International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:3, 2008

Evolutionary Search Techniques to Solve Set Covering Problems

Darwin Gouwanda, and S. G. Ponnambalam

***Abstract*—**Set covering problem is a classical problem in computer science and complexity theory. It has many applications, such as airline crew scheduling problem, facilities location problem, vehicle routing, assignment problem, etc. In this paper, three different techniques are applied to solve set covering problem. Firstly, a mathematical model of set covering problem is introduced and solved by using optimization solver, LINGO. Secondly, the Genetic Algorithm Toolbox available in MATLAB is used to solve set covering problem. And lastly, an ant colony optimization method is programmed in MATLAB programming language. Results obtained from these methods are presented in tables. In order to assess the performance of the techniques used in this project, the benchmark problems available in open literature are used.

***Keywords*—**Set covering problem, genetic algorithm, ant colony optimization, LINGO.

I. Introduction

SET covering problem is a classical problem in computer science and complexity theory. Set covering problem is one of most important discrete optimization problem because it serves as a model for real world problems. Real world problems that can be modeled as set covering problem include facility location problem, airline crew scheduling, nurse scheduling problem, resource allocation, assembly line balancing, vehicle routing, etc. Set covering problem is a problem of covering the rows of an m-row/n-column zero-one matrix with a subset of columns at minimal cost [1]. Set covering problem can be formulated as follows:

$$Min \sum_{j=1}^{n} c_j x_j \tag{1}$$

$$subject\ to \sum_{j=1}^{n} a_{ij} x_j \geq 1, \qquad i = 1,...,m \tag{2}$$

$$x_j \in (0,1) \qquad j = 1,...,n \tag{3}$$

Equation (1) is the objective function of set covering problem, where c_j is refer to weight or cost of covering j-column and x_j is decision variable. Equation (2) is a constraint to ensure that each row is covered by at least one column where a_{ij} is constraint coefficient matrix of size m x n whose elements comprise of either '1' or '0'. Lastly, equation (3) is the integrality constraint in which the value is represented as in (4).

$$x_j = \begin{cases} 1 & \text{if } j \in S, \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

Even though it may seem to be a simple problem by judging from the objective functions and constraints of the problem, set covering problem is a combinational optimization problem. It has been proven to be NP-Complete decision problem [2].

A number of heuristic algorithms for set covering problem have been reported in the literature. Beasley, as one of main researcher in set covering problem had implemented several algorithms in order to solve set covering problem. Beasley presented an algorithm that combines problem reduction tests with dual ascent, sub-gradient optimization and linear programming. This algorithm had performed well in solving set covering problem [2]. It was able to find feasible optimal solutions for all set covering problem sets. In a different literature, Beasley presented a paper which used Lagrangian relaxation and sub-gradient optimization approach to solve set covering problem [3]. But this method did not perform well compared to his previous method. It was unable to find optimal solutions for several set covering problems such as SCP-4.4, SCP-4.6, SCP-5.1, SCP-5.2, SCP-5.7, SCP-6.1, SCP-6.5, etc. Haddadi presented a simple Lagrangian heuristic to solve set covering problem [4]. The method is based on Lagrangian duality, greedy heuristic for set covering problem, sub-gradient optimization and redundant covers. This method had turn out to be efficient for low density set covering problem with a large number of variables with average deviation of 0.35%.

Beasley and Chu presented genetic algorithm for set covering problem [5]. They presented a new crossover-fusion operator, a variable mutation rate and a heuristic feasibility operator to improve the performance of genetic algorithm. This method performs well, for most of problems. Aickelin proposed an indirect genetic algorithm [5]. The indirect genetic algorithm comprises of three phases. In the first phase, genetic algorithm finds good permutation of the rows to be covered. In second phase, a decoder build a solution from the permutations using the parameter provided. And lastly, in the third phase, a hill-climber optimization method is used. Indirect genetic algorithm is able to solve the set covering problem in shorter computational time. Monfroglio proposed a linear programming relaxation model and improvement techniques based on simulated neural network [6]. This

Darwin Gouwanda is with School of Engineering, Monash University Sunway Campus, 46150, Bandar Sunway, Malaysia (e-mail: darwin_gouwanda@yahoo.com).

S. G. Ponnambalam is with School of Engineering, Monash University Sunway Campus, 46150, Bandar Sunway, Malaysia (phone: +60-3-55146203; e-mail: sgponnambalam@eng.monash.edu.my).

method is able to find solutions within 0.2% of optimal solution and increase the overall computational time. Vasko and Wolf adapted heuristic concentration approach to solve the weighted (non-unicost) set covering problem [7]. Their method is able to solve set covering problem and find solution with deviation of maximum of 3.27% from optimum solution.

In this paper, an attempt has been made to propose Genetic Algorithm (GA) and Ant Colony Optimization (ACO) technique and the performance of them are evaluated with an optimization solver LINGO. Matlab GA Toolbox is used in this paper.

II. Benchmark Problem Sets

The benchmark problem sets are publicly available in OR-Library [8]. The problem sets had been widely used by many researchers to verify their methods effectiveness in solving set covering problem. There are a total of eight problem sets considered in this paper for evaluation. The details of these problem sets are shown in Table I.

TABLE I
DETAILS OF BENCHMARK PROBLEM SETS

Problem Set	Problem type	Size of a_{ij} Matrix	Density	No. of datasets
SCP-4	Non-unicost	200 x 1000	2%	10
SCP-5	Non-unicost	200 x 2000	2%	10
SCP-6	Non-unicost	200 x 1000	5%	5
SCP-A	Non-unicost	300 x 3000	2%	5
SCP-B	Non-unicost	300 x 3000	5%	5
SCP-C	Non-unicost	400 x 4000	2%	5
SCP-D	Non-unicost	400 x 4000	5%	5

SCP-4, SCP-5 and SCP-6 are test problems produced by using scheme of Balas and Ho [9] while problem set SCP-A, SCP-B, SCP-C and SCP-D is randomly generated test problems. SCP-4 and SCP-5 has 10 datasets and the rest of problem sets i.e. SCP-6, SCP-A, SCP-B, SCP-C and SCP-D has 5 datasets. SCP-4, SCP-5, SCP-6, SCP-A, SCP-B, SCP-C and SCP-D are a non-unicost set covering problems. Non-unicost set covering problem, which also called weighted set covering problem, has various costs for each column. Density is the total number of integer '1' in the a_{ij} matrix. For example, problem set SCP-4 has a total of 4000 '1's in the a_{ij} matrix

Conversion of Problem Set

Datasets that are downloaded from OR-Library has information on size of a_{ij} matrix, the cost of each column, number of column in row and list of columns that cover row. The format of dataset is shown in Fig. 1.

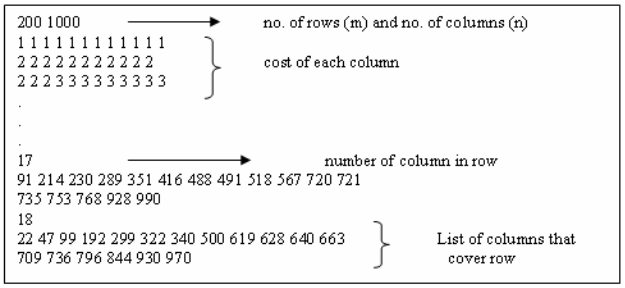

Fig. 1 Format of downloaded datasets

Adjustments/ conversions made are:

1) Separating the cost of each column from dataset manually and saving it under different file name (*.txt).
2) Arranging the columns that cover each row in one line based on total number of columns in that row and saving it under different file name as well (*.txt). (Please refer to Fig. 2).
3) Converting the previous file into excel format (*.txt is converted to *.xls)
4) Converting the columns that cover each row into zero one matrix (a_{ij}). (Please refer to Fig. 3).

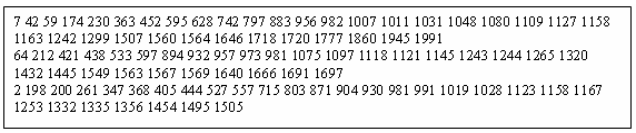

Fig. 2 Outcome of dataset arrangement

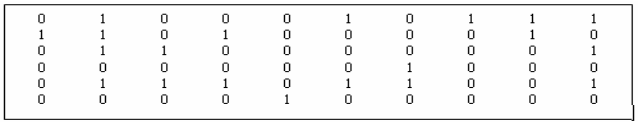

Fig. 3 Outcome of dataset conversion

III. Models Used to Solve Set Covering Problems

A. LINGO

One of LINGO powerful features is its mathematical modeling language [10]. Its modeling language enables users to express their problems in a natural manner that is very similar to standard mathematical notation. Another powerful feature is data section. Data section enables users to isolate model's data from formulation. This features offers flexibility to users to decrease or increase the data's size.

For set covering problem, LINGO categorizes it under PLIP (Pure Linear Integer Program) class. It solves set covering problem by using branch and bound manager

LINGO optimization model has following attributes:

1) Sets, which comprise of objects or variables in programming model.
2) Objective function of problem
3) Constraints of problem
4) Input data

Therefore by referring to properties of set covering problem which has one objective function, two sets, which are rows and columns, two constraints and a decision variable, x_j, a

model of set covering problem is expressed as follow, in LINGO.

```
Line1:      model:
Line2:
Line3:      sets:
Line4:      m /1..200/ :;
Line5:      n /1..1000/ :cost,x;
Line6:      links (m,n):matrix;
Line7:      endsets
Line8:
Line9:      min = @sum(n(j):cost(j)*x(j));
Line10:
Line11:     @for(m(i):@sum(n(j):matrix(i,j)*x(j))>= 1);
Line12:     @for(n(j):@bin (x(j)));
Line13:
Line14:     data:
Line15:     cost = xxxxxxx;
Line16:     matrix = xxxxx;
Line17:     enddata
Line18:
Line19:     end
```

Sets — Objective function — Constraints — Data

Fig. 4 A model of set covering problem in Lingo

Fig. 4 shows that a model has to be specified first by stating "model: "on first line and end it with "end" on the last line of the model. On next line, Line3, sets, which comprises of variables and their respective sizes are specified. Sets are initialized within "sets" and "endsets". The objective function of set covering problem is specified on Line9. The objective function can be specified by stating "min" for minimization problem or "max" for maximization problem. After stating the objective function, constraints of the set covering problem are specified on next lines, Line11 and Line12. Line11 states the constraint to ensure that each row is covered at by at least one column. Line12 states the integrality constraint. Lastly, data is inserted within "data" and "enddata". These data are the cost of columns (c_j) and a_{ij} matrix.

B. MATLAB's Genetic Algorithm Tool

Due to its superiority in solving problems which have complex fitness landscape and large search space, genetic algorithm was selected to solve set covering problem. And MATLAB Genetic Algorithm Tool (GA Tool) is selected to perform genetic algorithm [11]. In this section, the features of MATLAB GA Tool and its implementation to solve the set covering problem are discussed.

1) Representation scheme

Representation scheme is very important step in designing genetic algorithm for a particular problem. There are two possible representation schemes for set covering problem: column-based representation scheme and row-based representation scheme [12].

Due to nature of the benchmark problem sets, where the total number of rows of problem sets are smaller than total number of columns, row-based representation is chosen on the expectation that the computational time could be reduced. For example, problem set SCP4 has size of 200 x 1000. If column-based representation scheme is used, the length of solution will be 1000 bits. If row-based representation scheme is used, the length of solution will be 200.

2) Probabilistic Heuristic Initial Population

In this approach, genes for each chromosome are selected based on probability of coverage and cost. For each row, the total number of coverage and cost of each column are identified. Total number of coverage is total number of rows that is able to be covered by a specific column

3) Crossover

Scattered crossover is used in this paper [11]. Scattered crossover creates a random binary vector and selects the genes where the vector is a 1 from the first parent and the genes where the vector is a 0 from the second parent, and combines the genes to form the child. An example is presented in Fig. 5.

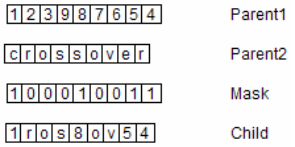

Fig. 5 Scattered crossover

The performance of scattered crossover is compared with single point and two point crossover for all the dataset considered in this study. It is observed that scattered crossover performs better in solving set covering problem. It enables the genetic algorithm to converge faster and produce better solution.

4) Mutation

Random mutation is used in this paper. Initially a gene is selected randomly. This gene will correspond to the row number of a particular dataset. In this row, a column that covers the row is selected randomly. An illustration of random mutation is presented in Fig. 6.

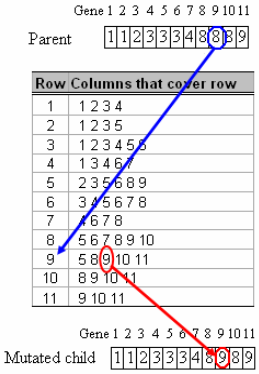

Fig. 6 Random mutation

From Fig.6, it can be seen that

1) Ninth gene of parent chromosome (which is equal to 8) is selected for mutation. Ninth gene is corresponding to ninth row of data.
2) In ninth row, a column that covers the row is randomly selected to replace the parent's gene. Column number 9 is selected.
3) Mutation of ninth gene in the parent, which has initial value of 8, is changed to 9.

Apart from normal mutation operator, variable mutation rate had been introduced It is able to introduce more diverse individuals in population which may lead to better solution when GA converges and when the crossover operator becomes less productive.

Initially constant mutation rate of 10 genes/chromosome is used. As genetic algorithm starts to converge, normally it occurs after 60th generation, an increase in mutation rate is introduced. The mutation rate is gradually increased by factor of 0.1. And from 150th generation onward, constant mutation rate of 20 genes/ chromosome is used because from this point onward, increase in mutation rate may degenerate the solutions due to high diversity in population

5) Stopping criteria

Stopping criteria chosen for solving set covering problem is 200 generation. It is observed that 200 generations is sufficient to produce optimal solutions for set covering problem.

The summary of optimal GA parameter set identified after extensive analysis is presented in Table II.

TABLE II
SUMMARY OF GA PARAMETERS USED TO SOLVE SET COVERING PROBLEM

Parameters	Description	
Population size	SCP-4 :	2000
	SCP-5 :	2000
	SCP-6 :	6000
	SCP-A :	6000
	SCP-B :	8000
	SCP-C :	8000
	SCP-D :	10000
Chromosome length	SCP-4 :	200
	SCP-5 :	200
	SCP-6 :	200
	SCP-A :	300
	SCP-B :	300
	SCP-C :	400
	SCP-D :	400
Fitness scaling	Top fitness scaling	
Selection	Tournament selection	
Crossover fraction	0.8	
Crossover	Scattered crossover	
Mutation	Random mutation	
Elite	5% of Population size	
Migration	-	
Algorithm settings	-	
Hybrid function	-	
Stopping criteria	200 generations	

In order to provide a reliable computational results produced by MATLAB GA Tool, genetic algorithm was run ten times for each problem sets.

C. Ant Colony Optimization

Ant colony optimization is a probabilistic construction heuristic that generates solutions iteratively, taking into account accumulated past search experiences: pheromone trails and heuristic information [13]. The search for solution composes of several iterations. Initially pheromone is initialized for first iteration. In next step, column is added into the solution according to its probability. The addition of column to the solution will continue until all rows in problems are covered.

The probability of each column to be chosen as ant path and added to solution is formulated as follows [14]:

$$P_{k_j} = \frac{\text{Phero}_j^{\alpha} \times H_j^{\beta}}{\sum_{i=1..n} \text{Phero}_j^{\alpha} \times H_i^{\beta}} \quad \text{if } j \notin S_k \tag{5}$$

S_k is set of columns belonging to the partial solution of k^{th} ant. Phero_j is pheromone intensity or pheromone trail of column j. H_j is a greedy heuristic ratio of cover value divided by cost [14]

$$H_{(j)} = \frac{\text{cov\_val}_j}{\text{cost}_j} \tag{6}$$

$$\text{cov\_ val}_j = \sum_{i \in \text{cov}(j,s)} \min\_\text{cost}(i) \tag{7}$$

cov(j,s) is the set of lines which are covered by the column j and not covered by the solution S, and min_cost(i) is the minimum cost of the columns that cover the line i. α and β are parameters which determine the relative influence of the pheromone trail and heuristic information. The parameters α and β can be varied accordingly in order to get an optimal result.

In next step, after all ants in colony construct feasible solutions, pheromone intensity is updated as follow [14]:

$$\text{Phero}_{j\,(t+1)} = (1-\rho) \times \text{Phero}_{j\,(t+1)} + \sum_{i=1..m} \Delta\text{Phero}_{k_j\,(t)} \tag{8}$$

where

$$\Delta\text{Phero}_j = \begin{cases} 1/\text{fitness value} & , \text{if } j \in S_k \\ 0 & , \text{otherwise} \end{cases}$$

ρ is pheromone evaporation coefficient. It determines the decreasing rate of pheromone (0< ρ< 1). In this project, the evaporation coefficient is set to 0.9.

Total number of iteration and number of ants are varied according to the problem size. Details of total number of iterations and ants used are shown in Table III.

TABLE III
NUMBER OF ITERATION AND NUMBER OF ANTS USED FOR THE BENCHMARK PROBLEMS

Problem Set	No. of ants	No. of iteration
SCP-4	2	5
SCP-5	4	5
SCP-6	4	5
SCP-A	6	7
SCP-B	6	7
SCP-C	7	8
SCP-D	7	8

Initial pheromone (τ_0) is used to start first iteration. Initial pheromone is calculated by following these steps:

1) Randomly construct a feasible solution
2) Calculate the fitness value of the solution
3) Determine the maximum cost of the columns
4) Calculate the ration of *Lnn* based on (9)

$$Lnn = \frac{\text{fitness value of solution + max cost of columns}}{2} \quad (9)$$

5) Calculate the initial pheromone τ_0 based on (10)

$$\tau_0 = \frac{1}{m \times Lnn} \quad , where \text{ m = total number of rows} \quad (10)$$

Ants will find columns that are able to cover all the rows in each iteration. Before ants select a column as partial solution, H_j will be calculated first. By having value of H_j, the probability of each column to be selected as partial solution is calculated by using (5). In the next step, a random number is generated. Based on the random number and probability, a column is selected as partial solution (roulette wheel principle). Calculation of H_j and column selections will continue until all the rows in set covering problem are covered. Once rows in set covering problem are covered, fitness value of ant path will be calculated by using (1). When all ants have found the solution, the pheromone intensity is updated according to (7) and next iteration starts.

D. Performance of the Models

The results obtained using LINGO, genetic algorithm and ant colony optimization are presented in Table IV. The objective function value (OFV) and the computational runtime are reported in the Table IV. It is observed that LINGO being an optimization solver performs better than the proposed GA and ACO. Between GA and ACO, ACO performs better except for the problem set SCP-4 with reference to OFV and GA performs better with reference to runtime.

IV. Conclusion

In this paper, three techniques are used to solve set covering problem: LINGO, genetic algorithm and ant colony optimization. In order to assess the performance of these techniques in solving set covering problem, a set of benchmark set covering problems are used. The problem size ranges from 200 x 1000 to 400 x 4000 with density of 2% and 5%.

From computational results, it is observed that LINGO as an optimization tool had performed well in solving set covering problem. It is able to find the optimal solution for benchmark problem sets that were tested in shorter run time compared with other techniques considered in this paper.

As second technique, genetic algorithm is a method that is based on natural selection which is a process that drives biological evolution. In this paper, genetic algorithm is performed by using MATLAB Genetic Algorithm Tool. MATLAB Genetic Algorithm Tool is a flexible tool. It provides various features that may assist users in solving various optimization problems.

Despite of its flexibility, genetic algorithm does not perform well in solving problems that have larger search space. GA Tool failed to find optimal or near to optimal solution for set covering problems that have larger problem size i.e. SCP-B, SCP-C and SCP-D.

Third technique considered is ant colony optimization. This is a method that imitates the behavior of ant colony in bringing their food from source to their nest. Ant colony optimization is able to find near to optimal solution based on the past experiences: pheromone trails and heuristic information. In this study, ant colony optimization is successfully coded and executed in MATLAB programming platform.

To conclude, LINGO being an optimization tool is able to obtain the optimal solution for all benchmark problem considered in this paper. Ant colony optimization has performed better than genetic algorithm. This method is able to find near to optimal solution for all benchmark set covering problems used in this project regardless of problem size. Despite of its superiority in solving set covering problem, ant colony optimization needs longer computational time in order to solve set covering problem.

References

[1] J.E. Beasley and P.C. Chu, "A genetic algorithm for the set covering problem", *European Journal of Operational Research*, vol. 94, 1996, pp. 392-404
[2] J.E. Beasley, "An algorithm for set covering problem", *European Journal of Operational Research,* vol. 31, 1987, pp 85-93
[3] J.E. Beasley, "A Lagrangian heuristic for set covering problems", *Naval Research Logistics*, Vol. 37, 1990, pp. 151-164
[4] S. Haddadi, "Simple Lagrangian heuristic for the set covering problem", *European Journal of Operational Research*, vol. 97, 1997, pp 200-204
[5] U. Aickelin, "An indirect genetic algorithm for set covering problem" *Journal of Operational Research Society*, vol. 53, 2002, pp. 1118-1126
[6] A. Monfroglio, "Hybrid heuristic algorithm for set covering", *Computers Operational Research*, vol. 25, 1998, pp. 441-445
[7] J.F. Vasko and F.E. Wolf, " A heuristic concentration approach for weighted set covering problems", *Locator: ePublication of Location Analysis,* vol. 2, 2001, no. 1, pp. 1-14
[8] *OR-Library:* http://people.brunel.ac.uk/~mastjjb/jeb/orlib/scpinfo.html
[9] E. Balas, and A. Ho, "Set covering algorithms: using cutting planes, heuristics and subgradient optimization: a computational study", *Mathematical Programming Study,* vol. 12, 1980, pp. 300-304.
[10] Lindo System, *Lingo 8.0 user's manual*, 2003
[11] The Mathworks, *Matlab's Genetic Algorithm and Direct Search Toolbox User's Guide,* The Matworks, 2005, Massachusetts
[12] M. Gen, R. Cheng, *Genetic Algorithms and Engineering Optimization*, John Wiley & Sons, 2000, Canada
[13] L. Lessing, I. Dumitrescu, and T. Stutzle, "A comparison between ACO algorithms for the set covering problem", *ANTS 2004, LNCS 3172,* 2004, pp. 1-12
[14] M. Rahoual, R. Hadji, and V. Bachelet, "Parallel ant system for the set covering problem" *ANTS 2002, LNCS 2463*, 2002 pp. 262 – 267.

Darwin Gouwanda received his degree in mechatronics engineering from Monash University Sunway Campus, Malaysia in 2006. The author is currently pursuing his Masters of Engineering Science (Research). His research areas include artificial intelligence, robotics, and biomechanics.

S. G. Ponnambalam is an Associate Professor in the School of Engineering at Monash University, Sunway Campus, Malaysia. He has a Ph.D in Computer Integrated Manufacturing form Bharathiar University, Coimbatore, India. He is teaching undergraduate and graduate courses in the areas of Mechatronics, manufacturing and manufacturing management and supervising number of doctoral students in the area of manufacturing automation and application of evolutionary optimization. He is the chair of IEEE-RAS, Malaysia Section.

He has published over 170 articles that include 42 referred journal publications and 128 refereed conference publications. His articles are published in different peer-reviewed journals that include *International Journal of Operations & Production Management, Production Planning and Control, International Journal of Advanced Manufacturing Technology, International Journal of Industrial Engineering, Robotics and Computer-Integrated Manufacturing, and Computers & Industrial engineering.*

TABLE IV
COMPUTATIONAL RESULTS OF THE MODELS

		LINGO		GA		ACO	
Problem	Optimal value	OFV	Run time	OFV	Run time	OFV	Run time
SCP-4.1	429	429	< 1	446	76.07	486	11024.8
SCP-4.2	512	512	< 1	549	75.78	556	405.12
SCP-4.3	516	516	< 1	554	75.84	591	80.5
SCP-4.4	494	494	< 1	551	77.64	577	69.53
SCP-4.5	512	512	< 1	531	74.65	545	52.79
SCP-4.6	560	560	< 1	574	76.1	620	66633.8
SCP-4.7	430	430	< 1	458	75.42	483	1519.63
SCP-4.8	492	492	< 1	522	75	540	1358.73
SCP-4.9	641	641	< 1	700	77.34	763	1568.72
SCP-4.10	514	514	< 1	548	76.04	596	128.7
SCP-5.1	253	253	≈ 2	295	89.89	284	1707.16
SCP-5.2	302	302	≈ 2	353	87.85	335	3259.46
SCP-5.3	226	226	≈ 2	264	90.59	245	7389.53
SCP-5.4	242	242	≈ 2	281	87.24	265	7389.53
SCP-5.5	211	211	≈ 2	245	87.91	230	4749.6
SCP-5.6	213	213	≈ 2	243	86.28	224	343.38
SCP-5.7	293	293	≈ 2	328	87.23	314	3165.61
SCP-5.8	288	288	≈ 2	326	88.77	315	4037.87
SCP-5.9	279	279	≈ 2	325	90.81	285	818.46
SCP-5.10	265	265	≈ 2	292	88.18	280	250.99
SCP-6.1	138	138	≈ 2	150	223.6	154	247.18
SCP-6.2	146	146	≈ 2	160	220.53	160	269.93
SCP-6.3	145	145	≈ 2	167	221.69	153	309.04
SCP-6.4	131	131	≈ 2	145	219.8	138	217.95
SCP-6.5	161	161	≈ 2	183	224.95	181	172.73
SCP-A.1	253	253	≈ 15	275	384.88	261	1317.89
SCP-A.2	252	252	≈ 15	289	370.67	266	1593.74
SCP-A.3	232	232	≈ 13	267	368.29	261	1689.36
SCP-A.4	234	234	≈ 8	257	377.4	257	1815.37
SCP-A.5	236	236	≈ 7	262	364.17	247	1465.29
SCP-B.1	69	69	≈ 26	107	538.1	74	1483.07
SCP-B.2	76	76	≈ 271	118	542.23	84	1626.83
SCP-B.3	80	80	≈ 56	119	552.46	87	2200.79
SCP-B.4	79	79	≈ 416	123	531.94	89	1956.24
SCP-B.5	72	72	≈ 38	106	528.72	79	1462.46
SCP-C.1	227	227	≈ 19	282	647.87	239	3994.02
SCP-C.2	219	219	≈ 59	281	692.86	241	4501.89
SCP-C.3	243	243	≈ 261	302	723.67	270	8443
SCP-C.4	219	219	≈ 46	265	664.86	240	3638.29
SCP-C.5	215	215	≈ 26	271	655.4	233	7558.4
SCP-D.1	60	60	≈ 415	139	952.63	69	4256.12
SCP-D.2	66	66	≈ 1071	157	933.78	69	3454.91
SCP-D.3	72	72	≈ 513	149	922.02	78	3837.55
SCP-D.4	62	62	≈ 853	151	948.31	68	4551.76
SCP-D.5	61	61	≈ 698	151	970.27	65	3481.71