A Hybrid Artificial Bee Colony Algorithm for the Terminal Assignment Problem

Jayalakshmi Banda and Alok Singh(✉)

School of Computer and Information Sciences,
University of Hyderabad, Hyderabad 500046, India
bjayalakshmi@uohyd.ac.in, alokcs@uohyd.ernet.in

Abstract. The terminal assignment (TA) problem is an important problem in the design of telecommunication networks. The problem consists in determining the best links for connecting a given set of terminals to a given set of concentrators so that a given cost function is optimized. In this paper, we have proposed an artificial bee colony algorithm based approach for solving the TA problem. In comparison with the best methods available in the literature, the proposed approach obtained better quality solutions in shorter time.

Keywords: Artificial bee colony algorithm · Heuristic · Swarm intelligence · Terminal assignment problem · Telecommunication networks

1 Introduction

Due to rapid growth of internet, many new problems arose in the field of telecommunication network design and management. Terminal assignment (TA) problem is one such problem. In large centralized computer networks, a central computers serves numerous terminals or workstations. In such cases, concentrators are used to increase the efficiency of the network. Instead of connecting the terminals directly to the central computers, the terminals are connected to the concentrators and concentrators are connected to the central computer. The terminals and concentrators have fixed and known locations. The capacity requirements of each terminal is known. This requirement may vary from terminal to terminal. The maximum capacity of each concentrator and the costs of connecting each terminal to different concentrators are also known. The objective of the TA problem is to connect a given set of N terminals to a given set of M concentrators in such a way that the total cost of the network thus formed is minimum according to a given objective function. The assignment of the terminals to the concentrators is done under following two constraints: First, each terminal must be connected to one and only one concentrator, second, the sumtotal of capacities of the terminals connected to a concentrator must not exceed the maximum capacity of that concentrator [7,9].

TA problem is solved in the literature with two different objectives: First, with the objective of minimizing the sumtotal of link costs alone and second

B.K. Panigrahi et al. (Eds.): SEMCCO 2014, LNCS 8947, pp. 134–144, 2015.
DOI: 10.1007/978-3-319-20294-5_12

with an objective which give consideration to equitable distribution of loads among concentrators in addition to link costs. Here, we have considered the latter objective. TA problem is proved $\mathcal{NP}$-Hard under the first objective [19]. The problem can be solved in polynomial time in the special case where all terminals have the same capacity requirements and all concentrators have the same maximum capacity. The TA problem is also $\mathcal{NP}$-Hard under the second objective as first objective can be considered as a special case of the second objective where no consideration is given to load on different concentrators. The problem is harder to solve under second objective because we can not compute the individual cost of assigning a terminal to a concentrator a priori, i.e., before the complete solution is constructed.

TA problem is also closely related to a number of classical combinatorial optimization problems in different fields. Bin packing problem, task assignment problem, problem of assigning cells to switches in mobile communication networks are few examples [8].

Many different approaches have been proposed in the literature to solve the TA problem. Abuali et al. [9] proposed a greedy heuristic and a greedy genetic algorithm for a restricted version of the problem where all concentrators have the same maximum capacity. Khuri and Chiu [7] proposed another greedy heuristic and two penalty based genetic algorithms. Both Abuali et al. [9] and Khuri and Chiu [7] considered the first objective as mentioned above. Salcedo-sanz and Yao [8] considered for the first time the second objective where a hybrid Hopfield network based genetic algorithm is presented. Xu et al. [4] presented a tabu serach based approach for TA problem. Bernardino et al. designed a local search genetic algorithm (LSGA) [5], a tabu search (TS) [3], a hybrid differential evolution algorithm (HDE) [16], a bees algorithm [15], an improved hybrid differential evolution algorithm with a multiple strategy (MHDE) [6] and a discrete differential evolution algorithm (DDE) [11] for solving the TA problem with second objective. The DDE algorithm is based on discrete differential evolution model proposed by Pan et al. [17]. DDE algorithm provides the better results in comparison to LSGA, TS and MHDE [11].

In this paper, we have proposed an artificial bee colony algorithm based approach for solving the TA problem. The artificial bee colony (ABC) algorithm is a recently developed swarm intelligence technique proposed by Karaboga [1]. ABC algorithm is inspired by intelligent foraging behaviour of honey bee swarm. ABC algorithm has already been applied successfully to solve numerous discrete optimization problems [12]. This has motivated us to develop an ABC algorithm for the TA problem. We have compared our ABC approach with 4 best approaches from the literature, viz. DDE, MHDE, TS and LSGA. In comparison to these approaches, our approach not only obtains solution of better quality, but is also faster.

The remaining part of this paper is organized as follows: In Sect. 2, we formally define the TA problem. Section 3 provides an overview of ABC algorithm. Section 4 describes our ABC approach for the TA problem. Section 5 reports the

computational results and compares our approach with other state-of-the-art approaches available in the literature. Finally, Sect. 6 contains some concluding remarks and directions for future research.

2 TA Problem

This section defines the TA problem formally. Given a set of terminals $T = \{T_1, T_2, \ldots, T_N\}$, a set of concentrators $C = \{C_1, C_2, \ldots, C_M\}$, a set of weights or capacities $W = \{W_1, W_2, \ldots, W_N\}$ associated with each terminal and capacities of concentrators $X = \{X_1, X_2, \ldots, X_M\}$. The capacity of terminals are such that $W_i < \min\{X_1, X_2, \ldots, X_M\}\ \forall\ T_i \in T$. The TA problem seeks an assignment of terminals to concentrators without violating the capacity constraint of concentrators such that the considered objective function is optimized. Hence, any feasible solution must satisfy the following two constraints:

$$\sum_{j=1}^{M} z_{ij} = 1 \quad \forall T_i \in T \tag{1}$$

$$\sum_{i=1}^{N} W_i z_{ij} \leq X_j \quad \forall C_j \in C \tag{2}$$

where binary variables z_{ij} indicate whether terminal T_i is assigned to concentrator C_j ($z_{ij} = 1$) or not ($z_{ij} = 0$). The Eq. 1 states that each terminal can be assigned to one and only one concentrator whereas Eq. 2 states that capacity constraint of no concentrator should be violated.

We have considered the same objective function for TA problem as used in [11]. This objective function considers the two factors:

- The total number of terminals assigned to each concentrator
- The distance between the terminals and their assigned concentrators

The objective is to minimize the Eq. 5. Equations 3 and 4 define terms needed for defining the objective function.

$$Total_{C_j} = \sum_{i=1}^{N} z_{ij} \quad \forall C_j \in C \tag{3}$$

$$Bal_{C_j} = \begin{cases} 10 & if\ (Total_{C_j} = round(\frac{N}{M}) + 1) \\ 20\times \mid (round(\frac{N}{M}) + 1 - Total_{C_j}) \mid & otherwise \end{cases} \quad \forall C_j \in C \tag{4}$$

$$objective\ function\ value = 0.9 \times \sum_{j=1}^{M} Bal_{C_j} + 0.1 \times \sum_{i=1}^{N} \sum_{j=1}^{M} D_{ij} z_{ij} \tag{5}$$

where D_{ij} is the distance between terminal T_i and concentrator C_j.

3 Overview of ABC Algorithm

The artificial bee colony (ABC) algorithm introduced by Dervis Karaboga in 2005 [1] is a population based meta-heuristic algorithm based on the foraging behavior of the real honey bees. In a bee colony, there are three types of bees: employed, onlooker and scout. Employed bees exploit the food sources. These bees bring loads of nectar to the hive and share the information about the food sources exploited by them with the onlooker bees which wait in the hive for this information to be shared. The onlooker bees tend to select a food source with a probability that depends on the quality of that food source with respect to other food sources. Once an onlooker bee selects a food source, it becomes employed. Scout bees search for new food sources in the vicinity of hive and as soon as they find a new food source they become employed. An employed bee whose food source becomes empty will turn either into a scout or an onlooker. Therefore, employed and onlooker bees are responsible for exploitation, whereas exploration is left for scout bees.

Inspired by this foraging behaviour, Karaboga developed ABC algorithm. This algorithm was initially developed for optimization in continuous domain only [1,12–14]. Later, it was extended to solve discrete optimization problems [2,10,18]. For a recent survey on ABC algorithm and its applications, interested readers may refer to [12]. Some recent applications of ABC algorithm can be found in [20–22].

In ABC algorithm, the food sources represent the possible solutions to the problem under consideration and their nectar content indicates the fitness of the solutions represented. The ABC algorithm also divides the colony of artificial bees into same three types with similar function. However, unlike real bees, a one-to-one correspondence is maintained between the food sources and employed bees by associating each employed bee with one and only one food source. Usually but not always, the number of onlooker bees is taken to be equal to number of employed bees. An employed bee whose food source becomes empty will turn only into a scout but never an onlooker. Such a scout is immediately made employed by generating a food source randomly and associating this scout with this newly generated food source. The ABC algorithm consists of an iterative search process. The algorithm is initialized by associating each employee bee with a randomly generated food sources (solutions). Then the algorithm repeats through the cycles of the employed bee and onlooker bee phases. In the employed bee phase, each employed bee determines a food source in the vicinity of its current food source and evaluates its nectar amount (fitness). If the nectar amount of the new food source is better than the current one then the employee bee moves to the new food source leaving the old one, otherwise it remains at the old one. When all the employee bees finish this process, they share the nectar information of the food sources with the onlookers, then the onlooker bee phase starts.

In the onlooker bee phase, onlookers select the food sources with a probability that depends on the nectar content of the food sources. Higher the nectar content of a food source, higher will be the chances of its selection. As a result of this

selection policy, good quality food sources attract more onlookers in comparison to worse ones. After all onlookers select the food sources, they determine the food sources in the vicinity of their selected food sources in a way similar to that of employee bees. Among all the new food sources determined in the vicinity of a food source i by the onlookers associated with food source i and the food source i, the best quality food source is determined. This best food source will be selected as the new location for food source i in the next iteration. The onlooker bee phase ends when all food sources are updated in the aforementioned manner and the next iteration of the ABC algorithm starts. The algorithm stops when the termination condition is satisfied. If the solution associated with a food source does not improve over some specific number of iterations say *limit* then that food source is considered as empty and is discarded by its associated employed bee. Then that employee bee becomes scout. A new food source is generated for this scout so as to make it again employed. This new food source is usually generated in the same manner as an initial solution.

In the employed bee phase every solution is given a fair chance to improve itself, whereas in the onlooker bee phase, good quality solution are given more chance to improve themselves in comparison to poor quality solutions. This is justified considering the fact that in the vicinity of good quality solutions, chances of finding even better solutions are higher. However, if a solution is locally optimal then no better solution exists in its vicinity and any attempt to improve it through employed or onlooker bee phases will fail. Here, the concept of scout bees plays its part by replacing the locally optimal solution with a new solution. In a robust search process the balance between the exploration and exploitation must be maintained. In the ABC algorithm, this balance depends on the parameter *limit*. Smaller value of *limit* favors exploration over exploitation whereas reverse is true for a higher value of *limit*. Therefore, the value of *limit* should be chosen with utmost care.

4 ABC Approach for the TA Problem

In this section, we present our ABC approach for TA problem. Subsequent subsections describe salient features of our proposed approach.

4.1 Solution Encoding

To encode a solution, we have used the terminal based representation proposed in the literature [11]. The value represented by position i specifies the concentrator to which the terminal i is assigned. Figure 1 explains this representation with the help of an example where there are 10 terminals and 3 concentrators. In this figure, terminals 1, 3 and 6 are assigned to concentrator 1, terminals 2, 4, 7 and 9 are assigned to concentrator 2, terminals 5, 8 and 10 are assigned to concentrator 3.

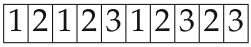

Fig. 1. Solution representation

4.2 Initial Solution Generation

Each initial solution is obtained by using a method which is partially greedy and partially random. In this method, with probability p_{asn}, the terminals are greedily assigned to the nearest available concentrator . Here, the availability refers to the capacity of the concentrator to serve the terminal capacity requirement. If the capacity is not satisfied then the terminal is assigned to the next nearest concentrator, and this process is repeated until an available concentrator is found or none exits. With probability $1 - p_{asn}$, terminals will be assigned to the available concentrators randomly. The algorithm iterates through this process until all the terminals are assigned. The terminal to be assigned next is selected randomly.

In case no available concentrator exists for a terminal then the solution is infeasible. This solution is discarded and we start afresh in a bid to generate a feasible solution. If we are not able to generate a feasible solution even after three attempts then the last infeasible solution is included in the population, but its fitness is penalised using a penalty term as explained in Sect. 4.5. In this infeasible solution, terminals which can not be assigned to any available concentrator are assigned to some randomly chosen concentrator.

4.3 Generation of Neighboring Solution

To generate a solution $S_i^{'}$ in the neighborhood or vicinity of a solution S_i, each terminal in S_i is reassigned with probability p_{rand} using a greedy approach. In the greedy approach the terminals are assigned to the nearest available concentrators. S_i is replaced with the neighboring solution $S_i^{'}$ if the fitness of $S_i^{'}$ is better than S_i. In case the neighboring solution is infeasible then we discard the solution. This can happen only when original solution is infeasible.

The pseudo code for generating a neighboring solution $S_i^{'}$ in the vicinity of a solution S_i is as follows:

4.4 Selecting a Food Source for an Onlooker Bee

Instead of using the commonly used roulette wheel selection method, we have used the binary tournament selection method for selecting a food source for an onlooker bee. In the binary tournament selection two food sources are selected randomly, and, the better of the two food sources are selected with the probability p_{onl} and worse of the two with the probability $1 - p_{onl}$.

Algorithm 1. generate_neighboring_solution(S_i)

```
for each terminal T_j ∈ T do
  p = generate a random number between 0 and 1
  if p <= p_rand then
    Assign T_j to the nearest available concentrator in S_i'
  else
    Assign T_j to the same concentrator in S_i' as in S_i
  end if
end for
return S_i'
```

4.5 Fitness of a Solution

To evaluate the fitness of a solution, we have used the same fitness function as used in [11]. This fitness function is a modification of the objective function given in Sect. 2. The fitness function adds a penalty term called *penalization* to the objective function for infeasible solutions. The *penalization* is computed as follows:

$$penalisation = \begin{cases} 0 & if\ (solution\ is\ feasible) \\ 500 \end{cases} \tag{6}$$

$$fitness = objective\ function\ value\ +\ penalisation. \tag{7}$$

This fitness needs to be minimized.

4.6 Other Features

We have used different number of employee and onlooker bees unlike the usual practice of using the same number of employed and onlooker bees. If a solution correlated with an employed bee does not improve for *limit* number of iterations then this employed bee becomes scout. There is no limit on the number of scouts in an iteration. The number of scouts in a particular iteration depends on how many employed bee solutions got improved *limit* iterations prior to current iteration.

5 Experimental Results

Our ABC approach has been implemented in C and executed on a Intel Core 2 Duo (E8400) system with 2 GB RAM running at 3.0 GHz under Fedora 12 release. In all our computational experiments, the number of employed bees (n_e) is taken to be 50 and the number of onlooker bees (n_o) is taken to be 100. We have used p_{rand}=0.2, p_{asn} =0.85, p_{onl}=0.85, $limit$=500 in all our experiments. Our ABC approach terminates after 1500 iterations. All these parameter values are chosen empirically, after executing the algorithm multiple times. In order to

Table 1. Solution quality of various approaches

	LSGA			TS			MHDE			DDE			ABC		
prob	BestF	AvgF	StdD	BestF	AvgF	StdD	BestF	AvgF	StdD	BestF	AvgF	StdD	BestF	AvgF	StdD
1	65.63	65.63	0.00	65.63	65.63	0.00	65.63	65.63	0.00	65.63	65.63	0.00	**65.63**	**65.63**	0.00
2	134.65	134.65	0.00	134.65	134.65	0.00	134.65	134.65	0.00	134.65	134.65	0.00	**133.41**	**133.41**	0.00
3	270.26	270.69	0.23	270.26	270.76	0.30	270.26	270.75	0.15	270.26	270.47	0.22	279.94	281.17	0.66
4	286.89	286.99	0.13	286.89	287.93	0.75	286.89	287.17	0.14	286.89	286.89	0.00	**286.61**	**286.65**	0.09
5	335.09	335.99	0.60	335.09	335.99	0.59	335.09	336.55	0.39	335.09	335.26	0.17	**335.07**	336.24	0.32
6	371.12	371.68	0.24	371.12	372.44	0.45	371.12	373.19	0.42	371.12	371.38	0.22	374.55	378.44	1.38
7	401.21	402.41	0.50	401.29	403.25	0.73	401.21	403.61	0.33	401.21	401.62	0.28	**399.85**	**400.59**	0.39
8	563.19	564.94	0.52	563.34	564.5	0.54	563.19	572.04	0.76	563.19	564.07	0.38	596.65	601.85	2.53
9	642.83	646.52	0.84	642.86	644.18	0.48	642.83	648.46	0.48	642.83	643.96	0.46	687.52	690.71	1.65

test the performance of our approach, we have used the 9 benchmark instances available in the literature [11]. On each instance, we have executed our approach 40 independent times. We compare the results of our approach with those of LSGA [5], TS [3], MHDE [6] and DDE [11]. Results of these four approaches are taken from [11].

Table 1 compares the different approaches in terms of best &average solution quality and standard deviation of solution values on each of the 9 benchmark instances. In this table, result of our approaches are shown in bold whenever they are as good as or better than previous 4 approaches. From this table, it can be clearly seen that performance of our approach is more-or-less comparable to these 4 state-of-the-art approaches. Our approach obtain new best solution values for 4 instances.

Table 2 reports the time taken by various approaches to reach the best solution. As the four previous approaches were executed on an Intel Core Duo (T2300) based system which is different from the system used to execute our ABC approach, therefore times can not be compared precisely. However, a rough comparison can always be made. Even after compensating for difference in

Table 2. Time taken to reach the best solution by various algorithm

prob	LSGA	TS	MHDE	DDE	ABC
1	<1s	<1s	<1s	<1s	<1s
2	<1s	<1s	<1s	<1s	<1s
3	<1s	<1s	<1s	<1s	<1s
4	<1s	<1s	<1s	<1s	<1s
5	<1s	<1s	<1s	<1s	<1s
6	1s	<1s	<1s	<1s	<1s
7	1s	1s	2s	<1s	<1s
8	7s	1s	10s	2s	<1s
9	7s	2s	15s	3s	<1s

Table 3. Influence of parameter settings on solution quality

Parameter	Value	Problem 4		Problem 5	
		BestF	AvgF	BestF	AvgF
n_e	25	286.61	286.87	335.07	336.19
	50	**286.61**	**286.65**	**335.07**	**336.23**
	75	286.61	286.65	335.07	336.13
	100	286.61	286.62	335.07	336.12
n_o	50	286.61	286.67	335.69	336.44
	75	286.61	286.68	335.69	336.41
	100	**286.61**	**286.65**	**335.07**	**336.23**
	125	286.61	286.69	335.56	336.20
p_{asn}	0.75	286.61	286.79	335.71	336.29
	0.8	286.61	286.76	335.07	336.19
	0.85	**286.61**	**286.65**	**335.07**	**336.23**
	0.9	286.61	286.67	335.07	336.31
	0.95	286.61	286.68	335.07	336.27
p_{onl}	0.75	286.61	286.68	335.07	336.25
	0.8	286.60	286.67	335.07	336.21
	0.85	**286.61**	**286.65**	**335.07**	**336.23**
	0.9	286.61	286.70	335.69	336.30
	0.95	286.61	286.66	335.07	336.11
p_{rand}	0.1	286.61	286.71	335.84	336.44
	0.15	286.61	286.74	335.69	336.33
	0.2	**286.61**	**286.65**	**335.07**	**336.23**
	0.25	286.61	286.69	335.07	336.16
	0.3	286.61	286.77	335.69	336.32
limit	300	286.61	286.68	335.52	336.12
	400	286.61	286.62	335.19	336.16
	500	**286.61**	**286.65**	**335.07**	**336.23**
	600	286.61	286.72	335.07	336.20

processing speed, we can safely say that our approach is faster on large instances. Our approach requires less than 1 s to reach the best solution on all 9 instances.

To investigate the influence of parameter settings on solution quality, we have taken two different instances, viz. Problem 4 and 5. We have varied all the parameters one by one while keeping all other parameters unchanged. In doing so all other parameters were set to their values reported at the start of this section. The results are reported in Table 3. Values in bold in this table show the results with original parameter values which are used in all the experiments involving our approach. From this table it can be seen that values chosen by us

provide either the best results or results which are very close to best results. In those cases where we have not got the best results with chosen parameter values, the parameter values chosen have provided best results on some other instances not included in this table.

6 Conclusions

In this paper, we have proposed an ABC algorithm based approach for the TA problem and compared it with the best methods proposed in the literature. The results shows the performance of the ABC algorithm is comparable with other methods. ABC algorithm provides good quality solutions in lesser execution times for majority of the instances.

As a future work, we plan to improve the performance of our ABC approach by hybridizing it with some local search procedure. We intend to investigate the performance of our ABC algorithm under different solution encoding schemes. Approaches similar to our ABC approach can be developed for other $\mathcal{NP}$-Hard assignment problems.

References

1. Karaboga, D., Basturk, K.: On the performance of artificial bee colony (ABC) algorithm. Appl. Soft Comput. **8**, 687–697 (2008)
2. Singh, A.: An artificial bee colony algorithm for the leaf constrained minimum spanning tree problem. Appl. Soft Comput. **9**, 625–631 (2009)
3. Bernardino, E., Bernardino, A., Sanchez- Perez, J., Vega-Rodriguez, M., Gomez-Pulido, J.: Tabu search vs hybrid genetic algorithm to solve the terminal assignment problem. In: International Conference on Applied Computing, pp. 404–409 (2008)
4. Xu, Y., Salcedo-Sanz, S., Yao, X.: Non-standard cost terminal assignment problems using tabu search approach. IEEE Conf. Evol. Comput. **2**, 2302–2306 (2004)
5. Bernardino, E., Bernardino, A., Sanchez-Perez, J., Vega-Rodriguez, M., Gomez-Pulido, J.: Solving the terminal assignment problem using a local search genetic algorithm. In: International Symposium on Distributed Computing and Artificial Intelligence, pp. 225–234, Springer, Heidelberg (2008)
6. Bernardino, E., Bernardino, A., Sanchez-Perez, J., Vega-Rodriguez, M., Gomez-Pulido, J.: A hybrid differential evolution algorithm with a multiple strategy for solving the terminal assignment problem. In: 6th Hellenic Conference on Artificial Intelligence. Springer, Heidelberg (2010)
7. Khuri, S., Chiu, T.: Heuristic algorithms for the terminal assignment problem. In: Proceedings of the ACM Symposium on Applied Computing, pp. 247–251 (1997)
8. Salcedo-Sanz, S., Yao, X.: A hybrid Hopfield network genetic algorithm approach for the terminal assignment problem. IEEE Trans. Syst. Man Cybern. **34**(6), 2343–2353 (2004)
9. Abuali, F., Schoenefeld, D., Wainwright, R.: Terminal assignment in a communications network using genetic algorithms. In: Proceedings of the 22nd Annual ACM Computer Science Conference, pp. 74–81. ACM press, Newyork (1994)

10. Singh, A., Sunder, S.: An artificial bee colony algorithm for the minimum routing cost spanning tree problem. Soft Comput. **15**, 2489–2499 (2011)
11. Bernardino, E.M., Bernardino, A.M., Sánchez-Pérez, J.M., Gómez-Pulido, J.A., Vega-Rodríguez, M.A.: Discrete differential evolution algorithm for solving the terminal assignment problem. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6239, pp. 229–239. Springer, Heidelberg (2010)
12. Karaboga, D., Gorkemli, B., Ozturk, C., Karaboga, N.: A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artif. Intell. Rev. **42**, 21–57 (2014)
13. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Global Optim. **39**, 459–471 (2007)
14. Karaboga, D., Basturk, B.: Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems. In: Melin, P., Castillo, O., Aguilar, L.T., Kacprzyk, J., Pedrycz, W. (eds.) IFSA 2007. LNCS (LNAI), vol. 4529, pp. 789–798. Springer, Heidelberg (2007)
15. Bernardino, E.M., Bernardino, A.M., Sánchez-Pérez, J.M., Gómez-Pulido, J.A., Vega-Rodríguez, M.A.: Using the bees algorithm to assign terminals to concentrators. In: García-Pedrajas, N., Herrera, F., Fyfe, C., Benítez, J.M., Ali, M. (eds.) IEA/AIE 2010, Part II. LNCS, vol. 6097, pp. 267–276. Springer, Heidelberg (2010)
16. Bernardino, E.M., Bernardino, A.M., Sánchez-Pérez, J.M., Gómez-Pulido, J.A., Vega-Rodríguez, M.A.: A Hybrid differential evolution algorithm for solving the terminal assignment problem. In: Omatu, S., Rocha, M.P., Bravo, J., Fernández, F., Corchado, E., Bustillo, A., Corchado, J.M. (eds.) IWANN 2009, Part II. LNCS, vol. 5518, pp. 179–186. Springer, Heidelberg (2009)
17. Pan, Q.-K., Tasgetiren, M. F., Liang, Y.C.: A discrete differential evolution algorithm for the permutation flowshop scheduling problem. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, pp. 126–133 (2007)
18. Pan, Q.-K., Tasgetiren, M.F., Suganthan, P.N., Chen, A.H.-L.: A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops. Inf. Sci. **181**, 3459–3475 (2011)
19. Kershenbaum, A.: Telecommunications Network Design Algorithms. McGraw-Hill, New York (1993)
20. Bose, D., Kundu, S., Biswas, S., Das, S.: Circular antenna array design using novel perturbation based artificial bee colony algorithm. In: Panigrahi, B.K., Das, S., Suganthan, P.N., Nanda, P.K. (eds.) SEMCCO 2012. LNCS, vol. 7677, pp. 459–466. Springer, Heidelberg (2012)
21. Biswas, S., Kundu, S., Bose, D., Das, S., Suganthan, P.N., Panigrahi, B.K.: Migrating forager population in a multi-population artificial bee colony algorithm with modified perturbation schemes. In: 2013 IEEE Symposium on Swarm Intelligence (SIS 2013), pp. 248–255 (2013)
22. Bose, D., Biswas, S., Vasilakos, A.V., Laha, S.: Optimal filter design using an improved artificial bee colony algorithm. Inf. Sci. **281**, 443–461 (2014)