE candidate solutions for the next iteration.

Also, at the end of every iteration, the best-so-far candidate solution and its fitness value are saved. This information is used for a stopping criterion, whose alternative is a maximum number of 1000 repetitions

IV. Experimental Results

We applied the PAE algorithms to real-world data sets from two universities, University of Barcelona [2] and Carleton University [1]. The data used is comprised of 11 subject cases described in Table 1. These cases include cases where the number of proctors is more than the number of examinations, and other cases where the number of examinations is more than the number of proctors.

We ran five algorithms on these data: Bee Colony (BC), Scatter Search (SS), Multi-Objective Scatter Search (MOSS), Partitioned Cplex, and Cplex. Each algorithm was executed ten times on each of the data sets and the worst, average, and best results were computed.

Table 2 shows the best, average, and worst results obtained from the five algorithms, where the entries are the values of the fitness function. We note that CPlex is deterministic and yields the same results for all of the ten runs. Based on these results we make the following comments:

- The Bee Colony algorithm produced better results than the other four algorithms in 8 out of the 11 cases in terms of the average values, in 9 out of 11 cases in terms of the best values and in 8 out of 11 cases in terms of the worst cases. The average parentage of improvement over the 11 cases yielded by the Bee Colony algorithm in comparison with CPlex is 5% in terms of the average values over ten runs.
- CPlex in its partitioned version was also able to solve all of the cases and produced better results than the scatter search and multi-objective scatter search. The degree of improvement in the results of CPlex with respect to the results of the multi-objective scatter search was around 27% in terms of the average values.
- Scatter search was able to find results for PAE but since the nature of the problem tends to be multi-objective, we find that the multi-objective version of scatter search yielded slightly better results. The degree of improvement between both algorithms was around 9.2% on average for all of the 11 test cases in terms of average values of 10 runs.
- However, CPlex in its non-partitioned version the values produced from the first 6 cases were the same the partitioned version. The quality of the results decreased as the problem instance size increased until it crashed in the last three cases.

TABLE I. PAE Subject Problems

Case	Number of Proctors	Examinations
1	23	21
2	59	52
3	59	44
4	25	21
5	46	42
6	150	300
7	230	690
8	370	1480
9	520	2600
10	740	4440
11	1500	15000

V. Conclusion

We have proposed a Bee Colony algorithm to produce sub-optimal assignments of proctors to examinations subject to the conditions of maximizing the preferred assignment of the proctors to examinations as well as preserving fairness in workload distribution. To validate our algorithm, we have used real data from two universities. Our experimental results show that the Bee Colony algorithm outperformed the scatter search and multi-objective scatter search algorithms as well as the CPlex. In particular, it produced an average improvement of 5% in comparison with the second best algorithm, partition CPlex.

Appendix: Multi-objective Scatter Search for PAE

The multi-objective solution for solving the proctor assignment problem involves two heuristic algorithms scatter search and tabu search algorithms. Tabu search is used to generate an initial set of diverse solution. Then we use the SS algorithm to enhance these solutions.

The procedure starts with the diversification generator to build a collection P of diversified candidate solutions. The dimension of this collection (PSize) should be large and is typically ten times greater than the reference set (RefSet). Then, the first reference set is constructed using the reference set update method by taking the best b distinct and diverse solutions from P. Here we employ tabu search again to select the variables as a basis for defining the move attributes.This is done to prevent moves that will produce close results to previously held ones. Specifically we consider a move as a tabu if it produces a solution that lies much closer than a specified tabu defined distance to any previously obtained/visited solution. Furthermore, an approach based on a weighted sum is built in making decisions to pick solutions as we need to pick the good solutions.

This is done by first taking the most attractive b_1 solutions from P which are included in the reference set and removed from P. Then, the remaining solutions in P are used to compute the least distance measure with respect to all solutions that make up the current reference set. The solution with the

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

greatest minimum distance gains entry to the reference set and is removed from P and the least distances measures are computed again. This procedure is iterated b_2 times knowing that $b_2 = b - b_1$. The resulting reference set would be made up of b_1 high-quality solutions and b_2 diverse solutions.These solutions are sorted with respect to the goodness of their objective/fitness function whereby the solution with the highest utility function figures in the first position. Next the lookup begins with setting the Boolean variable NewSolutions to TRUE.

Afterward, NewSubsets are constructed using the subset generation method constructed, and NewSolutions is assigned the value of FALSE. The subset generation method in its simplest form generates all pairs of reference solutions. That is, $(b^2 - b)/2$ NewSubsets of solutions are obtained. Then these pairs are subjected to the solution combination method to produce new trial solutions which are then subjected to the improvement method.

The reference set update method is run another time to get the new RefSet by taking the best solutions found from the current RefSet and the new trial solutions. These solutions are chosen according to their quality, i.e. according to their objective function. In case the RefSet changes at this stage, then NewSolutions is set to TRUE to indicate that a solution has gained membership in the reference set. The subset that was considered for combination is deleted from the collection of subsets or NewSubsets. Then all sub-collections constituting NewSubsets are combined, and if no better solution is added to the reference set, the scatter search is terminated.

It is worthwhile noting that after the combination method is applied, the reference set is updated in a static way. This is so because when trial solutions are generated by the means of the combination method, they are placed in a solution pool. Then, the b most attractive solutions found in the union of the reference set and the pool generated are chosen to construct the new reference set.

REFERENCES

[1] R. Marti, H. Lourenco, and M. Laguna, "Assigning proctors to exams with scatter search," *Journal of Computer Science Interfaces*, vol. 12, no. 37, pp. 215–227, 2000.

[2] R. Awad and J. Chinneck, "Proctor assignment at Carleton University," *Interfaces*, vol. 28, no. 2, pp. 58–71, 1998.

[3] A. Lamghari and J.A. Ferland, "Structured neighborhood tabu search for assigning judges to competitions," *Proceedings of IEEE Symposium on Computational Intelligence in Scheduling*, 2007.

[4] N. Mansour, V. Isahakian, and I. Ghalayini, "Scatter search technique for exam timetabling," *Journal of Applied Intelligence*, vol. 34, no. 2, pp. 299-310, 2011.

[5] A. Gunawan and K. Ng, "Solving the teacher assignment problem by two metaheuristics," *International Journal of Information and Management Sciences*, vol. 22, no. 2, pp. 73–86, 2011.

[6] M. Bowman, L.C. Briand, and Y. Labiche, "Solving the Class Responsibility Assignment Problem in Object-Oriented Analysis with Multi-Objective Genetic Algorithms," *IEEE Trans. Software Engineering*, vol. 36, no. 6, pp. 817–837, 2010.

[7] D. Karaboga, *An idea based on honey bee swarm for numerical optimization*, Tech. Rep. TR06, Engineering Faculty, Computer Engineering Department, Erciyes University, 2005.

[8] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: artificial bee colony (ABC) algorithm and applications," *Artificial Intelligence Review*, vol. 42, no. 1, pp. 21–57, 2014.

[9] J. Firoz, M. Rahman, & T. Saha, "Bee algorithms for solving DNA fragment assembly problem with noisy and noiseless data," *Proceedings of the 14th Conference on Genetic and Evolutionary Computation*, 2012, pp. 201–208.

[10] M. Alzaqebah and S. Abdullah, "Artificial bee colony search algorithm for examination timetabling problems," *International Journal of Physical Sciences*, vol. 6, no. 17, pp. 4264–4272, 2011.

[11] A. Bhagade and P. Puranik, "Artificial bee colony (ABC) algorithm for vehicle routing optimization problem," *International Journal of Soft Computing and Engineering*, vol. 2, no. 2, pp. 329–333, 2012.

[12] J. Q. Li, Q. K. Pan, S. X. Xie, and S. Wang, "A hybrid artificial bee colony algorithm for flexible job shop scheduling problems," *International Journal of Computers, Communications and Control*, vol. 6, no. 2, pp. 286–296, 2011.

[13] A. Banharnsakun, B. Sirinaovakul, and T. "Achalakul, "Job shop scheduling with the best-so-far ABC," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 3, pp. 583–593, 2012.

[14] E. Bernardino, A. Bernardino, J. Sánchez-Pérez, J. Gómez-Pulido, and M. Vega-Rodríguez, "Using the bees algorithm to assign terminals to concentrators," *Proceedings of the 23rd International Conference on Industrial Engineering and other Applications of Applied Intelligent Systems*, Berlin, Germany, 2010.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731