A Binary Firefly Algorithm for the Set Covering Problem

Broderick Crawford, Ricardo Soto, Miguel Olivares-Suárez and Fernando Paredes

Abstract The non-unicost Set Covering Problem is a well-known NP-hard problem with many practical applications. In this work, a new approach based on Binary Firefly Algorithm is proposed to solve this problem. The Firefly Algorithm has attracted much attention and has been applied to many optimization problems. Here, we demonstrate that is also able to produce very competitive results solving the portfolio of set covering problems from the OR-Library.

Keywords Set covering problem · Binary firefly algorithm · Metaheuristic

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 [1], facility location problem [2], and production planning in

B. Crawford (✉) · R. Soto · M. Olivares-Suárez
Pontificia Universidad Católica de Valparaíso, Valparaíso, Chile
e-mail: broderick.crawford@ucv.cl

R. Soto
e-mail: ricardo.soto@ucv.cl

M. Olivares-Suárez
e-mail: miguel.olivares.s@mail.pucv.cl

B. Crawford
Universidad Finis Terrae, Santiago, Chile

R. Soto
Universidad Autónoma de Chile, Santiago, Chile

F. Paredes
Universidad Diego Portales, Santiago, Chile
e-mail: fernando.paredes@udp.cl

R. Silhavy et al. (eds.), *Modern Trends and Techniques in Computer Science*,
Advances in Intelligent Systems and Computing 285, DOI: 10.1007/978-3-319-06740-7_6,

industry [3]. The SCP is a well-known NP-hard in the strong sense [4]. Many algorithms have been developed to solve it has been reported to literature. Exact algorithms are mostly based on branch-and-bound and branch-and-cut [5, 6]. 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 [7]. An improved greedy algorithm by incorporating randomness and memory into it and obtained promising results [8]. 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 [9, 10]. As top-level general search strategies, metaheuristics were also applied to the SCP. An incomplete list of this kind of heuristics for the SCP includes genetic algorithm [11], simulated annealing algorithm [12], tabu search algorithm [13], evolutionary algorithms [14], ant colony optimization (ACO) [15], electromagnetism (unicost SCP) [16], gravitational emulation search [17] and cultural algorithms [18]. A deeper comprehension of most of the effective algorithms for the SCP can be found in [19].

In this paper, a new approach based on Binary Firefly Algorithm for the SCP is presented. Firefly Algorithm (FA) is a recently developed, population-based metaheuristic [20, 21]. So far, it has been shown that firefly algorithm is very efficient in dealing with multimodal, global optimization problems. For a deeper comprehension of review of firefly advances and applications please refer to [22, 23]. Researches on FA for SCP have not been seen to date.

This paper is organized as follows: In Sect. 2, we formally describe the SCP. The Sect. 3, we present the overview of FA. The description of the proposed approach is described in Sect. 3. In Sect. 5, we present experimental results obtained when applying the algorithm for solving the 65 instances different of SCP. Finally, in Sect. 6 we conclude the paper.

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 $I = \{1, \ldots, m\}$ and $J = \{1, \ldots, n\}$ 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

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

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 Overview of Firefly Algorithm

Nature-inspired methodologies are among the most powerful algorithms for optimization problems. The Firefly Algorithm (FA) 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 [20, 21]. The pseudo code of the firefly-inspired algorithm was developed using these 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 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.
- Finally, 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 and vice versa.

As the attractiveness of a firefly is proportional to the light intensity seen by adjacent fireflies, we can now define the variation of attractiveness β with the distance r by

$$\beta = \beta_0 e^{-\gamma r^2} \tag{4}$$

where β_0 is the attractiveness at $r = 0$. The distance r_{ij} between two fireflies is determined by

$$r_{ij} = \left\| x^i - x^j \right\| = \sqrt{\sum_{k=1}^{d} \left(x_k^i - x_k^j \right)^2} \tag{5}$$

where x_k^i is the kth component of the spatial coordinate of the ith 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

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

where x_{ij} is the firefly position of the next generation. x_i^t and x_j^t are the current position of the fireflies and x_i^{t+1} is the ith firefly position of the next generation. The second term is due to attraction. The third term introduces randomization, with α being the randomization parameter and "rand" is a random number generated uniformly but distributed between 0 and 1. The value of γ determines the variation of attractiveness, which corresponds to the variation of distance from the communicated firefly. When $\gamma = 0$, there is no variation or the fireflies have constant attractiveness. When $\gamma = 1$, it results in attractiveness being close to zero, which again is equivalent to the complete random search. In general, the value of γ [20, 21] is in between [0, 10].

4 Description of the Proposed Approach

In this section, the FA is proposed to solve the SCP using binary representation.

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

Step 2 Initialization of firefly position. Initialize randomly $M = [X_1; X_2; \ldots; X_m]$ 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 the 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 SCP (Eq. 1).

Step 4 Modification of firefly position. A firefly produces a modification in the position based on the brightness between the fireflies. The new position is determined by modifying the value (old firefly position) using Eq. 6 for each dimension of a firefly. The result of the new component of the firefly, is probable to be a real number, to fix this, apply a threshold of 0 and 1. If x'_p is greater than the threshold, it is very likely to choose 1, otherwise 0. The threshold level can be made to range from 0 to 1, and in order to achieve this a tanh function is used as given in [24].

$$\tanh\left(\left|x'_p\right|\right) = \frac{\exp(2 * |x'_p|) - 1}{\exp(2 * |x'_p|) + 1} \tag{7}$$

Step 5 The new solution is subjected to an evaluation, if is not a feasible solution generated then is repaired. To make feasible solution is to determine which rows have not yet been covered and choose the columns needed for coverage. The search for these columns is based in: cost of a column/number

Table 1 Details of the test instances

Instance set	No. of instances	m	n	Cost range	Density (%)	Optimal solution
4	10	200	1,000	[1, 100]	2	Known
5	10	200	2,000	[1, 100]	2	Known
6	5	200	1,000	[1, 100]	5	Known
A	5	300	3,000	[1, 100]	2	Known
B	5	300	3,000	[1, 100]	5	Known
C	5	400	4,000	[1, 100]	2	Known
D	5	400	4,000	[1, 100]	5	Known
NRE	5	500	5,000	[1, 100]	10	Unknown
NRF	5	500	5,000	[1, 100]	20	Unknown
NRG	5	1,000	10,000	[1, 100]	2	Unknown
NRH	5	1,000	10,000	[1, 100]	5	Unknown

of rows not covered that cover the column *j*. Once the solution has become feasible applies optimization step to eliminate those redundant columns. A redundant column is that if removed, the solution remains feasible.

Step 6 Memorize the best solution achieved so far. Increment the generation count.

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

5 Experiments and Results

The performance of Binary Firefly Algorithm was evaluated experimentally using 65 SCP test instances from OR-Library of Beasley [25]. These instances are divided into 11 groups and each group contains 5 or 10 instances. Table 1 shows their detailed information where "Density" is the percentage of non-zero entries in the SCP matrix. The algorithm was coded in C in the development environment NetBeans 7.3 with support for C/C++ and run on a PC with a 1.8 GHz Intel Core 2 Duo T5670 CPU and 3.0 GB RAM, under Windows 8 system.

In all experiments, the Binary Firefly Algorithm is executed 50 generations, and 30 times each instance. This number was determined by the rapid convergence to a local optimal closest to global optimum. We used a population of 25 fireflies. The parameters γ, β_0 are initialized to 1. These parameters were selected empirically after a large number of tests and showed good results but may not be optimal for all instances.

Table 2 shows the results obtained of the 65 instances. Column "Optimum" reports the optimal or the best known solution value of each instance. Columns "Min. value found", "Max. value found" and "Average" reports the minimum, maximum, and average of the best solutions obtained in the 30 executions.

Table 2 Computational results on 65 instances of SCP

Instance	Optimum	Min. value found	Max. value found	Average
4.1	429	481	482	481.03
4.2	512	580	580	580.00
4.3	516	619	620	619.03
4.4	494	537	537	537.00
4.5	512	609	609	609.00
4.6	560	653	653	653.00
4.7	430	491	492	491.07
4.8	492	565	565	565.00
4.9	641	749	750	749.03
4.10	514	550	550	550.00
5.1	253	296	297	296.03
5.2	302	372	372	372.00
5.3	226	250	250	250.00
5.4	242	277	278	277.07
5.5	211	253	253	253.00
5.6	213	264	265	264.03
5.7	293	337	337	337.00
5.8	288	326	326	326.00
5.9	279	350	350	350.00
5.10	265	321	321	321.00
6.1	138	173	174	173.03
6.2	146	180	181	180.07
6.3	145	160	160	160.00
6.4	131	161	161	161.00
6.5	161	186	186	186.00
A.1	253	285	285	285.00
A.2	252	285	286	285.07
A.3	232	272	272	272.00
A.4	234	297	297	297.00
A.5	236	262	262	262.00
B.1	69	80	81	80.03
B.2	76	92	92	92.00
B.3	80	93	93	93.00
B.4	79	98	99	98.03
B.5	72	87	87	87.00
C.1	227	279	279	279.00
C.2	219	272	272	272.00
C.3	243	288	288	288.00
C.4	219	262	262	262.00
C.5	215	262	263	262.07
D.1	60	71	71	71.00
D.2	66	75	75	75.00
D.3	72	88	88	88.00
D.4	62	71	71	71.00
D.5	61	71	71	71.00
NRE.1	29	32	33	32.03

(continued)

Table 2 (continued)

Instance	Optimum	Min. value found	Max. value found	Average
NRE.2	30	36	36	36.00
NRE.3	27	35	35	35.00
NRE.4	28	34	34	34.00
NRE.5	28	34	34	34.00
NRF.1	14	17	18	17.03
NRF.2	15	17	17	17.00
NRF.3	14	21	21	21.00
NRF.4	14	19	19	19.00
NRF.5	13	16	16	16.00
NRG.1	176	230	231	230.03
NRG.2	154	191	191	191.00
NRG.3	166	198	198	198.00
NRG.4	168	214	214	214.00
NRG.5	168	223	223	223.00
NRH.1	63	85	86	85.07
NRH.2	63	81	82	81.03
NRH.3	59	76	76	76.00
NRH.4	58	75	75	75.00
NRH.5	55	68	68	68.00

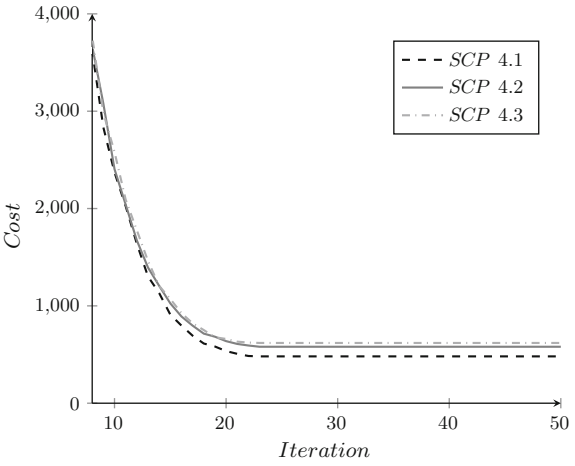

Fig. 1 Evolution of mean best values for SCP4.1, SCP4.2 and SCP4.3

In the Fig. 1 shows the evolution of mean best values for the instances 4.1, 4.2 and 4.3, which shows the rapid convergence of cost minimization.

6 Conclusions

As can be seen from the results obtained, the metaheuristic behaves of good way in almost all instances, with the first set columns instances between 1,000 and 2,000, there is a mean cost difference of 54 between the global optimum with the best optimum obtained, and starts to decrease. With a set of columns in 5,000, Firefly behaves very well coming to have a difference of 2 with respect to the best known solution value (NRF.2). This paper has demonstrated the Binary Firefly Algorithm is a valid alternative to solve the SCP, being that its main use is for continuous domains.

An interesting research direction to pursue in future work about the integration of autonomous search in the solving process, which in many cases has demonstrated excellent results [26–29].

Acknowledgements The author B. Crawford is supported by Grant CONICYT/FONDECYT/REGU- LAR/1140897. The author R. Soto is supported by Grant CONICYT/FON- DECYT/INICIACION/11130459. The author F. Paredes is supported by Grant CONICYT/FONDECYT/REGULAR/1130455.

References

1. Housos, E., Elmoth, T.: Automatic optimization of subproblems in scheduling airlines crews. Interfaces **27**(5), 68–77 (1997)
2. Vasko, F.J., Wilson, G.R.: Using a facility location algorithm to solve large set covering problems. Oper. Res. Lett. **3**(2), 85–90 (1984)
3. Vasko, F.J., Wolf, F.E.: Optimal selection of ingot sizes via set covering. Oper. Res. **35**(3), 346–353 (1987)
4. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co, New York (1990)
5. Balas, E., Carrera, M.C.: A dynamic subgradient-based branch-and-bound procedure for set covering. Oper. Res. **44**(6), 875–890 (1996)
6. Fisher, M.L., Kedia, P.: Optimal solution of set covering/partitioning problems using dual heuristics. Manage. Sci. **36**(6), 674–688 (1990)
7. Chvatal, V.: A greedy heuristic for the set-covering problem. Math. Oper. Res. **4**(3), 233–235 (1979)
8. Lan, G., DePuy, G.W.: On the effectiveness of incorporating randomness and memory into a multi-start metaheuristic with application to the set covering problem. Comput. Ind. Eng. **51**(3), 362–374 (2006)
9. Ceria, S., Nobili, P., Sassano, A.: A Lagrangian-based heuristic for large-scale set covering problems. Math. Program. **81**(2), 215–228 (1998)
10. Caprara, A., Fischetti, M., Toth, P.: A heuristic method for the set covering problem. Oper. Res. **47**(5), 730–743 (1999)
11. Beasley, J.E., Chu, P.C.: A genetic algorithm for the set covering problem. Eur. J. Oper. Res. **94**(2), 392–404 (1996)
12. 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)

13. Caserta, M.: Tabu search-based metaheuristic algorithm for large-scale set covering problems. In: Doerner, K., Gendreau, M., Greistorfer, P., Gutjahr, W., Hartl, R., Reimann, M. (eds.) Metaheuristics. Vol. 39 of Operations Research/Computer Science Interfaces Series, pp. 43–63. Springer, US (2007)
14. Crawford, B., Lagos, C., Castro, C., Paredes, F.: A evolutionary approach to solve set covering. In: Cardoso, J., Cordeiro, J., Filipe, J. (eds.) In: Proceedings of the 9th International Conference on Enterprise Information Systems (ICEIS '07), Vol. AIDSS, pp. 356-363. Funchal, Portugal. 12-16 June 2007
15. Ren, Z.G., Feng, Z.R., Ke, L.J., Zhang, Z.J.: New ideas for applying ant colony optimization to the set covering problem. Comput. Ind. Eng. **58**(4), 774–784 (2010)
16. 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)
17. Balachandar, S.R., Kannan, K.: A meta-heuristic algorithm for set covering problem based on gravity. J. Comput. Math. Sci. **4**, 223–228 (2010)
18. Crawford, B., Soto, R., Monfroy, E.: Cultural algorithms for the set covering problem. In: Tan, Y., Shi, Y., Mo, H. (eds.) ICSI (2). Vol. 7929 of Lecture Notes in Computer Science, pp. 27–34. Springer, Berlin (2013)
19. Caprara, A., Toth, P., Fischetti, M.: Algorithms for the set covering problem. Ann. Oper. Res. **98**, 353–371 (2000)
20. Yang, X.S.: Nature-Inspired Metaheuristic Algorithms. Luniver Press, UK (2008)
21. Yang, X.S.: Firefly algorithms for multimodal optimisation, In: Proceedings of the 5th International Conference on Stochastic Algorithms: Foundations and Applications. SAGA'09, pp. 169–178. Springer, Berlin (2009)
22. Fister, I., Fister Jr, I., Yang, X.S., Brest, J.: A comprehensive review of firefly algorithms. Swarm Evol. Comput. **13**, 34–46 (2013)
23. Yang, X.S., He, X.: Firefly Algorithm: Recent Advances and Applications. The Computing Research Repository, abs/1308.3898 (2013)
24. Chandrasekaran, K., Sishaj, P.S., Padhy, N.P.: Binary real coded firefly algorithm for solving unit commitment problem. Inf. Sci. **249**, 67–84 (2013)
25. Beasley, J.E.: A Lagrangian heuristic for set covering problems. Naval Res. Logistics **37**(1), 151–164 (1990)
26. Crawford, B., Soto, R., Monfroy, E., Palma, W., Castro, C., Paredes, P.: Parameter tuning of a choice-function based hyperheuristic using particle swarm optimization. Expert Syst. Appl. **40**(5), 1690–1695 (2013)
27. Soto, R., Crawford, B., Monfroy, E., Bustos, V.: Using autonomous search for generating good enumeration strategy blends in constraint programming. In: Proceedings of the 12th International Conference on Computational Science and Its Applications (ICCSA). Vol. 7335 of LNCS, pp. 607–617. Springer (2012)
28. Crawford, B., Soto R., Montecinos M., Castro C., Monfroy, E.: A framework for autonomous search in the eclipse solver. In: Proceedings of the 24th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE). Vol. 6703 of LNCS, pp. 79–84. Springer (2011)
29. Crawford, B., Soto R., Castro C., Monfroy, E.: Extensible CP-based autonomous search. In: Proceedings of HCI International. Vol. 173 of CCIS, pp. 561–565. Springer (2011)