Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 578541, 8 pages
http://dx.doi.org/10.1155/2015/578541

Research Article

A Binary Cat Swarm Optimization Algorithm for the Non-Unicost Set Covering Problem

Broderick Crawford,[1,2,3] Ricardo Soto,[1,4,5] Natalia Berríos,[1] Franklin Johnson,[1,6] Fernando Paredes,[7] Carlos Castro,[8] and Enrique Norero[9]

[1]*Pontificia Universidad Católica de Valparaíso, 2362807 Valparaíso, Chile*
[2]*Universidad San Sebastián, 8420524 Santiago, Chile*
[3]*Universidad Central de Chile, 8370178 Santiago, Chile*
[4]*Universidad Autónoma de Chile, 7500138 Santiago, Chile*
[5]*Universidad Cientifica del Sur, Lima 18 Lima, Peru*
[6]*Universidad de Playa Ancha, 2360003 Valparaíso, Chile*
[7]*Escuela de Ingeniería Industrial, Universidad Diego Portales, 8370109 Santiago, Chile*
[8]*Universidad Técnica Federico Santa María, 2390123 Valparaíso, Chile*
[9]*Facultad de Ingeniería, Universidad Santo Tomás, 2561694 Viña del Mar, Chile*

Correspondence should be addressed to Broderick Crawford; broderick.crawford@ucv.cl

Received 27 February 2015; Revised 27 May 2015; Accepted 22 June 2015

Academic Editor: Filippo Ubertini

The Set Covering Problem consists in finding a subset of columns in a zero-one matrix such that they cover all the rows of the matrix at a minimum cost. To solve the Set Covering Problem we use a metaheuristic called Binary Cat Swarm Optimization. This metaheuristic is a recent swarm metaheuristic technique based on the cat behavior. Domestic cats show the ability to hunt and are curious about moving objects. Based on this, the cats have two modes of behavior: seeking mode and tracing mode. We are the first ones to use this metaheuristic to solve this problem; our algorithm solves a set of 65 Set Covering Problem instances from OR-Library.

1. Introduction

The Set Covering Problem (SCP) [1–3] is a classic problem that consists in finding a set of solutions which allow to cover a set of needs at the lowest cost possible. There are many applications of these kinds of problems; the main ones are location of services, files selection in a data bank, simplification of boolean expressions, and balancing production lines, among others.

In the field of optimization, many algorithms have been developed to solve the SCP. Examples of these optimization algorithms include Genetic Algorithm (GA) [4–7], Ant Colony Optimization (ACO) [8, 9], and Particle Swarm Optimization (PSO) [10–13]. In this work we use a Cat Swarm Optimization (CSO) algorithm to solve the SCP.

By simulating the behavior of cats, CSO can solve optimization problems. It has been analysed that cats spend most of their time resting when they are awake. While they rest, they move from their position carefully and slowly. This behavioral mode is the one called seeking mode. In the tracing mode, a cat moves according to its own speed for all dimensions. This search method will be discussed in detail later in this paper.

The CSO was originally developed for continuous valued spaces. But there exist a number of optimization problems, as the SCP, in which the values are not continuous numbers but rather discrete binary integers. Sharafi et al. introduced a discrete binary version of CSO for discrete optimization problems: Binary Cat Swarm Optimization (BCSO) [14]. BCSO is based on CSO algorithm proposed by Chu et al.

in 2006 [15]. The difference is that in BCSO the vector position consists of ones and zeros, instead of the real numbers of CSO.

In this paper we use a BCSO algorithm to solve the Set Covering Problem. Our proposal is tested in different instances of SCP.

To the best of our knowledge, this is the first work solving SCP with BCSO.

This paper is organized as follows. In Section 2, there is a brief description of what Set Covering Problem is. Section 3 is about what BCSO is and the explanation and algorithm of behaviors. In Section 4, there is an explanation of how BCSO was used for solving the SCP. Section 5 discusses an analysis and results table. Finally, Section 6 is the conclusions.

2. Set Covering Problem

The SCP [16–18] 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 can cover a row 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

$$v(\text{SCP}) = \min \quad \sum_{j \in J} c_j x_j \tag{1}$$

$$\text{subject to} \quad \sum_{j \in J} a_{ij} x_j \geq 1, \quad \forall i \in I, \tag{2}$$

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

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

The SCP has been applied to many real world problems such as crew scheduling [19–21], location of emergency facilities [22, 23], production planning in industry [24–26], vehicle routing [27, 28], ship scheduling [29, 30], network attack or defense [31], assembly line balancing [32, 33], traffic assignment in satellite communication systems [34, 35], simplifying boolean expressions [36], the calculation of bounds in integer programs [37], information retrieval [38], political districting [39], stock cutting, crew scheduling problems in airlines [40], and other important real life situations. Because it has wide applicability, we deposit our interest in solving the SCP.

3. Binary Cat Swarm Optimization

Among the known felines, there are about thirty different species, for example, lion, tiger, leopard, and cat [41]. Though many have different living environments, cats share similar behavior patterns [42].

For wild cats, the hunting skill ensures their food supply and survival of their species [43]. Feral cats are groups with a mission to hunt for their food and are very wild feline colonies, with a range of 2–15 individuals [44].

Domestic cats also show the same ability to hunt and are curious about moving objects [45–47]. Watching the cats, you would think that most of the time is spent resting, even when awake [48, 49]. In this state of alertness they do not leave; they may be listening or with wide eyes looking around [50]. Based on all these behaviors we formulate BCSO.

Binary Cat Swarm Optimization [14] is an optimization algorithm that imitates the natural behavior of cats [51, 52]. Cats have curiosity about objects in motion and have a great hunting ability. It might be thought that cats spend most of the time resting, but in fact they are constantly alert and moving slowly. This behavior corresponds to the seeking mode. Furthermore, when cats detect a prey, they spend lots of energy because of their fast movements. This behavior corresponds to the tracing mode. In BCSO, these two behaviors are modeled mathematically to solve complex optimization problems.

In BCSO, the first decision is the number of cats needed for each iteration. Each cat, represented by cat_k, where $k \in [1, C]$, has its own position consisting of M dimensions, which are composed by ones and zeros. Besides, they have speed for each dimension d, a flag for indicating if the cat is on seeking mode or tracing mode, and finally a fitness value that is calculated based on the SCP. The BCSO keeps searching for the best solution until the end of iterations.

In BCSO the bits of the cat positions are $x_j = 1$ if column j is in the solution, 0 otherwise (1). Cat position represents the solution of the SCP and the constraint matrix ensures that each row i is covered by at least one column.

Next the BCSO general Algorithm 1 and diagram (Figure 1) are described where MR is a percentage that determines the number of cats that undertake the seeking mode.

3.1. Seeking Mode. This submodel is used to model the situation of the cat, which is resting, looking around, and seeking the next position to move to. Seeking mode has essential factors: PMO (Probability of Mutation Operation); CDC (Counts of Dimensions to Change) which indicate how many of the dimensions varied; SMP (Seeking Memory Pool) which is used to define the size of seeking memory for each cat. SMP indicates the points explored by the cat; this parameter can be different for different cats.

Algorithm 1 (BCSO()).

(1) Create C cats;

(2) initialize the cat positions randomly with values between 1 and 0;

(3) initialize velocities and flag of every cat;

(4) set some cats into seeking mode according to MR, and set the others into tracing mode;

(5) evaluate the cats according to the fitness function;

(6) keep the best cat which has the best fitness value into *bestcat* variable;

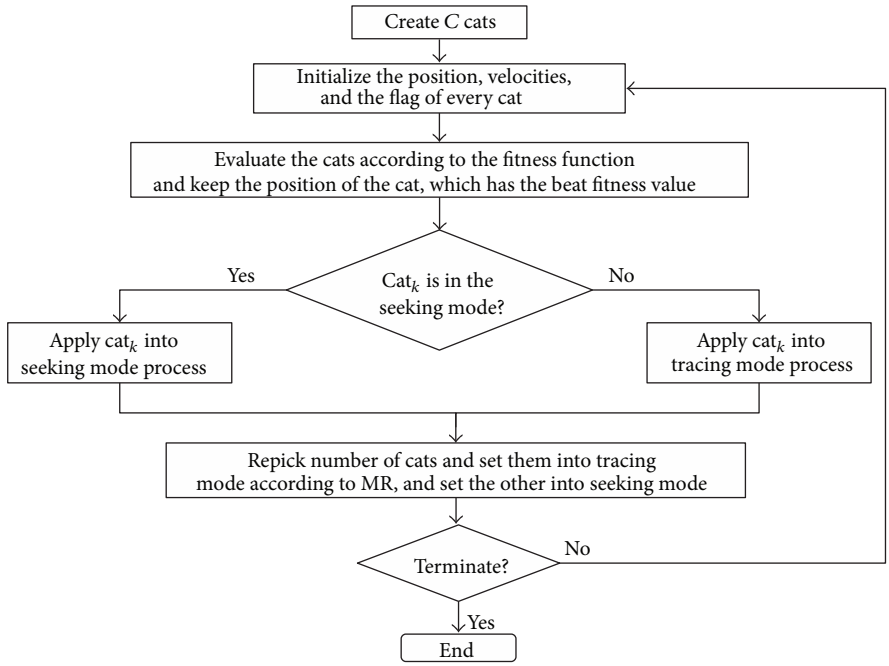

Figure 1: Binary Cat Swarm Optimization.

(7) move the cats according to their flags; if cat$_k$ is in seeking mode, apply the cat to the seeking mode process; otherwise apply it to the tracing mode process. The process steps are presented above;

(8) repick number of cats and set them into tracing mode according to MR; then set the other cats into seeking mode;

(9) check the termination condition; if satisfied, terminate the program, and otherwise repeat since step (5).

The following pseudocode and diagram (Figure 2) describe cat behavior seeking mode in which FS_i is the fitness of ith cat and $FS_b = FS_{\max}$ for finding the minimum solution and $FS_b = FS_{\min}$ for finding the maximum solution. To solve the SCP we use $FS_b = FS_{\max}$.

Step 1. Create SMP copies of cat$_k$.

Step 2. Based on CDC, update the position of each copy randomly according to PMO.

Step 3. Evaluate the fitness of all copies.

Step 4. Calculate the selecting probability of each copy according to

$$P_i = \frac{FS_i - FS_b}{FS_{\max} - FS_{\min}}. \tag{4}$$

Step 5. Apply roulette wheel to the candidate points and select one.

Step 6. Replace the current position with the selected candidate.

3.2. Tracing Mode. Tracing mode is the submodel for modeling the case of the cat in tracing targets. In the tracing mode, cats are moving towards the best target. Once a cat goes into tracing mode, it moves according to its own velocities for each dimension. Every cat has two velocity vectors which are defined as V_{kd}^1 and V_{kd}^0. V_{kd}^0 is the probability of the bits of the cat to change to zero while V_{kd}^1 is the probability that bits of cat change to one. The velocity vector changes its meaning to the probability of mutation in each dimension of a cat. The tracing mode action is described in the next steps and diagram (Figure 3).

Step 1. Calculate d_{kd}^1 and d_{kd}^0 where $X_{\text{best},d}$ is the d dimension of best cat, r_1 has random values in the interval of [0, 1], and c_1 is a constant which is defined by the user:

$$\begin{aligned} &\text{if } X_{\text{best},d} = 1 \text{ then } d_{kd}^1 = r_1 c_1,\ d_{kd}^0 = -r_1 c_1, \\ &\text{if } X_{\text{best},d} = 0 \text{ then } d_{kd}^1 = -r_1 c_1,\ d_{kd}^0 = r_1 c_1. \end{aligned} \tag{5}$$

Step 2. Update process of V_{kd}^1 and V_{kd}^0 is as follows, where w is the inertia weight and M is the column numbers:

$$\begin{aligned} V_{kd}^1 &= wV_{kd}^1 + d_{kd}^1 \\ V_{kd}^0 &= wV_{kd}^0 + d_{kd}^0 \\ &\qquad d = 1, \ldots, M. \end{aligned} \tag{6}$$

Step 3. Calculate the velocity of cat$_k$, V'_{kd}, according to

$$V'_{kd} = \begin{cases} V_{kd}^1 & \text{if } X_{kd} = 0 \\ V_{kd}^0 & \text{if } X_{kd} = 1. \end{cases} \tag{7}$$

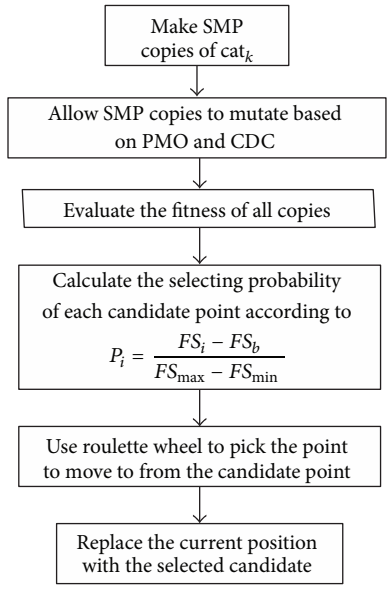

Figure 2: Seeking mode.

Step 4. Calculate the probability of mutation in each dimension; this is defined by parameter t_{kd}; t_{kd} takes a value in the interval of $[0, 1]$:

$$t_{kd} = \frac{1}{1 + e^{-V'_{kd}}}. \quad (8)$$

Step 5. Based on the value of t_{kd}, the new value of each dimension of cat is updated as follows where rand is an aleatory variable $\in [0, 1]$:

$$X_{kd} = \begin{cases} X_{\text{best},d} & \text{if rand} < t_{kd} \\ X_{kd} & \text{if } t_{kd} < \text{rand} \end{cases} \quad d = 1, \ldots, M. \quad (9)$$

The maximum velocity vector of V'_{kd} should be bounded to a value $V_{\max}$. If the value of V'_{kd} becomes larger than $V_{\max}$, $V_{\max}$ should be selected for velocity in the corresponding dimension.

4. Solving the Set Covering Problem

Next, the Solving SCP pseudocode is described.

Algorithm 2 (*solving* SCP()).

(1) Initialize parameters in cats;

(2) initialize cat positions: randomly initialize cat positions with values between 0 and 1;

(3) initialize all parameters of BCSO;

(4) evaluate the fitness of the population. In this case the fitness function is equal to the objective function of the SCP;

(5) change the position of the cat. A cat produces a modification in the position based on one of the behaviors, that is, seeking mode or tracing mode;

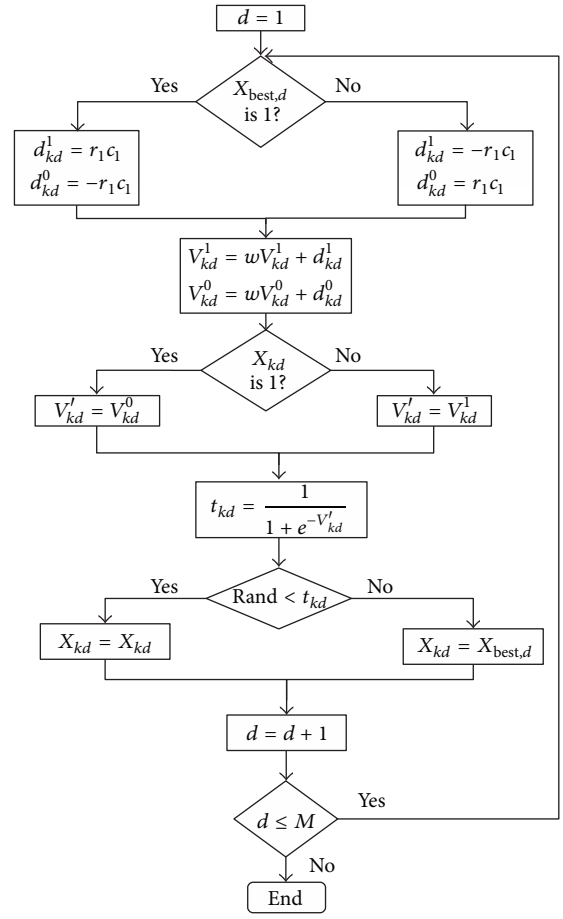

Figure 3: Tracing mode.

(6) if solution is not feasible then it is repaired. Each row i must be covered by at least one column; to choose the missing columns do the cost of a column/(number of not covered rows that can cover column j);

(7) eliminate the redundant columns. A redundant column is one that if removed the solution remains feasible;

(8) memorize the best found solution. Increase the number of iterations;

(9) stop the process and show the result if the completion criteria are met. Completion criteria used in this work are the number specified maximum of iterations. Otherwise, go to step (3).

5. Results

The BCSO performance was evaluated experimentally using 65 SCP test instances from the OR-Library of Beasley [53]. The optimization algorithm was coded in Java in NetBeans IDE 7.1 and executed on a computer with 2.53 GHz Intel Core i3 M380 CPU and 3.0 GB of RAM under Windows 7 Operating System.

Table 1: Parameter values.

Name	Parameter	Value
Number of cats	C	30
Mixture ratio	MR	0.5
Seeking Memory Pool	SMP	20
Probability of Mutation Operation	PMO	1
Counts of Dimensions to Change	CDC	0.001
Inertia weight	w	1
Factor c_1	c_1	1

Table 2: Computation times of BCSO and ABC.

Instance set	Number of instances	Time_{avg} BCSO (s)	Time_{avg} ABC (s)
4	10	4.5	5.1
5	10	5.6	6.6
6	5	7.0	11.0
A	5	11.7	11.2
B	5	25.8	37.9
C	5	20.0	17.7
D	5	49.4	72.2
NRE	5	71.6	96.6
NRF	5	52.9	314.2
NRG	5	151.8	94.8
NRH	5	213.4	545.7

In all experiments the BCSO was executed with 500 iterations and 30 times each instance. See Table 1 with the parameters of BCSO.

To select these parameters, the algorithm was executed 10 times, each one for the different population sizes of 1000, 100, 50, and 30 cats, keeping all other parameters constant. As the population size was decreased, the fitness value converges towards a minimum. The value of MR was varied while all other parameters are kept constant. The value 0.5 for MR was given best solutions. The value of SMP was varied between 1000, 500, 300, 100, and 20. All values gave the same result; for very large values good results were obtained in many iterations; small values of SMP to the same results were obtained in a few iterations. The values of w and c_1 were tested with values between 0 and 1, and both showed no influence on the results; w of 1 and c_1 of 1 are the reasonable choice of parameters.

Tables 3 and 4 show the results of the 65 instances. The Z_{opt} column reports the optimal value or the best known solution for each instance. The Z_{min}, Z_{max}, and Z_{avg} columns report the lowest cost, highest cost, and the average of the best solutions obtained in 30 runs, respectively. The quality of a solution is evaluated in terms of the percentage deviation relative (RPD) of the solutions reached Z_b and Z_{opt} (which can be either the optimal or the best known objective value). RPD_{min} was evaluated using $Z_b = Z_{\text{min}}$ and RPD_{avg} was

Table 3: Computational results on 65 instances of SCP: part 1.

Instance	Z_{opt}	Z_{min}	Z_{max}	Z_{avg}	RPD_{avg}	RPD_{min}	Time (s)
4.1	429	459	485	479.6	11.8	7.0	4.77
4.2	512	570	599	594.2	16.1	11.3	4.08
4.3	516	590	614	606.8	17.6	14.3	4.36
4.4	494	547	585	578.3	17.1	10.7	4.62
4.5	512	545	558	554.2	8.2	6.4	4.69
4.6	560	637	655	649.9	16.1	13.8	4.56
4.7	430	462	469	467.4	8.7	7.4	4.24
4.8	492	546	571	566.9	15.2	11.0	4.99
4.9	641	711	741	725.0	13.1	10.9	4.64
4.10	514	537	556	552.1	7.4	4.5	4.55
5.1	253	279	283	281.6	11.3	10.3	5.95
5.2	302	339	340	339.9	12.5	12.3	5.36
5.3	226	247	252	250.5	10.8	9.3	5.89
5.4	242	251	254	253.2	4.6	3.7	5.28
5.5	211	230	231	230.4	9.2	9.0	4.90
5.6	213	232	244	242.7	13.9	8.9	5.24
5.7	293	332	343	338.0	15.4	13.3	5.66
5.8	288	320	331	329.9	14.5	11.1	5.82
5.9	279	295	299	298.6	7.0	5.7	6.10
5.10	265	285	288	286.9	8.3	7.5	5.62
6.1	138	151	166	159.9	15.9	9.4	5.83
6.2	146	152	160	157.4	7.8	4.1	5.73
6.3	145	160	166	164.3	13.3	10.3	8.05
6.4	131	138	143	141.7	8.2	5.3	7.60
6.5	161	169	176	172.8	7.3	5.0	7.61
A.1	253	286	287	286.9	13.4	13.0	8.62
A.2	252	274	280	276.3	9.6	8.7	13.42
A.3	232	257	264	263.1	13.4	10.8	12.27
A.4	234	248	252	251.3	7.2	6.0	11.72
A.5	236	244	244	244	3.3	3.0	12.42
B.1	69	79	79	79	14.5	14.5	26.36
B.2	76	86	90	88.5	16.2	13.2	25.91
B.3	80	85	87	85.4	6.5	6.3	24.46
B.4	79	89	89	89	12.7	12.7	27.77
B.5	72	73	73	73	1.5	1.4	24.65
C.1	227	242	243	242.4	6.7	6.6	19.15
C.2	219	240	244	240.8	9.9	9.6	20.83
C.3	243	277	279	278	14.4	14.0	20.38
C.4	219	250	250	250	13.2	12.3	20.16
C.5	215	243	247	244.3	13.6	13.0	19.47
D.1	60	65	66	65.7	9.5	8.3	52.53
D.2	66	70	71	70.1	6.2	6.1	51.36
D.3	72	79	81	80.8	12.2	9.7	49.89
D.4	62	64	67	66.6	7.4	3.2	46.48
D.5	61	65	66	65.6	7.5	6.6	46.61

Table 4: Computational results on 65 instances of SCP: part 2.

Instance	Z_{opt}	Z_{min}	Z_{max}	Z_{avg}	RPD_{avg}	RPD_{min}	Time (s)
NRE.1	29	29	30	29.9	3.1	0.0	65.76
NRE.2	30	34	35	34.2	14.0	13.3	72.45
NRE.3	27	31	32	31.5	16.7	14.8	75.40
NRE.4	28	32	33	32.9	17.5	14.3	71.34
NRE.5	28	30	31	30.3	8.2	7.1	73.01
NRF.1	14	17	18	17.1	22.1	21.4	51.33
NRF.2	15	18	19	18.2	21.3	20.0	52.53
NRF.3	14	17	18	17.2	22.9	21.4	57.10
NRF.4	14	17	18	17.1	22.1	21.4	54.74
NRF.5	13	15	16	15.9	22.3	15.4	48.73
NRG.1	176	190	194	192.7	9.5	8.0	139.46
NRG.2	154	165	167	166	7.8	7.1	149.43
NRG.3	166	187	191	187.7	21.1	20.6	156.04
NRG.4	168	179	185	183.2	9.0	6.5	141.03
NRG.5	168	181	186	184.3	9.7	7.7	172.95
NRH.1	63	70	74	71.2	13.0	11.1	195.67
NRH.2	63	67	67	67	6.3	6.3	207.34
NRH.3	59	68	74	69.6	18.0	15.3	224.36
NRH.4	58	66	68	66.6	14.8	13.8	214.68
NRH.5	55	61	66	61.5	11.8	10.9	226.03

evaluated using $Z_b = Z_{avg}$. And finally the time (s) column reports the average computational time in seconds. Consider

$$\text{RPD} = \left(\frac{Z_b - Z_{opt}}{Z_{opt}} \right) * 100. \quad (10)$$

Table 2 shows the average computation times of the SCP instances set. Our proposed algorithm in the most instances was solved in less time than ABC [54]. The difference of seconds between BCSO and ABC is as follows: in the NRE instance there is a difference of 25 seconds, in NRF it is of 261.3 seconds, and in the NRH it is of 332.3. Except for the A, C, and NRG instances, all other instances were solved in less time than the ABC. Therefore, the computation time of our algorithm is much better than the ABC algorithm.

6. Conclusions

In this paper we use a binary version of Cat Swarm Optimization to solve SCP using its column based representation (binary solutions). In binary discrete optimization problems the position vector is binary. This causes significant change in BCSO with respect to CSO with real numbers. In fact in BCSO in the seeking mode the slight change in the position takes place by introducing the mutation operation. The interpretation of velocity vector in tracing mode also changes to probability of change in each dimension of position of the cats. The proposed BCSO is implemented and tested using 65 SCP test instances from the OR-Library of Beasley. As can be seen from the results, metaheuristic performs well in most of cases. This paper has shown that the BCSO is a valid alternative to solve the SCP. The algorithm performs well regardless of the scale of the problem.

We believe that the differences in the computation times between BCSO and ABC are in our favor; this difference is very obvious in the big sets; in NRH the difference of seconds is of 332.3. Table 2 shows that our proposed algorithm is better than the artificial bee colony algorithm (ABC) with respect to computation time.

We can see the premature convergence problem, a typical problem in metaheuristics, which occurs when the cats quickly attain to dominate the population, constraining it to converge to a local optimum. For future works the objective will make them highly immune to be trapped in local optima and thus less vulnerable to premature convergence problem. Thus, we could propose an algorithm that shows improved results in terms of both computational time and quality of solution.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The author Broderick Crawford is supported by Grant CONICYT/FONDECYT/REGULAR/1140897, Ricardo Soto is supported by Grant CONICYT/FONDECYT/INICIACION/11130459, and Fernando Paredes is supported by Grant CONICYT/FONDECYT/REGULAR/1130455.

References

[1] B. Crawford, R. Soto, and E. Monfroy, "Cultural algorithms for the set covering problem," in *Advances in Swarm Intelligence: Proceedings of the 4th International Conference, ICSI 2013, Harbin, China, June 12–15, 2013, Part II*, Y. Tan, Y. Shi, and H. Mo, Eds., vol. 7929 of *Lecture Notes in Computer Science*, pp. 27–34, Springer, Berlin, Germany, 2013.

[2] B. Crawford, R. Soto, R. Cuesta, and F. Paredes, "Application of the artificial bee colony algorithm for solving the set covering problem," *The Scientific World Journal*, vol. 2014, Article ID 189164, 8 pages, 2014.

[3] D. Gouwanda and S. Ponnambalam, "Evolutionary search techniques to solve set covering problems," *World Academy of Science, Engineering and Technology*, vol. 39, pp. 20–25, 2008.

[4] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, London, UK, 3rd edition, 1996.

[5] L. Han, G. Kendall, and P. Cowling, "An adaptive length chromosome hyper-heuristic genetic algorithm for a trainer scheduling problem," in *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning*, pp. 267–271, 2002.

[6] D. E. Goldberg, "Real-coded genetic algorithms, virtual alphabets, and blocking," *Complex Systems*, vol. 5, no. 2, pp. 139–167, 1990.

[7] U. Aickelin, "An indirect genetic algorithm for set covering problems," *Journal of the Operational Research Society*, vol. 53, no. 10, pp. 1118–1126, 2002.

[8] F. Amini and P. Ghaderi, "Hybridization of harmony search and ant colony optimization for optimal locating of structural dampers," *Applied Soft Computing*, vol. 13, no. 5, pp. 2272–2280, 2013.

[9] Z.-G. Ren, Z.-R. Feng, L.-J. Ke, and Z.-J. Zhang, "New ideas for applying ant colony optimization to the set covering problem," *Computers & Industrial Engineering*, vol. 58, no. 4, pp. 774–784, 2010.

[10] B. Crawford, R. Soto, E. Monfroy, W. Palma, C. Castro, and F. Paredes, "Parameter tuning of a choice-a function based hyper-heuristic using particle swarm optimization," *Expert Systems with Applications*, vol. 40, no. 5, pp. 1690–1695, 2013.

[11] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC '99)*, pp. 1945–1950, IEEE, Washington, DC, USA, July 1999.

[12] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science (MHS '95)*, pp. 39–43, IEEE, Nagoya, Japan, October 1995.

[13] P. Angeline, "Evolutionary optimization versus particle swarm optimization: philosophy and performance differences," in *Evolutionary Programming VII*, V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, Eds., Lecture Notes in Computer Science, pp. 601–610, Springer, 1998.

[14] Y. Sharafi, M. A. Khanesar, and M. Teshnehlab, "Discrete binary cat swarm optimization algorithm," in *Proceedings of the 3rd IEEE International Conference on Computer, Control and Communication (IC4 '13)*, pp. 1–6, September 2013.

[15] S.-C. Chu, P.-W. Tsai, and J.-S. Pan, "Cat swarm optimization," in *PRICAI 2006: Trends in Artificial Intelligence*, vol. 4099 of *Lecture Notes in Computer Science*, pp. 854–858, Springer, Berlin, Germany, 2006.

[16] A. Caprara, M. Fischetti, and P. Toth, "Algorithms for the set covering problem," *Annals of Operations Research*, vol. 98, pp. 353–371, 2000.

[17] J. E. Beasley and K. Jørnsten, "Enhancing an algorithm for set covering problems," *European Journal of Operational Research*, vol. 58, no. 2, pp. 293–300, 1992.

[18] L. Lessing, I. Dumitrescu, and T. Stutzle, "A comparison between ACO algorithms for the set covering problem," in *Ant Colony Optimization and Swarm Intelligence*, vol. 3172 of *Lecture Notes in Computer Science*, pp. 1–12, Springer, Berlin, Germany, 2004.

[19] A. I. Ali and H. Thiagarajan, "A network relaxation based enumeration algorithm for set partitioning," *European Journal of Operational Research*, vol. 38, no. 1, pp. 76–85, 1989.

[20] I. Bartholdi, "A guaranteed-accuracy round-off algorithm for cyclic scheduling and set covering," *Operations Research*, vol. 29, no. 3, pp. 501–510, 1981.

[21] M. Desrochers and F. Soumis, "A column generation approach to the urban transit crew scheduling problem," *Transportation Science*, vol. 23, no. 1, pp. 1–13, 1989.

[22] W. Walker, "Using the set-covering problem to assign fire companies to fire houses," *Operations Research*, vol. 22, no. 2, pp. 275–277, 1974.

[23] F. J. Vasko and G. R. Wilson, "Using a facility location algorithm to solve large set covering problems," *Operations Research Letters*, vol. 3, no. 2, pp. 85–90, 1984.

[24] F. J. Vasko, F. E. Wolf, and K. L. Stott, "Optimal selection of ingot sizes via set covering," *Operations Research*, vol. 35, no. 3, pp. 346–353, 1987.

[25] F. J. Vasko, F. E. Wolf, and J. Stott, "A set covering approach to metallurgical grade assignment," *European Journal of Operational Research*, vol. 38, no. 1, pp. 27–34, 1989.

[26] F. J. Vasko, F. E. Wolf, K. L. Stott, and J. W. Scheirer, "Selecting optimal ingot sizes for bethlehem steel," *Interfaces*, vol. 19, no. 1, pp. 68–84, 1989.

[27] M. L. Balinski and R. E. Quandt, "On an integer program for a delivery problem," *Operations Research*, vol. 12, no. 2, pp. 300–304, 1964.

[28] B. A. Foster and D. M. Ryan, "An integer programming approach to the vehicle scheduling problem," *Operational Research Quarterly*, vol. 27, no. 2, pp. 367–384, 1976.

[29] M. L. Fisher and M. B. Rosenwein, "An interactive optimization system for bulk-cargo ship scheduling," *Naval Research Logistics*, vol. 36, no. 1, pp. 27–42, 1989.

[30] M. Bellmore, H. J. Geenberg, and J. J. Jarvis, "Multi-commodity disconnecting sets," *Management Science*, vol. 16, no. 6, pp. B-427–B-433, 1970.

[31] M. Bellmore and H. D. Ratliff, "Optimal defense of multi-commodity networks," *Management Science*, vol. 18, pp. B174–B185, 1971.

[32] B. A. Freeman and J. V. Jucker, "The line balancing problem," *Journal of Industrial Engineering*, vol. 18, pp. 361–364, 1967.

[33] M. E. Salveson, "The assembly line balancing problem," *Journal of Industrial Engineering*, vol. 6, pp. 18–25, 1955.

[34] C. C. Ribeiro, M. Minoux, and M. C. Penna, "An optimal column-generation-with-ranking algorithm for very large scale set partitioning problems in traffic assignment," *European Journal of Operational Research*, vol. 41, no. 2, pp. 232–239, 1989.

[35] S. Ceria, P. Nobili, and A. Sassano, "A Lagrangian-based heuristic for large-scale set covering problems," *Mathematical Programming*, vol. 81, no. 2, pp. 215–228, 1998.

[36] M. A. Breuer, "Simplification of the covering problem with application to Boolean expressions," *Journal of the Association for Computing Machinery*, vol. 17, pp. 166–181, 1970.

[37] N. Christofides, "Zero-one programming using non-binary tree-search," *The Computer Journal*, vol. 14, pp. 418–421, 1971.

[38] R. H. Day, "Letter to the editor—on optimal extracting from a multiple file data storage system: an application of integer programming," *Operations Research*, vol. 13, no. 3, pp. 482–494, 1965.

[39] R. S. Garfinkel and G. L. Nemhauser, "Optimal political districting by implicit enumeration techniques," *Management Science*, vol. 16, no. 8, pp. B-495–B-508, 1970.

[40] E. Housos and T. Elmroth, "Automatic optimization of subproblems in scheduling airline crews," *Interfaces*, vol. 27, no. 5, pp. 68–77, 1997.

[41] V. Aspinall, *Complete Textbook of Veterinary Nursing*, Butterworth-Heinemann, Oxford, UK, 2006.

[42] B. Pallaud, "Hypotheses on mechanisms underlying observational learning in animals," *Behavioural Processes*, vol. 9, no. 4, pp. 381–394, 1984.

[43] J. Dards, "Feral cat behaviour and ecology," *Bulletin of the Feline Advisory Bureau*, vol. 15, no. 3, 1976.

[44] A. Yamane, T. Doi, and Y. Ono, "Mating behaviors, courtship rank and mating success of male feral cat (felis catus)," *Journal of Ethology*, vol. 14, no. 1, pp. 35–44, 1996.

[45] S. L. Crowell-Davis, "Cat behaviour: social organization, communication and development," in *The Welfare of Cats*, I. Rochlitz, Ed., vol. 3 of *Animal Welfare*, pp. 1–22, Springer, Dordrecht, The Netherlands, 2007.

[46] W. Sung, *Effect of gender on initiation of proximity in free ranging domestic cats (Felis catus) [M.S. thesis]*, University of Georgia, 1998.

[47] R. C. Wolfe, *The social organization of the free ranging domestic cat (Felis catus) [PhD dissertation]*, University of Georgia, 2001.

[48] R. E. Adamec, "The interaction of hunger and preying in the domestic cat (*Felis catus*): an adaptive hierarchy?" *Behavioral Biology*, vol. 18, no. 2, pp. 263–272, 1976.

[49] H. Adler, "Some factors of observational learning in cats," *The Journal of Genetic Psychology*, vol. 86, no. 1, pp. 159–177, 1995.

[50] B. Santosa and M. K. Ningrum, "Cat swarm optimization for clustering," in *Proceedings of the International Conference on Soft Computing and Pattern Recognition (SoCPaR '09)*, pp. 54–59, December 2009.

[51] S.-C. Chu and P.-W. Tsai, "Computational intelligence based on the behavior of cats," *International Journal of Innovative Computing, Information and Control*, vol. 3, no. 1, pp. 163–173, 2007.

[52] P.-W. Tsai, J.-S. Pan, S.-M. Chen, and B.-Y. Liao, "Enhanced parallel cat swarm optimization based on the Taguchi method," *Expert Systems with Applications*, vol. 39, no. 7, pp. 6309–6319, 2012.

[53] J. E. Beasley, "A Lagrangian heuristic for set-covering problems," *Naval Research Logistics*, vol. 37, no. 1, pp. 151–164, 1990.

[54] S. Sundar and A. Singh, "A hybrid heuristic for the set covering problem," *Operational Research*, vol. 12, no. 3, pp. 345–365, 2012.