Behavior of the Ant Colony Algorithm for the Set Covering Problem *

Dmitri Alexandrov, Munich
Yuri Kochetov, Novosibirsk

Summary: We develop the Ant Colony approach for the classical Set Covering problem. As artificial ants we use three randomized greedy heuristics: *GRASP* heuristic and two asymptotically exact heuristics *Random Neighborhood* and *Random Covering Set*. We study the influence of the main control parameters and present new ideas for accumulating and exploiting of statistical information. Computational results for difficult benchmarks are discussed.

1 Introduction

In this paper we study the behavior of the Ant Colony algorithm (AC) for the Set Covering Problem (SCP). The problem is formulated as follows. Given the sets $I = \{1, \dots, \mathbf{I}\}$, $J = \{1, \dots, \mathbf{J}\}$ and a boolean matrix (a_{ij}), $i \in I$, $j \in J$. Find a subset $S^* \subseteq J$ of minimal cardinality such that $\sum_{j \in S^*} a_{ij} \geq 1$ for all $i \in I$.

SCP is a well-known NP-hard problem. Considerable computational efforts are necessary for finding an optimal solution. Different algorithms are studied for this problem: randomized and deterministic approximation algorithms [8], Lagrangean heuristic [1], L-class enumeration algorithms [9], genetic algorithms [2, 5]. We apply AC approach [4] and present new ideas for accumulating and exploiting of statistical information. Computational results for difficult benchmarks indicate the advantage of the new algorithm.

2 The Ant Colony Algorithm

The main idea of the AC approach consists in exploiting of the statistical information as an optimization tool. During the search process we construct plenty of feasible solutions in a probabilistic way. These solutions influence on the statistical information and then this information influences on the construction of new solutions. In order to generate feasible solutions we apply three randomized greedy heuristics: *Random Neighborhood* (RN), *Random Covering Set* (RCS), and *Greedy Randomized Adaptive Search Procedure* ($GRASP$) [6].

RN algorithm can be stated as follows. For each subset $S \subset J$ and $j \notin S$ denote $I(S) = \{i \in I \mid \sum_{j \in S} a_{ij} = 0\}$, $I_j(S) = \{i \in I(S) \mid a_{ij} = 1\}$ and $\rho_j = \mid I_j(S) \mid$. A random neighborhood $L(p)$ with threshold p is a subset of $J \setminus S$. Each element $j \in J \setminus S$ is included in $L(p)$ randomly with probability p and independently from other elements. Algorithm RN starts with the empty set S. On each iteration the

*This work was supported by the RFBR grant 99-01-00510.

algorithm generates a random subset $L(p)$ and finds an element $j_0 \in L(p)$ with maximal value ρ_j. Then the element j_0 is added to the set S. The algorithm continues until $I(S) \neq \emptyset$.

Algorithm RN

1. Set $S_{RN} := \emptyset$.
2. While $I(S_{RN}) \neq \emptyset$ do
   2.1. Create the random subset $L(p)$.
   2.2. Find the best element $\rho_{j_0} = \max \{\rho_j,\ j \in L(p)\}$.
   2.3. Update $S_{RN} \leftarrow S_{RN} \cup \{j_0\}$.

The performance of algorithm RN strongly depends on the value of the parameter p. For $p = 1$ the algorithm is equivalent to the deterministic greedy algorithm of Chvátal [3]. Figure 1 shows results of 1000 trials of algorithm RN for different values of p. If $p = 1$, there is no effect of randomization and the algorithm always finds the same solution. The best results are obtained for $p \in [0.3, 0.5]$.

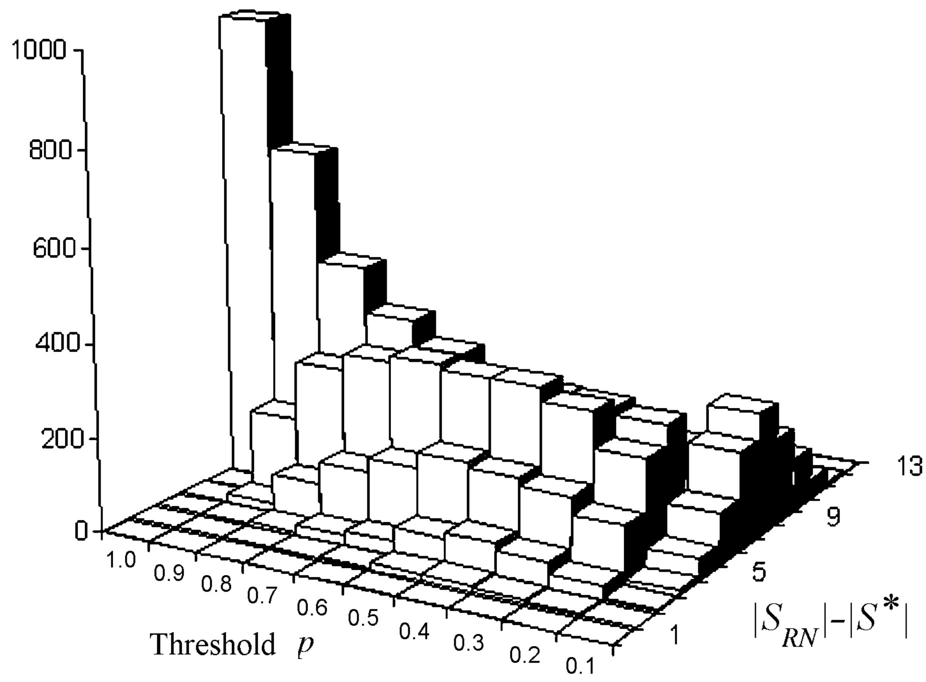

Figure 1: Influence of randomization on the performance of algorithm RN

Algorithm RN does not depend on the statistical information. All trials are independent. The main idea of algorithm AC is to memorize some information about the search process. The information is accumulated in the vector (τ_j), where $\tau_j \in [\tau_{min},\ \tau_{max}]$, $j \in J$. We change the vector (ρ_j) from algorithm RN to the vector $\rho'_j = \rho_j + \alpha_j \tau_j$. The parameter $\alpha \geq 0$ defines the relative importance of statistical information. Now we present a general framework for algorithm AC.

Algorithm AC

1. Set $\tau_j := \tau_{min}$, $S_{AC} := J$, $t := 0$.
2. While $t < T_{max}$ do
   2.1. Construct solutions S_t, $t = 1, \dots, N_1$.
   2.2. Select the best solutions S_r, $r = 1, \dots, N_2$.
   2.3. Update the statistical information (τ_j) and counter $t := t + 1$.
   2.4. If $|S_{AC}| > \min_r |S_r|$ then update S_{AC}
   and increase the corresponding values of (τ_j).

Algorithm AC constructs N_1 feasible solutions on each iteration. Then the algorithm selects N_2 solutions with minimal cardinalities and updates the current vector (τ_j) by the following rule:

$$\tau_j := \beta\tau_j + \gamma\delta_j,\ j \in J.$$

Parameter $0 \le \beta \le 1$ is used to imitate becoming out of date. Value of δ_j is the frequency of the j-th column in the solutions S_r, $r = 1, \dots, N_2$. In Figure 2 we can see the influence of the statistical information. Algorithm AC finds an optimal solution for $p \in [0.3, 0.7]$ and the errors decrease for values of $p < 1$.

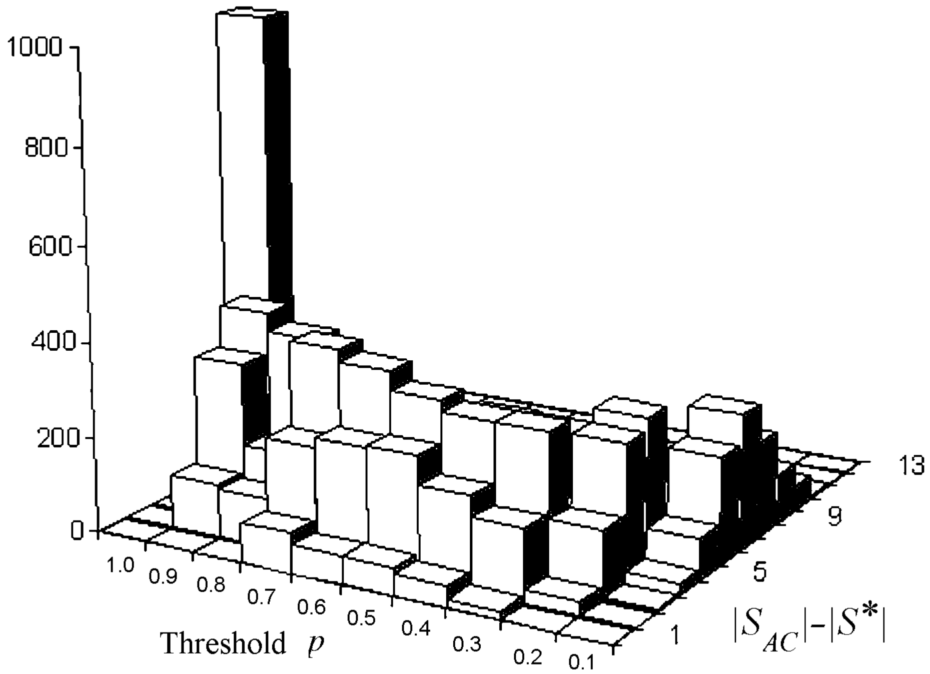

Figure 2: Influence of statistical information

We may run different randomized heuristics in step 2.1. It is easy to verify that if we use algorithms RN or RCS then probability $Pr(S_{AC} \neq S^*)$ converges to 0 at a geometrical rate as $T_{max} \to \infty$. This statement may be false for algorithm $GRASP$.

3 Experimental Analysis

We study the behavior of algorithm AC on four classes of test problems:

Class D5: $\mathbf{I} = \mathbf{J} = 100$. Density of (a_{ij}) is 5%.
Class D10: $\mathbf{I} = \mathbf{J} = 100$. Density of (a_{ij}) is 10%.
Class E5: $\mathbf{I} = \mathbf{J} = 100$. There are exactly five 1's for each row of (a_{ij}).
Class E3: $\mathbf{I} = 330, \mathbf{J} = 45$. There are exactly three 1's for each row of (a_{ij}).

All problems were randomly generated and solved exactly by the branch and bound method. Problems from the class E3 were generated using Steiner triples and are noted as computationally difficult benchmarks [7].

The performance of algorithm AC depends on the values of the control parameters. We investigate the influence of some of them. First of all, we study the importance of parameter β. If $\beta = 0$ we can not accumulate statistical information. If $\beta = 1$ we store all collected information and the values of τ_j increase monotonically. In the Table 1 we present computational results for $N_1 = 10$, $N_2 = 3$, $T_{max} = 50$. These results are averaged over 30 test problems from class E3. The column labeled by ε contains an average relative error $\varepsilon = 100\%(|S_{AC}| - |S^*|)/|S^*|$. The column labelled by K contains the average number of optimal solutions found. This number consists of two items $K = K_1 + K_2$, where K_1 is the average number of optimal solutions obtained during the first 25 iterations, and K_2 is the corresponding number for the last 25 iterations.

Table 1. Influence of the parameter β on performance of algorithm AC with GRASP heuristic, $\tau_{min} = 1$, $\tau_{max} = 15$, $\gamma = 0.2$

β	$\varepsilon(\%)$	K	K_1	K_2
0.00	6.9	0.4	0.3	0.1
0.50	5.4	1.4	1.3	0.1
0.80	3.1	29.8	11.7	18.1
0.85	2.6	46.5	13.8	32.7
0.90	2.8	59.3	17.6	41.7
1.00	3.3	3.8	2.1	1.7

The table shows that the performance of algorithm AC strongly depends on the value of the parameter β. For $\beta = 0$ the average error ε is equal to 6.9% and the probability to find an optimum is extremely small. Note that the optimal value for β is close to and less than 1. For $\beta = 0.9$ relative error is low $\varepsilon = 2.8\%$ and a positive influence of the statistical information is evident, $K_2 > K_1$.

The next aspect of our research is concerned with the effect of cooperation for a fixed number of trials $N_1 \times T_{max} = constant$. Figure 3 shows computational results

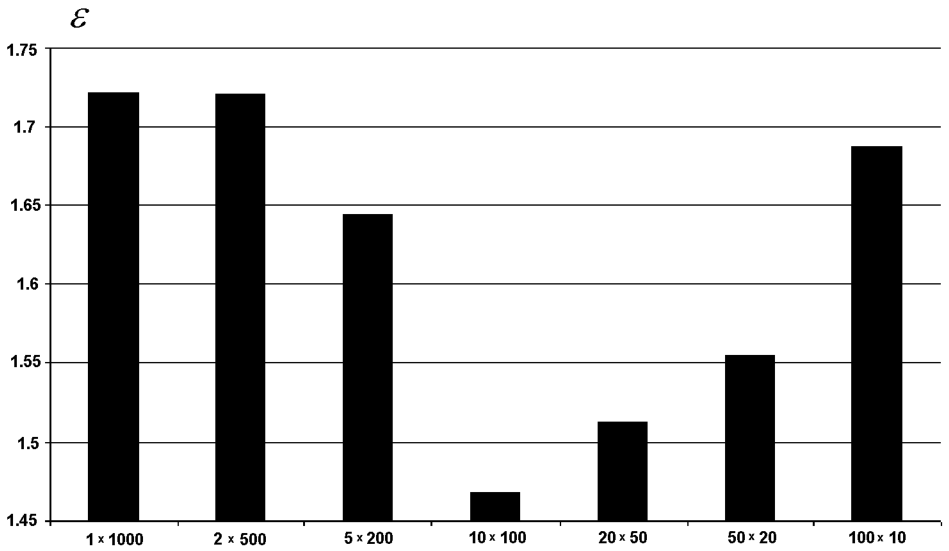

Figure 3: Average relative error as a function of $N_1 \times T_{max}$

for the class D5. If $N_1 = 1$, $T_{max} = 1000$, cooperation is high but the quality of the statistical information is low. If $N_1 = 100$, $T_{max} = 10$ then the situation is inverse. For $N_1 = 10$, $T_{max} = 100$ we get an optimal trade-off between both criteria.

Finally, we consider four criteria to evaluate the importance of statistical information. They are average relative error, standard deviation, percentage of finding optimal solutions, and percentage of approximation solutions with minimal absolute error. Computational results are shown in Table 2. If $\alpha = 0$ we get the probabilistic greedy algorithm. By comparing the columns for $\alpha = 0$ and $\alpha = 0.1$ we can see a positive influence of the statistical information. It is strongest for the class E3. This class is very difficult for algorithm AC. In this case using statistical information increases frequency of finding optimal solutions from 17% to 76%.

Table 2. Computational results

Benchmarks	D5		D10		E5		E3	
	$\alpha = 0$	$\alpha = 0.1$	$\alpha = 0$	$\alpha = 0.1$	$\alpha = 0$	$\alpha = 0.1$	$\alpha = 0$	$\alpha = 0.1$
Average error	2.66	1.49	3.39	2.47	2.91	2.29	5.57	2.2
Standard deviation	2.63	2.11	3.81	3.58	2.51	2.53	3.93	4.31
Percentage of S^*	44	66	56	68	43	55	17	76
Percentage of S^1	94	99	100	100	100	100	48	79

S^1 - approximation solution, $|S^1| = |S^*| + 1$.

4 Conclusion

Randomized greedy algorithms have an important advantage over deterministic greedy algorithms. We can run them several times and select the best solution. As a rule, this property allows us to prove asymptotic convergence of the best solution found towards the global optimum. In this paper we have studied the behavior of the Ant Colony algorithm. This approach improves the performance of randomized greedy algorithms due to using of the statistical information. We have investigated the influence of some important parameters and have found good values for them. The maximal effect of the Ant Colony approach we have obtained for the computationally difficult benchmarks.

References

[1] Balas, E.; Carrera, M. C. (1996): A Dynamic Subgradient-Based Branch and Bound Procedure for Set Covering. *Operations Research*, Vol. 44, pp. 875-890.

[2] Beasley, J. E.; Chu, P. C. (1996): A Genetic Algorithm for the Set Covering Problem. *European Journal of Operational Research*, Vol. 94, No. 2, pp. 394-404.

[3] Chvátal, V. (1979): A Greedy Heuristic for the Set Covering Problem. *Mathematics of Operations Research*, Vol. 4, No. 3, pp. 233-235.

[4] Dorigo, M.; Maniezzo, V.; Colorny, A. (1991): Ant System: An Autocatalytic Optimizing Process. Technical Report No. 91-016, Politecnico di Milano.

[5] Eremeev, A. V. (1999): Genetic Algorithm with a Non-Binary Representation for the Set Covering Problem. *Operations Research Proceedings 1998*, Springer-Verlag, pp. 175-181.

[6] Feo, T. A. (1995): Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, Vol. 6, pp. 109-133.

[7] Fulkerson, D. R.; Nemhauser, G. L.; Trotter Jr., L. E. (1974): Two Computationally Difficult Set Covering Problems that Arise in Computing the 1-Width of Incidence Matrices of Steiner Triple Systems. *Mathematical Programming Study*, Vol. 2, pp. 72-81.

[8] Grossman, T.; Wool, A. (1997): Computational Experience with Approximation Algorithms for the Set Covering Problem. *European Journal of Operational Research*, Vol. 101, No. 1, pp. 81-92.

[9] Zaozerskaya, L. A. (1998): Investigation and Solving of Some Classes of Integer Programming Problems on the Basis of Regular Partitions. PhD. Thesis, Omsk Branch of Sobolev Institute of Mathematics, SB RAS, (in Russian).