Set Covering Demo Help

Author:

Shayan Kavakeb

PhD Student

School of Computer Science and Electronic Engineering

University of Essex

Date:

27th April 2011

Contents

1. Introduction

In this report, I explain two algorithms for set covering problem and also the demo which has been developed for solving set covering problem. The first algorithm is a probabilistic greedy search which is presented by M.Haouari & JS.Chaouachi [2], and I implemented it as PROGRES algorithm. The other algorithm for set covering problem is Guided Local Search. Guided Local Search (GLS) has not been applied for set covering problem before this research, results show significantly improvement by applying GLS for set covering problem.
This report structured as follows. First, I explain features of the demo and I will show how the demo works. In section three, I briefly explain the PROGRES algorithm which originated from M.Haouari & JS.Chaouachi [2] and its window in the demo. In section four, neighbourhood function for Guided Local Search and related window in the demo has been introduced. Finally, I provide some conclusions.

2. Demo Overview

In this section, I describe how you can run the demo. This demo consists of two algorithms and some parameters related to the algorithms. Running the demo is totally straightforward. First, you need to brows the instance, after that select your algorithm and then, run the algorithm. Finally, the results will be shown; in addition, you can export the results into a text file. The steps for running the demo are summarized below. In order to clarifying the steps, I inserted some images from the demo.

Steps of Running the Demo:

1. Brows the instance:

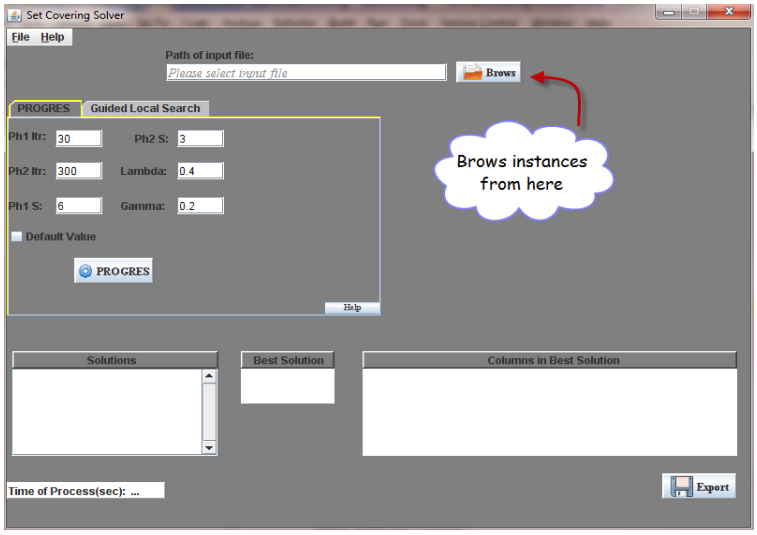

Figure 1 Brows Instance

2. Select the algorithm which you want to run:

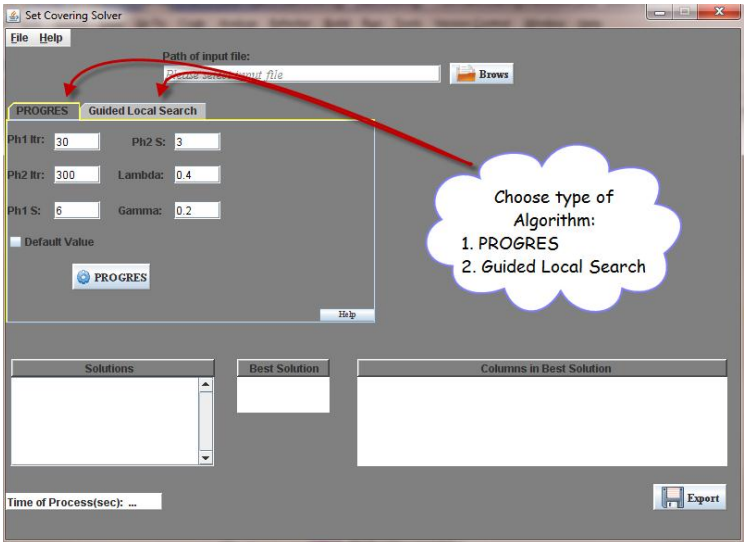

Figure 2 Selecting the algorithm

3. Run the algorithm:

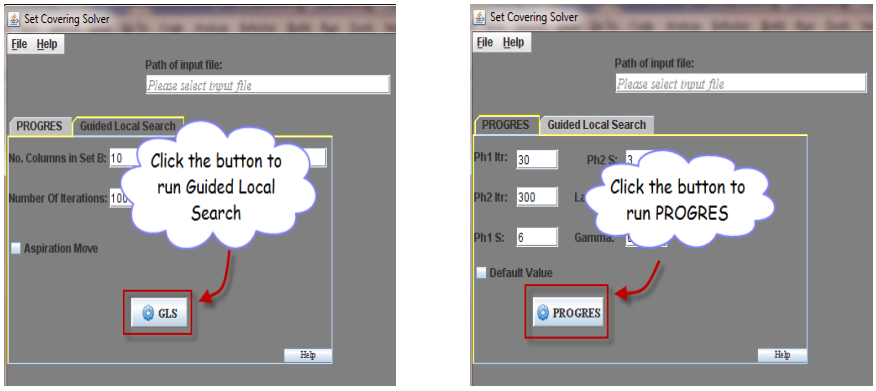

Figure 3 Algorithms button

4. See the results and you can export the results into a text file.

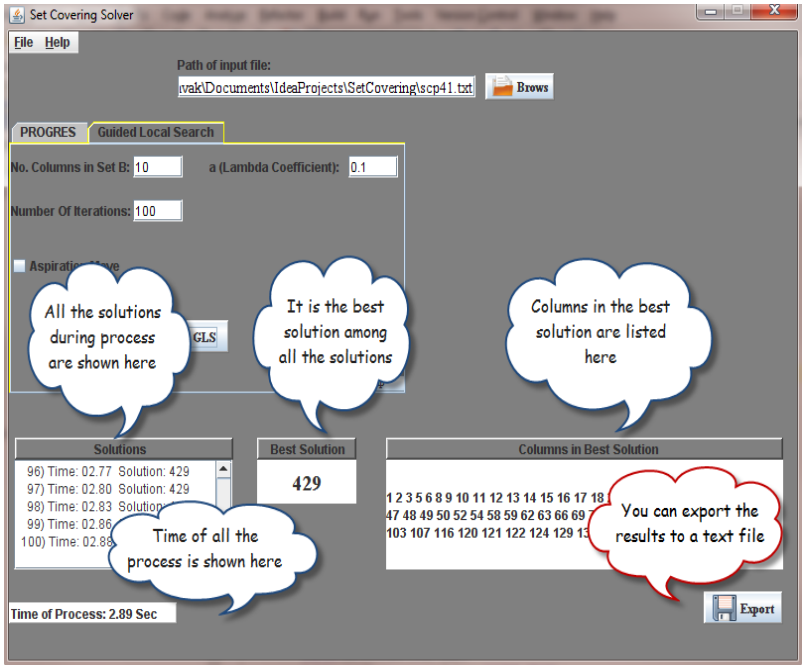

Figure 4 Results

3. PROGRES Algorithm

PROGRES algorithm is a probabilistic greedy search algorithm which presented by M. Haouari & JS. Chaouachi [2] for solving combinatorial optimization problems. To show performance and robustness of algorithm, they applied it for set covering problem.
To explain the PROGRES algorithm, first I need to introduce some notation from the paper:

A= (a_{ij}): 0-1 matrix whose columns are A_1, ..., A_n, respectively
I: set of rows of A
J: set of columns of A
I_j: set of rows such that a_{ij} = 1
C: set of columns in a solution
U: set of uncovered rows
α_j: number of rows in U such that a_{ij} = 1
β_i: number of columns in C that cover row i

This algorithm consists of two phases:

1. Pre-processing

2. Iterated Greedy Search.

In these two phases main procedure is *Probabilistic-Greedy*, before I explain this procedure, I need to declare formulation of the problem.

3.1 Formulation

This problem formulated as a matrix with zero and one elements. Each element $a_{ij}$with value 1 means that column i can cover row j. Each column j has a cost equal to c_j. The goal of the problem is to find a sub set from a set of columns in A to cover all the rows with minimum cost.

3.2 Probabilistic-Greedy

The most important part of this procedure is the way which it selects columns for the solution:

1. **Column Selection:** This procedure uses the $f_j = \frac{c_j}{\alpha_j}$ criteria for selecting the columns; where c_j is the cost of the column j and α_j is the number of rows that column j can cover. Columns with lower f_j have more chances to be selected for the solution.
2. **Randomization:** Instead of selecting a column with lowest f_j, the set B={A_i, i=1,2,..,s} has been defined to select columns from this set based on probability of columns which it can be calculated from the following function:

   $\pi_i = \frac{c_{max} - c_i + 1}{\sum_{j=1,s} c_{max} - c_j + 1}$

   The number of elements in set B can be changed in the demo.

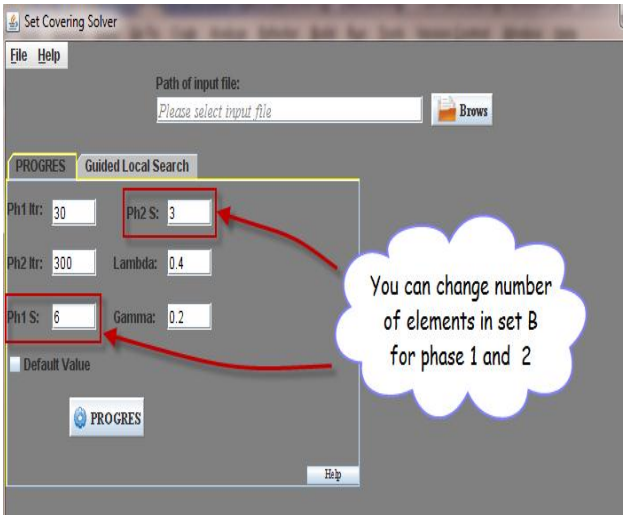

Figure 5 Number of elements in set B

The probabilistic greedy pseudo-code has been shown here:

procedure ProbabilisticGreedy(s,c,U)

begin

C ← ∅; /* set of columns in the solution*/

while (U≠ ∅)/*there is an uncovered row*/

begin

B← set of s best columns; /*columns with lower f_j */

π ← compute π_j for all elements of B;

C ← C ∪ $\{A_j\}$; /*A_j is selected randomly from set B based on π_j*/

β_i ← β_i+1 /*for all i ∈ I_j*/

U←U\$\{I_j\}$;

end

/*eliminate redundant columns*/

for all elements of C **do**

begin

if(β_i ≥2 for all $i \in I_j$**)**

begin

C←C\$\{A_j\}$;

end

end

end

3.3 Phase 1: Pre-processing

This phase aims to reduce number of columns in the instance to decrease search space. For number of iteration for example 60 times, it runs *Probabilistic-Greedy* procedure. Whenever a column is selected for the solution, then it will be labelled. After end of this phase, columns which have not been labelled yet, remove permanently from the instance.

To change number of iteration in phase 1, see the following picture.

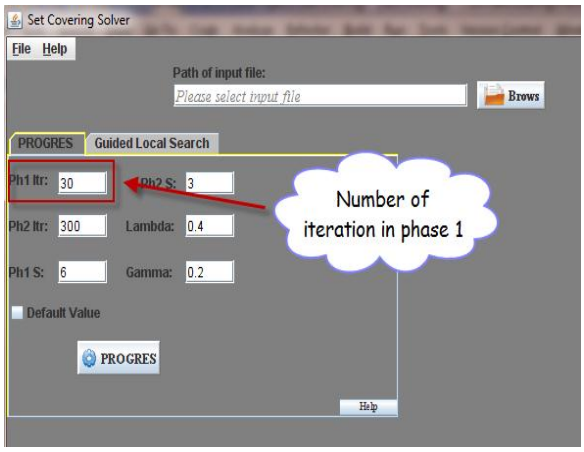

Figure 6 Phase1 iteration

3.4 Phase 2: Iterating Greedy Search

In this phase, partial destruction method has been implemented. In other words, instead of starting from zero point, (i.e. from a point that no row is covered) it destructs best solution which obtained yet by removing penalized columns and also other columns randomly until γ per cent of rows become uncover.

This phase starts by calling *Probabilistic-Greedy Procedure* and when it solves the problem, columns with maximum utility will be penalized (it is inspired from Guided Local Search) and cost of penalize columns augmented by $c_j = (1 + \lambda)c_j$ equation; where c_j is cost of column j and λ is the parameter of the algorithm which it determines effect of penalization on augmented cost function. Then, the original cost of this solution compared with the best solution, always in this phase we start from destructed best solution. Next, partial destruction method applied for best solution. And repeat this procedure for number of iteration which is given. Parameters in phase 2 are γ (Gamma), λ (Lambda), number of iteration (Ph2 Itr) and, number of elements in set B for phase 2 (Ph2 S), which you can change them in the demo.

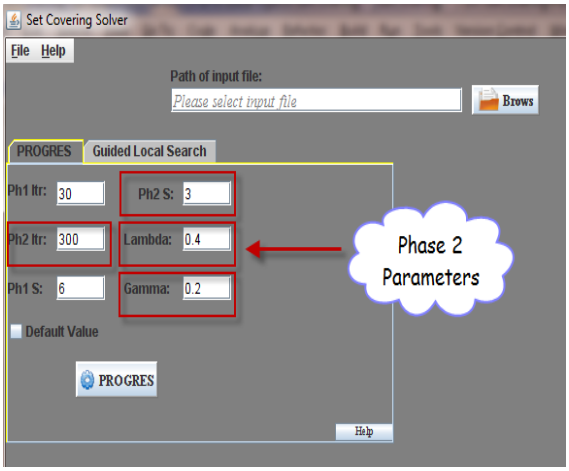

Figure 7 Phase 2 parameters

4. Guided Local Search

To implement GLS for set covering problem, we need to declare objective function, neighbourhood function, and solution features. Objective function in this problem is summation of cost of columns in the solution. Columns in the solution are considered as solution features of this problem.

Now, I introduce *1-column Drop Local Search* neighbourhood function. I will not explain Guided Local Search in this report, since it is supposed that the reader of this report is familiar with Guided Local Search concept and for further information I refer reader to C.Voudouris & E.Tsang [1] paper.

4.1 1-column Drop Local Search

At first step we start with a feasible solution, which it can be obtained by *Probabilistic-Greedy* Procedure. Then, we randomly drop one column from the solution and repair it by *Probabilistic-Greedy* to cover all the rows. If we couldn't obtain a better solution, pick another column and repair it again. We repeat this until we could find better solution. If we could not find a better solution, then we penalized columns with maximum utility and we start searching again with augmented cost function. The procedure has been summarized here.

Procedure 1-columnDropLocalSearch (C,h)

begin

C ← set of columns in the solution;

S ← set of columns which their neighbourhoods have been search;

Randomly find two columns A_i from C;

$\acute{C}$ ← C\{A_i};

S← {A_i};

While better solution found **do**

Probabilistic-Greedy(s, c, U)

if h($\acute{C}$)< h(C) **then**

begin

C ← $\acute{C}$;

break;

end

else

begin

Randomly find two columns A_i from C where $A_i \notin S$

$\acute{C}$ ← C\{A_i}

S← {A_i}

end

end

return C**;**

end

Where, h(C) is augmented cost function.

GLS has three parameters which they must be set before running the demo. Parameters are number of columns in set B, lambda coefficient (a) and, number of iteration. Number of columns in set B is the one which I declared it in PROGRES algorithm, lambda coefficient (a) is a real number between [0, 1] applied for obtaining λ in the $\lambda = a\frac{g(local\ minimum)}{N}$ equation, where N is number of columns in the solution and g (local minimum) is value of local minimum which reached so far.

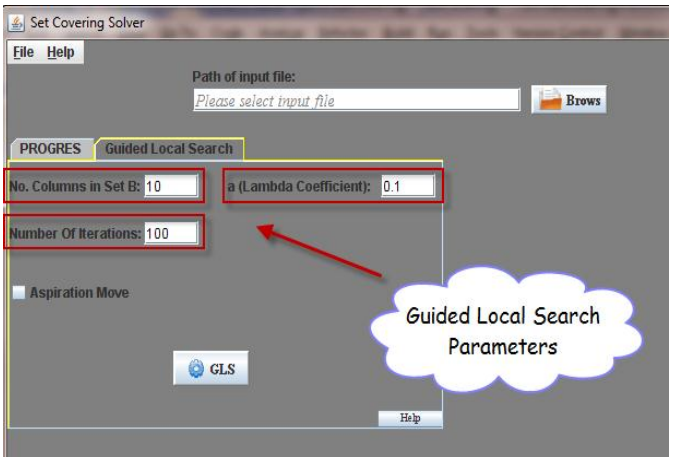

Figure 8 Guided Local Search parameters

4.2 Aspiration Move

You can apply extended guided local search which is introduced by P.Mills et al [3]. To apply extended guided local search you can just check aspiration move.

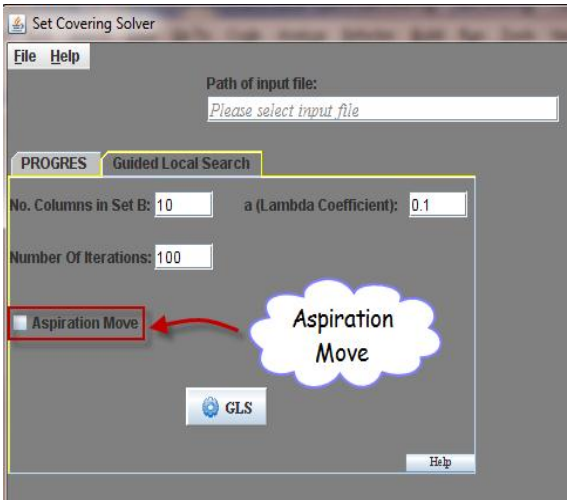

Figure 9 Aspiration move

5. Conclusion

In this research, I have implemented the PROGRESS algorithm. Then, I have applied Guided Local Search to achieve competitive results. I introduced a new neighbourhood functions which is 1-colum drop for Guided Local Search.

I develop a demo for set covering problem and I explained how you can run the demo.

6. References

1. Tsang E and Voudouris C (1999). Guided local search and its application to the travelling salesman problem. Eur J Opl Res 113: 469-499
2. M. Haouari and J.S. Chaouachi, A probabilistic greedy search algorithm for combinatorial optimisation with application to the set covering problem, Journal of the Operational Research Society 53 (2002), pp. 792–799.
3. Mills P., Tsang E., Ford J.: Applying an extended guided local search to the quadratic assignment problem. Ann. Oper. Res. 118(1-4), 121–135 (2003)