An Ant System for the Maximum Independent Set Problem

Guillermo Leguizamón
LIDIC - Departamento de Informática
Universidad Nacional de San Luis
San Luis, Argentina
legui@unsl.edu.ar

Zbigniew Michalewicz
University of North Carolina
Charlotte, NC 28223, USA
zbyszek@uncc.edu *and*
Institute of Computer Science
Polish Academy of Sciences

Martin Schütz
NuTech Solutions GmbH
Martin-Schmeisser-Weg 15
44227 Dortmund, Germany
martin.schuetz@nutechsolutions.de

Abstract

Early applications of Ant Colony Optimization (ACO) have been mainly concerned with solving ordering problems (e.g., the Traveling Salesperson Problem). More recently, promising results were obtained for solving the Multiple Knapsack Problem by introducing a modification of the standard Ant System algorithm. In this paper we extend our study on the applicability of the ACO approach to subset problems. The computational study involves its applicability for solving the Maximum Independent Set Problem (MISP). The set of instances tested were either randomly generated by specific methods or taken from the so-called DIMACS benchmark graphs. The reported results which are comparable with different state-of-the-art algorithms show the potential of the ACO approach for solving the MISP.

Keywords: *ant colony optimization, maximum independent set problem, combinatorial optimization, metaheuristics.*

1 Introduction

The Ant Colony Optimization (ACO) technique has emerged recently (Dorigo et al. [9, 11, 12]) as a new metaheuristic for hard combinatorial optimization problems. ACO algorithms, that is, instances of the ACO metaheuristics, are basically a multi-agent system where low level interactions between single agents (called artificial ants) result in a complex behavior of the whole system. ACO algorithms have been inspired by colonies of real ants [9], which deposit a chemical substance (called *pheromone*) on the ground. This substance influences the choices they make: the larger the amount of pheromone on a particular path, the larger the probability that an ant selects the path. Artificial ants, in ACO algorithms, behave in a similar way.

ACO algorithms can be directly applied to discrete optimization problems that can be characterized as a graph $G = (C, L)$, where C is a finite set of *components* and $L \subseteq C \times C$ the set of *connections* between the components (see [9] for a complete description). The solutions to the optimization problem can be expressed in terms of feasible paths on the graph G. Thus, ACO algorithms can be used to find minimum cost paths (sequences) feasible with respect to the constraints Ω[1].

In ACO algorithms a population (colony) of agents (ants) collectively solve the optimization problem under consideration by using the above graph representation. Information collected by the ants during the search process

[1]For example, in the traveling salesperson problem defined in Section 2, C is the set of cities, L is the set of arcs connecting cities, and Ω indicates that a particular solution is a Hamiltonian circuit.

is encoded in *pheromone trails* τ_{ij} associated with connection l_{ij}[2]. Pheromone trails encode a long-term memory about the whole ant search process. Depending on the problem representation chosen, pheromone trails can be associated with all arcs, or only to some of them. Arcs can also have an associated *heuristic value* η_{ij} representing *a priori* information about the problem instance definition or run-time information provided by a source different from the ants.

Early experiments with ACO algorithms were connected with ordering problems such as the Traveling Salesperson [11, 12] Problem, the Quadratic Assignment Problem [14], as well as the Job Shop Scheduling, Vehicle Routing, Graph Coloring and Telecommunication Network Problem [9]. More recently promising results were reported in [8, 17] from the application of a new version of an Ant System to the Multiple Knapsack Problem, an example of a non-ordering problem. This paper aims to go further in this direction in order to evaluate the feasibility of applying an ACO algorithm to a different subset problem according to the general concept behind an ACO heuristic. The ACO algorithm thereby takes into account that in subset problems there are no connections between the problem components.

The Maximum Independent Set Problem (MISP), the subset problem considered in this paper, is computationally intractable by its nature, or sufficiently large to preclude the use of exact algorithms. In such cases, for the MISP as well as other combinatorial optimization problems, heuristic methods are usually employed to find good, but not necessarily optimal solutions. The effectiveness of these methods depends upon their ability to adapt to a particular solution, avoid entrapment to local optima, and exploit the basic structure of the problem, such as a network or a natural ordering among its components or even a combination of those components. Various heuristic search techniques have been developed that have demonstrably improved their ability to obtain good solutions to difficult combinatorial optimization problems. Such techniques include simulated annealing, tabu search, greedy randomized adaptive search procedures (GRASP), evolutionary algorithms and more recently ant colony optimization.

With regards to the MISP, several algorithms have been developed in the last years by attacking this problem through different heuristic methods. Khuri et al. [2] developed a genetic algorithm for the MISP that used a graded penalty function which was applied to some small instances of the MISP. A recent work by Aggarawal et al. [1] proposes a genetic algorithm which is compared with state-of-the-art methods for the MISP problem. The instances tested in this work were taken from the Second DIMACS challenge [10]. In Resende et al. [20], GRASP is applied to a set of larger and difficult instances of MISP generated according to a method proposed by Bollobas [5] for building random graphs for which it is possible to know in advance the approximate cardinality of the maximum independent set. Friden et al. [15] applied a tabu search approach called STABULUS to the same types of random graphs tested in [20]. Besides these algorithms based on meta heuristic approaches, a number of algorithms have been proposed for the maximum clique problem which has a close relation to the MISP. Many of these algorithms use partially enumerative techniques or branch and bound methods as well as evolutionary algorithms which work well for many cases of this type of graphs. Some of them are proposed by Balas et al. [3], Gibbons et al. [16], Rossi [21], Bonze et al. [6], and Marchiori [18].

The rest of the paper is organized as follows. In section 2 we will illustrate the basic concepts of the original Ant System algorithm, the first ACO algorithm introduced by Dorigo, Maniezzo, and Colorni [13], using the Traveling Salesperson Problem (TSP) as an example. Further sections of this paper investigate the applicability of the ACO algorithm for solving subset problems and its applicability to the Maximum Independent Set Problem. The section on experiments contains an important number of instances having different characteristics and sizes. Finally, we discuss the behavior and the performance of the Ant System on the MISP with regards to other state-of-the-art algorithms and benchmarks.

2 Ant System for the TSP

Given a set C of n cities and a set of distances between them, the Traveling Salesperson Problem (TSP) is the problem of finding a minimum cost closed path (a *tour*), which visits every city exactly once (Hamiltonian

[2]Here we simplify notation by setting $l_{c_i c_j} = l_{ij}$, where $c_i, c_j \in C$.

circuit). Thus, we have to minimize

$$COST(i_1, \ldots, i_n) = \sum_{j=1}^{n-1} d(C_{i_j}, C_{i_{j+1}}) + d(C_{i_n}, C_{i_1}), \tag{1}$$

where $d(C_x, C_y)$ is the distance between the cities x and y.

Let $b_i(t)$ denote the number of ants in city i $(i = 1, ..., n)$ at time t and $a = \sum_{i=1}^{n} b_i(t)$ is the total number of ants. The variables $\tau_{ij}(t)$ denote the intensity of pheromone trail on connection (i, j) at time t and are defined as

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t), \tag{2}$$

where $0 < \rho \leq 1$ is a coefficient which represents pheromone evaporation[3]. $\Delta\tau_{ij}(t) = \sum_{k=1}^{a} \Delta\tau_{ij}^{k}(t)$, where $\Delta\tau_{ij}^{k}(t)$ is the quantity per unit of length of trail substance (pheromone in real ants) laid on connection (i, j) by the k-*th* ant at time t and is given by the following formula:

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{Q}{L_k} & \text{if ant k-}th\text{ uses edge (i,j) on its tour} \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

where Q is a constant and L_k is the tour length found by the k-*th* ant. For each edge, the intensity of trail at time 0 ($\tau_{ij}(0)$) is set to a very small value.

While building a tour, the probability that ant k in city i visits city j is

$$P_{ij}^{k}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{h \in allowed_k(t)}[\tau_{ih}(t)]^{\alpha}[\eta_{ih}]^{\beta}}, & \text{if } j \in allowed_k(t) \\ 0, & \text{otherwise,} \end{cases} \tag{4}$$

where $allowed_k(t)$ is the set of cities not visited by ant k at time t, and η_{ij} represents a local heuristic. For the TSP, this heuristic (called 'visibility') is $\eta_{ij} = \frac{1}{d(C_i, C_j)}$.

The parameters α and β control the relative importance of pheromone trail versus visibility. Hence, the transition probability is a trade-off between visibility, which says that closer cities should be chosen with a higher probability, and trail intensity associated to connection (i, j) is intended to represent the learned desirability of choosing city j when in city i.

A data structure, called a *tabu list*, is associated to each ant in order to avoid that ants visit a city more than once. This list $tabu_k(t)$ maintains a set of visited cities up to time t by the k-th ant. Therefore, the set $allowed_k(t)$ can be defined as follows: $allowed_k(t) = C - tabu_k(t)$. When a tour is completed, the $tabu_k(t)$ list $(k = 1, \ldots, a)$ is emptied and every ant is free again to choose an alternative tour for the next cycle.

The above definitions allow us to describe the Ant System algorithm (Figure 1) characterizing the ants behavior in the following way: they build a solution in an incremental way using a stochastic decision rule (in the repeat-until loop) starting from a randomly chosen city. When all ants have completed a solution, the pheromone is deposited on the connections.

3 Ant System for Subset Problems

The subset-based[4] and permutation-based Ant Systems have many features in common. However, in the permutation-based Ant System the pheromone is laid on *paths* while for subset problems no path exists connecting the items. A subset-based Ant System takes advantage of one of the central ideas involved in the selection process of a permutation-based ant system: "the more amount of pheromone on a particular *path*, the more profitable is

[3] There exist other approaches for pheromone update trail. See [9].

[4] Given a combinatorial problem P which includes a set of components $S = \{1, ..., n\}$, , $n \in \mathbb{N}$. We say that P is a subset problem if a solution for P can be obtained by combining elements of S, that is, the solution is an element of 2^S. See also section 4 for a special subset problem.

1. initialize
2. **for** $t = 1$ **to** number of cycles **do**
3. **for** $k = 1$ **to** a **do**
4. **repeat** until $allowed_k$ is empty
5. select city j to be incorporated with probability P_{ij}^k given by Eq.(4)
6. **end**
7. calculate L_k, the cost of the generated solution
8. save the best solution so far
9. **end**
10. **update** the trail levels τ_{ij} on all paths according to Eq.(2)
11. **end**
12. print the best solution found

Figure 1: General outline of an Ant System.

that *path*". This idea was adapted here in the following way: "the more pheromone on a particular *item*, the more profitable that *item* is." In other words, we move the pheromone from paths to items. At the same time, a local heuristic is also used in the new version, but now it considers *items* only instead of *connections* between them. Let us explore these similarities and differences in more detail. Out of a set S of n items we have to select the best subset of s items, possibly satisfying some additional constraints. There is no concept of a path here, so it is not clear how to apply the concepts described in the previous section directly to subset problems.

The main difference is the following: In ordering problems, the sequence $\tilde{Sq} =< i_1, i_2, ..., i_j >$ and the set $R = \{i_{j+1}, i_{j+2}, ..., i_n\}$ represent a partial solution of the problem and the set of remaining items to be considered in order to complete the ordering of n items from the set S, respectively. The selection process of the next item from the set R involves probabilities $P_{i_j i_p}^k(t)$ (e.g., Eq. (4)) ($p \in \{j+1, j+2, ..., n\}$), which depend on the amount of pheromone $\tau_{i_j i_p}$ on the edge (i_j, i_p) and the local heuristic measure $\eta_{i_j i_p}$ (Figure 2).

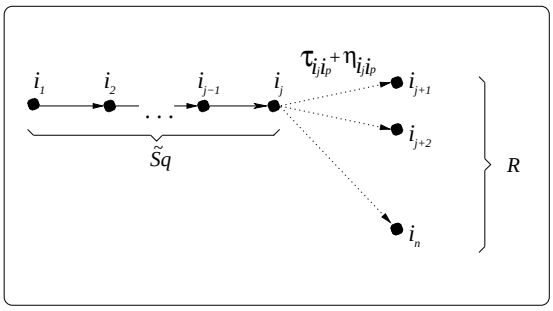

Figure 2: A sequence representing a partial solution $\tilde{Sq}$ at step j during a particular cycle.

On the other hand, in subset problems we are not interested in solutions giving a particular order (e.g., a tour in the TSP). Therefore, a partial solution is represented by the set $\tilde{S} = \{i_1, i_2, ..., i_j\}$ and the most recent element incorporated in $\tilde{S}$, that is i_j, need not be involved in the process for selecting the next element (Figure 3).

Thus, the original Ant System must be modified accordingly. First of all, the pheromone trail is now laid on each element from set S, with the intended meaning that elements with a higher pheromone level are more profitable. Therefore, the *intensity of pheromone trail* on item i at time $t+1$ is given by:

$$\tau_i(t+1) = (1-\rho)\tau_i(t) + \Delta\tau_i(t), \tag{5}$$

where $\Delta\tau_i(t) = \sum_{i=1}^{a} \Delta\tau_i^k(t)$, i.e. the accumulated amount of the contribution $\Delta\tau_i^k(t)$ from each ant in the system. In other words, that contribution is the quantity of pheromone trail laid on item i by the k-th ant at time

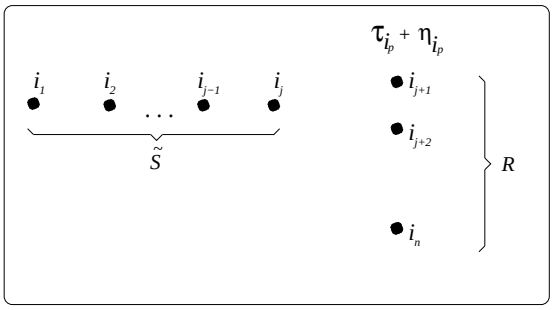

Figure 3: A set representing a partial solution $\tilde{S}$ at step j during a particular cycle.

t. This quantity is given by the following formula:

$$\Delta\tau_i^k(t) = \begin{cases} G(L_k), & \text{if } k-\text{th ant incorporates item i} \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

In Equation 6 the function G depends on the problem and gives the amount of pheromone added to item i. Usually, $G(L_k) = Q/L_k$ or $G(L_k) = QL_k$ for minimization or maximization problems respectively (Q is a constant). L_k is the cost obtained by the k-th ant. Further, the local heuristic should assign a value to each element without any considerations about possible connections between them (ordering is not important any longer).

For the subset problem studied in this paper, our version of the Ant System considers a special type of heuristic which takes into account both, problem knowledge and the partial solution being built by a particular ant k. Thus, we define the heuristic value for the item i as a function (see Eq. 7) of the partial solution $\tilde{S}_k(t)$ at time t where $i \in S - \tilde{S}_k(t)$.

Then, for a partial solution $\tilde{S}_k(t) = \{i_1, ..., i_j\}$ being built by the k-th ant, the probability $P_{i_p}^k(t)$ of selecting i_p as the next item ($p \in \{j+1, j+2, ..., n\}$) is given as

$$P_{i_p}^k(t) = \begin{cases} \frac{[\tau_{i_p}(t)]^\alpha [\eta_{i_p}(\tilde{S}_k(t))]^\beta}{\sum_{j \in allowed_k(t)} [\tau_j(t)]^\alpha [\eta_j(\tilde{S}_k(t))]^\beta}, & \text{if } i_p \in allowed_k(t) \\ 0, & \text{otherwise,} \end{cases} \tag{7}$$

where $allowed_k(t) \subseteq S - \tilde{S}_k(t)$ is the set of remaining feasible items, $\tau_i(t)$ is the amount of pheromone on item i, and $\eta_{i_p}(\tilde{S}_k(t))$ represents the heuristic value for item i based on the solution being built by the k-th ant. Thus, the higher the value of τ_{i_p} and/or $\eta_{i_p}(\tilde{S}_k(t))$, the more profitable it is to include item i_p in the partial solution. Therefore, a general version of an Ant System for subset problems can be obtained from the general algorithm in Figure 1 by changing the lines 5 and 10 as follows:

5. select item i to be incorporated with probability P_i^k given by Eq.(7)
10. **update** the trail levels τ_i on all items according to Eq.(5)

4 Formulation of an Ant System for the MISP

The maximum independent set problem consists of finding the largest subset of vertices of a graph such that none of these vertices are connected by an edge (i.e., all vertices are independent of each other). Thus, if $G = (V, E)$ denotes a graph where V is the set of nodes and E the set of edges, the problem is to determine a set $V^* \subseteq V$ such that $\forall i, j \in V^*$ the edge $\langle i, j \rangle \notin E$ and $|V^*|$ is maximum.

In this section we provide implementational details of an Ant System applied to the MISP. Taking into account the implementation for the Multiple Knapsack Problem in [17], it is important to emphasize the robustness of the

Ant System regarding a particular subset problem. The only variable component is the knowledge about the problem, i.e., the *local heuristic* involved in the probability of item selection.

For $V = \{1, ..., n\}$ the Ant System tries to find the maximal independent set $V^* \subseteq V$Let $F_k(t)$ be the set of remaining feasible items with respect to $\tilde{S}_k(t)$: the solution being built by ant k at time t. The *local heuristic* for the MISP can be defined as:

$$\eta_i(\tilde{S}_k(t)) = |F_i|, \tag{8}$$

where F_i represents $F_k(t+1)$ in case item i is added to $\tilde{S}_k$. Then the local heuristic aims at assigning higher scores to that item (say i) which yields a larger F_i. Thus, the larger $F_k(t+1)$, the larger the set of remaining items for completing $\tilde{S}_k$. The probability for item selection is given by Eq.(7) where $allowed_k = V - \tilde{S}_k(t) - U_k(t)$ and $U_k(t) = \{j|((j,i) \in E \ \vee \ (i,j) \in E) \wedge i \in \tilde{S}_k(t)\}$, i.e., the set of infeasible items with respect to $\tilde{S}_k(t)$. Function G is defined as $G(L_k) = QL_k$ where $Q = 1/|V|$ and L_k, the objective value, is the cardinality of the set of vertices conforming the solution obtained by the ant k.

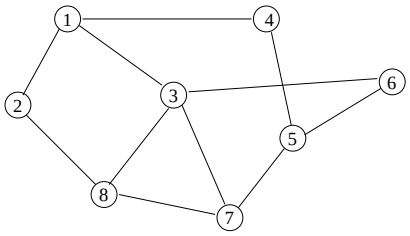

Figure 4: Instance of MISP

Let us consider the following example concerning the heuristic defined above. Figure 4 shows a small MISP instance where $|V| = 8$. Let us suppose that in time t the partial solution being built by the k-th ant is $\tilde{S}_k(t) = \{2\}$, then $F_k(t) = V - \{2\} - \{1, 8\} = \{3, 7, 4, 5, 6\}$. In the last expression, the set $\{1, 8\}$ represents the subset of infeasible items due to the inclusion of item 2 in the partial solution. Now the subset $\{3, 7, 4, 5, 6\}$ is the set of current feasible items and the corresponding heuristic values are as follows:

$$\tau_3(S_k(t)) = |F_3| = |\{4, 5\}| = 2 \quad \tau_4(S_k(t)) = |F_4| = |\{3, 6, 7\}| = 3 \quad \tau_5(S_k(t)) = |F_5| = |\{3\}| = 1$$
$$\tau_6(S_k(t)) = |F_6| = |\{4, 7\}| = 2 \quad \tau_7(S_k(t)) = |F_7| = |\{4, 6\}| = 2$$

Therefore, the highest score is obtained by item $i = 4$ possessing the biggest set of feasible items for the next selection step according to Eq. (7).

5 Experiments and Results

The Ant System was coded in C. All runs were performed on a SunW, Ultra-1 with 140MHz and 128M RAM. Although there exists a parallel version of the Ant System, we show the results for the serial version called AS-MISP. Suitable parameter settings for AS-MISP were defined in some preliminary experiments. We choose $\alpha = 1$, $\beta = 1$ and $\rho = 0.5$. It is important to note that these parameter settings are similar to those found in earlier studies of Dorigo [9], where $\alpha = 1$, $\beta = 5$ and $\rho = 0.5$, however, with $\beta = 5$, AS-MISP suffered of premature convergence, i.e., the algorithm got stuck in a local optimum on many of the considered test cases. On the other hand, the number of ants and the number of cycles was set to 10 and 200 respectively. Additionally, we show at the end of this section that for a set of instances the setting $\beta = 0$ improves the performance of AS-MISP. In this case the search is based strictly on blind cooperation using the trail information. The vector representing the trail substance was initialized randomly with $\tau_i \in (0, 1)$, $i \in \{1, \ldots, n\}$ for all experiments.

Three groups of instances were considered for this problem. The first group was generated according to two different methods as used by Khuri et al. [2]. One method consists in the following algorithm in order to generate

randomly select $V^* = \{i_1, ..., i_m\} \subseteq V = \{1, ..., n\}$
for i=1 **to** n **do**
  for j=i+1 **to** n **do**
    if $((Rnd(0,1) < d)$ **and** $((i \notin V^*)$ **or** $(j \notin V^*)))$[5]
    then $e_{ij} = 1$
    else $e_{ij} = 0$
  done
done

Figure 5: Algorithm for the generation of random graphs which preselects an independent set of size m.

graphs having n nodes, a density d ($d \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ in our experiments) and a known maximum independent set m. We denote the instances generated in this way by *Mn-d-m*.

The other method builds a *scalable* graph (see Figure 6), which can be constructed for an even number of nodes n ($n > 6$). If n is a multiple of 4, two equivalent global maxima of value $|V^*| = n/2$ are obtained by partitioning the set of vertices into those of even (respectively odd) node numbers. Otherwise, the unique global maximum is given by $V^* = \{1, 3, ..., n/2, n/2+1, ..., n\}$, with $|V^*| = n/2+1$. Accordingly, this type of graphs is denoted by *scal-n*.

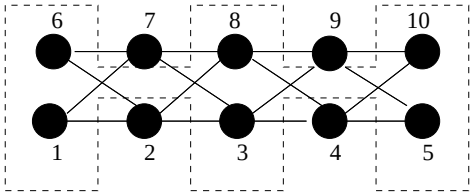

Figure 6: Scalable graph with $n = 10$ nodes where $V^* = \{1, 3, 5, 6, 8, 10\}$ is indicated by the dashed lines.

The second group of MISP instances was generated following the process described in [5, 20, 19]: Select each $e_{ij} = 1$ if and only if `Rnd(0,1)` $\leq p$, where p is the probability for the existence of an edge connecting the nodes i and j. This group of instances corresponds to the family of undirected random graphs $G_{|V|,p}$ where $n = |V|$. Afterwards in this section we analyze in detail the expected number of independent sets that can be found in this family of graphs.

Finally, the third group of instances was obtained from the DIMACS [10] directories. Actually, this group of instances corresponds to instances of the Max-Clique problem, but by a reduction process[6], it is possible to obtain instances of MISP for which the size of the maximum independent set is the same as the max-clique of the reverse graph $\bar{G}$. Therefore, the results obtained by applying some heuristic can be compared with the best results for the original instances for the Max-Clique problem [1].

The results for the three groups of instances are shown in tables in which the data are organized as follows: The set of columns for the AS-MISP displays the values obtained by the Ant System with $(\alpha = 1, \beta = 1)$. Although all tables offered some differences, a set of column labels exists that are used with the same meaning for each instance. Special columns will be described with the respective experiment. Thus, BF represents the maximum independent set found; avg(BF) is the respective average out of 10 runs; #hits represents the number of successful results. On the other hand, we show two more columns, *Cycles to Best* which displays the minimum number of cycles (Min) to obtain the best result, with its respective average value (Avg) and similarly *Time to*

[6] If $G = \langle E, V \rangle$ is an acyclic graph, and $N \leq |V|$ is the maximum clique for G, then N represents the cardinality of the maximum independent set for the reverse graph $\bar{G} = \langle \bar{E}, V \rangle$.

Best displays the minimum elapsed time[7] (Min) to get the best result and its respective average value (Avg).

Instance	AS-MISP							GA		
	BF	avg(BF)	#hits	Cycles to Best		Time to Best		BF	avg(BF)	#hits
				Min	Avg	Min	Avg			
M100-0.1-45	46	46.00	100	5	9	0.14	0.30	47	37.39	1
M100-0.2-45	45	45.00	100	1	2	≈0	0.05	45	37.25	34
M100-0.3-45	45	45.00	100	1	1	≈0	≈0	45	41.38	77
M100-0.4-45	45	45.00	100	1	1	≈0	≈0	45	44.20	96
M100-0.5-45	45	45.00	100	1	1	≈0	≈0	45	44.72	99
M200-0.1-90	90	90.00	100	2	3	0.18	0.39	90	68.75	4
M200-0.2-90	90	90.00	100	1	2	≈0	0.20	90	81.05	54
M200-0.3-90	90	90.00	100	1	2	≈0	0.11	90	88.22	93
M200-0.4-90	90	90.00	100	1	2	≈0	0.03	90	90.00	100
M200-0.5-90	90	90.00	100	1	1	≈0	≈0	90	90.00	100
M300-0.1-135	135	135.00	100	1	2	1.70	0.91	NA	NA	NA
M300-0.2-135	135	135.00	100	1	2	≈0	0.61	NA	NA	NA
M300-0.3-135	135	135.00	100	1	2	≈0	0.55	NA	NA	NA
M300-0.4-135	135	135.00	100	1	2	≈0	0.36	NA	NA	NA
M300-0.5-135	135	135.00	100	1	1	≈0	0.12	NA	NA	NA
scal-100	50	50.00	100	4	28	0.19	0.87	NA	NA	NA
scal-200	100	98.10	74	6	65	0.50	9.89	NA	NA	NA
scal-300	150	143.70	27	16	120	12.81	40.61	NA	NA	NA
scal-102	52	50	49	1	25	≈0	1.06	50	44.94	0
scal-202	102	99.1	13	7	70	0.98	9.60	96	88.90	0
scal-302	152	147.5	15	26	87	8.40	32.10	NA	NA	NA

Table 1: Results of AS-MISP compared with results from Khuri et al. [2] on different graphs (First group of instances of MISP).

Table 1 displays the results obtained for the instances of the MISP corresponding to the first group. In addition, the last column GA shows the results reported by Khuri et al. in [2] which include the best found (BF), the corresponding average (avg(BF)), and the number of hits (#hits). The entry NA stands for *not available*. For each instance of this first group we run AS-MISP 100 times in order to be comparable with the results from Khuri. The results in table 1 indicate that the AS-MISP performed very well for all instances of type *Mn-d-m* for which the optimal solution was found in the very early cycles of the run. Therefore, no cooperation took place and using only the initial value for the trail (τ) and the heuristic information was enough for solving these instances. Also, it is worth comparing these results with those results reported by Khuri et al. [2] for the same type of instances. Besides *M100-0.1-45* AS-MISP offered in all instances a better performance than the GA. So, in contrast to the GA, AS-MISP found the defined local maximum in all runs. Only in case of *M100-0.1-45* the GA (BF = 47) outperformed the best result of AS-MISP (BF = 46) [8] in one run. In our opinion this does not build a contradiction to the overall positive performance of the ACO algorithm, which in any case found a subset of size 46, while the GA found values better than 46 only in two out of 100 runs. For the instances of size 200 and 300 AS-MISP showed a similar performance as for the test cases of size 100. Therefore, an increment on the number of nodes did not decrease its performance on this type of instance. Particularly for the instances of size 200 the ANT-MISP found results superior to the GA regarding the number of hits out of the 100 runs.

For *scal-n* instances, AS-MISP increases the number of cycles in order to get the best solution — i.e., the ACO algorithm converged slower than before (instances *Mn-d-m*) implying that an interaction between the ants took place during the experiment. However, the *scal-n* instances represent more difficult test cases since the average performance of AS-MISP is a little bit inferior with respect to its performance on instances of type *Mn-d-m*. Nevertheless, it is still able to find the optimal values and a good overall performance as indicated by BF, #hits and avg(BF) respectively.

[7] The time measure, here in seconds, is only used to give an impression of the order of magnitude of the running time and not useful for any type of comparisons.

[8] According to the design of the *Mn-d-m* graphs m is a known optimum, but not necessarily the global one.

For the second group we considered a set of instances from the family of graphs $G_{|V|,p}$ [5] with different sizes ($n \in \{200, 400, 600\}$) and probabilities ($p \in \{0.2, 0.5, 0.6, 0.85, 0.9\}$) – i.e., for each size we considered different degrees of density. For this type of random graphs we can calculate the expected number of independent sets of size $1 \leq k \leq n$ [5]. Let X_k be a stochastic variable denoting the number of independent sets of size k of a particular instance of a random graph. Table 2 shows the expected values around the size of the maximum independent sets we can find in each one of them. Thus, for a graph with $|V| = 200$ and $p = 0.2$ we can determine that the expected value for $E(X_{25}) = 382$ and $E(X_{26}) = 9$. This means that we expect to find around 382 and 9 independent sets of size 25 and 26 respectively. For the same example, the expected value for X_k with $k = 27$ is near 0 and, on the other hand for $k \leq 24$, the expected value increases significantly, that is, independent sets of size 27 are very rare, but still they could exist and independent sets of size 24 or less are abundant. A similar situation may be observed for the other combination of $|V|$ and p. Therefore, we expect that any algorithm that performs well on this kind of instances should obtain results as close as possible to the larger k for each combination of $|V|$ and p displayed in Table 2.

$\|V\|$	Probability value p														
	0.2			0.5			0.6			0.83			0.9		
200	X_{26}	X_{25}	X_{24}	X_{11}	X_{10}	X_9	X_9	X_8	X_7	X_6	X_5	X_4	X_5	X_4	X_3
	10	382	$> 10^4$	11	638	$> 10^4$	6	397	$> 10^4$	≈0	51	1561	≈ 0	65	1313
400	X_{31}	X_{30}	X_{29}	X_{13}	X_{12}	X_{11}	X_{11}	X_{10}	X_9	X_7	X_6	X_5	X_5	X_4	X_3
	15	995	$> 5 \cdot 10^4$	3	402	$> 2 \cdot 10^4$	≈0	32	3116	≈0	16	1678	8	1051	$> 10^4$
600	X_{34}	X_{33}	X_{32}	X_{14}	X_{13}	X_{12}	X_{11}	X_{10}	X_9	X_7	X_6	X_5	X_6	X_5	X_4
	16	1522	$> 10^5$	3	609	$> 5 \cdot 10^4$	11	1913	$> 10^5$	≈0.5	181	$> 10^4$	≈0	64	5346

Table 2: Some expected values for X_k variables in random graphs.

We denote the instances of this group as *Mn-p*, where n is the number of nodes and p the probability used in the generation process. The total number of instances tested was 75 according to the values of $n \in \{200, 400, 600\}$ and $p \in \{0.2, 0.5, 0.6, 0.83, 0.9\}$[9] provided that for each combination of n and p we generated 5 random graphs by using different seeds. Table 3 shows the results for the randomly generated graphs. The second column, $N_{(k,r)}$, displays for each instance four pairs (k, r) where each pair indicates that AS-MISP found an independent set of size k, r-times. The value of k at the leftmost pair represents the size of the rarest independent set. In the subsequent columns, k is decremented by 1 until the value $k - 3$ — i.e., the size of more abundant independent sets.

For this group, the best performance of AS-MISP is reached for all graphs for which $p \in \{0.5, 0.6, 0.83, 0.9\}$. It is important to note that beside the graph size, considering the above values for p, the AS-MISP found solutions representing the rarest independent set or very close to them. On the other hand, the graphs generated with a probability $p = 0.2$ were the hardest, independent of the size of the graph. For example, for the instances of size 200, 400 and 600, the performance is still acceptable but we observe (Table 3) that the independent sets found are biased to the more abundant ones. Thus, for test cases *M200-0.2* the AS-MISP found one solution of size 26, 29 of size 25, 18 of size 29 and so on. A similar situation can be found for *M400-0.2* and *M600-0.2*. Additionally, the difference concerning the value in *Cycles to Best* and *Time to Best* columns between graphs generated with $p = 0.2$ and those generated with $p \in \{0.5, 0.6, 0.83, 0.9\}$ is remarkable. As mentioned before these types of instances were tested by Resende et al. [20, 19] where they applied GRASP to the following type of instances: *M400-0.6*, *M400-0.83*, *M600-0.83*, *M1000-0.2*, *M1000-0.5*, *M1000-0.83*, *M2000-0.5* and *M3500-0.5*. The results reported are comparable with the results obtained by the AS-MISP, sizes 400 and 600. For the larger graphs, we run the AS-MISP algorithm for graphs *M1000-0.2*, *M1000-0.5*, *M1000-0.83*, *M2000-0.5* and *M3500-0.5*. Again, we found that the hardest test case was represented by a low density graph (*M1000-0.2*) for which the AS-MISP never reached the values 38 or 37 whereas GRASP did. The values found for this instance were $\{(36, 5), (35, 14), (34, 24), (33, 7)\}$[10]. For the remaining instances the AS-MISP behaved in a similar way

[9] The same values were used in [20, 19].

[10] This notation has the same meaning as in Table 3.

as for the other medium or high density graphs tested.

The corresponding results for the third group are shown in Table 4 where two additional columns are included, OCH and CBH. Column OCH (Optimized Crossover Heuristic) corresponds to the best results reported by Aggarawal et al. [1] by applying a genetic algorithm to a set of instances including those in Table 4. The other column, CBH stands for Continuous Based Heuristic which is a deterministic heuristic for the independent set problem (Gibbons et al. [16]). CBH was used for comparison in the experimental study where OCH was proposed [1]. Concerning the performance of AS-MISP, we should consider dividing this group of instances in two subgroups. The first one includes the instances for which AS-MISP performed successfully. These instances include the type *c-fat*, *johnson* and *p-hat.* It is worth mentioning that for these instances OCH and CBH also performed optimally in nearly all cases. For example, AS-MISP outperformed OCH and CBH on instance *p-hat-500-3*; CBH on *p-hat-700-3*; and finally, OCH on *p-hat-500-2* and *p-hat-700-1.* In this subgroup we can include the instance *keller4* for which AS-MISP and OCH solved optimally.

The second subgroup deserves a particular consideration. There are some instances which were not solved optimally by the AS-MISP, but they were by OCH (*san200_0.7_1*, *san400_0.7_1*, *san400_0.7_2*). However, for the above instances AS-MISP performed better than CBH. On the other hand, some instances exist for which no optimal solution was found by applying either AS-MISP, OCH or CBH. Nevertheless, for these instances (*brock400_2*, *brock400_4* and *brock800**) AS-MISP outperformed OCH. Additionally, for the instances *sanr400_0.5*, *sanr400_0.7*, *brock400_1* and,*brock400_3* ANT-MISP performed better than OCH and CBH respectively. An exception is *brock200_2* were AS-MISP and CBH found the same value. For test case MANN_a27, AS-MISP performed in between OCH and CBH, whereas for MANN_a45, AS-MISP performed worse than OCH and CBH.

Instance	AS-MISP							
	$N_{(k,r)}$				Cycles to Best		Time to Best	
					Min	Avg	Min	Avg
M200-0.2	(26,1)	(25,29)	(24,18)	(23,2)	0.88	11.48	5	51.8
M200-0.5	(11,39)	(10,11)	(9,0)	(8,0)	≈0	5.47	1.8	28.46
M200-0.6	(9,49)	(8,1)	(7,0)	(6,0)	≈0	4.68	1.2	24.58
M200-0.83	(5,50)	(4,0)	(3,0)	(2,0)	≈0	0.31	1	1.79
M200-0.9	(5,10)	(4,40)	(3,0)	(2,0)	≈0	.31	1	18.08
M400-0.2	(31,0)	(30,6)	(29,29)	(28,15)	33	89	9.67	25.03
M400-0.5	(13,1)	(12,39)	(11,10)	(10,0)	8	41	1.9	15.8
M400-0.6	(11,2)	(10,21)	(9,27)	(8,0)	2	32	0.5	7.5
M400-0.83	(7,0)	(6,47)	(5,3)	(4,0)	1	17	≈0	4.83
M400-0.9	(5,40)	(4,10)	(3,0)	(2,0)	1	25	≈0	7.82
M600-0.2	(34,1)	(33,5)	(32,22)	(31,14)	44	96	22.34	63.02
M600-0.5	(14,0)	(13,27)	(12,23)	(11,0)	7	68	2.5	34.01
M600-0.6	(11,9)	(10,41)	(9,0)	(8,0)	5	29	1.89	17.21
M600-0.83	(7,1)	(6,49)	(5,0)	(4,0)	2	7	≈0	2.73
M600-0.9	(6,6)	(5,44)	(4,0)	(3,0)	2	5	≈0	1.35

Table 3: Results from AS-MISP applied to different random graphs (Second group of instances of MISP).

Similarly to the instances of the first group (*Mn-p-m*) we observed an analogous situation concerning the convergence of the AS-MISP. See instances *c-fat**, *johnson**, *san400** and, *MANN**. However this premature convergence does not affect the performance of the AS-MISP for instances *c-fat** and *johnson** which seem to be rather easy problems if we consider the reported results for this instances from CBH, OHC and many others algorithms. Unfortunately, for some of the *san400** and *MANN_a45* the AS-MISP got stuck in a local optimum.

It is important to note the similarities, as was noted by Bonabeau et al. [7], between ACO algorithms and a class of evolutionary algorithms which incorporate a population-based incremental learning (PBIL). See [4] for further information. More precisely, in the Ant System for subset problems we have a direct connection between the generation vector as called in PBIL, and the trail information which also is represented by a real vector. However, an Ant System, besides the trail, uses heuristic information to direct the search while PBIL does not.

Instance	Size	Best Value	AS-MISP							OCH	CBH
						Cycles to best		Time to Best			
			BF	avg(BF)	#hits	Min	Avg	Min	Avg		
c-fat-200-1	200	12	12	12	10	1	1	≈0	≈0	12	12
c-fat-200-2	200	24	24	24	10	1	1	≈0	0.72	24	24
c-fat-200-5	200	58	58	58	10	1	1	≈0	≈0	58	58
c-fat-500-1	500	14	14	14	10	1	1	≈0	≈0	14	14
c-fat-500-10	500	126	126	126	10	1	1	≈0	≈0	126	126
c-fat-500-2	500	26	26	26	10	1	1	≈0	≈0	26	26
c-fat-500-5	500	64	64	64	10	1	1	≈0	≈0	64	64
johnson16-2-4	120	8	8	8	10	1	1	≈0	≈0	8	8
johnson32-2-4	196	16	16	16	10	1	1	≈0	≈0	16	16
johnson8-2-4	28	4	4	4	10	1	1	≈0	≈0	4	4
johnson8-4-4	70	14	14	14	10	1	1	≈0	≈0	14	14
p-hat-300-1	300	8	8	8	10	3	7	0.85	2.97	8	8
p-hat-300-2	300	25	25	25	10	5	9	2.10	5.88	25	25
p-hat-300-3	300	36	36	36	10	15	43	9.82	27.72	36	36
p-hat-500-1	500	9	9	9	10	6	21	5.68	23.15	9	9
p-hat-500-2	500	36	36	36	10	8	19	13.12	34.27	35	36
p-hat-500-3	500	≥ 50	50	49.6	6	17	58	29.52	105.92	49	49
p-hat-700-1	700	11	11	10.3	5	4	72	6.47	155.15	9	11
p-hat-700-2	700	44	44	44	10	11	42	38.79	164.10	44	44
p-hat-700-3	700	≥ 62	62	60.7	4	14	62	49.66	235.85	62	60
keller4	171	11	11	11	10	2	7	0.14	0.81	11	10
keller5	776	27	26	23.79	0	8	95	17.92	240.65	25	21
san200_0.7_1	200	30	24	18.6	0	1	60	0.25	15.20	30	15
san200_0.7_2	200	18	15	15	1	11	65	2.09	12.5	15	12
san200_0.9_1	200	70	70	51.9	1	5	79	1.7	26.89	70	46
san200_0.9_2	200	60	60	50.59	5	2	76	2.34	20.48	60	36
san200_0.9_3	200	44	44	38.29	2	13	64	2.94	15.71	36	30
san400_0.5_1	400	13	13	8.7	1	1	25	≈0	21.33	13	8
san400_0.7_1	400	40	22	22	0	1	3	≈0	3.39	40	20
san400_0.7_2	400	30	18	17.19	0	1	3	≈0	2.0	30	15
san400_0.7_3	400	22	16	15.5	0	1	1	≈0	21.85	16	14
san400_0.9_1	400	100	100	63	2	1	23	≈0	26.81	100	50
sanr200_0.7	200	18	18	17.8	8	4	83	0.66	17.33	18	18
sanr200_0.9	200	≥ 42	42	41.2	2	11	61	2.91	16.14	42	41
sanr400_0.5	400	13	13	12.4	4	7	75	5.18	55.54	12	12
sanr400_0.7	400	≥ 21	21	20.2	3	17	91	11.65	63.28	20	20
brock200_1	200	21	21	20.1	4	5	27	1.08	6	21	20
brock200_2	200	12	12	10.7	6	1	4	≈0	3.72	11	12
brock200_3	200	15	14	13.1	0	1	47	≈0	46.29	14	14
brock200_4	200	17	16	16	1	10	45	2.96	8.33	16	16
brock400_1	400	25	25	23.9	0	7	56	4.55	36.2	24	23
brock400_2	400	29	25	23.9	4	17	101	13.01	79.13	24	24
brock400_3	400	25	25	23.9	0	37	87	28.08	67.29	24	23
brock400_4	400	33	26	24	6	28	56	20.29	40.46	24	24
brock800_1	800	21	20	19.2	20	15	59	16.24	68.36	19	20
brock800_2	800	21	20	19.5	0	12	61	12.87	70.47	19	19
brock800_3	800	21	20	18.86	0	21	68	23.5	78.72	19	20
brock800_4	800	21	20	18.9	0	18	56	19.88	64.08	19	19
MANN_a27	378	126	123	122.19	0	1	18	≈ 0	17.32	126	121
MANN_a45	1035	345	335	334.29	0	1	1	≈0	≈ 0	343	336

Table 4: Results from AS applied to different graphs (Third group of instances of MISP). Additional results are showed in columns OCH and CBH.

But for some instances (highlighted in boldface - Table 4) from the three groups of all instances tested, AS-MISP improved its performance without using any heuristic value, i.e., parameter $\beta = 0$. For these test cases the

AS-MISP got stuck in a local optimum perhaps due to the greedy component used in the item selection process. However, by setting $\beta = 0$ the algorithm only uses the trail information to direct the search. Consequently AS-MISP (see Table 5) could escape from the local optimum and get the best solution (*san200_0.7_1, san400_0.7_1, brock200_3 and brock200_4*), improve the results (*MANN_a27 and MANN_a45*) or increase the number of hits (*san200_0.9_1*). However, the best overall performance of the AS-MISP was achieved with $\alpha = 1$ and $\beta = 1$ — i.e., a fair trade-off between the importance of the trail and the heuristic value.

Instance	Size	Best Value	AS-MISP							OCH	CBH
						Cycles to best		Time to Best			
			BF	avg(BF)	#hits	Min	Avg	Min	Avg		
san200_0.7_1	200	30	30	29.6	9	25	102	5.16	12.80	30	15
san200_0.9_1	200	70	70	60.9	6	13	98	2.82	23.27	70	46
san400_0.7_1	400	40	40	28.6	2	35	116	12.9	115.01	40	20
brock200_3	200	15	15	13.9	5	5	50	1.8	8.02	14	14
brock200_4	200	17	17	15.3	1	6	63	0.82	10.32	16	16
MANN_a27	378	126	125	124.1	0	12	100	10.51	94.59	126	121
MANN_a45	1035	345	339	337.6	0	1 20	100	146.86	768.77	343	336

Table 5: Some improved results from AS-MISP without using any heuristic value. Additional results are shown in columns OCH and CBH.

6 Conclusions

In this paper we presented a new version of an Ant System extended to handle subset problems. In the proposed version of the system, a pheromone trail is put to the problem's components instead of the problem's connections. The extended Ant System was applied to several instances of the Maximum Independent Set Problem generated under methods previously used as well as a well known set of instances for the Maximum Clique problem which were taken from the DIMACS 2^{nd} challenge directory. For some test cases of the three groups considered a few cycles of the algorithm were necessary for obtaining the optimal value. However, on a few problems the AS-MISP exhibits premature convergence. Concerning the random graphs tested, the best performance was achieved for those graphs generated with higher probability values — highly connected graphs — except for the graphs with the low density. Nevertheless, the performance of the AS-MISP for this kind of instances was as good as the performance of GRASP according to the reported results for this method. Also for DIMACS instances, the AS-MISP showed a performance comparable to several specific algorithms either of the MISP or the Clique Problem. Additionally, some improvement were obtained for some instances just directing the search based on trail information. This experiment (using $\beta = 0$) was not intended to show that an Ant System using only trail information is equivalent to PBIL. On the contrary, we only described that some similarities exist and therefore they can be exploited to improve the performance if possible. The general results indicate the potential power of the ACO approach for solving constrained subset problems. Future extensions of this work include mainly an analysis of the influence of the density of graph's on the performance of the Ant System.

References

[1] C. Aggarwal, J. Orlin, and R. Tai. Optimized Crossover for the Independent Set Problem. *Operations Research*, 45(2):226–234, 1997.

[2] Th. Bäck and S. Khuri. An Evolutionary Heuristic for the Maximum Independent Set Problem. In M. Michalewicz et. al., editor, *Proceedings of the First International Conference on Evolutionary Computation*, pages 531–535, Piscataway, NJ, 1994. IEEE Press.

[3] E. Balas and W. Niehaus. Finding Large Cliques by Bipartite Matching. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 26:29–52, 1996.

[4] S. Baluja. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical Report Tech. Rep. No. CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, PA, 1994.

[5] B. Bollobás. *Random Graphs.* Academic Press, 1985.

[6] I.M. Bomze, M. Budinich, M. Pelillo, and C. Rossi. Annealed replication: A new heuristic for the maximum clique problem. *Discrete Applied Mathematics, Special issue on "Discrete vs analog computation*, 1998.

[7] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence - From Natural to Artificial Systems.* Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press, 1999.

[8] M. Cena, M.L. Crespo, C. Kavka, and G. Leguizamón. A Study of Performance of an Ant Colony System applied to Multiple Knapsack Problem. In E. Apayd, editor, *Proceedings of EIS-98*, pages 567–573. ICSC Academic, 1998.

[9] D. Cornea, M. Dorigo, and F. Glover, editors. *New Ideas in Optimization.* McGraw-Hill International, 1999.

[10] DIMACS directories. Second DIMACS implementation challenge. http://dimacs.rutgers.edu /Challenges/index.html.

[11] M. Dorigo, G. Di Caro, and L.M. Gambardella. Ant Algorithms for Discrete Optimization. *Artificial Life*, 5(2):137–172, 1996. Also available as Tech. Rep. IRIDIA/98-10, Universitè Libre de Bruxelles, Belgium.

[12] 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.

[13] M. Dorigo, V. Maniezzo, and A. Colorni. Positive feedback as a search strategy. Technical Report Tech. Rep. No. 91-016, Politecnico di Milano, Italy, 1991.

[14] M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System Applied to the Quadratic Assignment Problem. Technical Report Technical Report IRIDIA 94/28, Université Libre de Bruxelles, Belgium, 1994.

[15] C. Friden, A. Hertz, and D. de Werra. STABULUS: A Technique for Finding Stable Sets in Large Graphs with Tabu Search. *Computing*, 42:35–40, 1989.

[16] L.E. Gibbons, D.W. Hearn, and P.M. Pardalos. A continuous Based Heuristic for the Maximum Clique Problem. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 21:103–124, 1996.

[17] G. Leguizamón and Z. Michalewicz. A New Version of Ant System for Subset Problems. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 1459–1464. IEEE Press, Piscataway, NJ, 1999.

[18] E. Marchiori. A simple heuristic based genetic algorithm for the maximum clique problem. In *Proc. ACM Symp. Appl. Comput.*, pages 366–373, 1998.

[19] C. Resende, T. Feo, and G. Mauricio. Greedy Randomized Adaptive Search Problems. *Journal of Global Optimization,* 6:109–133, 1994.

[20] C. Resende, T. Feo, G. Mauricio, and S. Smith. A Greedy Randomized Adaptive Search Procedure for Maximum Independent Set. *Oparations Research*, 42:860–878, 1994.

[21] C. Rossi. A Replicator Equations based Evolutionary Algorithm for the Maximum Clique Problem. In *Congress on Evolutionary Computation (CEC2000)*, CA, USA, 2000.