World Academy of Science, Engineering and Technology
International Journal of Mathematical and Computational Sciences
Vol:10, No:3, 2016

Solving the Set Covering Problem Using the Binary Cat Swarm Optimization Metaheuristic

Broderick Crawford, Ricardo Soto, Natalia Berrios, Eduardo Olguín

Abstract—In this paper, we present a binary cat swarm optimization for solving the Set covering problem. The set covering problem is a well-known NP-hard problem with many practical applications, including those involving scheduling, production planning and location problems. Binary cat swarm optimization is a recent swarm metaheuristic technique based on the behavior of discrete cats. Domestic cats show the ability to hunt and are curious about moving objects. The cats have two modes of behavior: seeking mode and tracing mode. We illustrate this approach with 65 instances of the problem from the OR-Library. Moreover, we solve this problem with 40 new binarization techniques and we select the technical with the best results obtained. Finally, we make a comparison between results obtained in previous studies and the new binarization technique, that is, with roulette wheel as transfer function and V_3 as discretization technique.

Keywords—Binary cat swarm optimization, set covering problem, metaheuristic, binarization methods.

I. Introduction

THE Set Covering Problem (SCP) [15], [14], [26] 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 kind of problems, the main ones are: location of services, files selection in a data bank, simplification of boolean expressions, 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) [25], [1], Ant Colony Optimization (ACO) [3], [30], Particle Swarm Optimization (PSO) [14], [16], Firefly Algorithm [17], [18], Shuffled Frog Leaping [19], and Cultural Algorithms [15] have been also successfully applied to solve the SCP. Our proposal of algorithm uses cat behavior to solve optimization problems, it is called Binary Cat Swarm Optimization (BCSO) [32].

BCSO refers to a serie of heuristic optimization methods and algorithms based on cat behavior in nature. Cats behave in two ways: seeking mode and tracing mode. BCSO is based in CSO [29] algorithm, proposed by Chu and Tsai in 2006 [12]. The difference is that in BCSO the vector position consists of ones and zeros, instead the real numbers of CSO.

Broderick Crawford is with the Pontificia Universidad Católica de Valparaíso, Universidad San Sebastián, and Universidad Central de Chile, Chile (e-mail: broderick.crawford@ucv.cl).

Ricardo Soto is with the Pontificia Universidad Católica de Valparaíso, Universidad Autónoma de Chile, Chile, and Universidad Cientifica del Sur, Lima, Perú (e-mail: ricardo.soto@ucv.cl).

Natalia Berrios is with the Pontificia Universidad Católica de Valparaíso, Chile (e-mail: natalia.berriosp.p@mail.pucv.cl).

Eduardo Olguín is with the Universidad San Sebastián, Chile.

This paper is an improvement of previous work [13], this seeks to get better results for each instace of OR-Library. We use a new method of setting parameters, which we choose different parameters for each instances set. Moreover, we tested 40 new techniques binarization [20], then we analyze the results to choose the method with which the best results. To use the binarization technique, we change the technique usually proposed for tracing mode, and we discover if this could help to improve results.

This paper is structured as follows: In Section II, a brief description of what SCP is given. Section III gives: what BCSO is, the explanation and algorithm of behaviors. In Section IV, an explanation of how was BCSO used for solving the SCP is presented. Section V gives an analysis and results table. Finally, conclusions are given in Section VI.

II. Set Covering Problem

The SCP [9], [6], [28] 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=$\{1,...,m\}$ 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

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

subject to

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

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

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 [2], location of emergency facilities [35], production planning in industry [34], vehicle routing [4], ship scheduling [22], network attack or defense [7], assembly line balancing [23], traffic assignment in satellite communication systems [31], simplifying boolean expressions [8], the calculation of bounds in integer programs [10], information retrieval [21], political districting [24], stock cutting, crew scheduling problems in airlines [27] and other important real life situations. Because it has wide applicability, we deposit our interest in solving the SCP.

III. Binary Cat Swarm Optimization

Among the known felines, there are about thirty different species, e.g., lion, tiger, leopard, cat, among others. Though many have different living environments, cats share similar behavior patterns.

For wild cats, the hunting skill ensures their food supply and survival of their species. Feral cats are groups with a mission to hunt their food. They are very wild feline colonies, ranging from 2-15 individuals.

Binary Cat Swarm Optimization [32] is an optimization algorithm that imitates the natural behavior of cats [11], [33]. Cats have curiosity by 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. Based on all these behaviors we formulate BCSO.

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 to search 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 ensure that each row i is covered by at least one column. Next is described the BCSO general pseudocode where MR is a percentage that determine the number of cats that undertake the seeking mode.

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 the cats into seeking mode according to MR, and the others set 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;
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: Re-pick 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;

A. Seeking Mode

This sub-model 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: Probability of Mutation Operation (PMO); Counts of Dimensions to Change (CDC), it indicates how many of the dimensions varied; Seeking Memory Pool (SMP), it 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.

The following pseudocode describe cat behavior seeking mode. In which FS_i is the fitness of i^{th} 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 $FS_b = FS_{max}$.

Algorithm 2 $BCSO()$

1: Create SMP copies of cat_k
2: Based on CDC update the position of each copy by randomly according to PMO
3: Evaluate the fitness of all copies
4: Calculate the selecting probability of each copy according to (4)
5: Apply roulette wheel to the candidate points and select one
6: Replace the current position with the selected candidate

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

B. Tracing Mode

Tracing mode is the sub-model 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 vector are defined as V^1_{kd} and V^0_{kd}. V^0_{kd} is the probability that the bits of the cat change to zero and V^1_{kd} 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 pseudocode:

Step1: Calculate d^1_{kd} and d^0_{kd} where $X_{best,d}$ is the d-th dimension of the best cat, r_1 has a random values in the interval of [0,1] and c_1 is a constant which is defined by the user

$$\begin{array}{l} \text{if } X_{best,d} = 1 \text{ then } d^1_{kd} = r_1 c_1 \text{ and } d^0_{kd} = -r_1 c_1 \\ \text{if } X_{best,d} = 0 \text{ then } d^1_{kd} = -r_1 c_1 \text{ and } d^0_{kd} = r_1 c_1 \end{array} \tag{5}$$

Step2: Update process of V^1_{kd} and V^0_{kd} are as follows, where w is the inertia weight and M is the column numbers

$$\begin{array}{ll} V^1_{kd} = wV^1_{kd} + d^1_{kd} & \\ V^0_{kd} = wV^0_{kd} + d^0_{kd} & \text{d = 1,...,M} \end{array} \tag{6}$$

Step3: Calculate the velocity of cat_k, $V^{'}_{kd}$, according to

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

Step4: Calculate the probability of mutation in each dimension, this is defined by parameter t_{kd}, t_{kd} takes a value in the inverval of [0,1]

$$t_{kd} = \frac{1}{1 + e^{-V^{'}_{kd}}} \tag{8}$$

Step5: Based on the value of t_{kd} the new value of each dimension of cat is update as follows where $rand$ is an aleatory variable $\in$ [0,1]

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

The maximun velocity vector of $V_{kd}^{'}$ should be bounded to a value V_{max}. If $V_{kd}^{'}$ value becomes higher than V_{max}, a new velocity should be assigned to $V_{kd}^{'}$ or the V_{max} value.

C. New Binarization Technique

In this work, experiments were performed with 40 different binarization techniques [20] using 65 OR -Library instances. A selection of smaller costs obtained is performed for each instance for get the results of all the experiments. Finally, to determine the binarization technique with better results, the technique smallest results obtained by all the instances are selected. Thus we obtained that roulette wheel in the case of the discretization method and V_3 for transfer function.

The transfer functions define a probability to change an element of solution from 1 to 0, or vice versa. In this case we select the transfer function v_3 (10).

$$\mathbf{T}(V_i^d) = \left| \frac{V_i^d}{\sqrt{1 + (V_i^d)^2}} \right| \qquad (10)$$

In addition to the Transfer functions, the discretization method was selected, Roulette wheel (11).

Roulette :

$$p_i = \frac{f_i}{\sum_{j=1}^{k} f_j} \qquad (11)$$

IV. Solving the Set Covering Problem

Next is described the Solving SCP pseudocode:

Algorithm 3 *Solving SCP()*

1: Initialize parameters in cats;
2: Initialization of cat positions, randomly initialize cat positions with values between 0 and 1;
3: Initialization of all parameter of BCSO;
4: Evaluation of the fitness of the population. In this case the fintess function is equal to the objective function of the SCP;
5: Change of the position of the cat. A cat produces a modification in the position based in one of the behaviors. i.e. seeking mode or tracing mode;
6: If solution is not feasible then repaired. Eeach row i must be covered by at least one columns, to choose the missing columns do: the cost of a column/(number of not covered row 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;

A. Parameter Setting

All the algorithms were configured before performing the experiments. To this end and starting from default values, a parameter of the algorithm is selected to be turned. Then, 10 independent runs are performed for each configuration of the parameter. Next, the configuration which provides the best performance on average is selected. Next, another parameter is selected so long as all of them are fixed. Table I shows the range of values considered and the configurations selected. These values were obtained experimentally.

This procedure was performed for each set of instances, Table I. In all experiments the BCSO was executed with 1000 iterations. Moreover, the results of the eight different transfer functions and five discretization techniques were considered to select the final parameter.

TABLE I
Parameter Values

Name	Parameter	Instance Set	Selected	Range
Number of Cats	C	4, 5 and 6	100	[10,20,...,1000]
		A and B	50	
		C and D	30	
		NRE and NRF	25	
		NRG and NRH	20	
Mixture Ratio	MR	4 and 5	0.7	[0.1,0.2,..,0.9]
		A and B	0.65	
		C and D	0.5	
		NRE and NRF	0.5	
		NRG and NRH	0.5	
Seeking Memory Pool	SMP	4 and 5	5	[5,10,...,100]
		A and B	5	
		C and D	10	
		NRE and NRF	15	
		NRG and NRH	20	
Probabily of Mutation Operation	PMO	4 and 5	0,97	[0.10,0.97,...,1.00]
		A and B	0,93	
		C and D	0,9	
		NRE and NRF	1	
		NRG and NRH	1	
Counts of Dimension to Change	CDC	4 and 5	0,001	[0.001,0.01,..,0,9]
		A and B	0,001	
		C and D	0,002	
		NRE and NRF	0,002	
		NRG and NRH	0,01	
Inertia Weight	w	4 and 5	1	[0.1,0.25,..,5]
		A and B	1	
		C and D	1	
		NRE and NRF	1	
		NRG and NRH	1	
Factor c_1	c_1	4 and 5	1	[0.1,0.25,..,5]
		A and B	1	
		C and D	1	
		NRE and NRF	1	
		NRG and NRH	1	

V. Results

The BCSO performance was evaluated experimentally using 65 SCP test instances from the OR-Library of Beasley [5]. The Table II and III shows the results of the 65 instances. The Z_{Opt} column reports the optimal value or the best known solution for each instance. The Z_{Best} and Z_{Avg} columns report the lowest 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 solution reached Z_b and Z_{opt} (which can be either the optimal or the best known objective value). RPD was evaluated using $Z_b = Z_{Best}$.

$$RPD = \left(\frac{Z_b - Z_{opt}}{Z_{opt}} \right) * 100 \quad (12)$$

About the solutions obtained we reach 6 optimum, all in Table II. The others results are very close to optimum values. If the best binarization technique was chosen for each set of instances is very pobable that better results are obtained.

Comparing the RPD_{Best} average of each instances set with the obtained results in previous work [13], where transfer function and discretization technique are not used, it can be seen that in the most of cases the results were improved. In Table IV the best difference is in instance set with 8,91 RPD, where Average of the New RPD is 0,82. Other instances set have a difference between about 3,0 and 7,0 RPD. The bad result is in the instance set NRF, with -0,12 of difference, worst results were obtained. The most important thing is that in most cases the results were improved. This shows that using the transfer function, discretization technique and use the new setting parameters achieves better results.

TABLE II
KNOWN OPTIMUM INSTANCES RESULTS

Instance	Z_{Opt}	Z_{Best}	Z_{Avg}	RPD_{Best}	RPD_{Avg}
SCP 41	429	432	441,6	0,7	2,9
SCP 42	512	516	532,2	0,8	3,9
SCP 43	516	520	554,9	0,8	7,5
SCP 44	494	497	514,5	0,6	4,1
SCP 45	512	515	527	0,6	2,9
SCP 46	560	560	568,7	0,0	1,6
SCP 47	430	434	437,5	0,9	1,7
SCP 48	492	494	518	0,4	5,3
SCP 49	641	658	678,4	2,7	5,7
SCP 410	514	518	526,6	0,8	2,4
SCP 51	253	261	263,3	3,2	3,9
SCP 52	302	303	315,2	0,3	4,4
SCP 53	226	229	235,3	1,3	4,1
SCP 54	242	242	245,7	0,0	1,5
SCP 55	211	216	220,5	2,4	4,4
SCP 56	213	213	224,1	0,0	5,2
SCP 57	293	298	307	1,7	4,7
SCP 58	288	299	307	3,8	6,4
SCP 59	279	280	281,7	0,4	1,0
SCP 510	265	271	275,6	2,3	3,9
SCP 61	138	143	146,9	3,6	6,2
SCP 62	146	146	149,1	0,0	2,1
SCP 63	145	149	152,5	2,8	5,0
SCP 64	131	133	135,2	1,5	3,2
SCP 65	161	164	168,9	1,9	4,8
SCP A1	253	271	275,4	7,1	8,3
SCP A2	252	260	265,3	3,2	5,1
SCP A3	232	235	243,5	1,3	4,9
SCP A4	234	244	246,4	4,3	5,1
SCP A5	236	237	239,2	0,4	1,4
SCP B1	69	71	74,9	2,9	8,3
SCP B2	76	79	85,5	3,9	12,0
SCP B3	80	80	83,2	0,0	4,0
SCP B4	79	81	84,4	2,5	6,7
SCP B5	72	73	73	1,4	1,4
SCP C1	227	231	234,8	1,8	3,4
SCP C2	219	224	230,3	2,3	5,0
SCP C3	243	258	265,6	6,2	8,8
SCP C4	219	233	239,5	6,4	8,8
SCP C5	215	224	228,8	4,2	6,2
SCP D1	60	60	64,6	0,0	7,7
SCP D2	66	69	69,9	4,5	5,7
SCP D3	72	76	78,8	5,6	8,9
SCP D4	62	63	65,7	1,6	5,9
SCP D5	61	63	65,2	3,3	6,7

TABLE III
UNKNOWN OPTIMUM INSTANCES RESULTS

Instance	Z_{Opt}	Z_{Best}	Z_{Avg}	RPD_{Best}	RPD_{Avg}
SCP NRE1	29	30	30	3,4	3,3
SCP NRE2	30	34	34,5	13,3	13,2
SCP NRE3	27	30	32,8	11,1	19,3
SCP NRE4	28	32	33	14,3	15,6
SCP NRE5	28	30	30	7,1	6,7
SCP NRF1	14	16	17	14,3	18,8
SCP NRF2	15	18	18	20,0	16,7
SCP NRF3	14	17	17	21,4	17,6
SCP NRF4	14	17	17,3	21,4	19,4
SCP NRF5	13	16	16	23,1	18,8
SCP NRG1	176	191	194,1	8,5	9,5
SCP NRG2	154	166	167,7	7,8	8,3
SCP NRG3	166	182	182,5	9,6	9,1
SCP NRG4	168	180	183,2	7,1	8,4
SCP NRG5	168	182	184,7	8,3	9,2
SCP NRH1	63	70	72,6	11,1	13,7
SCP NRH2	63	67	67	6,3	6,0
SCP NRH3	59	66	68,8	11,9	14,8
SCP NRH4	58	66	67,1	13,8	13,8
SCP NRH5	55	61	61	10,9	9,8

TABLE IV
COMPARISON OF NEW AND OLD RPD [13]

Instance Set	Avg. New RPD	Avg. Old RPD	Difference
4	0,82	9,73	8,91
5	1,53	9,11	7,58
6	1,95	6,82	4,87
A	3,26	8,30	5,04
B	2,15	9,62	7,47
C	4,16	11,10	6,94
D	3,00	6,78	3,78
NRE	9,86	9,90	0,04
NRF	20,04	19,92	-0,12
NRG	8,29	9,98	1,69
NRH	10,81	11,48	0,67

VI. 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 cases observed according to old RPD works [13]. 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. Moreover, it could also better solutions using different parameter setting for each set of instances.

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 be 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.

ACKNOWLEDGMENTS

The author Broderick Crawford is supported by grant CONICYT/FONDECYT/REGULAR/1140897 and Ricardo Soto is supported by grant CONICYT/FONDECYT/INICIACION/11130459

REFERENCES

[1] U. Aickelin. An indirect genetic algorithm for set covering problems. *Journal of the Operational Research Society*, pages 1118–1126, 2002.

[2] A. I. Ali and H. Thiagarajan. A network relaxation based enumeration algorithm for set partitioning. *European Journal of Operational Research*, 38(1):76–85, 1989.

[3] F. Amini and P. Ghaderi. Hybridization of harmony search and ant colony optimization for optimal locating of structural dampers. *Applied Soft Computing*, pages 2272–2280, 2013.

[4] M. L. Balinski and R. E. Quandt. On an integer program for a delivery problem. *Operations Research*, 12(2):300–304, 1964.

[5] J. Beasley. A lagrangian heuristic for set covering problems. *Naval Research Logistics*, 37:151–164, 1990.

[6] J. Beasley and K. Jornsten. Enhancing an algorithm for set covering problems. *European Journal of Operational Research*, 58(2):293–300, April 1992.

[7] M. Bellmore and H. D. Ratliff. Optimal defense of multi-commodity networks. *Management Science*, 18(4-part-i):B174–B185, 1971.

[8] M. A. Breuer. Simplification of the covering problem with application to boolean expressions. *Journal of the Association for Computing Machinery*, 17(1):166–181, Jan. 1970.

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

[10] N. Christofides. Zero-one programming using non-binary tree-search. *Computer Journal*, 14(4):418–421, 1971.

[11] S. Chu and P. Tsai. Computational intelligence based on the behavior of cats. *International Journal of Innovative Computing, Information and Control*, pages 163 – 173, 2007.

[12] S. Chu, P. Tsai, and J. Pan. Cat swarm optimization. In *Trends in Artificial Intelligence*, pages 854–858. Springer-Verlag, Berlin, Heidelberg, 2006.

[13] B. Crawford, R. Soto, N. Berrios, F. Johnson, F. Paredes, C. Castro, and E. Norero. A binary cat swarm optimization algorithm for the non-unicost set covering problem. *Mathematical Problems in Engineering*, 2015(Article ID 578541):1–8, 2015.

[14] 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*, 2014(Article ID 189164):1–8, 2014.

[15] B. Crawford, R. Soto, and E. Monfroy. Cultural algorithms for the set covering problem. In Y. Tan, Y. Shi, and H. Mo, editors, *Advances in Swarm Intelligence, 4th International Conference*, volume 7929 of *Lecture Notes in Computer Science*, pages 27–34. Springer, Harbin, China, 2013.

[16] B. Crawford, R. Soto, E. Monfroy, W. Palma, C. Castro, and F. Paredes. Parameter tuning of a choice-a function based hyperheuristic using particle swarm optimization. *Expert Systems with Applications*, pages 1690–1695, 2013.

[17] B. Crawford, R. Soto, M. Olivares-Suárez, W. Palma, F. Paredes, E. Olguin, and E. Norero. A binary coded firefly algorithm that solves the set covering problem. volume 17, pages 252–264, 2014.

[18] B. Crawford, R. Soto, M. Olivares-Suárez, and F. Paredes. A binary firefly algorithm for the set covering problem. In *3rd Computer Science On-line Conference 2014, Modern Trends and Techniques in Computer Science*, volume 285, pages 65–73. Springer, 2014.

[19] B. Crawford, R. Soto, C. Peña, W. Palma, F. Johnson, and F. Paredes. Solving the set covering problem with a shuffled frog leaping algorithm. In N. T. Nguyen, B. Trawinski, and R. Kosala, editors, *Intelligent Information and Database Systems - 7th Asian Conference*, volume 9012 of *LNCS*, pages 41–50, Bali, Indonesia, 2015. Springer.

[20] B. Crawford, R. Soto, M. Riquelme-Leiva, C. Pena, C. Torres-Rojas, F. Johnson, and F. Paredes. Set covering problem solved by new binary firefly algorithm. In *10th Iberian Conference on Information Systems and Technologies*, pages 1–4. 2015.

[21] R. H. Day. Letter to the editoron optimal extracting from a multiple file data storage system: An application of integer programming. *Operations Research*, 13(3):482–494, 1965.

[22] M. L. Fisher and M. B. Rosenwein. An interactive optimization system for bulk-cargo ship scheduling. *Naval Research Logistics*, 36(1):27–42, 1989.

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

[24] R. S. Garfinkel and G. L. Nemhauser. Optimal political districting by implicit enumeration techniques. *Management Science*, 16(8):B495–B508, 1970.

[25] D. Goldberg. Real-coded genetic algorithms, virtual alphabets, and blocking. *Complex Systems*, pages 139–167, 1990.

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

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

[28] L. Lessing, I. Dumitrescu, and T. Stutzle. A comparison between aco algorithms for the set covering problem. In *Ant Colony Optimization and Swarm Intelligence*, pages 1–12. 2004.

[29] G. Panda, P. Pradhan, and B. Majhi. Iir system identification using cat swarm optimization. *Expert Systems with Applications*, 38:12671–12683, 2011.

[30] Z. Ren, Z. Feng, L. Ke, and Z. Zhang. New ideas for applying ant colony optimization to the set covering problem. *Computers & Industrial Engineering*, pages 774 – 784, 2010.

[31] 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*, 41(2):232–239, 1989.

[32] Y. Sharafi, M. Khanesar, and M. Teshnehlab. Discrete binary cat swarm optimization algorithm. *Computer, Control and Communication*, pages 1–6, 2013.

[33] P. Tsai, J. Pan, S. Chen, and B. Liao. Enhanced parallel cat swarm optimization based on the taguchi method. *Expert Systems with Applications*, 39:6309–6319, 2012.

[34] F. J. Vasko, F. E. Wolf, and K. L. Stott. Optimal selection of ingot sizes via set covering. *Operations Research*, 35(3):346–353, June 1987.

[35] W. Walker. Using the set-covering problem to assign fire companies to fire houses. *Operations Research*, 22:275–277, 1974.