Ant Colony System for Graph Coloring Problem

Malika Bessedik, Rafik Laib, Aissa Boulmerka et Habiba Drias,
Institut National d'Informatique, INI, Alger, ALgérie
m_bessedik@ini.dz

Abstract

In this paper, we present a first ACO approach, namely Ant Colony System (ACS) for the graph colouring problem (GCP). We implemented two strategies of ACS for the GCP; construction strategy and improvement strategy. In construction strategy, the algorithm iteratively constructs feasible solutions. The phase of construction is carried out by a specific constructive method for the problem, that is: Recursive Largest First (RLF) or DSATUR. These two constructive methods take into account different updates of pheromone trails and the heuristic information . The improvement strategy uses a local search, namely Tabu Search, to improve the best solution obtained at each iteration of the algorithm. To test the efficiency of our approach, we also implement best-known algorithms for the GCP. That is, Scatter Search (SS) which, integrates also Tabu Search (TS), Ant System (AS), RLF and DSATUR. This paper report experimental results on some well studied Dimacs graphs. A comparison between the different algorithms shows that the algorithm we called ACS1_R (construction strategy, construction done by RLF) gives best results. It approaches the best colouring algorithms and outperforms some hybrid algorithms on some large instances of the famous Dimacs benchmarks.

1. Introduction

The graph colouring problem (GCP) is one of the most studied NP-hard problems in graph's theory, completeness theory and operational research. Its importance is justified by its divers and interesting applications such as timetabling and resource assignment. Before introducing the GCP, we first define a graph k-colouring which can be stated as follows: given an undirected graph G with a set of vertices V, and a set of edges E, a k-colouring of G consists of affecting to each vertex of V a colour such that any two adjacent vertices have different colours. Formally, a k-colouring of G=(V,E) can be stated as a function C from V to a set of colours K (colours are generally represented by integers) such that $|K|=k$ and $C(u)\neq C(v)$ whenever E contain an edge (u,v) for any two vertices u and v of V (assignment approach). The graph k-colouring can also be stated as a partition of V into k stables called colour sets S1,…, Sk where every vertex in Si has the same color i. (partition approach). The minimal number of colours k for which a k-colouring exists is called the chromatic number of G and is denoted by χ (G). An optimal colouring is one that uses exactly χ (G) colours. The GCP is the optimisation problem that consists of finding an optimal coloring for a given graph G. Since the GCP is NP complete [1], we need to use heuristics methods to solve it. There are many resolutions methods for this problem:
greedy constructive approaches, such as, DSATUR [2] RLF[2], local search algorithms, such as Tabu Search [3] or Simulated Annealing[4], evolutionary methods, eg Genetic Algorithm [5], Scatter Search [6], Ant System[7]and hybrid Strategies [8], [9].
In this paper, we give the main stages of a new ACO approach namely Ant Colony System (ACS) for the GCP with two strategies of construction and improvement. For the first strategy, at each stage of its phase of construction, a partial solution will be added to the current solution until obtaining a complete solution. This is done by either RLF or DSATUR. Whereas, in the second strategy, we deal with complete solutions that are improved iteratively using Tabu Search. In section 2, we briefly outline the concept of Ant Colony Optimisation (ACO).In section 3, we present the general principle of the methods used in improvement of our ACS algorithm. In section 4, we give the framework and the components of our approach ACS for the GCP. In section 5 and 6, we present our adaptation of ACS for the GCP, construction strategy and improvement strategy. In section 7, we present results and finally, in section 9, we draw some conclusions and give an outlook of future works.

2. Ant colony optimisation

Ant Colony Optimisation (ACO) is a recently proposed metaheuristic approach for solving hard combinatorial optimization problems [10], [11]. It's an evolutionary method inspired from the foraging behaviour of real ants that enables them to find shortest paths between a food source and their nest. This method differs from other evolutionary methods such as, Genetic Algorithms (GA) and Scatter Search (SS) by the fact that at each stage of the solution construction, it takes into account information resulting from the preceding iterations and the desirability of each element that can be added to the current solution. In an ACO algorithm, a complete graph denoted by $\mathcal{G}=(\mathcal{C},\mathcal{L})$ whose vertices are the solution components is associated to the problem. It is called "*construction graph"*. Moreover, in an ACO algorithm, simple agents called *artificial ants* communicate indirectly to find good solutions for the optimization problem. Informally, the behaviour of ants in ACO algorithms can be summarised as follows: The ants of a colony concurrently and independently move through adjacent states of the problem on the *construction graph*, applying a stochastic local decision. While moving, ants incrementally build solutions to the optimisation problem. Typically, good quality solutions emerge as the result of the collective

Proceedings of the 2005 International Conference on Computational Intelligence for Modelling, Control and Automation, and International Confe
Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'05)
0-7695-2504-0/05 $20.00 © 2005 IEEE

interaction of the ants, which is obtained via indirect communication (*pheromone trails*). During the construction of the solution, ants evaluate the partial solution and deposit pheromone trails on components or connections it used (*online update*). This information will guide the future ants search. To move on the construction graph, ants will make decision based on *pheromone trails* (τ) and an information specific to the problem (η). In many cases, η is the cost, or an estimate of the cost. These values are used by the ant's heuristic rule to make probabilistic decisions on how to move on the construction graph. The probabilities involved in this case are commonly called *transition probabilities.* The goal in combinatory optimisation is not to reproduce the biological model but to be inspired usefully. Thus, besides ant's activities, an ACO algorithm includes two additional procedures: *pheromone trail evaporation* and *daemon actions* (this last component being optional). Pheromone evaporation is the process by means of which the pheromone trail intensity on the components decreases over time. Formally, pheromone evaporation is a useful form of *forgetting*, favouring the exploration of new search space areas. Daemon actions can be used to implement centralized actions that cannot be performed by an ant. For example, the daemon can observe the path (solution) found by each ant of the colony and deposit extra pheromone (additional pheromone) on the components used by the ant that built the best solution. Pheromone updates performed by the daemon are called *off-line pheromone updates*.

The first ACO algorithm proposed was Ant System (AS) [12]. AS was applied to some rather small instances of the travelling salesman problem (TSP) with up to 75 cities. It was able to reach the performance of other general-purpose heuristics like evolutionary computation [12]. Despite these initial encouraging results, AS could not prove to be competitive with state-of-the-art algorithms specifically designed for the TSP when attacking large instances. Therefore, a substantial amount of recent research has focused on ACO algorithms that show better performance than AS when applied, for example, to the TSP such as *MAX-MIN* Ant System (*MM*AS) and Ant Colony System (ACS) [11]. In this paper, we will be interest by this last approach.

3. Improvement methods

Before giving the main stages of our ACO approach namely Ant Colony System (ACS) for the GCP with its two strategies of construction and improvement, we present the general principle of the methods used in improvement of our algorithm. In construction strategy, the self-adaptation phase uses a problem specific constructive method (RLF or DSATUR) to create a new solutions population on the basis of the global memory . These two constructive methods take into account the different updates of pheromone trails and thus the heuristic information relative to the problem. Indeed, two decisions have to be taken at each step of a constructive method. The first is about the choice of the next vertex to colour and the second is about the colour to assign to the chosen vertex. In an ordinary constructive method, these decisions are made in a myopic way by completing the current partial solution at best [13]. In addition, ACO algorithms perform best when hybrid with local search algorithms, (which is a particular form of daemon action) [10]. In the improvement strategy, we conceived the principle of hybridization as follows: at each iteration of the algorithm, the best solution found is improved using Tabu Search and these locally optimised solutions are used in pheromone updates. Moreover, the Daemon can improve each ant solution.

3.1. Constructive methods: RLF and DSATUR

Constructive methods are largely used in combinatory optimization. Their efficiency is justified by their short execution time and their facility of implementation. Constructive methods are also used to find an upper bound for the chromatic number. A constructive method adapted to the GCP run over the vertices set sequentially, and assigns to each vertex the smallest possible colour (colours: 1, 2..., k) until obtaining a complete solution. In construction strategy of ACS, the self-adaptation phase uses a constructive method specific to the considered problem. This method will take into account the different updates of pheromone and the heuristic information of the problem. We will choose two famous methods, Recursive Largest First (RLF) and DSATUR, which are considered among the best resolution methods for the GCP. We will often use these two methods, particularly DSATUR, to generate an upper bound of the chromatic number. This value is used to initialize another metaheuristic whose function is to find a k-colouring of G with k fixed. For example, we use this upper bound to determine the number of colours of each solution in the initial population in SS. We explain below the general principle of these two methods. For RLF, classes of colours are built sequentially. Once a vertex v$\in$A where A is the set of vertices which have not been coloured yet, such that maximum degree of A is $deg_A(v)$ has been selected , the current stable is augmented by inserting as long as possible the vertex v$\in$AUB with maximum degree $deg_B(v)$. The set B contains every uncoloured vertex, which cannot belong to the stable under construction. For DSATUR, vertices are ordered by choosing at each step a vertex v with a maximum degree of saturation dsat(v), where dsat(v) is the number of different colours already assigned to the neighbours of v. If v is not unique, we choose the vertex whose $deg_A(v)$ is minimum. The complexity of DSATUR is about $O(n^2)$, and $O(n^3)$ for RLF. This difference appears in the execution time, which is higher for RLF on large graphs.

3.2. Tabu Search for GCP

Let us recall that Tabu search (TS) explores, at each stage, a random subset V*$\subset$ N(v) (N(v) is the set of the neighbours of v), moving from a solution to another with an aim of obtaining a better solution.

For the GCP, TS moves from a colouring to another improving the quality of this. The neighbourhood for the assignment approach is defined as being the permutation of two successive vertices or not in the vector (solution).

Concerning the partition approach, a neighbour is obtained starting from a given solution by moving a vertex from a class of colour (Stable) S_i to another class of colour S_j. The elements of the tabu list T are pairs (v, c), such that once a vertex v is coloured with a colour c, one prohibits oneself that it will receive this colour during the nearest |T| stages.

4. Ant Colony System for GCP

We give below the framework and the components of our approach ACS for the GCP.

4.1. Update of pheromone trails

Values of pheromone are associated to pairs of nonadjacent vertices having the same colour. Formally, the value $\tau^k(v_i, v_j)$ corresponds to the trace left by a given ant "k" having assigned the same colour to the vertices v_i and v_j ($1 \leq i \neq j \leq n$). Therefore, at the end of a cycle of the algorithm, $\tau(v_i, v_j)$ is the value of pheromone associated to the couple (v_i, v_j) for all colourings (ants) which coloured v_i and v_j with the same colour. Let us note $\Delta\tau(v_i, v_j)$ values of pheromone added to $\tau(v_i, v_j)$ by all the ants during a cycle. The values $\tau(v_i, v_j)$ are stored in a square matrix of order n and initialized like this:

$$\tau(v_i, v_k) = \begin{cases} 1 \text{ si } (v_i, v_k) \notin E. \\ 0 \text{ si } (v_i, v_k) \in E. \end{cases} \quad \ldots\ldots(1)$$

. Stage by stage update: for each ant k, we have:

$\forall (v_i, v_j) \in S_k : \tau(v_i, v_j) = (1-\rho) * \tau(v_i, v_j) + \rho * \tau_0$. …… …(2)

.Online delayed update: $\tau(v_i, v_j) = \tau(v_i, v_j) + \rho * \partial(S^*)$… (3)

. Offline update (Daemon action):

$\forall (v_i, v_j) \in S^*: \tau(v_i, v_j) = (1-\varepsilon) * \tau(v_i, v_j) + \varepsilon * \partial(S^*)$. …… (4)

. Evaporation is carried out according to this rule:

$\tau(v_i, v_j) = (1-\rho) * \tau(v_i, v_j)$……(5)

Where: S_k is the solution built by the ant "k"; S^* the best solution found until there; $\rho \in [0,1]$ pheromone trails persistence; (1-ρ) the evaporation coefficient; $\varepsilon \in [0,1]$ a parameter indicating the decline of the pheromone; τ_0 the initial value of the pheromone; $\partial(c)$ function of the quality of colouring C and $\partial(S^*) = 1/\text{cost}(S^*)$.

Cost(C) is defined in two possible ways, as follows:

Cost1(C)=**nc** (nc:a number of colours used in C)or Cost2(C)=**nsc** (nsc: the number of vertices in conflict in C). Note that, Cost1 is used in first strategy. Cost2 is used in second strategy and it is more expensive in time than Cost1.

4.2. Heuristic information

For the first strategy, heuristic information is defined according to the used constructive method. If an ant is implemented as RLF, heuristic information $\eta(v_i, v_j)$ relative to the choice of the vertex v_j starting from the current vertex v_i is defined in three possible ways: $\eta(v_i, v_j) = \deg_B(v_j)$. $\eta(v_i, v_j) = |A| - \deg_A(v_j)$. $\eta(v_i, v_j) = \deg_{A \cup B}(vj)$.

Where A and B are defined in section 3. However, if an ant behaves like DSATUR, heuristic information is defined simply as the degree of saturation of the vertex in question. $\eta(v_i, v_j) = \text{DSAT}(v_j)$. (Let us recall that DSAT (v) is the number of colours already assigned to the neighbours of v).

4.3. Transition Rule

ACS improves AS algorithm by giving more importance to information collected by previous ants with respect to exploration of the search space. This is

ACS improves AS algorithm by giving more importance to information collected by previous ants with respect to exploration of the search space. This is acheaved using two mechanisms. First, a strong elitist strategy is used to update pheromone trails (2), second, ants choose the next vertex "v" to move to (colour) applying a pseudo- random proportional rule: With probability q_0 they move to the vertex j for which the product between pheromone trail and heuristic information is maximum, that is, v= arg $\max\{(\tau_{ij})^{\alpha}(t) . (\eta_{ij})^{\beta}(t)\}$, while with probability $(1- q_0)$ they operate a biased exploration in which the probability $p_{ij}^k(t)$ is

The same as in AS given by:

$$P_{ij}^k(t) = \begin{cases} \dfrac{(\tau_{ij})^{\alpha}(t).(\eta_{ij})^{\beta}(t)}{\sum_{l \in N_i^k} (\tau_{il})^{\alpha}(t).(\eta_{il})^{\beta}(t)} & \text{If } l \notin \text{tabu\_list}k \\ 0 & \text{if not} \end{cases}$$

Where τ_{ij} is the value of pheromone on the edge (i,j) at the iteration "t", $\eta_{ij}(t)$ the heuristic information associated to the vertex j at the iteration "t", q a random variable uniformly distributed on [0,1], q_0 a parameter of the algorithm, $0 \leq q_0 \leq 1$. It determines the relative importance between exploitation and exploration. α and β are two parameters which determine the relative influence of pheromone trail and heuristic information, and N_i^k is the feasible neighbourhood of ant *k*; that is, the set of vertices which ant has not yet visited.

4.4. Candidate list

The candidate list is an intelligent strategy used mainly when the size of the instance (order of the graph) is very large. This technique consists in not considering the totality of the neighbourhood but rather a subset of this (for example, containing the best solutions), which can lead search towards promising areas. In addition, this allows to accelerate the solutions construction. Inspiration of other techniques as in Tabu Search, where candidate's lists are used, could be useful for the development of this strategy efficiently in ACS.

5. Construction Strategy

In this strategy, each ant is initially put on a vertex of the construction graph randomly, or according to a well-defined criterion (the vertex having the maximum degree). Pheromone values on all connections are initialized to τ_o. Each ant repeatedly construct a feasible solution while inserting into each stage a component, couple (vertex, colour), in the current partial solution until obtaining a complete solution according to the constructive method implementing the ant (RLF or DSATUR). This construction is done by repeating the following stages:
(*i*) The next vertex to be coloured is chosen by observing the rule of transition (*). It is selected among the vertices of the candidate list if this one is considered.
(*ii*)The vertex chosen in the stage (*i*) is put in the ant Tabu list, this list is used to save the path (solution) built by the ant. Tabu list is also used to make sure that a vertex already coloured will not be coloured a second time, and consequently to guarantee the feasibility of the built solution.
(*iii*)This stage (*Update Online stage by stage*) consists in decreasing the pheromone values associated to pairs of vertices having the same colour (the vertex that has been just added to the stable in construction) according to the equation (1); this to avoid fast convergence to the same solution. This update replaces the phase of evaporation in the algorithm. As long as the ant did not build a complete solution yet, it repeats the phases of construction. After all the ants established their solutions, the best one is saved and compared with that of the preceding iteration for a possible improvement of the objective function according to the equation (3) (online delayed update). After this, the daemon adds extra pheromone to only the best solution found in the preceding stage, and precisely to all connections that constitute it according to the equation (4) (offline update). If the same solution appears in several iterations of the algorithm (stagnation), then, the daemon decides to make evaporation (5). If the stopping criterion (a maximum number of iterations defined preliminary, or a maximum execution time) is not filled, the algorithm is started again. We give below, the pseudo code of ACS algorithm, construction strategy, for the GCP.

Begin ACS1
- *Initialization ; (Pheromone and parametres)... (1)*
  - ***While*** *(stop criterion not satisfied) do*
    - *Position ants on starting vertices*
      - ***Repeat***
        - ***For*** *(each ant)* ***do***
          - *Choose a vertex to colour by applying the transition rule (*);*
          - *Tabu list Update;*
          - *Update Online stage by stage of pheromone trails (2);*
        - ***End for***
      - ***Until*** *(each ant construct a solution)*
        - *Select the best solution ;*
        - *Pheromone Offline update of the best solution (4)*
        - *Evaporation if necessary (5)*

End While

End ACS1

6. Improvement strategy

The strategy of improvement can be summarised as follows: Pheromone values are initialized to a value τ_0 on all connections. We assign to each ant a starting vertex chosen randomly or by a well-defined criterion. Staring from an initial colouring (obtained either in a random way or by a constructive method), each ant tries to improve it by recolouring (change the colour) some vertices in conflict. Improvement is carried out by choosing the next vertex to recolour as the vertex having a maximum number of violations among vertices of the condidate list. If such a vertex does not exist (current solution is feasible), a vertex is selected arbitrarily. If Iter_Max (a certain fixed number of iteration) is not reached yet, the process of improvement is repeated again. After this, Tabu Search makes an improvement to the best solution found in the preceding stage (online delayed update). The daemon adds extra pheromone (Offline update) to the best solution and precisely all connections that compose it according to the equation (4). If for a situation of stagnation, the same solution appears during several iterations of the algorithm, the daemon decides to make evaporation using the equation (5). If the stop criterion (a maximum number of iterations defined preliminary, or a maximum execution time) is not satisfied, the algorithm is started again. We give below the pseudo code of the ACS improvement strategy for the GCP.

Begin ACS2
- *Initialization ; (Pheromone et parameters)*
- ***While*** *(stop criterion not satisfied) Do*
  - *Position ants on starting vertices;*
  - ***For*** *(each ant)*
    - ***Generate*** *an initial colouring for the graph G.*
  - ***Repeat***
    - *Choose the next vertex to recolour among vertices of the condidate list and according to the transition rule (*).*
    - *Change its colour so that conflicts are minimised.*
  - ***Until*** *(Iter_Max reached)*
    - *Online delayed Update of pheromone trails (3)*
  - ***End For***
  - *Select the best solution;*
  - *Improvement of the best solution (TS);*
  - *Evaporation if necessary (5);*

End While

End ACS2

8. Test Results

We give in this section some preliminary execution results obtained with our approach of ACS for the GCP. We have done our tests on different graphs types from the second DIMAC challenge. We have implemented several algorithms to do comparisons: That is, RLF, DSATUR, AS_D(AS, ant as DSATUR), AS_R(ant as RLF) [2],

0-7695-2504-0/05 $20.00 © 2005 IEEE

ACS1_D(first strategy, ant as DSATUR), ACS1_R(first strategy, ant as RLF), ACS2 (second strategy, standart), ACS2_H(second strategy, Hybrid) SS and SS_H(SS hybrid)[6]. We have considered various parameters, number of vertices, several densities (0.4, 0.5, and 0.6) and several executions for each graph. For example, for each density, we did 20 executions for graphs of 100 vertices and this for each method. We test all methods on the same graphs. We denote by **n_a,** the size of the colony, **n_it** the number of iterations, **b_s** number of colours of the best solution, **T(s)** execution time in seconds. **n_m** the number of moves in an iteration of TS and by **n_it,** the number of iterations in TS. The results table 1 are the averages (**evr**) of the colours numbers of five tests for each line of the table. We give in table1 and 2 results of ACS1_R, and ACS_H for Random graphs of 100 vertices. Each line corresponds to a parameter value (in table 2, ρ=0,5; τ_0=1 and q_0=0,5). The last column of table 1 gives results of SS (the best results between those of SS standard and SS hybrid).

	d	α	β	ρ	τ_0	q_0	n_a	n_it	evr	SS
d=0,4		2	2	0,5	1	0,5	40	5	13	15
		2	2	0,5	1	0,5	5	2	13,8	15
		4	2	0,5	1	0,5	20	2	13	15
		2	2	0,5	1	0,5	10	2	13	15
d=0,5		2	2	0,5	1	0,5	20	5	16,8	20
		4	2	0,5	1	0,5	20	2	16,2	20
		2	4	0,5	1	0,5	20	2	16,6	20
		2	2	0,5	1	0,5	40	5	16,4	20
d=0,6		2	2	0,5	1	0,5	20	5	20,4	24
		2	2	0,5	1	0,5	5	2	20,6	24
		4	2	0,5	1	0,5	20	2	21	24
		2	2	0,5	1	0,5	40	5	20	24

Table1: Results of ACS1_R for Random graphs (100 vertices).

d	α	B	n_a	n_it	lt_ts	n_m	b_s	T(s)
0.4	2	2	20	10	10	10	9	63.31
0.4	2	2	40	10	10	10	9	124.27
0.6	1	1	20	10	10	10	11	129.30
0.4	1	3	20	10	10	10	9	62.36
0.5	1	3	20	10	10	10	11	93.42
0.6	1	3	20	10	10	10	13	128.28

Table2: : Results of ACS2_H for Random graphs (100 vertices).

	|V|	|E|	opt	DSAT	R LF	AS_D	AS_R	ACS 1_D	ACS 1_R	AC S2
G_1	128	1170	20	**20**	**20**	**20**	**20**	**20**	**20**	**20**
G_2	128	2113	32	32	32	**31**	33	32	**31**	33
G_3	128	3216	42	43	43	44	44	43	**42**	44
G_4	128	5198	73	**73**	**73**	**73**	**73**	**73**	**73**	**73**

Table 3: Results for queen graphs

(G_1=miles 500; G_2=miles 750; G_3=miles 1000; G_4=miles 1500)

In tables 3 and 4, the column "optimal" means the best-known results for some of the DIMACS benchmark graphs[14]. ACO algorithms appear more powerful than other methods, particularly ACS1_R algorithm that gives best results mainly for large graphs. Indeed, while referring to table1, we note that ACS1_R enables us to decrease the number of colours by 3 colours compared to SS. From table3, we observed that ACS1_R outperforms the best-known result for miles 750. In addition to that, ACO algorithms are faster than other implemented methods mainly the ACS_R. ACS2_H seems quite slow than other ACO algorithms. This is natural since ACS2_H is a hybrid algorithm.

|V|	|E|	opt	D SAT	R LF	AS _D	AS _R	ACS 1_D	ACS 1_R	AC S2
50	1099	17	17	17	17	17	17	17	17
50	1006	11	11	11	11	11	11	11	11
50	825	8	9	8	8	8	8	8	8
50	568	4	5	5	5	5	5	4	5
100	4455	20	20	21	21	20	20	20	20
100	4007	12	13	14	13	14	12	12	14
100	3186	9	10	10	10	10	9	9	9
100	2090	4	4	4	4	5	4	4	4

Table4: Results for Geometrical graphs

Such as: (G_1=gr(50, 100), gr(50,150), gr(50,200), gr(50,300), gr(100, 200), gr(100, 300))

9. Conclusions and outlook for future works

In this work, we have developed the first adaptation of Ant Colony Optimisation for the GCP. We presented two different methods. The first one constructs repeatedly feasible solutions that are improved locally either by RLF or DSATUR. These two constructive methods take into account the different pheromone updates and thus the heuristic information relative to the problem. In the second method, the best solution found at each iteration is improved using Tabu Search and these locally optimised solutions are used in different pheromone updates. Moreover, the possibility of improving the solution of each ant is carried out by the Daemon. The preliminary results we obtained on some well-studied DIMACS graphs show that our approach may be effective. Indeed, ACS1_D, ACS1_ R, ACS2 and ACS2_H reached most of the best-known results. ACS1_R obtains also better results than those of SS and SS_H on some graphs. In addition to that, ACS with its two strategies is quite fast when compared with other methods existing in literature. Since the nature of ACO algorithms lends them to be parallelized, we are now working on adaptation of a parallel ACO algorithm for the GCP and some of its generalisation such as list colouring and T-coloration.

10. References

[1]M.R. GAREY, D.S. JOHNSON, *Computers and intractability: a guide to the theory of NP-completeness*, W.H. Freeman and Company, New York, 1979.

[2] D.BRÉLAZ,New méthods to color vertices of a graph. Communications of ACM 22: 251-256, 1979.

[3] A. HERTZ, D. De WERRA, Using Tabu search techniques for graph coloring. *Computing* 39 : 345-351, 1987.

[4] M. CHAMS, A. HERTZ, D. de WERRA, Some experiments with simulated annealing for coloring graphs. *EJOR* 32 : 260-266, 1987.

[5]C. FLEURENT, J.A. FERLAND, Genetic and hybrid algorithms for graph coloring. Dans G. Laporte, I.H. Osman,

(Eds.), *Metaheuristics in Combinatorial Optimization, Annals of Operations Research*, 63 : 437-441, 1996.
[6]J-P. HAMIEZ AND J.K. HAO, *Scatter Search For graph coloring*, *Lecture Notes in Computer Science 2310*: 168-179, Springer, 2002.
[7] M. DORIGO, V. MANIEZZO, and A. COLORNI. The ant system: An Autocatalytic optimizing process. Technical Report 91-016 Revised,
[8]P. GALINIER. *Hybrid Evolutionary Algorithms for graph coloring. J.*Combin. Optim. 3(4) (1999) 379-397.
[9]R. DORNE, J.K. HAO, A new genetic local search algorithm for graph coloring. *Lecture Notes in Computer Science 1498 :* 745-754, Springer-Verlag, 1998.
[10]F. COMELLAS AND J. OZAN, *Ant Algorithms for Discrete Optimization* in D. Corne, M. Dorigo and F. Glover editors, New ideas in Opt pages 11-32 Mc Graw Hill London. UK, 1999.
[11] M.DORIGO, G. DI CARO AND L.M. GAMBARDELLA., *Ant algorithms for discrete mathematics. Artifitial life,* 5(2): 137-172, 1999.
[12]M. DORIGO, *Ant Colonies for the Travelling Salesman Problem*. Bio systems, 43:73-81, 1997.
[13]D. COSTA, A. HERTZ, *Ants Can Colour Graphs*, The journal of the operational Research Society. (1997)
[14] D.S. JOHNSON, M.A. TRICK (Eds.), 2nd DIMACS implementation challenge –cliques, coloring and satisfiability. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science 26*, American Mathematical Society, 1996.

Proceedings of the 2005 International Conference on Computational Intelligence for Modelling, Control and Automation, and International Confe
Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'05)
0-7695-2504-0/05 $20.00 © 2005 IEEE