# Set Covering Problem Resolution by Biogeography-Based Optimization Algorithm

Broderick Crawford[1,2,3], Ricardo Soto[1,4,5], Luis Riquelme[1(✉)], Eduardo Olguín[3], and Sanjay Misra[6]

[1] Pontificia Universidad Católica de Valparaíso, Valparaíso, Chile
lriquelme@outlook.com
[2] Universidad Central de Chile, Santiago, Chile
[3] Facultad de Ingeniería y Tecnología, Universidad San Sebastián, Santiago, Chile
[4] Universidad Autónoma de Chile, Santiago Metropolitan, Chile
[5] Universidad Cientifica del Sur, Lima, Peru
[6] Covenant University, Ota, Nigeria

**Abstract.** The research on Artificial Intelligence and Operational Research has provided models and techniques to solve many industrial problems. For instance, many real life problems can be formulated as a Set Covering Problem (SCP). The SCP is a classic NP-hard combinatorial problem consisting in find a set of solutions that cover a range of needs at the lowest possible cost following certain constraints. In this work, we use a recent metaheuristic called Biogeography-Based Optimization Algorithm (BBOA) inspired by biogeography, which mimics the migration behavior of animals in nature to solve optimization and engineering problems. In this paper, BBOA for the SCP is proposed. In addition, to improve performance we provide a new feature for the BBOA, which improve stagnation in local optimum. Finally, the experiment results show that BBOA is a excellent method for solving such problems.

**Keywords:** Biogeography-Based Optimization Algorithm · Set Covering Problem · Metaheuristics

## 1 Introduction

Different solving methods have been proposed in the literature to solve Combinatorial Optimization Problems. Exact algorithms are mostly based on Branch-and-Bound and Branch-and-Cut techniques, Linear Programing and Heuristic methods [2,16]. However, these algorithms are rather time consuming and can only solve instances of very limited size. For this reason, many research efforts have been focused on the development of heuristics to find good results or near-optimal solutions within a reasonable period of time.

In Artificial Intelligence (AI) heuristics are used, that in a very generic way, is a set of techniques or methods for solve a problem more quickly, finding an approximate solution when classic methods fail to find any exact solution. From


O. Gervasi et al. (Eds.): ICCSA 2016, Part I, LNCS 9786, pp. 153–165, 2016.
DOI: 10.1007/978-3-319-42085-1_12

this point of view, the heuristic is a procedure which tries to give good solutions, with quality and good performance.

Metaheuristics, as the prefix says, are upper level heuristics. They are intelligent strategies to design or improve general heuristic procedures with high performance. In their original definition, metaheuristics are general purpose approximated optimization algorithms; they find a good solution for the problem in a reasonable time (not necessarily the optimal solution). They are iterative procedures that smartly guide a subordinate heuristic, combining different concepts to suitably explore and operate the search space. Over time, these methods have also come to include any procedures that employ strategies for overcoming the trap of local optimum in complex solution spaces, especially those procedures that utilize one or more neighborhood structures as a means of defining admissible moves to transition from one solution to another, or to build or destroy solutions in constructive and destructive processes.

To get good solutions, any search algorithm must establish an adequate balance between two overlayed process characteristics:

- Intensity: Effort put on local space search (space exploitation).
- Diversity: Effort put on distant space search (space exploration).

This balance is needed to quickly identify regions with good quality solutions and to not spend time in promising or visited regions.

The metaheuristics are categorized based on the procedures types which it refers. Some of the fundamental types of metaheuristics are:

- Constructive heuristics: Start from an empty solution and go adding components until a solution is built.
- Trajectory methods: Start from an initial solution and then, iteratively, try to replace it with a better one from their neighborhood.
- Population-based methods: Iteratively evolve a solution-population.

One of the fairly new and existing metaheuristics is the Biogeography-Based Optimization Algorithm (BBOA). It is based on the behavior of natural migration of animals, considering emigration, immigration and mutation factors. This is a population algorithm for binary and real problems, and it's useful for maximizing and minimizing problems [28]. In general, BBOA is based on the concept of Habitat Suitability Index (HSI) which it is generated from the characteristics of an habitat, where the habitat that has better characteristics have a higher HSI and worst features, lower HSI. It is also considered that the more HSI have an habitat, more species inhabit it, contrary to lower HSI [28,36]. Each habitat also has a single rate of immigration, emigration and mutation probabilities, which come from the habitat number of species.

This metaheuristic is applied for solving the Set Covering Problem (SCP), whose aim is to cover a range of needs at the lowest cost, following certain restrictions on the context of the problem where the needs are constraints. SCP can be applied for location services, selection of files in a database, simplifying boolean expressions, slot allocation, among others [3]. Currently, there is extensive literature on methods for SCP resolutions. They are the exact methods as mentioned

in [2,16], and heuristic methods to solve a range of problems such in [21]. In case of SCP, this is solved by a variety of heuristics, so there is considerable literature. Among the metaheuristics that has tried to solve the SCP, they are: hybrid algorithms [15], hybrid ant algorithm [13], binary cat swarm optimization [7], bat algorithm [10], cuckoo search [30], artificial bee colony algorithm [8], binary firefly algorithm [11], among others.

BBOA has been used to solve other problems of optimization, among them are the classic and one of the most important optimization problems: The Traveling Salesman Problem of NP-hard class, which it is to find the shortest route between a set of points, visiting them all at once and returning to the starting point [26]. This was solved by using BBOA in [25], demonstrating that behaves very effectively for some combinations of optimization and even outperforms other traditional methods inspired by nature. Also, BBOA has been used to solve constraint optimization problems such as in [23], where indicate that BBO generally performs better than Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) in handling constrained single-objective optimization problems. Undoubtedly, the BBOA is a method that may have great potential to solve the SCP.

The remaining of this document is structured as follows: a description of the SCP, then the technique (BBOA) used to solve SCP. Then, the changes to the algorithm to relate and integrated at problem. Subsequently, results of experiments comparing with known global optimums and, finally, the corresponding conclusions.

## 2 Set Covering Problem

The Set Covering Problem is a popular NP-hard problem [19] that has been used to a wide range of airlines and buses crew scheduling [29], location of emergency facilities [32], railway crew management [5], steel production [33], vehicle scheduling [18], ship scheduling [17], etc.

The SCP consists of finding a set of solutions which covers a range of needs at the lowest cost. In a zero-one matrix view ($a_{ij}$), the needs correspond to m-rows (constraints), while the whole solution is the selection of n-columns that optimally cover the rows. Among the real-world applications in which it applies are: location of emergency facilities, steel production, vehicle routing, network attack or defense, information retrieval, services location, among others [3].

The SCP was also successfully solved with metaheuristics such as taboo search [6], simulated annealing [4,31], genetic algorithm [20,22,24], ant colony optimization [1,24], swarm optimization particles [9], artificial bee colony [12,35] and firefly algorithms [14].

### 2.1 Formal Definition

The SCP is mathematically modeled as follows.

$$Minimize\, Z = \sum_{j=1}^{n} c_j x_j. \tag{1}$$

Subject to:

$$\begin{aligned} \sum_{j=1}^{n} a_{ij} x_j \geq 1 \qquad & \forall i \in \{1, 2, 3, ..., m\} \\ x_j \in \{0, 1\} \qquad & \forall j \in \{1, 2, 3, ..., n\}. \end{aligned} \tag{2}$$

where Eq. (1) minimizes the number of sets, analogous to obtain the minimum cost ($c_j$). Subject to Eq. (2), ensuring that each m-row is covered by at least one n-column. Where the domain constraint $x_j$ is 1 if the column belongs to solution and 0 otherwise.

## 3 Biogeography-Based Optimization Algorithm

Biogeography studies the migration between habitats, speciation and extinction of species. Simon (2008) proposes the BBOA by mathematical models of biogeography made in the 1960s [28]. This says that areas that are well adapted as a residence for biological species have a high HSI. Some features are related to this index; precipitation, vegetation diversity, diversity of topography, land surface and temperature. Variables that characterize the habitability are called Suitability Index Variables (SIV). SIVs can be considered the independent variables of the habitat, and HSI can be considered the dependent variable [28].

Then, based on the species number, it is possible to predict the rate of immigration and emigration: habitats that are more HSI have higher rate of emigration, since the big population causes that species migrate to neighboring habitats. They also have a low inmigration rate because they are already nearly saturated with species. Furthermore, habitats with a low HSI have a high species immigration rate because of their sparse populations and a high rate of emigration, as conditions cause rapid way or species extinction. This behavior is shown in Fig. 1.

Where $I$ and $E$ are the highest rates of immigration and emigration, the same for simplicity. $S_{max}$, the maximum amount of species and $S_0$ the equilibrium number of species. Finally, $\lambda$ is the inmigration rate and $\mu$ is the emigration rate. $k$ is the habitat species number.

### 3.1 Migration Operator

As mentioned in biogeography, species can migrate between habitats. In BBO, the characteristics of the solutions may affect others and themselves, using immigration and emigration rates to share information between them probabilistically.

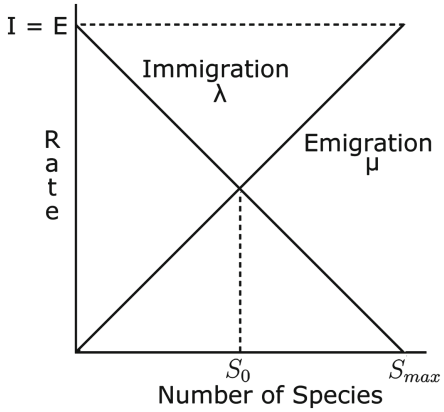


**Fig. 1.** Species model of a single habitat

In BBOA and based on Fig. 1, immigration curve is used to probabilistically decide whether or not to immigrate each feature in each solution. If a characteristic of solution is selected to immigrate, a solution to migrate one of its features are probabilistically selected randomly. Based on above description the main steps in the BBOA are detailed in Algorithm 1. Also, note that "probability $\lambda_i$" and "probability $\mu_j$" are a random number (0 to 1) compared with the respective rate.

**Algorithm 1.** Migration operator

```
{N the size of the population}
for i=1 to N do
  Select H_i with probability λ_i
  if H_i is selected then
    {D the solution length}
    for k=1 to D do
      Select H_j with probability μ_j
      if H_j is selected then
        Select random k ϵ [1, D]
        Set H_ik = H_jk
      end if
    end for
  end if
end for
```

We note that the best solutions are the least likely to immigrate characteristics, since immigration rates are lower. Opposite of this the solutions with lower fitness are more likely to immigrate, given their high rates of immigration. However, the solutions they provide to their emigration to these worst solutions are those having good fitness for its high rate of emigration [36].

### 3.2 Mutation Operator

A natural habitat may be affected by cataclysmic events drastically changing its HSI. This could cause a count of species that is different from its equilibrium value (species arriving from neighboring habitats, diseases, natural disasters and others). Thus, the HSI of habitat could suddenly change due to random events. In BBOA likely number of species ($P_s$) is used to determine mutation rates. These are determined by the balance between immigration and emigration rates (Fig. 1) as a balance between these rates, the probability that $S$ number of species is greater: immigrating species at a rate that is similar to the number of species that migrate in the same habitat. Given that, the best and worst habitats are less likely to have $S$ number of species. This is mentioned in detail in [28]. Then, the mutation rate is calculated as Eq. (3)

$$m(s) = m_{max}\left(\frac{1 - P_s}{P_{max}}\right), \tag{3}$$

where $m_{max}$ is a maximum probability of mutation given by parameter, and $P_{max}$ the probability of $S$ maximum existing. Then, the algorithm 2 explain this operator: where for each habitat the probability of $S$ species is calculated, and then for each feature if selected to be mutated by this probability, it is replaced with another SIV random.

**Algorithm 2.** Mutation Operator

```
{M the size of the solution}
for j=1 to M do
    Calculate probability of mutation P_i based on (3)
    Select SIV H_i(j) with probability P_i(j)
    if H_i(j) is selected then
        Replace H_i(j) with a randomly generated SIV.
    end if
end for
```

Note that in binary problems, the mutation operator to exchange a SIV does so that $H_i(j) = 1 - H_i(j)$ [36]

### 3.3 Algorithm Description

The features and steps are described in general terms of the BBOA:

1. Initialize parameters. Mapping SVI and habitats according to problem solutions. Initialize a maximum of species $S_{max}$ (for simplicity, matching with the size of the population); immigration, emigration and mutation maximum rates. An elitist parameter to save the best solutions.
2. Initialize set of habitats, where each habitat corresponds to a possible solution of the problem.

3. For each habitat, calculate the HSI and accordingly, the number of species (A greater HSI, the greater the number of species). Then calculate rates of immigration and emigration.
4. Probabilistically using rates of immigration and emigration to modify habitats (Migration operator).
5. For each habitat, update the probability of number of species. Then mutate based on their probability of mutation (Mutation Operator).
6. Back to step 3 and finish until a stopping criterion is satisfied.

Note that after each habitat is modified (steps 2, 4, and 5), its feasibility as a problem solution should be verified. If it does not represent a feasible solution, then some method needs to be implemented in order to map it to the set of feasible solutions [28].

## 4 Biogeography-Based Optimization Algorithm for the SCP

After the description of the problem and the technique to use, finally it continues with the implementation and adaptation of BBOA to obtain acceptable results for the SCP.

### 4.1 General Considerations

As general considerations of the algorithm implementation, we can highlight:

- The long (SIVs) of each solution is the same size as the amount of costs instance of SCP.
- Repair function for infeasible solutions is used.
- A parameter of elitism, which stores the 2 solutions with the lowest cost over the generations is used.
- The stop criterion is a maximum number of generations.
- We use an optimized stagnation in local optimum by a created method for this purpose.

### 4.2 Fitness

An important point of the implemented algorithm is the calculation of the HSI, also called fitness in other population optimization algorithms. BBOA indicates that greater HSI solutions are best; and lower HSI, the worst. In addition, these estimates based on the costs of the problem being optimized. This contradicts the SCP, since this is minimization. It must find a solution with the lowest cost; therefore, lower cost solution is the best. Given that the fitness is calculated as:

$$HSI = \frac{1}{total\ cost\ solution} \tag{4}$$

Thus, at lower cost, the greater the value of the HSI. And higher cost, lower it.

### 4.3 Repair Infeasible Solutions

Due to changes in the solution features, some of these may be unfeasible for the instance of SCP; this means that the solution generated not comply with constraints. To resolve this problem, repair function is used. The repair is based on analyzing the solution in each constraint (row) verifying the feasibility; i.e., occurs at least one active column covering the restriction. If not exists a column that covers the row, then it is considered infeasible. To fix this, sought and activated columns from unfeasible rows with lower cost that will make the solution becomes feasible.

### 4.4 Delete Redundant Columns

Other technique for improving solutions is delete redundant columns [27]. A column is considered as redundant, w.r.t a given solution, if after deleting it the solution remains feasible. Therefore, we check the columns of the solution to find possible removals. With this, the solutions costs is reduced without losing feasibility.

### 4.5 Optimize Stagnation in Local Optimum

In BBOA, a very high maximum mutation rate allowed varied solutions, affecting the cost of these. For this, the value in parameter tends to lower numbers (0.0005 to 0.004 approximately). In the convergence of the BBOA, solutions generally stagnate in a local optimum, losing valuable iterations. When this happens, we created and applied a method that increase the maximum mutation rate, adding diversity and avoiding long stagnation.

The maximum rate of mutation should be increased to allow for new solutions when there is stagnation. For this, we calculated a percentage of 10 % of deadlock over the missing iterations. If this is true, the maximum mutation rate is increased in a 0.0009 over the rate (the latter parameter value subject to more experimentation). Then, if the percentage of stagnation continues to increase up to 20 %, the rate increases again and so that the local optimum change. By applying this method, the maximum mutation rate, which is a fixed parameter of BBOA, becomes variable.

We created this method over experimentation, nothing improvements in results. The values mentioned in this section, are very subject to more experimentation, since they could improve much more the solutions and avoiding the stagnation.

## 5 Experiments and Results

For the experiments, the optimization algorithm was implemented in Java programming language. In addition, they were carried out on a laptop with Windows 8.1 operating system, Intel Core i3 2.50 GHz with 6 GB of RAM. Moreover, we

**Table 1.** Results of preprocessed instances experiments - Problem Set 4

| Instance | Optimal | Best R | Worst R | Average |
|---|---|---|---|---|
| mscp41 | 429 | 430 | 433 | 430,83 |
| mscp410 | 514 | 514 | 519 | 516,53 |
| mscp42 | 512 | 512 | 512 | 512,00 |
| mscp43 | 516 | 516 | 521 | 516,53 |
| mscp44 | 494 | 495 | 495 | 495,00 |
| mscp45 | 512 | 514 | 517 | 516,50 |
| mscp46 | 560 | 560 | 570 | 561,47 |
| mscp47 | 430 | 430 | 433 | 431,73 |
| mscp48 | 492 | 493 | 499 | 498,20 |
| mscp49 | 641 | 641 | 656 | 646,07 |

**Table 2.** Results of preprocessed instances experiments - Problem Set 5

| Instance | Optimal | Best R | Worst R | Average |
|---|---|---|---|---|
| mscp51 | 253 | 253 | 263 | 255,70 |
| mscp510 | 265 | 265 | 267 | 265,87 |
| mscp52 | 302 | 305 | 307 | 305,70 |
| mscp53 | 226 | 226 | 230 | 228,07 |
| mscp54 | 242 | 242 | 243 | 242,37 |
| mscp55 | 211 | 211 | 212 | 211,50 |
| mscp56 | 213 | 213 | 216 | 213,57 |
| mscp57 | 293 | 293 | 301 | 294,53 |
| mscp58 | 288 | 288 | 299 | 289,13 |
| mscp59 | 279 | 279 | 287 | 280,27 |

used preprocessed [34] instances for SCP, obtained from OR-Library [3]. The table columns are formatted following: the first, for instance executed; the second, the global optimum known; the third best result obtained; fourth worst result and in the fifth the average of the results obtained.

The next parameters, obtained through experimentation was used: Population size = 15, maximum mutation probability = 0.004, maximum immigration probability = 1, maximum emigration probability = 1 and a maximum number of iterations = 6000. Each instance was executed 30 times. We divide the results on instances set. This can be seen in Tables 1, 2, 3, 4, 5, 6, 7 and 8.

Given the above results, we can see an excellent performance with the preprocessed instances. Getting the global optimum in 41 of 48 instances, and low cost average. Furthermore, thanks to the preprocessed method, this instances allow numerous experiments because to the speed of execution.

**Table 3.** Results of preprocessed instances experiments - Problem Set 6

| Instance | Optimal | Best R | Worst R | Average |
|---|---|---|---|---|
| mscp61 | 138 | 138 | 148 | 142,57 |
| mscp62 | 146 | 146 | 151 | 149,90 |
| mscp63 | 145 | 145 | 148 | 146,60 |
| mscp64 | 131 | 131 | 134 | 131,10 |
| mscp65 | 161 | 161 | 169 | 164,83 |

**Table 4.** Results of preprocessed instances experiments - Problem Set A

| Instance | Optimal | Best R | Worst R | Average |
|---|---|---|---|---|
| mscpa1 | 253 | 253 | 258 | 255,33 |
| mscpa2 | 252 | 252 | 261 | 255,73 |
| mscpa3 | 232 | 232 | 239 | 234,00 |
| mscpa4 | 234 | 234 | 235 | 234,60 |
| mscpa5 | 236 | 236 | 238 | 236,70 |

**Table 5.** Results of preprocessed instances experiments - Problem Set B

| Instance | Optimal | Best R | Worst R | Average |
|---|---|---|---|---|
| mscpb1 | 69 | 69 | 75 | 70,37 |
| mscpb2 | 76 | 76 | 80 | 76,50 |
| mscpb3 | 80 | 80 | 82 | 80,77 |
| mscpb4 | 79 | 79 | 83 | 80,53 |
| mscpb5 | 72 | 72 | 74 | 72,13 |

**Table 6.** Results of preprocessed instances experiments - Problem Set C

| Instance | Optimal | Best R | Worst R | Average |
|---|---|---|---|---|
| mscpc1 | 227 | 227 | 233 | 229,93 |
| mscpc2 | 219 | 219 | 225 | 221,13 |
| mscpc3 | 243 | 248 | 255 | 250,40 |
| mscpc4 | 219 | 219 | 227 | 221,20 |
| mscpc5 | 215 | 215 | 218 | 216,83 |

**Table 7.** Results of preprocessed instances experiments - Problem Set D

| Instance | Optimal | Best R | Worst R | Average |
|---|---|---|---|---|
| mscpd1 | 60 | 60 | 62 | 60,27 |
| mscpd2 | 66 | 66 | 69 | 67,43 |
| mscpd3 | 72 | 72 | 76 | 73,83 |
| mscpd4 | 62 | 62 | 65 | 63,37 |
| mscpd5 | 61 | 61 | 64 | 61,57 |

**Table 8.** Results of preprocessed instances experiments - Problem Set NR

| Instance | Optimal | Best R | Worst R | Average |
|---|---|---|---|---|
| mscpnre1 | 29 | 29 | 32 | 29, 63 |
| mscpnrf1 | 14 | 14 | 15 | 14, 47 |
| mscpnrg1 | 176 | 177 | 190 | 181, 77 |

## 6 Conclusion

After analyzed the problem and the technique to solve it, the algorithm is implemented, showing good results with full experiments; finding some low-cost solutions and low average cost. We created and implemented a technique that occur very good behavior in the algorithm, adding diversity and avoiding long stagnation. This, together with delete redundant columns and a simple repair method, allowed improve the results and algorithm performance. This type of algorithm modifications were made in order to obtain better quality results, shown results with 41 optimum solutions of 48 instances, including big instances.

Undoubtedly, new methods applied had great impact on the quality of results, due to the native algorithm not shown as good behavior. We could carry out more experiments, with new good repair methods, since even a basic repair method is used; as to find more precise parameters in the change of maximum mutation rate or BBOA input. This could generate a full optimum table. Finally, we can say that BBOA is very good to solve the SCP.

**Acknowledgements.** The author Broderick Crawford is supported by grant CONICYT/FONDE-CYT/REGULAR/1140897 and Ricardo Soto is supported by grant CONICYT/FONDECYT/REGULAR/1160455.

## References

1. Amini, F., Ghaderi, P.: Hybridization of harmony search and ant colony optimization for optimal locating of structural dampers. Appl. Soft Comput. **13**(5), 2272–2280 (2013)
2. Balas, E., Carrera, M.C.: A dynamic subgradient-based branch-and-bound procedure for set covering. Oper. Res. **44**(6), 875–890 (1996)

3. Beasley, J.E., Jornsten, K.: Enhancing an algorithm for set covering problems. Eur. J. Oper. Res. **58**(2), 293–300 (1992). http://ideas.repec.org/a/eee/ejores/v58y1992i2p293-300.html
4. Brusco, M.J., Jacobs, L.W., Thompson, G.M.: A morphing procedure to supplement a simulated annealing heuristic for cost- and coverage-correlated set-covering problems. Ann. Oper. Res. **86**, 611–627 (1999)
5. Caprara, A., Fischetti, M., Toth, P., Vigo, D., Guida, P.L.: Algorithms for railway crew management. Math. Program. **79**(1–3), 125–141 (1997). http://dx.doi.org/10.1007/BF02614314
6. Caserta, M.: Tabu search-based metaheuristic algorithm for large-scale set covering problems. In: Doerner, K.F., Gendreau, M., Greistorfer, P., Gutjahr, W., Hartl, R.F., Reimann, M. (eds.) Metaheuristics. OR/CSIS, vol. 39, pp. 43–63. Springer, Heidelberg (2007). http://dx.doi.org/10.1007/978-0-387-71921-4_3
7. Crawford, B., Soto, R., Berríos, N., Johnson, F., Paredes, F.: Solving the set covering problem with binary cat swarm optimization. In: Tan, Y., Shi, Y., Buarque, F., Gelbukh, A., Das, S., Engelbrecht, A. (eds.) ICSI-CCI 2015. LNCS, vol. 9140, pp. 41–48. Springer, Heidelberg (2015)
8. Crawford, B., Soto, R., Cuesta, R., Paredes, F.: Application of the artificial bee colony algorithm for solving the set covering problem. Sci. World J. **2014**(189164), 1–8 (2014)
9. Crawford, B., Soto, R., Monfroy, E., Palma, W., Castro, C., Paredes, F.: Parameter tuning of a choice-function based hyperheuristic using particle swarm optimization. Expert Syst. Appl. **40**(5), 1690–1695 (2013)
10. Crawford, B., Soto, R., Olea, C., Johnson, F., Paredes, F.: Binary bat algorithms for the set covering problem. In: 2015 10th Iberian Conference on Information Systems and Technologies (CISTI), pp. 1–4, June 2015
11. Crawford, B., Soto, R., Olivares Suarez, M., Paredes, F., Johnson, F.: Binary firefly algorithm for the set covering problem. In: 2014 9th Iberian Conference on Information Systems and Technologies (CISTI), pp. 1–5, June 2014
12. Crawford, B., Soto, R., Cuesta, R., Paredes, F.: Application of the artificial bee colony algorithm for solving the set covering problem. Sci. World J. **2014**, 1–8 (2014)
13. Crawford, B., Soto, R., Monfroy, E., Paredes, F., Palma, W.: A hybrid ant algorithm for the set covering problem. Int. J. Phys. Sci. **6**, 4667–4673 (2011)
14. Crawford, B., Soto, R., Olivares-Suárez, M., Paredes, F.: A binary firefly algorithm for the set covering problem. In: Silhavy, R., Senkerik, R., Oplatkova, Z.K., Silhavy, P., Prokopova, Z. (eds.) Modern Trends and Techniques in Computer Science. AISC, vol. 285, pp. 65–73. Springer, Heidelberg (2014). http://dx.doi.org/10.1007/978-3-319-06740-7_6
15. Eremeev, A.V., Kolokolov, A.A., Zaozerskaya, L.A.: A hybrid algorithm for set covering problem. In: Proceedings of International Workshop Discrete Optimization Methods in Scheduling and Computer-Aided Design, pp. 123–129 (2000)
16. Fisher, M.L., Kedia, P.: Optimal solution of set covering/partitioning problems using dual heuristics. Manage. Sci. **36**(6), 674–688 (1990)
17. Fisher, M.L., Rosenwein, M.B.: An interactive optimization system for bulk-cargo ship scheduling. Naval Res. Logistics (NRL) **36**(1), 27–42 (1989)
18. Foster, B.A., Ryan, D.: An integer programming approach to the vehicle scheduling problem. Oper. Res. **27**, 367–384 (1976)
19. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA (1990)

20. Goldberg, D.E.: Real-coded genetic algorithms, virtual alphabets, and blocking. Complex Syst. **5**, 139–167 (1990)
21. Guanghui Lan, A., Depuy, G.W.B., G.E.W.C.: Discrete optimization an effective- and simple heuristic for the set covering problem abstract (2005)
22. Han, L., Kendall, G., Cowling, P.: An adaptive length chromosome hyperheuristic genetic algorithm for a trainer scheduling problem. In: Proceedings of the fourth Asia-Pacific Conference on Simulated Evolution And Learning, (SEAL 2002), Orchid Country Club, Singapore, pp. 267–271 (2002)
23. Ma, H., Simon, D.: Biogeography-based optimization with blended migration for constrained optimization problems. In: Pelikan, M., Branke, J. (eds.) Genetic and Evolutionary Computation Conference, GECCO 2010, Proceedings, Portland, Oregon, USA, July 7–11, 2010. pp. 417–418. ACM (2010). http://doi.acm.org/10.1145/1830483.1830561
24. Michalewicz, Z.: Genetic algorithms + data structures = evolution programs, 3rd edn. Springer-Verlag, London, UK (1996)
25. Mo, H., Xu, L.: Biogeography migration algorithm for traveling salesman problem. In: Tan, Y., Shi, Y., Tan, K.C. (eds.) ICSI 2010, Part I. LNCS, vol. 6145, pp. 405–414. Springer, Heidelberg (2010). http://dx.doi.org/10.1007/978-3-642-13495-1_50
26. Mudaliar, D., Modi, N.: Unraveling travelling salesman problem by genetic algorithm using m-crossover operator. In: 2013 International Conference on Signal Processing Image Processing Pattern Recognition (ICSIPR), pp. 127–130, February 2013
27. Naji-Azimi, Z., Toth, P., Galli, L.: An electromagnetism metaheuristic for the unicost set covering problem. Eur. J. Oper. Res. **205**(2), 290–300 (2010). http://EconPapers.repec.org/RePEc:eee:ejores:v:205:y:2010:i:2
28. Simon, D.: Biogeography-based optimization. Evol. Comput. IEEE Trans. **12**(6), 702–713 (2008)
29. Smith, B.M.: Impacs - a bus crew scheduling system using integer programming. Math. Program. **42**(1), 181–187 (1988). http://dx.doi.org/10.1007/BF01589402
30. Soto, R., Crawford, B., Olivares, R., Barraza, J., Johnson, F., Paredes, F.: A binary cuckoo search algorithm for solving the set covering problem. In: Vicente, J.M.F., Álvarez-Sánchez, J.R., López, F.P., Toledo-Moreo, F.J., Adeli, H. (eds.) Bioinspired Computation in Artificial Systems. LNCS, vol. 9108, pp. 88–97. Springer, Heidelberg (2015). http://dx.doi.org/10.1007/978-3-319-18833-1_10
31. Thomson, G.: A simulated annealing heuristic for shift-scheduling using non-continuously available employees. Comput. Oper. Res. **23**, 275–288 (1996)
32. Toregas, C., Swain, R., ReVelle, C., Bergman, L.: The location of emergency service facilities. Oper. Res. **19**(6), 1363–1373 (1971). http://dx.doi.org/10.1287/opre.19.6.1363
33. Vasko, F.J., Wolf, F.E., Stott, K.L.: A set covering approach to metallurgical grade assignment. Eur. J. Oper. Res. **38**(1), 27–34 (1989). http://EconPapers.repec.org/RePEc:eee:ejores:v:38:y:1989:i:1:p:27-34
34. Xu, Y., Kochenberger, G., Wang, H.: Pre-processing method with surrogate constraint algorithm for the set covering problem (2008)
35. Zhang, Y., Wu, L., Wang, S., Huo, Y.: Chaotic artificial bee colony used for cluster analysis. In: Chen, R. (ed.) ICICIS 2011 Part I. CCIS, vol. 134, pp. 205–211. Springer, Heidelberg (2011). http://dx.doi.org/10.1007/978-3-642-18129-0_33
36. Zhao, B.B., Deng, C., Yang, Y., Peng, H.: Novel binary biogeography-based optimization algorithm for the knapsack problem. In: Tan, Y., Shi, Y., Ji, Z. (eds.) ICSI 2012, Part I. LNCS, vol. 7331, pp. 217–224. Springer, Heidelberg (2012). http://dx.doi.org/10.1007/978-3-642-30976-2_26