A Binary Particle Swarm Optimization Algorithm for a Variant of the Maximum Covering Problem

Bruno Prata * † Jorge Pinho de Sousa † Teresa Galvão †

April 14, 2009

* Center of Technological Sciences, University of Fortaleza - UNIFOR
Avenida Washington Soares, 1321, CEP 60811-905. Fortaleza, CE - Brazil
Email: baprata@unifor.br

† FEUP, Faculdade de Engenharia da Universidade do Porto
Rua Dr. Roberto Frias, 4200-465, Porto, Portugal
Email: {bruno.prata, tgalvao, jsousa}@fe.up.pt

1 Introduction

The Maximum Covering Problem (MCP) is a widely studied Combinatorial Optimization problem, with several applications, such as facility location (including health centers, emergency vehicles and commercial bank branches) and scheduling (flexible manufacturing systems, mass transit services, telecommunications)[1], [3], [4]. Practical instances of the problem are in general quite difficult to solve. Thus, approximate heuristic methods are used to achieve satisfactory solutions in acceptable computational times.

A GRASP approach for the MCP is presented in [1], while in [3] and [4] Genetic Algorithms are applied for the problem. The GRASP metaheuristic proposed by Feo and Resende [2] has been successfully tested in randomly generated instances of MCP. Arakaki and Lorena [3] propose a constructive Genetic Algorithm for the Maximum Covering Location Problem, using both real and theoretical instances in their computational experiments. Park and Ryu [4] present a Genetic Algorithm with unexpressed genes for the MCP, using real instances of a subway system. In the unexpressed genes, the chromosomes consist of an expressed part and an unexpressed part. The expressed part is used in evaluation of an individual and the unexpressed part preserve the information of an individual, maintaining the diversification.

Park and Ryu [4] describe a MCP approach for the crew pairing problem. The MCP can be a good model for the Vehicle and Crew Scheduling Problem (VCSP), if the changeovers (the change of a vehicle of a driver) are forbidden.

Given a matrix A, in which $a_{ij} \in \{0, 1\}$, the MCP consists in covering the largest number of rows of matrix A with a number of columns of matrix A less or equal to d. The variables y_i are associated with the rows of matrix A, so that $y_i = 1$ if the i^{th} row is not covered in a solution, being $y_i = 0$ otherwise. The variables x_j represent the columns of matrix A, so that $x_j = 1$ if the j^{th} column is part of the solution, being $x_j = 0$ otherwise. Obviously the x_j are decision variables, the y_i being auxiliary variables required for the definition of the objective function.

It should be noted that our version is a simplified version of the MCP, as the weights of the rows (considered by other authors) are here ignored. In this formulation, a maximum cover occurs when a minimum number of rows are left uncovered.The abovementioned formulation was adopted in scheduling problems of transport planning [4].

The variant of the MCP can be formulated mathematically as follows:

$$\min z = \sum_{i=1}^{n} y_i \tag{1}$$

subject to,

$$\sum_{j=1}^{m} x_j \leq d \tag{2}$$

$$\sum_{j=1}^{n} a_{ij} x_j + y_i \geq 1, \quad i = 1, \ldots, m \tag{3}$$

$$x_j \in \{0, 1\} \tag{4}$$

$$y_i \in \{0, 1\} \tag{5}$$

$$\tag{6}$$

The objective function represented by equation (1) aims at minimizing the number of uncovered rows of matrix A. The set of constraints (2) impose that a maximum number of d columns of matrix A be selected in the solution. The constraints of type (3) are used to define the auxiliary variables y_i. It is easy to show that constraints (5) are redundant and can therefore be replaced by simply bounds on the variables y_i. Therefore in our computational implementation, the number of binary variables is m.

The MCP is similar to the Set Covering Problem (SCP). While in the SCP the goal is to cover the rows of matrix A with minimum cost, in the MCP the goal is to minimize the number of uncovered rows of A, with no type of cost being measured. It seems that constraint (2) makes the problem really difficult to solve.

The objective of the work reported on this paper was to create a Binary Particle Optimization Algorithm for the Maximum Covering Problem. This work should obviously be viewed as a preliminary stage of a larger research project.

2 Particle Swarm Optimization

Evolutionary Algorithms are inspired by the theory of evolution of species, by recombining solutions, and thus aiming at performing an intelligent search of the solution space. As examples we have Genetic Algorithms (GA) and optimization based on colonies of socials insects, such as ants, bees, and termites.

Kennedy and Ebehart [5] proposed a new meta-heuristic for solving unconstrained problems of non-linear optimization they have called Particle Swarm Optimization (PSO). The philosophy of PSO is the following: an initial population of particles is generated, each one in an initial position of the space of solutions and with a certain initial velocity. Each particle i is characterized by its position and a vector of change in position called velocity [9].

A particle is a candidate solution of the swarm in a step of the search. The swarm is the whole set of particles in a given iteration. The velocity is a number that leads the movement of the particles. Each particle will go through the n-dimensional search space and will have its velocity updated according to the velocity of the other particles.

For each particle, its best fitness in the search process is stored, and this value is named pbest. The overall pbest value, obtained evaluating all the particles in the population, is the best solution of the search, which is called gbest. The values of pbest and gbest are used for the updating of velocities of the particles in the search space. The particles that are far from the promising regions of the search space will have their velocity increased, while the other particles will have their velocity decreased.

PSO has a search strategy extremely efficient [5], [6]: it has an easy computational implementation, requiring few lines of code; it uses little memory and requires few processing speed; and the search process is enhanced by the continuous learnship of the particles. After the promising results achieved by PSO in non-linear programming problems, its application to combinatorial optimization problems showed also to be very promising.

Kennedy and Ebehart [7] developed a binary version for PSO and concluded that the meta-heuristic is also flexible and robust for this class of problems. Tasgetiren and Liang [8] applied a binary PSO for the Lot Sizing Problem, obtaining results of better quality than a Genetic Algorithm.

3 Proposed heuristic

For the resolution of the MCP with PSO, the following coding was used: each particle consists of a binary vector of m elements, representing the values of the x_j. Therefore, the solution space is $x = \{x_1, x_2, \ldots, x_m\}, x_j \in \{0, 1\}$. It should be noted that many solutions may be unfeasible for the original problem, due the constraints (4) and (5). The neighborhoods are the solutions obtained from x by a moving. A movement is a flip in a bit of the binary vector. Therefore, the total number of solutions is 2^m.

Since it is an evolutionary algorithm, PSO operates on a population of solutions (particles), with size *maxpop*. The search is performed in *maxgen* iterations (generations). The initial population is generated in a random way. The values $P[x_j = 1] = 0.05$ and $P[x_j = 0] = 0.95$ were adopted, because the vector of decision variables is very sparse. Therefore, it is likely that constraint (2) is satisfied.

This policy for the generation of the initial population does not prevent the generation of unfeasible solutions. However, it hopefully guarantees a good diversification of the search. An infeasible solution can be a neighbor of a high quality solution.

The evaluation function (fitness of a solution) is the number of rows of matrix A uncovered in a certain solution, to be minimized. In order not to violate constraint (2), a penalty factor equal to 20 was adopted, per extra column, to penalize particles that use more than d columns in a solution. This penalty factor was adjusted empirically, after some initial computational experiments.

The velocities are stored in a matrix of $maxpop \times m$ order. The values are generated through the following expression: $velocity[i, j] = vmin + (vmax - vmin) * random[0, 1]$. The parameters *vmax* and *vmin* are the maximum and minimum values allowed for the velocity of each particle (the following values have been adopted: $vmax = ln(m)$ and $vmin = -ln(m)$).

Be *pbest* and *gbest* the best solutions obtained, respectively, by the ith particle and by the swarm. Be x_{pbest} and x_{gbest} the positions of such solutions, the velocity variation is given through the expression:

$$\Delta v[i, j] = c_1 * random[0, 1] * (x_{pbest} - x) + c_2 * random[0, 1] * (x_{gbest} - x)$$

c_1 is the cognitive constant and c_2 is the social constant. The first is related to the learnship of the particle, based on its own move in the search space, while the social constant is related to the learnship of the particle concerning the behavior of the swarm as a whole. A position of a particle is a binary vector in the solution space is x.

As in a binary optimization problem the decision variables only take the values 0 and 1, a function

that transforms a real velocity, defined in the interval $[vmin, vmax]$, into the interval $[0, 1]$ is required. The sigmoid function is usually used for this purpose [7], [8]:

$$sigmoid(v[i, j]) = \frac{1}{1 + e^{-v[i,j]}}$$

Thus, the value of the sigmoid for the velocity of the i^{th} particle in the j^{th} dimension (bit) is calculated. If this value is smaller than a random number uniformly distributed, the j^{th} bit takes the value 1; otherwise, it takes the value 0. Algorithm 1 presents the pseudo code for updating the position of the particles.

```
for i = 1 to maxpop do
    for j = 0 to m do
        r ← [0, 1];
        if r < sigmoid(v[i, j]) then
            x[i, j] ← 1;
        else
            x[i, j] ← 0;
        end
    end
end
```

Algorithm 1: Movement of particles

After the accomplished moves, the fitness of each particle is calculated and the unfeasible solutions are penalized. If the value of gbest does not improve during k iterations of the algorithm, a restart of the velocity of the particles, except from xgbest. After that, at Algorithm 2, the pseudo code of the heuristic proposed is presented.

According to [1], both constructive algorithms and local search can play an important role in the performance of algorithms for the MCP. However, the authors employed a PSO without the hybridization of such techniques aiming to demonstrate that the mechanisms of social learnship of PSO, by themselves, can incur in the obtaining of good solutions in combinatorial binary optimization problems, as it is the case of MCP.

```
Generate initial population ;
Generate initial velocities ;
Generate initial values for pbest and gbest;
while gen ≤ maxgen do
    Find pbest and gbest ;
    Calculate velocity of the i^th particle in the j^th dimension;
    Update velocity of the ith particle in the j^th dimension;
    Update each bit in string using a sigmoid function (update position) ;
    Evaluate fitness of each particle ;
    Assign penalization to unfeasible solutions ;
    Restart velocities for each k iterations without improvement of gbest;
    gen ← gen + 1 ;
end
```

Algorithm 2: Pseudo code of proposed PSO heuristic

Instance	n	m	d	ρ	Global Optimum	t(s)
MCP01	200	200	16	4.02	48	5434
MCP02	200	200	10	3.82	91	48
MCP03	200	200	18	3.94	37	5906
MCP04	200	200	18	3.83	40	2896
MCP05	200	200	16	3, 93	44	232
MCP06	200	200	6	9.22	65	396
MCP07	200	200	17	3.96	41	443
MCP08	200	200	32	3.83	0	1621
MCP09	200	200	16	3.95	52	2148
MCP10	200	200	23	4.12	15	19193

Table 1: Test instances

4 Computational Results

For the computational experiments, 10 instances of medium size were generated randomly. These instances were solved in an exact way with LINGO 8.0. The values n=m=200 have been adopted to allow the resolution of the instances to optimality. These instances are presented in Table 1 as follows: number of rows of matrix A (n), number of columns of matrix A (m), maximum number of columns to be taken into the solution (d) and density of the matrix A (ρ), i.e. number of "1"s divided by the total number of elements, and t(s) is the time needed to solve the problem.

The algorithm was implemented in Pascal. The experiments were made in an AMD Semprom 2400 + 1.67 GHz, 504MB RAM. The following parameters were utilized: $maxpop = 15$, $maxgen = 2500$ and $c1 = c2 = 1$. For the restart routine of the velocities, $k = 500$ generations was adopted. It should be emphasized that the performance of PSO is not so much dependent on parameters.

For each instance, 20 runs of the algorithm were made. The results are presented in Table 2. The first column of the table identifies the instance. The second and third columns present, respectively, the best and the worst results obtained by the heuristic. The fourth and the fifth column present the mean and the standard deviation of the results obtained by the algorithm, for the 20 runs. In the sixth column, we have the number of times that the algorithm found the optimal solution of the instance. Finally, in the seventh column, the average computational times are presented.

Based on these results, we can say that, although the initial solutions generated are of low quality and there is no repair heuristics, PSO could obtain good solutions in acceptable computational times. For 5 out of the 10 instances considered, the optimal solutions were achieved. Such fact confirms the potential of PSO in solving this type of Combinatorial Optimization problems.

The strategy of generating initial solutions in a random way, without taking into consideration the specific characteristics of the problem, turned out to be very inefficient. i.e., the quality of the initial solutions has a considerable impact on the quality of the final solutions.

Constraint (2) leads to feasible vectors x with a few number of "1"s. The several local optima are therefore quite "different", being separated by a large Hamming distance. Thus, migration to global optima becomes, in general, quite difficult.

The quality of a solution for the MCP can be assessed through the calculation of the covering percentage p, determined by the following expression, in which n is the number of rows of matrix A and z is the value of the objective function obtained, which represents the number of uncovered rows:

Instance	Global Optimum	Best solution	Worst solution	Mean	Std dev.	n.opt. sol. in 20 runs	tm(s)
MCP01	48	48	55	50.10	1.76	4	23.71
MCP02	91	92	96	93.90	1.22	0	23.78
MCP03	37	37	44	40.35	1.77	2	23.74
MCP04	40	42	46	44.10	1.70	0	23.77
MCP05	44	44	50	46.45	1.60	4	23.94
MCP06	65	65	74	70.15	3.35	5	23.85
MCP07	41	41	48	43.70	2.85	9	23.86
MCP08	0	4	8	5.85	1.39	0	23.79
MCP09	52	54	58	55.65	1.62	0	23.60
MCP10	15	19	24	21.20	1.03	0	23.82

Table 2: Computational Results

Instance	Global Optimum	poptimum	pbest	pmean	pworst	gapbest	gapmean	gapworst
MCP01	48	76.00	76.00	74.95	72.50	0.00	1.05	3.50
MCP02	91	54.50	54.00	53.05	52.00	0.50	1.45	2.50
MCP03	37	81.50	81.50	79.83	78.00	0.00	1.67	3.50
MCP04	40	80.00	79.00	77.95	77.00	1.00	2.05	3.00
MCP05	44	78.00	78.00	76.78	75.00	0.00	1.23	3.00
MCP06	65	67.50	67.50	64.93	75.00	0.00	2.58	4.50
MCP07	41	79.50	79.50	78.15	63.00	0.00	1.35	3.50
MCP08	0	100.00	98.00	97.08	76.00	2.00	2.93	4.00
MCP09	52	74.00	73.00	72.18	71.00	1.00	1.83	3.00
MCP10	15	92.50	90.50	89.40	88.00	2.00	3.10	4.50

Table 3: Comparison between heuristic results and optimal solutions.

$$p = \frac{(n - z)}{n}$$

Based on the mean of the values of the solutions obtained by the heuristic, it was possible to calculate the deviation (gap) between the covering percentage p obtained by PSO and by the exact method, for the instances analyzed. For this class of instances the results seem to be very promising (Table 3).

5 Conclusions

The main goal of this work was to demonstrate the potential of PSO in solving the Maximum Covering Problem. This new way to handle the MCP is interesting because the MCP can be quite useful in modeling important practical problems such as those occurring in simultaneous vehicle and crew scheduling.

The heuristic proposed was tested in a set of 10 randomly generated instances of medium size that were solved to optimality with LINGO 8.0.

The accomplished computational experience showed that the algorithm could produce solutions of good quality with a low computational cost. The process of collective learnship of the particles seems

to be a good way to guarantee a comprehensive search of the solution space.

Unfortunately, there are not available instances for the variant of Maximum Covering Problem studied in the paper. Therefore, the comparison to other meta-heuristic is difficult. This comparison is a topic for future studies.

Future research should also cover the design of heuristics for repairing unfeasible solutions, the generation of an initial population, and the hybridization of the PSO heuristic with local search procedures.

Comprehensive tests with larger and heterogeneous instances are required, in order to fully validate and tune the proposed general approach. In line with this research area, the authors are developing an application of PSO for the Vehicle and Crew Scheduling Problem (VCSP). The MCP model for the VCSP is likely to have a good performance, when compared with the more traditional Set Covering/Set Partitioning approaches.

References

[1] Mauricio G. C. Resende. *Computing approximate solutions of the maximum covering problem with GRASP. Journal of Heuristics*, 4: 161-177, 1998.

[2] T. A. Feo and Mauricio G. C. Resende. *A probabilistic heuristic for a computationally difficult set covering problem. Journal of Operations Research Letter* , 8: 67-71, 1989.

[3] R. G. I. Arakaki and L. A. N. Lorena. *A constructive Genetic Algorithm for the Maximal Covering Location Problem.* In Proceedings of 4th Metaheuristics International Conference, 13-17, Porto, 2001.

[4] T. Park and K. R. Ryu *Crew pairing optimization by a genetic algorithm with unexpressed genes. Journal of Intelligent Manufacturing*, 17: 375-383, 2006.

[5] J. Kennedy and R. C. Eberhart. *Particle swarm optimization.* In Proceedings of the IEEE International Conference on Neural Networks, Piscataway, 1995.

[6] J. Kennedy and R. C. Eberhart. *A new optimizer using particle swarm theory.* In Proceedings of the 6th International Symposium on Micromachine and Human Science, Nagoya, 1995.

[7] J. Kennedy and R. C. Eberhart. *A discrete binary version of the particle swarm algorithm.* In Conference on Systems, Man and Cybernetics, Hyatt, 1997.

[8] M. F. Tasgetiren and Y. C. Liang. *A binary Particle Swarm Optimization Algorithm for the lot sizing problem. Journal of Economic and Social Research*, 5: 1-20, 2003.

[9] J. Dréo, A. Pétrowksi, P. Siarry and E. Taillard. *Metaheuristics for hard optimization.* Springler, Heidelberg, 2006.

[10] N. Nedjah and L. M. Mourelle (Eds.). *Swarm Intelligent Systems.* Springler, Heidelberg, 2006.