Ant Colonies using Arc Consistency Techniques for the Set Partitioning Problem

Broderick Crawford[1,2] and Carlos Castro[2]

[1] Escuela de Ingeniería Informática, Pontificia Universidad Católica de Valparaíso, Chile
[2] Departamento de Informática, Universidad Técnica Federico Santa María, Chile
broderick.crawford@ucv.cl Carlos.Castro@inf.utfsm.cl

Abstract. In this paper, we solve some benchmarks of Set Covering Problem and Equality Constrained Set Covering or Set Partitioning Problem. The resolution techniques used to solve them were Ant Colony Optimization algorithms and Hybridizations of Ant Colony Optimization with Constraint Programming techniques based on Arc Consistency. The concept of Arc Consistency plays an essential role in constraint satisfaction as a problem simplification operation and as a tree pruning technique during search through the detection of local inconsistencies with the uninstantiated variables. In the proposed hybrid algorithms, we explore the addition of this mechanism in the construction phase of the ants so they can generate only feasible partial solutions. Computational results are presented showing the advantages to use this kind of additional mechanisms to Ant Colony Optimization in strongly constrained problems where pure Ant Algorithms are not successful.

1 Introduction

Set Covering Problem (SCP) and Set Partitioning Problem (SPP), or Equality Constrained Set Covering, are two types of problems that can model different real life situations [2, 3, 11, 21]. In this work, we solve some benchmarks of them with Ant Colony Optimization (ACO) algorithms and some hybridizations of ACO with Constraint Programming based on Arc Consistency.

There exist problems for which ACO is of limited effectiveness, among them the very strongly constrained problems. They are problems for which neighborhoods contain few solutions, or none at all, and local search is of very limited use. Probably, the most significant of such problems is the SPP [18]. A direct implementation of the basic ACO framework is incapable of obtaining feasible solutions for many standard tested instances of SPP. The best performing metaheuristic for SPP is a genetic algorithm due to Chu and Beasley [5, 6]. There exist already some first approaches applying ACO to the SCP. In [1, 16] ACO has been used as a construction algorithm and the approach has only been tested on some small SCP instances. Others works [15, 17] apply Ant Systems to the SCP and use techniques to remove redundant columns and local search to improve solutions. In [14] there is a very interesting work of ACO solving the related problem Set Packing.

In this paper we explore the addition of a mechanism, usually used in complete techniques, in the construction phase of ACO algorithms in a different

Please use the following format when citing this chapter:

Crawford, B., Castro, C., 2006, in IFIP International Federation for Information Processing, Volume 218, Professional Practice in Artificial Intelligence, eds. J. Debenham, (Boston: Springer), pp. 295–301.

way of the used form in [20] where was proposed a lookahead function evaluating pheromone in a supersequence problem, and in [13] where was introduced a lookahead mechanism to estimate the quality of the partial solution.

The remainder of the paper is organised as follows: Section 2 is devoted to the presentation of the problems and their mathematical models. In Section 3, we describe the applicability of the ACO algorithm for solving SPP and SCP. In Section 4, we present the basic concepts to adding Constraint Programming techniques to the two basic ACO algorithms: Ant System and Ant Colony System. In Section 5, we present results when adding Constraint Programming techniques to the two basic ACO algorithms to solve some benchmarks from ORLIB. Finally, in Section 6 we conclude the paper and give some perspectives for future research.

2 Equality Constrained Set Covering Problem

Equality Constrained Set Covering Problem, or Set Partitioning Problem, is the NP-complete problem of partitioning a given set into mutually independent subsets while minimizing a cost function defined as the sum of the costs associated to each of the eligible subsets. SPP importance derives from the fact that many combinatorial optimization problems (such as, crew scheduling, vehicle routing, project scheduling, and warehouse location problems, to name a few) can be modeled as SPP with maybe some additional constraints. In SPP we are given a $m \times n$ matrix $A = (a_{ij})$ in which all the matrix elements are either zero or one. Additionally, each column is given a non-negative cost c_j. We say that a column j can cover a row i if $a_{ij} = 1$. Let J denotes a subset of the columns and x_j a binary variable which is one if column j is chosen and zero otherwise. The SPP can be defined formally as follows:

$$Minimize \quad f(x) = \sum_{j=1}^{n} c_j \times x_j$$

Subject to

$$\sum_{j=1}^{n} a_{ij} \times x_j = 1; \forall i = 1, \ldots, m$$

These constraints enforce that each row is covered by exactly one column. The SCP is a SPP relaxation. The goal in the SCP is to choose a subset of the columns of minimal weight which covers every row. The SCP can be defined formally using constraints to enforce that each row is covered by at least one column as follows:

$$\sum_{j=1}^{n} a_{ij} \times x_j \geq 1; \forall i = 1, \ldots, m$$

3 Ant Colony Optimization for Set Covering Problems

ACO can be applied in a very straightforward way to the SCP and SPP. The columns are chosen as the solution components and have associated a cost and a pheromone trail [10]. Each column can be visited by an ant once and only once and that a final solution has to cover all rows. A walk of an ant over the graph representation corresponds to the iterative addition of columns to the partial solution obtained so far. Each ant starts with an empty solution and adds columns until a cover is completed. A pheromone trail and a heuristic information are associated to each eligible column j. A column to be added is chosen with a probability that depends of pheromone trail and the heuristic information. In the application of ACO to other problems, such as the TSP, there are some differences. For example, the SPP/SCP solution construction of the individual ants does not necessarily end after the same number of steps of each ant, but only when a cover is completed. One of the most crucial design decisions to be made in ACO algorithms is the modeling of the set of pheromones. In the original ACO implementation for TSP the choice was to put a pheromone value on every link between a pair of cities, but for other combinatorial problems often can be assigned pheromone values to the decision variables (first order pheromone values) [10]. In this work the pheromone trail is put on the problems componentes (each elegible column J) instead of the problems connections. And setting good pheromone quantity is a non trivial task too. The quantity of pheromone trail laid on columns is based on the idea: *the more pheromone trail on a particular item, the more profitable that item is* [16]. Then, the pheromone deposited in each component will be in relation to its frequency in the ants solutions. In this work we divided this frequency by the number of ants obtaining better results. We use a dynamic heuristic information that depends on the partial solution of an ant. It can be defined as $\eta_j = \frac{e_j}{c_j}$, where e_j is the so called cover value, that is, the number of additional rows covered when adding column j to the current partial solution, and c_j is the cost of column j [10]. In other words, the heuristic information measures the unit cost of covering one additional row. An ant ends the solution construction when all rows are covered. Figure 1 describe the pure ACO algorithm to solve SCP and SPP.

In this work, we use two instances of ACO: Ant System (AS) and Ant Colony System (ACS) algorithms, the original and the most famous algorithms in the ACO family [9, 8, 10]. ACS improves the search of AS using: a different transition rule in the constructive phase, exploting the heuristic information in a more rude form, using a list of candidates to future labeling and using a different treatment of pheromone. ACS has demostrated better performance than AS in a wide range of problems.

Trying to solve larger instances of SPP with the original ACO implementation derives in a lot of unfeasible labeling of variables, and the ants can not obtain complete solutions. In this paper we explore the addition of a mechanism in the construction phase of ACO in order to that only feasible partial solutions are generated.

```
1  Procedure ACO_for_SCP_and_SPP
2   Begin
3    InitParameters();
4     While (remain iterations) do
5      For k := 1 to nants do
6       While (solution is not completed) do
7        AddColumnToSolution(election)
8        AddToTabuList(k);
9       EndWhile
10     EndFor
11     UpdateOptimum();
12     UpdatePheromone();
13    EndWhile
14  Return best_solution_founded
15 End.
```

Fig. 1. Pure ACO algorithm for SCP and SPP

4 ACO with Constraint Programming

Recently, some efforts have been done in order to integrate Constraint Programming techniques to ACO algorithms [19, 12].An hibridization of ACO and CP can be approached from two directions: we can either take ACO or CP as the base algorithm and try to embed the respective other method into it. A form to integrate CP into ACO is to let it reduce the possible candidates of the not yet instantiated variables participating in the same constraints that the actual variable. A different approach would be to embed ACO within CP. The obvious point at which ACO can interact with CP is during the labeling phase, using ACO to learn a value ordering that is more likely to produce good solutions.

In this work, ACO uses CP in the variable selection (when adding columns to partial solution). The CP algorithm used in this paper is Arc Consistency Technique and Chronological Backtracking [7]. In the construction phase ACO performs Arc Consistency between pairs of a not yet instantiated variable and an instantiated variable, i.e., when a value is assigned to the current variable, any value in the domain of a future variable which conflicts with this assignment is removed from the domain.

Adding Arc Consistency Technique and Backtracking to ACO for SPP means that columns are chosen if they do not produce any conflict with respect to the next column to be chosen, trying to assure the possibilities of ants in order to complete the solutions. Figure 2 describe the hybrid ACO with CP.

5 Experiments and Results

Table 1 presents results when adding Forward Checking to the basic ACO algorithms for solving standard benchmarks taken from the ORLIB [4]. The first four columns of the Table 1 present the problem code, the number of rows (constraints), the number of columns (decision variables), and the best known solution for each instance, respectively. The next two columns present the cost obtained when applying AS and ACS, and the last two columns present the results combining AS and ACS with Forward Checking. Considering several tests

```
1  Procedure ACO+CP_for_SPP
2   Begin
3    InitParameters();
4    While (remain iterations) do
5       For k := 1 to nants do
6          While (solution is not completed) and TabuList <> J do
7             Choose next Column j with Transition Rule Probability
8             For each Row i covered by j do                /* j constraints          */
9               feasible(i):= Posting(j);                   /* Constraint Propagation */
10            EndFor
11            If feasible(i) for all i then AddColumnToSolution(j)
12                                     else Backtracking(j); /* set j uninstantiated   */
13            AddColumnToTabuList(j);
14         EndWhile
15      EndFor
16     UpdateOptimum();
17     UpdatePheromone();
18   EndWhile
19   Return best_solution_founded
20  End.
```

Fig. 2. Hybrid ACO+CP algorithm for SPP

and published experimental results [16, 17, 10] we use the following parameters values for the algorithms: evaporation rate = 0.4, number of iterations = 160, number of ants = 120, influence of pheromone (alpha) = 1.0, influence of heuristic information (beta) = 0.5, for ACS the list size = 500 (in scp41, scp42, scp48, scp61, scp62, and scp63), for ACS Qo = 0.5.

Algorithms were implemented using ANSI C, GCC 3.3.6, under Microsoft Windows XP Professional version 2002.

Problem	Rows	Columns	Optimum	AS	ACS	AS + FC	ACS + FC
sppnw39	25	677	10080	11670	10758	11322	10545
sppnw34	20	899	10488	13341	11289	10713	10797
sppnw26	23	771	6796	6976	6956	6880	6880
sppnw23	19	711	12534	14304	14604	13932	12880
scp41	200	1000	429	473	463	458	683
scp42	200	1000	512	594	590	574	740
scp48	200	1000	492	524	522	537	731
scp51	200	1000	253	289	280	289	464
scp61	200	1000	138	157	154	155	276
scp62	200	1000	146	169	163	170	280
scp63	200	1000	145	161	157	161	209

Table 1. ACO with Forward Checking

The effectiveness of ACO improved with Arc Consistency is shown to the SPP, the strongly constrained characteristic of this problem does the stochastic behavior of pure unsuitable for solve it. In the original ACO implementation the SPP solving derives in a lot of unfeasible labeling of variables, and the ants can not complete solutions. For SCP, the huge size of the search space and the relaxation of the constraints does ACO algorithms work better than ACO+CP considering the same execution conditions.

6 Conclusions and Future Directions

The concept of Arc Consistency plays an essential role in Constraint Programming as a problem simplification operation and as a tree pruning technique during search through the detection of local inconsistencies among the uninstantiated variables. We have shown that it is possible to add Arc Consistency to any ACO algorithms. The computational results confirm that the performance of ACO is possible to improve with some types of hibridization.

Our goal was to demonstrate that ACO is possible to improve with CP in some kind of problems. Future versions of the algorithm will study the pheromone treatment representation and the incorporation of available local search techniques in order to reduce the input problem (Pre Processing) and improve the solutions given by the ants (Post Processing). The ants solutions may contain redundant components which can be eliminated by a fine tuning after the solution, then we will explore Post Processing procedures, which consist in the identification and replacement of the columns of the ACO solution in each iteration by more effective columns.

References

1. D. Alexandrov and Y. Kochetov. Behavior of the Ant Colony Algorithm for the Set Covering Problem. In *Proc. of Symp. Operations Research*, pp 255–260. Springer Verlag, 2000.
2. E. Andersson, E. Housos, N. Kohl and D. Wedelin. Crew Pairing Optimization. *In Yu G.(ed.)Operations Research in the Airline Industry*,Kluwer Academic Publishing, 1998.
3. E. Balas and M. Padberg. Set Partitioning: A Survey. *SIAM Review*, 18:710–760, 1976.
4. J. E. Beasley. OR-Library:Distributing test problem by electronic mail. *Journal of Operational Research Society*, 41(11):1069–1072, 1990.
5. J. E. Beasley and P. C. Chu. A genetic algorithm for the set covering problem. *European Journal of Operational Research*, 94(2):392–404, 1996.
6. P. C. Chu and J. E. Beasley. Constraint handling in genetic algorithms: the set partitoning problem. *Journal of Heuristics*, 4:323–357, 1998.
7. R. Dechter and D. Frost. Backjump-based Backtracking for Constraint Satisfaction Problems. *Artificial Intelligence*, 136:147–188, 2002.
8. M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant Algorithms for Discrete Optimization. *Artificial Life*, 5:137–172, 1999.
9. M. Dorigo and L. M. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.
10. M. Dorigo and T. Stutzle. *Ant Colony Optimization*. MIT Press, USA, 2004.
11. A. Feo, G. Mauricio, and A. Resende. A Probabilistic Heuristic for a Computationally Difficult Set Covering Problem. *OR Letters*, 8:67–71, 1989.
12. F. Focacci, F. Laburthe and A. Lodi. Local Search and Constraint Programming. *Handbook of metaheuristics*,Kluwer, 2002.

13. C. Gagne, M. Gravel and W.L. Price. A Look-Ahead Addition to the Ant Colony Optimization Metaheuristic and its Application to an Industrial Scheduling Problem. In J.P. Sousa et al., eds., *Proceedings of the fourth Metaheuristics International Conference MIC'01*, July 16-20, 2001. Pages 79-84.
14. X. Gandibleux, X. Delorme and V. T'Kindt. An Ant Colony Algorithm for the Set Packing Problem. In M. Dorigo et al., editor, *ANTS 2004*, vol 3172 of *LNCS*, pp 49–60. SV, 2004.
15. R. Hadji, M. Rahoual, E. Talbi, and V. Bachelet. Ant colonies for the set covering problem. In M. Dorigo et al., editor, *ANTS 2000*, pp 63–66, 2000.
16. G. Leguizamón and Z. Michalewicz. A new version of Ant System for subset problems. In *Congress on Evolutionary Computation, CEC'99*, pp 1459–1464, Piscataway, NJ, USA, 1999. IEEE Press.
17. L. Lessing, I. Dumitrescu, and T. Stutzle. A Comparison Between ACO Algorithms for the Set Covering Problem. In M. Dorigo et al., editor, *ANTS 2004*, vol 3172 of *LNCS*, pp 1–12. SV, 2004.
18. V. Maniezzo and M. Milandri. An Ant-Based Framework for Very Strongly Constrained Problems. In M. Dorigo et al., editor, *ANTS 2002*, vol 2463 of *LNCS*, pp 222–227. SV, 2002.
19. B. Meyer and A. Ernst. Integrating ACO and Constraint Propagation. In M. Dorigo et al., editor, *ANTS 2004*, vol 3172 of *LNCS*, pp 166–177. SV, 2004.
20. R. Michel and M. Middendorf. An Island model based Ant system with lookahead for the shortest supersequence problem. *Lecture notes in Computer Science, Springer Verlag*, 1498:692–701, 1998.
21. R. L. Rardin. *Optimization in Operations Research.* Prentice Hall, 1998.