Two Swarm Intelligence Algorithms for the Set Covering Problem

Broderick Crawford[1,2], Ricardo Soto[1,3], Rodrigo Cuesta[1], Miguel Olivares-Suárez[1], Franklin Johnson[4] and Eduardo Olguín[5]

[1]*Pontificia Universidad Católica de Valparaíso, Av. Brasil 2950, Valparaíso, Chile*
[2]*Universidad Finis Terrae, Av. Pedro de Valdivia 1509 Providencia, Santiago, Chile*
[3]*Universidad Autónoma de Chile, Av. Pedro de Valdivia 641 Providencia, Santiago, Chile*
[4]*Universidad de Playa Ancha, Avda. González de Hontaneda 855, Valparaíso, Chile*
[5]*Universidad San Sebastián, Bellavista 7 Recoleta, Santiago, Chile*

Keywords: Weighted Set Covering Problem, Metaheuristics, Firefly Algorithm, Artificial Bee Colony Algorithm, Swarm Intelligence.

Abstract: The Weighted Set Covering problem is a formal model for many industrial optimization problems. In the Weighted Set Covering Problem the goal is to choose a subset of columns of minimal cost in order to cover every row. Here, we present its resolution with two novel metaheuristics: Firefly Algorithm and Artificial Bee Colony Algorithm. The Firefly Algorithm is inspired by the flashing behaviour of fireflies. The main purpose of flashing is to act as a signal to attract other fireflies. The flashing light can be formulated in such a way that it is associated with the objective function to be optimized. The Artificial Bee Colony Algorithm mimics the food foraging behaviour of honey bee colonies. In its basic version the algorithm performs a kind of neighbourhood search combined with random search. Experimental results show that both are competitive in terms of solution quality with other recent metaheuristic approaches.

1 INTRODUCTION

The Set Covering Problem (SCP) is a class of representative combinatorial optimization problem that has been applied to many real world problems, such as crew scheduling in airlines (Housos and Elmroth, 1997), facility location problem (Vasko and Wilson, 1984), and production planning in industry (Vasko et al., 1987).

The SCP is a well-known NP-hard in the strong sense (Garey and Johnson, 1990). Many algorithms have been developed to solve it and has been reported to literature. Exact algorithms are mostly based on branch-and-bound and branch-and-cut (Balas and Carrera, 1996; Fisher and Kedia, 1990). 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 or near-optimal solutions within a reasonable period of time.

Classical greedy algorithms are very simple, fast, and easy to code in practice, but they rarely produce high quality solutions for their myopic and deterministic nature (Chvatal, 1979). Compared with classical greedy algorithms, heuristics based on Lagrangian relaxation with subgradient optimization are much more effective. The most efficient ones are those proposed in (Ceria et al., 1998; Caprara et al., 1999). As top-level general search strategies, metaheuristics such as genetic algorithms (Beasley and Chu, 1996), simulated annealing (Brusco et al., 1999a), tabu search (Caserta, 2007), evolutionary algorithms (Crawford et al., 2007), ant colony optimization (ACO) (Ren et al., 2010; Crawford et al., 2013b), electromagnetism (unicost SCP) (Naji-Azimi et al., 2010), gravitational emulation search (Balachandar and Kannan, 2010) and cultural algorithms (Crawford et al., 2013a) have been also successfully applied to solve the SCP.

In this paper, we propose to solve the SCP with two recent metaheuristics: Firefly Algorithm and Artificial Bee Colony Algorithm. The Firefly Algorithm (FA) is a recently developed, population-based metaheuristic (Yang, 2010; Yang, 2009) where the objective function of a given optimization problem is based on differences of light intensity. Thus, fireflies are

Crawford B., Soto R., Cuesta R., Olivares-Suárez M., Johnson F. and Olguín E..
Two Swarm Intelligence Algorithms for the Set Covering Problem.
DOI: 10.5220/0005093500600069
In *Proceedings of the 9th International Conference on Software Engineering and Applications* (ICSOFT-EA-2014), pages 60-69
ISBN: 978-989-758-036-9

characterized by their light intensity which helps fireflies to change their position iteratively towards more attractive locations in order to obtain optimal solutions. The canonical FA algorithm is developed to tackle continuous optimization problems (Fister et al., 2013; Yang and He, 2013). However, the effectiveness of the FA algorithm to solve discrete NP-hard problems such as image compression and processing (Horng, 2012), shape and size optimization (Miguel and Fadel Miguel, 2012) and manufacturing cell problems (Sayadi et al., 2013) encourage researchers to design novel FAs for discrete optimization problems. The approach developed in this paper focus on transfer functions which force fireflies to move in binary space. To the best of our knowledge, this is the first work proposing a binary coded FA to solve the SCP. Artificial Bee Colony Algorithm (ABC) is one of the most recent algorithms in the domain of the collective intelligence. Created by Dervis Karaboga in 2005, who was motivated by the intelligent behavior observed in the domestic bees to take the process of foraging (Karaboga and Basturk, 2007). ABC mimics the foraging strategy of honey bees to look for the best solution to an optimization problem. Each candidate solution is thought of as a food source and a colony of bees is used to search in the solution space.

The rest of this paper is organized as follows. In Section 2, we give a formal definition of the SCP. The Section 3 describes FA and the Section 4 describes ABC. In Section 5, we present experimental results obtained when applying the algorithm for solving the 65 instances of SCP contained in the OR-Library. Finally, in Section 6 we conclude and highlight future directions of research.

2 PROBLEM DESCRIPTION

The Set Covering Problem (SCP) can be formally defined as follows. Let $A = (a_{ij})$ be an m-row, n-column, zero-one matrix. We say that a column j covers a row i if $a_{ij} = 1$. Each column j is associated with a nonnegative real cost c_j. Let $I = \{1, ..., m\}$ and $J = \{1, ..., n\}$ be the row set and column set, respectively. The SCP calls for a minimum cost subset $S \subseteq J$, such that each row $i \in I$ is covered by at least one column $j \in S$. A mathematical model for the SCP is

$$Minimize \quad f(x) = \sum_{j=1}^{n} c_j x_j \tag{1}$$

subject to

$$\sum_{j=1}^{n} a_{ij} x_j \geq 1, \quad \forall i \in I \tag{2}$$

$$x_j \in \{0, 1\}, \quad \forall j \in J \tag{3}$$

The goal is to minimize the sum of the costs of the selected columns, where $x_j = 1$ if the column j is in the solution, 0 otherwise. The restrictions ensure that each row i is covered by at least one column.

3 THE FIREFLY ALGORITHM

Nature-inspired methodologies are among the most powerful algorithms for optimization problems. The Firefly Algorithm is a novel nature-inspired algorithm inspired by the social behavior of fireflies. By idealizing some of the flashing characteristics of fireflies, a firefly-inspired algorithm was presented in (Yang, 2010; Yang, 2009).

The canonical FA was developed using the following three idealized rules:

- All fireflies are unisex and are attracted to other fireflies regardless of their sex.
- The degree of the attractiveness of a firefly is proportional to its brightness, and thus for any two flashing fireflies, the one that is less bright will move towards to the brighter one. More brightness means less distance between two fireflies. However, if any two flashing fireflies have the same brightness, then they move randomly.
- The brightness of a firefly is determined by the value of the objective function. For a maximization problem, the brightness of each firefly is proportional to the value of the objective function.

As the attractiveness of a firefly is proportional to the light intensity seen by adjacent fireflies, the attractiveness β of a firefly is defined as follows:

$$\beta(r) = \beta_0 e^{-\gamma r^m}, \; m \geq 1 \tag{4}$$

where r is the distance between two fireflies, β_0 is the attractiveness at $r = 0$ and γ is a fixed light absorption coefficient. The distance r_{ij} between two fireflies i and j at positions x_i and x_j is determined by

$$r_{ij} = ||x_i - x_j|| = \sqrt{\sum_{k=1}^{d} (x_i^k - x_j^k)^2} \tag{5}$$

where x_i^k is the current value of the k_{th} dimension of the i^{th} firefly and d is the number of dimensions. The movement of a firefly i is attracted to another more attractive (brighter) firefly j is determined by

$$x_i^k(t+1) = x_i^k(t) + \beta_0 e^{-\gamma r_{ij}^2}(x_j^k(t) - x_i^k(t)) + \alpha(rand - \frac{1}{2}) \tag{6}$$

where the first term $x_i^k(t)$ is the current value (current position) of the k_{th} dimension of the firefly i at iteration t. The second term denotes the firefly attractiveness where γ characterizes the variation of the attractiveness typically varying from 0.1 to 10 (Yang, 2010), and the last term introduces randomization, with $\alpha \in [0,1]$ being the randomization parameter and *rand* is a random number generator uniformly distributed between 0 and 1.

3.1 Description of the Firefly Approach

In this section, a discrete FA is proposed to solve the SCP.

Step 1. Initialization of firefly parameters (γ, β_0, size for the firefly population and the maximum number of generations for the termination process).

Step 2. Initialization of firefly position. Initialize randomly $M = [X_1; ...; X_m]$ of m solutions or firefly positions in the multi-dimensional search space, where m represents the size of the firefly population. Each solution of X is represented by a d-dimensional binary vector.

Step 3. Evaluation of fitness of the population. For this case the function of fitness is equal to the objective function of the SCP model (Eq. 1).

Step 4. Modification of firefly position. A firefly produces a modification in its position based on the brightness w.r.t other fireflies. Using Eq. 6 the new position is determined by modifying the value of each dimension of a firefly. To move from a continuous search space to a discrete one we work with the following update rule:

$$x_i^k(t+1) = \begin{cases} x_*^k & \text{if } rand < T(x_i^k(t+1)) \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

where *rand* is a uniform random number between 0 and 1, x_*^k is the best firefly so far, and $T(x)$ is the binary transfer function (Mirjalili and Lewis, 2013). The transfer function forces the values of the dimensions of fireflies to move in a binary space. In this work we use:

$$T(x) = \left| \frac{2}{\pi} \arctan(\frac{\pi}{2}x) \right| \quad (8)$$

Step 5. Evaluation. The new solution is evaluated and if it is not a feasible solution then it is repaired. In order to make feasible solutions we determine which rows have not yet been covered and choose the columns needed for coverage. The criteria used to choose these columns is based in the cost of a column/number of rows not covered that cover the column j. Once the solution has become feasible we apply an optimization step in order to eliminate those redundant columns. A column is redundant when it is removed and the solution remains feasible.

Step 6. Memorization of the best solution achieved so far and increment the counter of generations.

Step 7. Stop the process and display the result if the termination criteria is satisfied. Termination criteria used in this work is the maximum number of generations. Otherwise, go to step 3.

The following algorithm shows the pseudo code of the steps proposed.

Algorithm: FA pseudo-code.

```
1 Begin
2   Initialize parameters
3   Evaluate the light intensities
4   while $t < MaxGeneration$ do
5     for $i = 1 : m$ ($m$ fireflies) do
6       for $j = 1 : m$ ($m$ fireflies) do
7         if ($I_j < I_i$) then
8           $movement$ = calculates value according to Eq. 6
9           if (rand() < T($movement$)) then
10              $fireflies[i][j] = bestFirefly[j]$
11          else
12              $fireflies[i][j] = 0$
13          end if
14        end if
15        Repair solutions
16        Update attractiveness
17        Update light intensity
18      end for $j$
19    end for $i$
20    $t = t + 1$
21  end while
22  Output the results
23 End
```

4 THE ARTIFICIAL BEE COLONY ALGORITHM

ABC is one of the most recent algorithms in the domain of the collective intelligence. Created by Dervis Karaboga in 2005, who was motivated by the intelligent behavior observed in the domestic bees to take the process of foraging (Karaboga and Basturk, 2007).

ABC is an algorithm of combinatorial optimization based on populations, in which the solutions of the problem of optimization, the sources of food, are modified by the artificial bees, that fungen as operators of variation. The aim of these bees is to discover the food sources with major nectar.

In the ABC algorithm, an artificial bee moves in a multidimensional search space choosing sources of nectar depending on its past experience and its companions of beehive or fitting his position. Some bees (exploratory) fly and choose food sources randomly without using experience. When they find a source of major nectar, they memorize their positions and forget the previous ones. Thus, ABC combines methods of local search and global search, trying to balance the process of the exploration and exploitation of the search space.

Although, the performance of different optimization algorithm is dependent on applications some recent works demonstrate that the Artificial Bee Colony is faster than either Genetic Algorithm or Particle Swarm Optimization solving certain problems (Zhang and Wu, 2012; Zhang and Wu, 2011; Zhang et al., 2011b; Zhang et al., 2011a; Zhang et al., 2011c). Additionally, ABC has demonstrated an ability to attack problems with a lot of variables (high-dimensional problems) (Akay and Karaboga, 2009).

The pseudocode of Artificial Bee Colony is as follows.

Algorithm: ABC pseudo-code.

```
1 Begin
2   InitPopulation()
3   while remain interations do
4     Select sites for the local search
5     Recruit bees for the selected sites and to
          evaluate fitness
6     Select the bee with the best fitness
7     Assign the remaining bees to looking
          for randomly
8     Evaluate the fitnes of remaining bees
9     UpdateOptimum()
10  end while
11  return BestSolution
12 End
```

The procedure for determining a food source in the neighborhood of a particular food source which depends on the nature of the problem. Karaboga (Karaboga, 2005) developed the first ABC algorithm for continuous optimization. The method for determining a food source in the neighborhood of a particular food source is based on changing the value of one randomly chosen solution variable while keeping other variables unchanged. This is done by adding to the current value of the chosen variable the product of a uniform variable in [-1, 1] and the difference in values of this variable -current food source - and some other randomly chosen food source. This approach can not be used for discrete optimization problems for which it generates at best a random effect.

Singh (Singh, 2009) subsequently proposed a method, which is appropriate for subset selection problems. In his model, to generate a neighboring solution, an object is randomly dropped from the solution and in its place another object, which is not already present in the solution, it is added. The object to be added is selected from another randomly chosen solution. If there are more than one candidate objects for addition then ties are broken arbitrarily.

This approach is based on the idea that if an object is present in one good solution then it is highly likely that this object is present in many good solutions. This method provides another advantage, consisting in that if the method fails to find an object different from the others objects in the original solution it means that the two solutions are equal. Then, the employed bee associated with the original solution is converted in a scout bee eliminating duplication.

4.1 Description of the Bee Approach

In this section, an ABC is proposed to solve the SCP.

Step 1. Initialization.
To initialize the parameters of ABC as size of the colony, number of workers and curious (onlookers or "in wait") bees, limit of attempts and maximum number of cycles.

Step 2. Generation of initial population.
To generate the initial population by every row (or SCP constraint) a column (or SCP variable) is selected at random from the set of columns with covering possibilities. After we run a duplicates drop process, we check that there are no columns duplicated. A solution is represented by means of an entire vector like appears in Figure 1 staying the columns considered in the solution (a "-1" means that the column was removed due to duplication). Then, we use an integer encoding as the encoding rule.

333	10	5	300	-1	...	657	99

Figure 1: Representation of a solution.

Step 3. Evaluation of the fitness of the population.
The fitness function is equal to the objective function of the SCP.

Step 4. Modification of position and selection of sites for worker bees.

A hard-working bee modifies its position by means of the creation of a new solution based on a different food source selected randomly. It sees if at least it has a different column, in case of having not even a different column, the hard-working bee is transformed in an explorer in order to eliminate duplicated solutions. In opposite case, it proceeds to add a certain random number of columns between 0 and the maximum numbers columns to add.

After this, it proceeds to eliminate a certain random number of columns between 0 and the maximum numbers columns to eliminate. In case that new solution does not meet constraints, it is repaired. The fitness of the solution is evaluated, if the fitness (cost) is minor that the solution had in a beginning, the solution is replaced. In opposite case, it increases the number of attempts for improving this solution (parameter limit of attempts).

Step 5. Recruitment of curious bees for the selected sites.
A curious bee evaluates the information of the nectar through the workers and it chooses a source of food with the fitness proportionate selection method or roulette-wheel selection.

Step 6. Modification of position for the curious bees. They work alike to hard-working bees in Step 4.

Step 7. To leave a source exploited by the bees.
If the solution representing a source of food does not improve for a predetermined number of attempts (limit), then the source of food is left and is replaced by a new source of food generated as in Step 1.

Step 8. Memorization of the best solution and to increase the counter of the cycle.

Step 9. The process stops if the criteria of satisfaction expires, in opposite case to return to Step 3.

5 EXPERIMENTAL RESULTS

In order to test the effectiveness of FA and ABC they were tested using the 65 SCP test instances from OR-Library (Beasley, 1990). These instances are divided into 11 groups and each group contains 5 or 10 instances. Table 3 shows their detailed information where "Density" is the percentage of non-zero entries in the SCP matrix. The algorithms were implemented using C language and conducted on a 1.8 GHz Intel Core 2 Duo T5670 CPU with 3GB RAM running Windows 8.

In all experiments, the algorithms were executed 30 times over each SCP instance. Parameter values have a profound influence on the performance of ABC. The parameters were empirically adjusted, we determined their values in an experimental way, for each parameter, a set of candidate values were considered. We modified the value of one parameter while keeping the others fixed. According to the best results, as parameter values in our experiments, we used:

In FA the maximum number of generations was set to 50. We used a population of 25 fireflies and the values of γ, β_0 are initialized to 1. ABC runs 1000 iterations with a population of 200 bees, where 100 corresponds to hard-working and 100 to curious. Limit = 50, maximum number of columns to add = $0,5\%$ of columns in the SCP instance, maximum number of columns to eliminate = $1,2\%$ of the SCP instance. These parameters showed good results but they cannot be the ideal ones for all the instances.

Tables 1 and 4 show the results obtained over the 65 instances. The quality of a solution is evaluated using the relative percentage deviation (*RPD*). The RPD value quantifies the deviation of the objective value Z from Z_{opt} which in our case is the best known cost value for each instance (see the second column). We report the minimum cost reached, maximum, and average of the obtained solutions. To compute *RPD* we use $Z = Min$. This measure is computed as follows:

$$RPD = (Z - Z_{opt})/Z_{opt} \times 100 \qquad (9)$$

The results expressed in terms of the *RPD* show the effectiveness of our approach. It provides high quality near optimal solutions and it has the ability to generate them for a variety of instances.

5.1 Comparison with Other Works

In comparison with very recent works solving SCP - - with Cultural algorithms (Crawford et al., 2013a) and Ant Colony + Constraint Programming techniques (Crawford et al., 2013c) - our ABC proposal performs better with SCP instances reported in those works.

In order to bring out the efficiency of our proposal the solutions of the complete set of instances have been compared with other metaheuristics. We compared our algorithm solving the complete set of 65 standard non-unicost SCP instances from OR Library with the newest ACO-based algorithm for SCP in the literature: Ant-Cover + Local Search (ANT+LS) (Ren et al., 2010), Genetic Algorithm (GA) proposed by Beasley and Chu (1996) (Beasley and Chu, 1996) and

Simulated Annealing (SA) proposed by Brusco et al. (1999) (Brusco et al., 1999b).

Tables 1 and 4 show the detailed results obtained by the algorithms. Column 2 reports the optimal or the best known solution value of each instance. The third and fourth columns show the best value and the average obtained by our ABC algorithm in the 30 runs (trials). The fifth and sixth columns show the best value and the average obtained by our FA in the 30 runs (trials). The next columns show the average values obtained by GA, SA and ANT+LS respectively. The last columns show the Relative Percentage Deviation (RPD) value over the instances tested with ABC and FA.

In relation with ABC, we can observe in Tables 1 and 4 that:

- ABC is able to find the optimal solution consistently - i.e. in every trial- for 43 of 65 problems.
- ABC is able to find the best known value in all instances of Table 4.
- ABC is able to find the best known value in all trials of Table 4.
- ABC has higher success rate compared to genetic algorithm, simulated annealing and ants in sets NRE, NRF, NRG and NRH. The *RPD* of BEE is 0,00%, the *RPD* of GA is 1,04%, the *RPD* of SA is 0,72% and the *RPD* of ANT+LS is 0,86%.
- ABC can obtain optimal solutions in some instances where the others metaheuristics failed.

In relation with FA, we can observe in Tables 1 and 4 that:

- A direct implementation of Binary FA shows good results considering its simplicity. FA is capable of solving problems which have continuous search space. However, there are many optimization problems as SCP, which have discrete binary search spaces. They need binary algorithms to be solved. In the original FA, fireflies can move around the search space because of having position vectors with continuous real domain. Consequently, the concept of position updating can be easily implemented for fireflies based on the brightness w.r.t. other fireflies using 6. However, the meaning of position updating is different in a discrete binary space. In binary space, due to dealing with only two numbers: “0” and “1”, the position updating process cannot be done using 6. Therefore, we have to find a way to use the movement equation to change the positions from “0” and “1” or vice versa. This switching should be done based on the attractiveness between the fireflies. The idea is to change position of a firefly with the probability of its movement to another more attractive (brighter). In order to do this, a transfer function is needed to map the movement values to probability values for updating the positions.
- A novel v-shape transfer function with a new rule (7) was used in our Binary FA. Traditionally, it is used a s-shape transfer function as S2 in Table 2. Transfer functions force fireflies to move in a binary space. The transfer function should be able to provide a high probability of changing the position for a large absolute value of the movement. It should also present a small probability of changing the position for a small absolute value of the movement. Moreover, the range of a transfer function should be bounded in the interval [0,1] and increased with the increasing of movement. The binary version of FA proposed here uses the transfer function V4 in Table 2. For us, this function had presented better experimental results than others transfer functions showed.

5.2 Computational Cost and Convergence to the Best Solution

Our Binary FA shows an excellent tradeoff between the quality of the solutions obtained and the computational effort required. In all cases, FA converged very quickly (mainly from the 3th iteration) and its computation time in the runs was less than 1 second (except for NRG and NRH instances where the computation time was less than 3 secs).

Due to a more sophisticated implementation, our ABC algorithm requires a major computational effort. In our ABC approach we considered specific operators and parameters to try with SCP. Anyway, we believe it is worth given the high quality of the solutions obtained. In all cases, ABC converged quickly (mainly from the 10th iteration) and its computation time in the runs was less than 2 seconds (except for NRG and NRH instances where the computation time was less than 30 secs).

6 CONCLUSION

In this paper we have presented two recent swarm based metaheuristics for optimizing the Weighted Set Covering Problem. We have performed experiments throught several ORLIB instances. Our approach has demostrated to be very effective, providing unattended solving methods, for quickly producing solutions of a good quality. Experiments shown interesting results in terms of robustness, where using the same parameters for different instances giving good results.

The promising results of the experiments open up opportunities for further research. We visualize different directions for future work:

- The fact that the presented algorithm is easy to implement, clearly implies that ABC could also be effectively applied to other combinatorial optimization problems.
- An interesting proposal by Teodor Crainic et al. at (Glover and Kochenberger, 2003) involves parallelizing strategies for metaheuristics. The author sets a basis on the idea that the central goal of parallel computing is to speed up computation by dividing the work load among several threads of simultaneous execution, then a type of metaheuristic parallelism could come from the decomposition of the decision variables into disjoint subsets. The particular heuristic is applied to each subset and the variables outside the subset are considered fixed.
- An interesting extension of this work would be related to hybridization with other metaheuristics or to apply a hyperheuristic approach (Valenzuela et al., 2012).
- The use of Autonomous Search (AS), AS represents a new research field, and it provides practitioners with systems that are able to autonomously selftune their performance while effectively solving problems. Its major strength and originality consist in the fact that problem solvers can now perform self-improvement operations based on analysis of the performances of the solving process (Crawford et al., 2013d; Monfroy et al., 2013; Crawford et al., 2012).
- Furthermore, we are considering to use different preprocessing steps from the OR literature, which allow to reduce the problem size (Krieken et al., 2003).

REFERENCES

Akay, B. and Karaboga, D. (2009). Parameter tuning for the artificial bee colony algorithm. In Nguyen, N., Kowalczyk, R., and Chen, S.-M., editors, *Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems*, volume 5796 of *Lecture Notes in Computer Science*, pages 608–619. Springer Berlin Heidelberg.

Balachandar, S. R. and Kannan, K. (2010). A meta-heuristic algorithm for set covering problem based on gravity. 4(7):944–950.

Balas, E. and Carrera, M. C. (1996). A dynamic subgradient-based branch-and-bound procedure for set covering. *Operations Research*, 44(6):875–890.

Beasley, J. and Chu, P. (1996). A genetic algorithm for the set covering problem. *European Journal of Operational Research*, 94(2):392 – 404.

Beasley, J. E. (1990). A lagrangian heuristic for set-covering problems. *Naval Research Logistics (NRL)*, 37(1):151–164.

Brusco, M., Jacobs, L., and Thompson, G. (1999a). A morphing procedure to supplement a simulated annealing heuristic for cost- and coverage-correlated set-covering problems. *Annals of Operations Research*, 86(0):611–627.

Brusco, M. J., Jacobs, L. W., and Thompson, G. M. (1999b). A morphing procedure to supplement a simulated annealing heuristic for cost- and coverage-correlated set-covering problems. *Annals of Operations Research*, 86:611–627.

Caprara, A., Fischetti, M., and Toth, P. (1999). A heuristic method for the set covering problem. *Operations Research*, 47(5):730–743.

Caserta, M. (2007). Tabu search-based metaheuristic algorithm for large-scale set covering problems. In Doerner, K., Gendreau, M., Greistorfer, P., Gutjahr, W., Hartl, R., and Reimann, M., editors, *Metaheuristics*, volume 39 of *Operations Research/Computer Science Interfaces Series*, pages 43–63. Springer US.

Ceria, S., Nobili, P., and Sassano, A. (1998). A lagrangian-based heuristic for large-scale set covering problems. *Mathematical Programming*, 81(2):215–228.

Chvatal, V. (1979). A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235.

Crawford, B., Castro, C., Monfroy, E., Soto, R., Palma, W., and Paredes, F. (2012). A hyperheuristic approach for guiding enumeration in constraint solving. In *Proceedings of EVOLVE*, volume 175 of *Advances in Intelligent Systems and Computing*, pages 171–188. Springer.

Crawford, B., Lagos, C., Castro, C., and Paredes, F. (2007). A evolutionary approach to solve set covering. In Cardoso, J., Cordeiro, J., and Filipe, J., editors, *ICEIS (2)*, pages 356–363.

Crawford, B., Soto, R., and Monfroy, E. (2013a). Cultural algorithms for the set covering problem. In Tan, Y., Shi, Y., and Mo, H., editors, *ICSI (2)*, volume 7929 of *Lecture Notes in Computer Science*, pages 27–34. Springer.

Crawford, B., Soto, R., Monfroy, E., Castro, C., Palma, W., and Paredes, F. (2013b). A hybrid soft computing approach for subset problems. *Mathematical Problems in Engineering.*, Article ID 716069:1–12.

Crawford, B., Soto, R., Monfroy, E., Castro, C., Palma, W., and Paredes, F. (2013c). A hybrid soft computing approach for subset problems. *Mathematical Problems in Engineering*, 2013(Article ID 716069):1–12.

Crawford, B., Soto, R., Monfroy, E., Palma, W., Castro, C., and Paredes, F. (2013d). Parameter tuning of a choice-function based hyperheuristic using particle swarm optimization. *Expert Systems with Applications*, 40(5):1690–1695.

Fisher, M. L. and Kedia, P. (1990). Optimal solution of set covering/partitioning problems using dual heuristics. *Management Science*, 36(6):674–688.

Fister, I., Jr., I. F., Yang, X.-S., and Brest, J. (2013). A comprehensive review of firefly algorithms. *Swarm and Evolutionary Computation*, 13(0):34–46.

Garey, M. R. and Johnson, D. S. (1990). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA.

Glover, F. and Kochenberger, G. A. (2003). *Handbook of metaheuristics*. Springer.

Horng, M.-H. (2012). Vector quantization using the firefly algorithm for image compression. *Expert Syst. Appl.*, 39(1):1078–1091.

Housos, E. and Elmroth, T. (1997). Automatic optimization of subproblems in scheduling airline crews. *Interfaces*, 27(5):68–77.

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. *Technical Report TR06. Computer Engineering Department, Erciyes University, Turkey*.

Karaboga, D. and Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *J. Global Optimization*, 39(3):459–471.

Krieken, M. v., Fleuren, H., and Peeters, M. (2003). Problem reduction in set partitioning problems. Discussion Paper 2003-80, Tilburg University, Center for Economic Research.

Miguel, L. F. F. and Fadel Miguel, L. F. (2012). Shape and size optimization of truss structures considering dynamic constraints through modern metaheuristic algorithms. *Expert Syst. Appl.*, 39(10):9458–9467.

Mirjalili, S. and Lewis, A. (2013). S-shaped versus v-shaped transfer functions for binary particle swarm optimization. *Swarm and Evolutionary Computation*, 9(0):1 – 14.

Monfroy, E., Castro, C., Crawford, B., Soto, R., Paredes, F., and Figueroa, C. (2013). A reactive and hybrid constraint solver. *Journal of Experimental and Theoretical Artificial Intelligence*, 25(1):1–22.

Naji-Azimi, Z., Toth, P., and Galli, L. (2010). An electromagnetism metaheuristic for the unicost set covering problem. *European Journal of Operational Research*, 205(2):290–300.

Ren, Z.-G., Feng, Z.-R., Ke, L.-J., and Zhang, Z.-J. (2010). New ideas for applying ant colony optimization to the set covering problem. *Computers & Industrial Engineering*, 58(4):774 – 784.

Sayadi, M. K., Hafezalkotob, A., and Naini, S. G. J. (2013). Firefly-inspired algorithm for discrete optimization problems: An application to manufacturing cell formation. *Journal of Manufacturing Systems*, 32(1):78 – 84.

Singh, A. (2009). An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem. *Appl. Soft Comput.*, 9(2):625–631.

Valenzuela, C., Crawford, B., Soto, R., Monfroy, E., and Paredes, F. (2012). A 2-level metaheuristic for the set covering problem. *INT J COMPUT COMMUN*, 7(2):377–387.

Vasko, F. J. and Wilson, G. R. (1984). Using a facility location algorithm to solve large set covering problems. *Operations Research Letters*, 3(2):85–90.

Vasko, F. J., Wolf, F. E., and Stott, K. L. (1987). Optimal selection of ingot sizes via set covering. *Oper. Res.*, 35(3):346–353.

Yang, X.-S. (2009). Firefly algorithms for multimodal optimization. In *Proceedings of the 5th International Conference on Stochastic Algorithms: Foundations and Applications*, SAGA'09, pages 169–178, Berlin, Heidelberg. Springer-Verlag.

Yang, X.-S. (2010). *Nature-Inspired Metaheuristic Algorithms*. Luniver Press.

Yang, X.-S. and He, X. (2013). Firefly algorithm: Recent advances and applications. *CoRR*, abs/1308.3898.

Zhang, Y., L., W., and Wang, S. (2011a). Magnetic resonance brain image classification by an improved artificial bee colony algorithm. *Progress In Electromagnetics Research*, 116:65–79.

Zhang, Y. and Wu, L. (2011). Optimal multi-level thresholding based on maximum tsallis entropy via an artificial bee colony approach. *Entropy*, 13(4):841–859.

Zhang, Y. and Wu, L. (2012). Artificial bee colony for two dimensional protein folding. *Advances in Electrical Engineering Systems*, 1(1):19–23.

Zhang, Y., Wu, L., Wang, S., and Huo, Y. (2011b). Chaotic artificial bee colony used for cluster analysis. In Chen, R., editor, *Intelligent Computing and Information Science*, volume 134 of *Communications in Computer and Information Science*, pages 205–211. Springer Berlin Heidelberg.

Zhang, Y. D., Wu, L. N., and Wang, S. H. (2011c). Ucav path planning based on fscabc. *Information-an International Interdisciplinary Journal*, 14(3):687–692.

APPENDIX

Table 1: Experimental results. - Instances with optimal.

Prob.	Opt.	ABC Min	ABC Avg	FA Min	FA Avg	GA Avg	SA Avg	ANT-LS Avg	ABC RPD	FA RPD
4.1	429	430	430.5	481	481.03	429.7	-	429	0.35	12.12
4.2	512	512	512	580	580	512	-	512	0	13.28
4.3	516	516	516	619	619.03	516	-	516	0	19.96
4.4	494	494	494	537	537	494.8	-	494	0	8.7
4.5	512	512	512	609	609	512	-	512	0	18.94
4.6	560	561	561.7	653	653	560	-	560	0.30	16.6
4.7	430	430	430	491	491.07	430.2	-	430	0	14.18
4.8	492	493	494	565	565	492.1	-	492	0.41	14.83
4.9	641	643	645.5	749	749.03	643.1	-	641	0.70	14.84
4.10	514	514	514	550	550	514	-	514	0	7
5.1	253	254	255	296	296.03	253	-	253	0.79	16.99
5.2	302	309	310.2	372	372	303.5	-	302	2.72	23.17
5.3	226	228	228.5	250	250	228	-	226	1.11	10.61
5.4	242	242	242	277	277.07	242.1	-	242	0	14.46
5.5	211	211	211	253	253	211	-	211	0	19.9
5.6	213	213	213	264	264.03	213	-	213	0	23.94
5.7	293	296	296	337	337	293	-	293	1.02	15.01
5.8	288	288	288	326	326	288.8	-	288	0	13.19
5.9	279	280	280	350	350	279	-	279	0.36	25.44
5.10	265	266	267	321	321	265	-	265	0.75	21.13
6.1	138	140	140.5	173	173.03	138	-	138	1.81	25.36
6.2	146	146	146	180	180.07	146.2	-	146	0	23.28
6.3	145	145	145	160	160	145	-	145	0	10.34
6.4	131	131	131	161	161	131	-	131	0	22.9
6.5	161	161	161	186	186	161.3	-	161	0	15.52
A.1	253	254	254	285	285	253.2	-	253	0.40	12.64
A.2	252	254	254	285	285.07	253	-	252	0.79	13.09
A.3	232	234	234	272	272	232.5	-	232.8	0.86	17.24
A.4	234	234	234	297	297	234	-	234	1.10	26.92
A.5	236	237	238.6	262	262	236	-	236	0	11.01
B.1	69	69	69	80	80.03	69	-	69	0	15.94
B.2	76	76	76	92	92	76	-	76	0	21.05
B.3	80	80	80	93	93	80	-	80	0	16.25
B.4	79	79	79	98	98.03	79	-	79	0	24.05
B.5	72	72	72	87	87	72	-	72	0	20.83
C.1	227	230	231	279	279	227.2	-	227	1.76	22.9
C.2	219	219	219	272	272	220	-	219	0	24.2
C.3	243	244	244.5	288	288	246.4	-	243	0.62	18.51
C.4	219	220	224	262	262	219.1	-	219	2.28	19.63
C.5	215	215	215	262	262.07	215.1	-	215	0	21.86
D.1	60	60	60	71	71	60	-	60	0	18.33
D.2	66	67	67	75	75	66	-	66	1.52	13.63
D.3	72	73	73	88	88	72.2	-	72	1.39	22.22
D.4	62	63	63	71	71	62	-	62	1.61	14.51
D.5	61	62	62	71	71	61	-	61	1.64	16.39
NRE.1	29	29	29	32	32.03	29	29	29	0	10.34
NRE.2	30	30	30	36	36	30.6	30	30	0	20
NRE.3	27	27	27	35	35	27.7	27	27	0	29.62
NRE.4	28	28	28	34	34	28	28	28	0	21.42
NRE.5	28	28	28	34	34	28	28	28	0	21.42
NRF.1	14	14	14	17	17.03	14	14	14	0	21.42
NRF.2	15	15	15	17	17	15	15	15	0	13.33
NRF.3	14	14	14	21	21	14	14	14	0	50
NRF.4	14	14	14	19	19	14	14	14	0	35.71
NRF.5	13	13	13	16	16	13.7	13.7	13.5	0	23.07
NRG.1	176	176	176	230	230.03	177.7	176.6	176	0	30.68
NRG.2	154	154	154	191	191	156.3	155.3	155.1	0	24.02
NRG.3	166	166	166	198	198	167.9	167.6	167.3	0	19.27
NRG.4	168	168	168	214	214	170.3	170.7	168.9	0	27.38
NRG.5	168	168	168	223	223	169.4	168.4	168.1	0	32.73
NRH.1	63	63	63	85	85.07	64	64	64	0	34.92
NRH.2	63	63	63	81	81.03	64	63.7	67.9	0	28.57
NRH.3	59	59	59	76	76	59.1	59.4	59.4	0	28.81
NRH.4	58	58	58	75	75	58.9	58.9	58.7	0	29.31
NRH.5	55	55	55	68	68	55.1	55	55	0	23.63

Table 2: S-shaped and v-shaped transfer functions.

S-shaped family		V-shaped family	
Name	Transfer function	Name	Transfer function
S1	$T(x)=\frac{1}{1+e^{-2x}}$	V1	$T(x)=\left\lvert \mathrm{erf}\left(\frac{\sqrt{2}}{\pi}x\right)\right\rvert = \left\lvert \frac{\sqrt{2}}{\pi}\int_{o}^{\frac{\sqrt{2}}{\pi}x} e^{-t^2}dt\right\rvert$
S2	$T(x)=\frac{1}{1+e^{-x}}$	V2	$T(x)=\lvert\tanh(x)\rvert$
S3	$T(x)=\frac{1}{1+e^{-x/2}}$	V3	$T(x)=\left\lvert\frac{x}{\sqrt{1+x^2}}\right\rvert$
S4	$T(x)=\frac{1}{1+e^{-x/3}}$	V4	$T(x)=\lvert\frac{2}{\pi}\arctan(\frac{\pi}{2}x)\rvert$

Table 3: Details of the test instances.

Instance set	No. of instances	m	n	Cost range	Density (%)	Optimal solution
4	10	200	1000	[1, 100]	2	Known
5	10	200	2000	[1, 100]	2	Known
6	5	200	1000	[1, 100]	5	Known
A	5	300	3000	[1, 100]	2	Known
B	5	300	3000	[1, 100]	5	Known
C	5	400	4000	[1, 100]	2	Known
D	5	400	4000	[1, 100]	5	Known
NRE	5	500	5000	[1, 100]	10	Unknown
NRF	5	500	5000	[1, 100]	20	Unknown
NRG	5	1000	10000	[1, 100]	2	Unknown
NRH	5	1000	10000	[1, 100]	5	Unknown

Table 4: Experimental results - Instances with Best Known Solution.

Prob.	Opt.	ABC Min	ABC Avg	FA Min	FA Avg	GA Avg	SA Avg	ANT-LS Avg	ABC RPD	FA RPD
NRE.1	29	29	29	32	32.03	29	29	29	0	10.34
NRE.2	30	30	30	36	36	30.6	30	30	0	20
NRE.3	27	27	27	35	35	27.7	27	27	0	29.62
NRE.4	28	28	28	34	34	28	28	28	0	21.42
NRE.5	28	28	28	34	34	28	28	28	0	21.42
NRF.1	14	14	14	17	17.03	14	14	14	0	21.42
NRF.2	15	15	15	17	17	15	15	15	0	13.33
NRF.3	14	14	14	21	21	14	14	14	0	50
NRF.4	14	14	14	19	19	14	14	14	0	35.71
NRF.5	13	13	13	16	16	13.7	13.7	13.5	0	23.07
NRG.1	176	176	176	230	230.03	177.7	176.6	176	0	30.68
NRG.2	154	154	154	191	191	156.3	155.3	155.1	0	24.02
NRG.3	166	166	166	198	198	167.9	167.6	167.3	0	19.27
NRG.4	168	168	168	214	214	170.3	170.7	168.9	0	27.38
NRG.5	168	168	168	223	223	169.4	168.4	168.1	0	32.73
NRH.1	63	63	63	85	85.07	64	64	64	0	34.92
NRH.2	63	63	63	81	81.03	64	63.7	67.9	0	28.57
NRH.3	59	59	59	76	76	59.1	59.4	59.4	0	28.81
NRH.4	58	58	58	75	75	58.9	58.9	58.7	0	29.31
NRH.5	55	55	55	68	68	55.1	55	55	0	23.63