

Solving the Regenerator Location Problem using Swarm Intelligence Algorithms

Pedro Ferreira

*School of Technology and Management
Polytechnic Institute of Leiria
Leiria, Portugal
2151163@my.ipleiria.pt*

Rodrigo Pessoa

*School of Technology and Management
Polytechnic Institute of Leiria
Leiria, Portugal
2150662@my.ipleiria.pt*

Anabela Bernardino

*School of Technology and Management
Computer Science and Communication
Research Centre
Polytechnic Institute of Leiria
Leiria, Portugal
anabela.bernardino@ipleiria.pt*

Eugénia Bernardino

*School of Technology and Management
Computer Science and Communication
Research Centre
Polytechnic Institute of Leiria
Leiria, Portugal
eugenia.bernardino@ipleiria.pt*

Beatriz Piedade

*School of Technology and Management
Computer Science and Communication
Research Centre
Polytechnic Institute of Leiria
Leiria, Portugal
beatriz.piedade@ipleiria.pt*

Alexandrino Gonçalves

*School of Technology and Management
Computer Science and Communication
Research Centre
Polytechnic Institute of Leiria
Leiria, Portugal
alex@ipleiria.pt*

Abstract—In optical networks, due to deficiencies in the fibre (attenuation, dispersion, conversion) the signal strength deteriorates as it gets further away from its source. For that reason, the distance that a signal can travel without losing or corrupting information is limited. Therefore, it is necessary to regenerate the signals periodically using regenerators. Given an optical network, the regenerator location problem tries to install a subset of regenerators with the minimum possible cost, in a way that each pair of nodes can communicate with each other. In this paper, two swarm optimisation algorithms: Ant Colony Optimisation and Bees Algorithm are used to solve this problem. We compare our results with other algorithms used in literature to solve the same problem. Simulation results obtained using 480 different instances prove their efficiency in solving the regenerator location problem.

Keywords—Swarm Optimisation Algorithms, Regenerator Location Problem, Optical Networks.

I. INTRODUCTION

With the evolution of communication technologies, there is a growing need for a fast, secure and flawless service. In response to this need, the fibre optic networks have arisen. However, the further away the optical signal is from its source, the more it is subjected to loss of quality. This happens due to fibre faults. In order to solve this problem, there are equipment, called regenerators, whose purpose is to restore signal quality [1]. Unfortunately, these devices have a very high cost.

The objective of this work is to solve the Regenerator Location Problem (RLP). RLP seeks to place the minimum number of regenerators in an optical network to facilitate the communication of a signal between any node pair. Our goal is to obtain solutions with the lowest possible cost, in the lower computational time [2] [3] [4] [5].

The problem studied is classified as NP-Complete (Non-deterministic Polynomial time) [2] [4]. To solve this type of problems, approximation algorithms can be used to find a good solution, not necessarily optimal, however within a certain range of error. In some cases, finding a good approximation is enough to solve the problem, but not all NP-Complete problems have good approximation algorithms. In many cases metaheuristics such as evolutionary algorithms

and swarm optimisation algorithms can be used, finding good solutions in a reasonable computational time.

In order to solve the problem, two algorithms were implemented: Ant Colony Optimisation (ACO) and Bees Algorithm (BA) [6] [7]. These modern search paradigms rely on the collective intelligence of many single agents to find good solutions. These algorithms were applied with success by the same authors to other optimisations problems and that was the motivation to apply them to solve the RLP.

This paper presents the first attempt (to the authors' knowledge) to use a bees inspired algorithm to optimise RLP. 480 instances were used to prove the efficiency of the algorithms in terms of solution cost and response time.

We compare the performance of our algorithms with other heuristics from literature implemented by the same authors.

This article is organised as follows: Section 2 describes the problem; in section 3 the applied algorithms are described; Section 4 presents the results; and finally, in section 5, the conclusions and directions for future work are presented.

II. REGENERATOR LOCATION PROBLEM

The RLP intends to ensure that in a fibre optic network the signals are regenerated in order to maintain the signal quality, which deteriorates proportionally to the distance it travels through the network. The problem was formally defined in [2]. The RLP proved to be NP-Complete [2] [4]. As far as we know, the best algorithm in the state of the art for the RLP was presented in [5] - the Greedy Randomized Adaptive Search Procedure (GRASP) algorithm.

A generalised version of this problem, called Generalized Regenerator Localization, was recently proposed by Chen et al. [8]. In [9] a GRASP algorithm was proposed to solve the problem.

The instances used are in the format of text files and were obtained from the website of the Computer Centre of the University of Vienna [3]. The main information contained in these files is a matrix that represents the links between all the nodes of a problem. It also contains the number of nodes and the number of connections. Some of the instances also include a vector with weights, referring to the cost of placing a regenerator in each specific node.

In order to illustrate the problem, we present an example of an instance with 7 nodes, in Fig. 1.

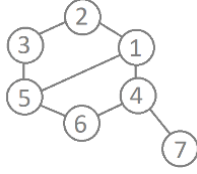


Fig. 1. Instance with 7 nodes

To find the solution to this problem it is necessary to discover an arrangement of signal regenerators in a set of interconnected nodes, so that all nodes contain, or, are connected to at least one regenerator. This process will have to be done using as few regenerators as possible (random instances). In the case of problems in which the placement of regenerators in a specific node has different costs (weighted instances), the objective is to have the lowest possible cost.

Fig. 2 represents the text file of an instance with weights (weighted instance) and Fig. 3 an instance without weights (random instance).

```
nNodes = 7
nPairs = 8
Weight
3
2
3
4
2
3
4
Cost
1 1 0 1 1 0 0
1 1 1 0 0 0 0
0 1 1 0 1 0 0
1 0 0 1 0 1 1
1 0 1 0 1 1 0
0 0 0 1 1 1 0
0 0 0 1 0 0 1
NDC Pairs
<1 2>
<1 4>
<1 5>
<2 3>
<3 5>
<4 6>
<4 7>
<5 6>
```

Fig. 2. Instance with weights

Each instance file starts with “nNodes” which is the number of total nodes and then “nPairs” which is the number of not directly connected (NDC) node pairs, followed by a $nNodes \times nNodes$ matrix, denoted by “Cost”, and followed by a list of NDC node pairs, denoted by “UnPairedArc”. If the instances are weighted, before providing the “Cost” matrix, a list of node weights, denoted by “Weight” is given [3].

All nodes must be connected to a regenerator. The regenerator can be on the node itself or on one of the adjacent nodes, that is, on one of the nodes connected to that node.

Fig. 4 shows the representation of an infeasible solution (node 7 is not connected to a regenerator), a non-optimised

solution (3 regenerators) and an optimised solution (2 regenerators). To represent the solutions, we used a binary vector with a number of positions equal to the total number of nodes. If a position has the value 1 it means that the corresponding node has a regenerator.

```
nNodes=7;
nPairs=8;
Cost=
1 1 0 1 1 0 0
1 1 1 0 0 0 0
0 1 1 0 1 0 0
1 0 0 1 0 1 1
1 0 1 0 1 1 0
0 0 0 1 1 1 0
0 0 0 1 0 0 1
UnPairedArc=
1 2
1 4
1 5
2 3
3 5
4 6
4 7
5 6
```

Fig. 3. Instance without weights

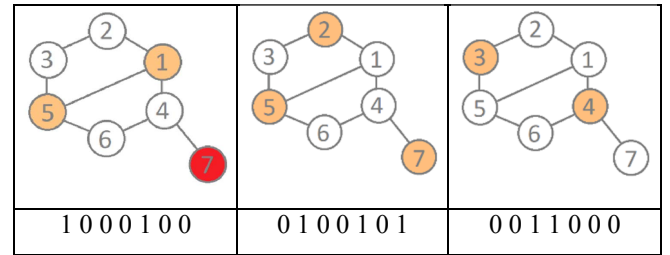


Fig. 4. Representation of the solution a) infeasible, b) non-optimised, c) optimised

To determine the quality of the solution a fitness value is calculated using one of the following equations:

$$\text{fitness} = DN * 500 + RU * 100 \quad (1)$$

or

$$\text{fitness} = DN * 500 + \sum RU W_i * 100 \quad (2)$$

Equation (1) is used for problems without weights and (2) for problems with weights. DN corresponds to the number of nodes disconnected, RU to the number of regenerators used and W_i corresponds to the weight of the regenerator i .

The goal is to minimise the fitness value and to find a solution where all nodes are connected to a regenerator. In instances without weights, the goal is only to reduce the number of regenerators used. In the instances with weights it is intended at the same time to reduce the number of regenerators and to place the regenerators in the nodes with a lower weight.

III. ALGORITHMS

The area of Swarm Intelligence (SI) relies on the collective intelligence of agents that interactively explore the search space. Some of the best-known areas of swarm intelligence are ACO, Particle Swarm Optimisation and bees-inspired algorithms [6] [7].

The ACO algorithm is essentially a system based on agents which simulate the natural behaviour of ants, including mechanisms of cooperation and adaptation. The ACO algorithm was presented by Dorigo, Maniezzo and Colnani [10] [11], and Dorigo [12]. This metaheuristic has shown to be both robust and versatile. The ACO algorithm and related variants have been successfully applied to a range of different CO problems [13] [14] [15] [16].

Honey bees live in hives around the world in well organised colonies. These colonies are characterised by the division of labour, where specific bees perform specific tasks. Bees' swarming around their hive is another example of SI [7]. A review of literature on algorithms inspired by the behaviour of bees [7] suggests that the topic is evolving and that there is no consensus on a single descriptive title for metaheuristics based on bees' behaviour. In literature, it is possible to find several bees inspired algorithms that use different algorithm models [7]. In this paper we consider the BA. This algorithm simulates the foraging behaviour of bees.

A. Bees Algorithm

The BA proposed by Pham et al. [15] is a SI algorithm which mimics the foraging behaviour of honey bees. In its basic version, the algorithm performs a kind of neighbourhood search combined with a random search. For neighbourhood selection, the smallest fitness bees are chosen as selected bees. For recruitment, bees are assigned based on the fitness values associated with the sites they are visiting. Only the best bees are selected in order to form the next generation of the colony.

Our work presents the first application of BA to solve the RLP. Based on previous experiences, performed by some authors of this paper, with the application of BA in solving SONET ring problems [16] and the Terminal Assignment Problem [17] we decide to apply it to RLP.

1) Algorithm description

In BA, after creating the initial population of bees (ns), bees are ranked according to the fitness values of their sites [18]. The best nss of the ns sites are classified as "selected sites", and the best nbs of the nss sites are classified as "best sites". The $nbbs$ bees are sent to the "best sites", and the $nbbs$ bees are sent to the remaining ($nss-nbs$) "selected sites". These bees ($nbbs$ and $nbbs$) produce new sites in the neighbourhood of the "selected sites". Searches in the neighbourhood of the best sites, which represent more promising solutions, are made by recruiting more bees to follow them than the other selected bees. The remaining bees ($ns-nss$) are classified as scout and randomly assigned. The basic form of the BA uses a neighbourhood search to explore through the selected sites [15].

The algorithm conducts searches in the neighbourhood of the selected sites, assigning more bees to search near to the best nbs sites.

A neighbour is obtained by performing multiple moves to find a new solution. The algorithm repeats the neighbourhood search until at least one exchange with improvement is made or until a predeterminate number of modifications (nm) is reached. Optimising a solution consists of removing or adding a regenerator until the individual's fitness improves.

If the fitness does not improve, the individual is not changed. The probability of removing a regenerator is twice the probability of adding a regenerator. Optimisation can be done in two ways: consider any regenerator (connected or

disconnected) as an optimisation target or, as the number of modifications increases, the same number of best and worst regenerators are considered to be modified (the best regenerators are those that are disconnected and can connect more nodes and, the worse regenerators are those that are connected and connect less nodes).

The main steps of the BA are given below.

```

Initialization of parameters
Initialise Solutions (bees)
Evaluate Solutions
For each iteration
    Compute Probabilities
    For each selected bee
        Calculate the number of
            optimisation attempts based
            on probabilities
        For each attempt
            Optimise solution
    Evaluate the new population
Return the best solution found

```

For more details about the main steps and the parameters used by this algorithm see [16] [17].

B. Ant Colony Optimisation

ACO is a population-based optimisation method to solve hard combinatorial optimisation problems. Since the first ACO algorithm was presented by Dorigo, Maniezzo and Colnani [10] [11], Dorigo [12] many diverse variants of the basic principle have been reported in literature [13].

ACO is essentially a system based on agents that simulate the natural behaviour of ants, including mechanisms of cooperation and adaptation. This metaheuristic has shown to be both robust and versatile. Our algorithm is based on the algorithm proposed by Gambardella et al. [14] to solve the Quadratic Assignment Problem (QAP). The ACO uses pheromone trail information to perform modifications on RLP solutions, unlike traditional ant systems that use pheromone trail information to construct complete solutions.

1) Algorithm description

In real life, ants indirectly communicate among them (with each other) by depositing pheromone trails on the ground, influencing the decision processes of other ants. This simple form of communication between individual ants causes complex behaviours and capabilities of the colony as a whole. The real ants behaviour is transposed into an algorithm by making an analogy between:

- The real ants search and the set of feasible solutions to the problem;
- The amount of food in a source and the fitness function;
- The pheromone trail and an adaptive memory [14].

The pheromone trails in ACO serve as a distributed, numerical information which the ants use to probabilistically construct solutions to the problem to be solved and which they adapt during the algorithm execution to reflect their search experience.

The simplest way to exploit the ants search experience is to make the pheromone updating process a function of the solution quality achieved by each particular ant. In ACO, only the best solution found during the search process contributes to pheromone trail updating. This makes the search more

aggressive and requires less time to reach good solutions. Moreover, this has been strengthened by an intensification mechanism that allows it to return to previous best solutions [14].

The algorithm proposed by Gambardella et al. [14] also performs a diversification mechanism after performing a predefined number of iterations without improving the best solution found so far. This algorithm was applied with success to the SONET ring problems and TAP [17] [19].

The algorithm performs nm modifications. A modification consists of putting / taking off a regenerator of a node. First, a node n is randomly chosen (between 1 and m) and after is decided to put or not a regenerator (yes – 1, no – 0) – value r . A random number x is generated between 0 and 1. If x is smaller than q (parameter), the best value r is chosen in a way that Trn is the maximum. This policy consists in exploiting the pheromone trail. If x is higher than q , the value r is chosen with a probability proportional to the values contained in the pheromone trail. This consists in exploring the solution space.

The main steps of the ACO are given bellow.

```

Initialise Parameters
Initialise Solutions (ants)
Evaluate Solutions
Initialise Pheromone Trails
For each iteration
    FOR each Solution in Population
        Modify Solution using Pheromone Trails
        Evaluate Solution
        Apply Intensification Mechanism
    Evaluate population
    Update Pheromone Trails
Output the best solution

```

For more details about the equations used to initialise / update pheromone trails, description of the main steps and the parameters used in the algorithm see [14] [17] [19].

IV. RESULTS

The instances used to solve the RLP are available on the website of the computer centre of the University of Vienna, as previously mentioned [3].

The results presented in this section were obtained by running each algorithm using 500 iterations / generations, with an Intel i5-8600k processor running at 4.6GHz frequency. The results obtained were treated with the RapidMiner tool and are presented in the form of tables.

A. Test results for parameters

Different combinations of parameters were tested for each algorithm. These different combinations were applied to 36 instances of different dimensions.

Based on preliminary observations, the best intervals for each input parameter were determined for each algorithm and based on these intervals were performed several automated tests that use different parameter values. The objective was to determine the best combination of parameters to perform the final tests and compare the performance of the two algorithms.

In order to obtain the best BA parameter values, the following interval values were used:

- Selected Value: from 20 to 50 with increments of 10;
- Best Value: from 30 to 60 with increments of 10;

- Change Value: from 3 to 7 with increments of 1.

The five best combinations of parameters for the BA are presented in Table I (D. - Disconnected nodes).

TABLE I. BA FIVE BEST COMBINATION OF PARAMETERS

Select	Best	Change	Time(ms)	Regenerators	D.
30	50	5	3001.944	4.666666667	0
50	60	4	5773.667	4.666666667	0
40	50	5	5588.444	4.666666667	0
50	50	7	1860.083	4.694444444	0
30	60	6	1979.583	4.694444444	0

To obtain the best combination of parameters for the ACO, the following interval values were used:

- Q probability: from 60% to 100% with increments of 20%;
- Modifications: 5 to 7 with increments of 1;
- Evaporation rate: from 30% to 70% with increments of 20%;
- Influence: from 60% to 100% with increments of 20%.

In Table II ($Qprob.$ – Q probability, $M.$ – Modifications, $Evap.$ – Evaporation rate, $Inf.$ – Influence rate, $D.$ – Disconnected nodes) is presented the five best combinations of parameters for the ACO.

TABLE II. ACO FIVE BEST COMBINATION OF PARAMETERS

Qprob.	M.	Evap.	Inf.	Time(ms)	Regenerators	D.
80	5	50	60	733.3889	4.861111111	0
60	5	50	80	1997.361	4.861111111	0
60	5	50	100	1999.028	4.861111111	0
80	5	50	80	1225.833	4.888888889	0
80	5	50	100	1230.194	4.888888889	0

Based also on preliminary observations the initial size of the population was set to 50 in ACO and 40 in BA.

B. Results for all instances

In this section we present the results obtained using 480 instances (280 random instances and 200 weighted instances) with 50 different seeds: The combinations of parameters are presented in Table III.

TABLE III. PARAMETER VALUES

ACO	BA
Population: 50	Population: 40
Iterations: 500	Iterations: 500
Q probability: 80%	Select Size: 20
Q: 35	Best Size: 10
Modifications: 5	Select Value: 30
Evaporation rate: 50%	Best Value: 50
Influence rate: 60%	Change Value: 5
Seeds: [2, 100]	Seeds: [2, 100]

Tables IV and V present the average results obtained for each group of problems (*GP*) – random and weighted (*Regen.* – Average number of regenerators and *Disc.* – Disconnected nodes by the BA).

TABLE IV. BA RESULTS FOR RANDOM INSTANCES

Size	GP	Time(ms)	Fitness	Regen.	Disc.
40	1	2623.128	1081	10.81	0
40	2	1466.244	602.8	6.028	0
40	3	381.772	450.2	4.502	0
40	4	167.908	380	3.8	0
40	5	83.19	300	3	0
40	6	74.468	300	3	0
40	7	68.824	200	2	0
40	8	64.652	200	2	0
40	9	63.036	200	2	0
60	1	10748.53	1139.6	11.396	0
60	2	3159.484	701.6	7.016	0
60	3	747.09	500	5	0
60	4	283.144	400	4	0
60	5	3650.714	314.4	3.144	0
60	6	90.18	300	3	0
60	7	378.57	210	2.1	0
60	8	74.18	200	2	0
60	9	73.254	200	2	0
80	1	26429.82	1282.8	12.828	0
80	2	17422.69	737.6	7.376	0
80	3	9921.9	525.4	5.254	0
80	4	8733.152	408.4	4.084	0
80	5	8692.816	352.2	3.522	0
80	6	118.148	300	3	0
80	7	230.792	280	2.8	0
80	8	83.828	200	2	0
80	9	75.30924	200	2	0
100	1	39227.02	1353	13.53	0
100	2	19045.99	784	7.84	0
100	3	7832.966	590.6	5.906	0
100	4	11006.81	454.8	4.548	0
100	5	235.316	400	4	0
100	6	290.66	300	3	0
100	7	1053.738	270.2	2.702	0
100	8	196.108	200	2	0
100	9	80.628	200	2	0

Table VI presents the average results obtained for the weighted instances by the ACO. Because of the available space of this article and since the ACO results are slightly poor in comparison with the BA we do not present the ACO results obtained for the random instances for each GP.

TABLE V. BA RESULTS FOR WEIGHTED INSTANCES

Size	GP	Time(ms)	Fitness	Regen.	Disc.
40	1	2640.256	2984.4	11.094	0
40	3	468.35	1060	4.714	0
40	5	319.794	670	3.022	0
40	7	63.806	470	2.132	0
40	9	46.198	400	2	0
60	1	12428.27	2936.4	11.756	0
60	3	9988.838	1141.6	5.202	0
60	5	973.772	741.2	3.502	0
60	7	351.09	520	2.376	0
60	9	63.964	400	2	0
80	1	26007.45	3275	13.256	0
80	3	16247.17	1274.8	5.896	0
80	5	2714.716	782.2	3.822	0
80	7	234.65	600	2.986	0
80	9	75.51807	400	2	0
100	1	43405.91	3382.8	13.958	0
100	3	22089.21	1337	6.17	0
100	5	8551.19	803	4	0
100	7	431.136	590	2.9	0
100	9	183.85	400	2	0

TABLE VI. ACO RESULTS FOR WEIGHTED INSTANCES

Size	GP	Time(ms)	Fitness	Regen.	Disc.
40	1	1682.534	3012.4	11.132	40
40	3	683.708	1083	4.708	40
40	5	970.106	781	3.384	40
40	7	259.832	544	2.532	40
40	9	147.856	406.2	2	40
60	1	8609.668	3055.2	11.77	60
60	3	7654.778	1281.4	5.46	60
60	5	6795.852	882.6	3.938	60
60	7	3320.496	603.2	2.86	60
60	9	1522.784	422.2	2	60
80	1	19679.34	3640.8	13.646	80
80	3	23094.08	1468.6	6.074	80
80	5	15789.94	939	4.034	80
80	7	5147.544	650.6	3	80
80	9	1583.245	435.94	2	80
100	1	40550.72	3911.1	14.660	100
100	3	55088.56	1655.8	6.702	100
100	5	30588.99	1042.4	4.232	100
100	7	9464.18	678.6	3.002	100
100	9	1140.922	421.8	2	100

Table VII presents the average results for each size of problems, using the BA and the ACO algorithm. The algorithm that presents the best results in terms of the quality of the obtained solution is the BA. Were obtained valid solutions (solutions where all nodes are connected to at least one regenerator) for all random and weighted instances.

TABLE VII. RESULTS FOR SEVERAL GROUPS OF PROBLEMS

Size	BA		ACO	
	Time(ms)	Fitness	Time (ms)	Fitness
40	609.40	664.17	364.75	685.87
60	3072.22	693.20	2777.48	743.83
80	8356.28	758.46	6901.07	837.18
100	10973.61	790.39	16118.50	904.81

We also compare our results with other algorithms from literature that were implemented and tested by the same authors using the same equipment and platform – the Custom Algorithm (CA) and the Genetic Algorithm (GA) [20]. The comparisons between the different algorithms can be seen in Table 7, where the average results for each problem size are presented. Both implemented algorithms were able to find the best-known solutions in the literature. However, the BA obtained better solutions in terms of fitness.

TABLE VIII. AVERAGE FITNESS FOR THE DIFFERENT ALGORITHMS

Algorithms	CA	GA	BA	ACO
Fitness	738.99	777.28	726.55	792.92
Regenerators	4.92	5.19	4.82	5.07

In this article it was not possible to compare the results obtained with several other algorithms implemented by other authors to solve the same problem. To make an effective comparison with that algorithms we will need to implement the algorithms and test them on the same platform.

V. CONCLUSION

In this article we solve the Regenerator Location Problem. Two Swarm optimisation algorithms, BA and ACO, were used to solve the problem. We solved 480 instances (280 random and 200 weighted) and the results prove the effectiveness of both algorithms in solving the problem.

The results show that the best algorithm in terms of the solution quality is the BA. To our knowledge this was the first implementation of a BA to solve the RLP.

With more optimisation (better adaptations to the code of the algorithms) we think that it will be possible to improve the results obtained and also reduce the execution time. With the use of more iterations will also be possible to obtain better solutions, because the optimum solution is not always found in the number of iterations specified. For future work we also intend to compare the results with other algorithms already applied in the literature by other authors to the same problem.

ACKNOWLEDGMENT

This work was supported by national funds through the Portuguese Foundation for Science and Technology (FCT) under the project UID/CEC/04524/2019.

REFERENCES

- [1] E. Yetginer, and E. Karasan, "Regenerator Placement and Traffic Engineering with Restoration in GMPLS Networks," *Photonic Network Communications* 6 (2) (2003) 139–149.
- [2] S. Chen, I. Ljubic, and S. Raghavan, "The regenerator location problem," *Networks* 55 (3) (2010) 205–220.
- [3] S. Chen, I. Ljubic, and S. Raghavan, "The regenerator location problem," [Online]. Available: <http://homepage.univie.ac.at/ivana.ljubic/research/rlp/>.
- [4] M. Flammini, A. Marchetti-Spaccamela, G. Monaco, L. Moscardelli, and S. Zaks, "On the Complexity of the Regenerator Placement Problem in Optical Networks," *IEEE/ACM Trans. Netw.* 19 (2) (2011) 498–511.
- [5] A. Duarte, R. Martí, M. Resende, and R. Silva, "Improved heuristics for the regenerator location problem," *International Transactions in Operational Research* 21 (4) (2014) 541–558.
- [6] J. Kennedy, R.C. Eberhart, and Y. Shi, *Swarm intelligence*, 1st ed., Morgan Kaufmann, San Francisco, CA, 2001.
- [7] D. Karaboga, B. Basturk, *Artificial Bee Colony (ABC)*, "Harmony Search and Bees Algorithms on Numerical Optimization," in *Innovative Production Machines and Systems Virtual Conference*, 2009.
- [8] S. Chen, I. Ljubic and S. Raghavan, "The Generalized Regenerator Location Problem," *INFORMS Journal on Computing* 27 (2) (2015) 204–220.
- [9] J. Quintana, J. Sánchez-Oro, and A. Duarte, "Efficient Greedy Randomized Adaptive Search Procedure for the Generalized Regenerator Location Problem," *International Journal of Computational Intelligence Systems* 9 (6) (2016) 1016–1027.
- [10] M. Dorigo, V. Maniezzo, and A. Colomi, "Positive feedback as a search strategy," *Technical Report 91-016*, Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy, 1991.
- [11] M. Dorigo, V. Maniezzo, and A. Colomi, "The ant system: Optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics* 26(1) (1996) 29–41.
- [12] M. Dorigo, "Ottimizzazione, apprendimento automatico, ed algoritmi basati su metafora naturale (Optimisation, learning and natural algorithms)," *Doctoral dissertation*, Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy, 1991.
- [13] Ant Colony Optimization Website, <http://iridia.ulb.ac.be/dorigo/ACO/ACO.html>
- [14] L.M. Gambardella, E.D. Taillard, M. Dorigo, Ant colonies for the quadratic assignment problem, *Journal of the Operational Research Society* 50(2) (1999) 167–76.
- [15] D.T. Pham, A. Ghanbarzadeh, E. Koç, S. Otri, S. Rahim, and M. Zaidi, "The Bees Algorithm, Technical Note," *Manufacturing Engineering Centre*, Cardiff University, UK, 2005.
- [16] E.M. Bernardino, A.M. Bernardino, J.M. Sánchez-Pérez, M.A. Vega-Rodríguez, and J.A. Gómez-Pulido, "Solving large-scale SONET Network Design Problems using Bees-inspired Algorithms," *Optical Switching and Networking* 9(2) (2012) 97–117.
- [17] E.M. Bernardino, A.M. Bernardino, J.M. Sánchez-Pérez, M.A. Vega-Rodríguez, J.A. Gómez-Pulido, "Swarm optimisation algorithms applied to large balanced communication networks," *Journal of Network and Computer Applications* 36(1) (2013) 504–522.
- [18] D.T. Pham, A.A. Afify, E. Koç, Manufacturing cell formation using the Bees Algorithm, in: *Innovative Production Machines and Systems Virtual Conference*, Cardiff, UK, 2007.
- [19] A.M. Bernardino, E.M. Bernardino, J.M. Sánchez-Pérez, M.A. Vega-Rodríguez, J.A. Gómez-Pulido, "Solving ring loading problems using Bio-inspired algorithms," *Journal of Network and Computer Applications* 34(2) (2011) 668–685.
- [20] P. Ferreira, R. Pessoa, A.M. Bernardino, E.M. Bernardino, and B. Piedade, "Solving the Regenerator Location Problem using bio-inspired algorithms," in *proceedings of the 14th Iberian Conference on Information Systems and Technologies (CISTI)*, 2019.