A New Genetic Local Search Algorithm for Graph Coloring

Raphaël Dorne and Jin-Kao Hao

LGI2P/EMA-EERIE
Parc Scientifique Georges Besse
F-30000 Nîmes-France
email: {dorne,hao}@eerie.fr

Abstract. This paper presents a new genetic local search algorithm for the graph coloring problem. The algorithm combines an original crossover based on the notion of union of independent sets and a powerful local search operator (tabu search). This new hybrid algorithm allows us to improve on the best known results of some large instances of the famous Dimacs benchmarks.

1 Introduction

The graph coloring problem is one of the most studied NP-hard problems and can be defined informally as follows. Given an undirected graph, one wishes to color with a minimal number of colors the nodes of the graph in such a way that two colors assigned to two adjacent nodes must be different. Graph coloring has many practical applications such as timetabling and resource assignment. Given the NP-completeness of the coloring problem, it becomes natural to design heuristic methods. Indeed many heuristic methods have been developed, constructive methods in the 60's and 70's [1, 12], local search meta-heuristics [10, 11, 2] and hybrid algorithms [6, 14, 3] in the 80's and 90's.

In the field of combinatorial optimization, the best solutions are often obtained by specialized local search algorithms or population-based algorithms without crossover. However, there are cases where hybrid evolutionary algorithms have produced very competitive results, for example for the traveling salesman problem [7], the quadratic assignment problem [13] and the bin-packing problem [5]. In general, these hybrid algorithms rely on a "meaningful" specialized crossover which combines high quality solutions produced by an "efficient" local search method. It should be clear that random crossovers are hardly meaningful for any combinatorial optimization problem, therefore are little helpful to producing high quality solutions. The main point here is that the crossover must be specialized to, and meaningful for the problem.

In this paper, we present an original crossover tailored to the coloring problem. This crossover, that we call UIS, is based on the notion of Union of Independent Sets. The UIS crossover, combined with a tabu search method leads to a simple, yet very powerful algorithm. Indeed, this new algorithm allows us to improve on the best known results of some large benchmarks.

2 Graph coloring

Definition: Given an undirected graph G=(V, E) with V=$\{v_1, ..., v_n\}$ being the set of nodes and E=$\{e_{ij} | \ \exists$ an edge between v_i and $v_j\}$ the set of edges. The graph coloring problem is the following optimization problem: to determine a partition of V in a *minimum* number (the *chromatic number*) of color classes $C_1, C_2, ..., C_k$ such that for each edge $e_{ij} \in E$, v_i and v_j are not in the same color class [15]. Such a color class is called an *independent set*.

Let $c(v_i)$ be the color (represented by a positive integer) assigned to the node v_i, a *proper coloring* must verify the following *constraint*:

$$\forall e_{ij} \in E, \quad c(v_i) \neq c(v_j) \tag{1}$$

Benchmarks: In this paper, we use well known benchmarks (random graphs) from the 2nd Dimacs Challenge[1]. A random graph is denoted by $G_{n,d}$ where n represents the number of nodes and $d \in [0, 1]$ the density of edges defined on the graph, which means that the number of edges of the graph is about to be $d.(n.(n-1)/2)$. Random graphs are difficult to color. It is believed that none of today's algorithms is able to color optimally such graphs having more than 100 nodes with a 0.5 density [11]. In this study, we are interested in some large and hard graphs (500 or 1000 nodes).

Johnson et al. : a set of 6 instances of 500 and 1000 nodes with edge densities of 0.1, 0.5 and 0.9, denoted by DSJC500.1.col, DSJC500.5.col, DSJC500.9.col, DSJC1000.1.col, DSJC1000.5.col, DSJC1000.9.col.

Hertz & De Werra : a set of 7 instances of 500 and 1000 nodes with edge density of 0.5, denoted by ggg1, ..., ggg5, gggg1 and gggg2.

3 Related work

Hertz and De Werra are the first who have applied tabu search to the coloring problem [10]. They proposed a simple coding for solutions and a natural 1-change neighborhood. In addition, to color large graphs ($\geq$300 nodes), they introduced a pre-processing technique which removes some independent sets from a graph leading to a reduced residual graph. Their algorithm, combined with this pre-processing, has produced excellent results on random graphs. Similarly, a simulated annealing algorithm is reported in [2].

Johnson et al. gave the first systematic investigation about the performance of simulated annealing, *Dsatur* [1] and *RLF* [12] on random graphs. A new algorithm called *XRLF* was also proposed. Extensive tests of these algorithms on various random graphs have led to mixed results: no clear dominance of a single algorithm was observed over all the tested instances.

Fleurent and Ferland investigated a tabu algorithm and in particular a hybrid algorithm combining genetic algorithm and tabu search [6]. They replaced random

[1] Available via ftp: dimacs.rutgers.edu/pub/challenge/graph/benchmarks/.

mutation by tabu search and developed a specialized crossover operator based on conflicting nodes (adjacent nodes having the same color). They employed the preprocessing technique of *Hertz and De Werra* to remove independent sets. Their hybrid algorithm has produced excellent results on the 2nd Dimacs challenge benchmarks. However, the computing times to obtain these results were very high for some large instances. *Costa* presented a similar hybrid algorithm called *EDM* and obtained comparable results [3].

Morgenstern proposed a distributed, population based and multi-strategy method [14]. He introduced a special neighborhood which is different from all the other previous studies. This highly specialized method has produced the best results for a large number of Dimacs challenge benchmarks.

4 A genetic local search algorithm

4.1 General algorithm

First, recall that the graph coloring problem consists in minimizing the number of colors k used. Before we present our algorithm, let us explain first the minimization process used. Following [10], we fix k, the number of available colors, to a given value and use our hybrid algorithm to search for a proper k-coloring. This is done by minimizing the number of violated constraints to 0 (f^* =0) according to Eq.(1). If such a coloring is found, we decrease k and repeat the above search process until no more possible proper k-coloring is found within the allowed iterations. Therefore, the goal of our hybrid algorithm is to find a proper k-coloring for a fixed k. The general algorithm is given as follows.

```
Algorithm 1: Genetic local search algorithm for coloring
Data: G, a graph
Result: the number of conflicts with k fixed colors
% f,f* : fitness function and its best value encountered so far
% s* : best individual encountered so far
% i, MaxIter : the current and maximum number of iterations allowed
% best(P) : returns the best individual of the population P
begin
  i = 0
  generate(P_0)
  s* = best(P_0)
  f* = f(s*)
  while ( f* > 0 and i < MaxIter) do
    P'_i = crossing(P_i,T_x); /*using UIS crossover */
    P_{i+1} = mutation(P'_i), /*using tabu search */
    if (f(best(P_{i+1})) < f*) then
      s* = best(P_{i+1})
      f* = f(s*)
    i=i+1
  return f*
end
```

As we can see, the algorithm follows a simple evolution cycle. It crosses two individuals from time to time and then it mutates the individuals of the population. More precisely, at each generation, the UIS crossover (Section 4.4) is applied with probability T_x to each possible pair (Ind_1, Ind_2) (determined randomly) of individuals of the population: p/2 pairs for a population of p individuals. If a crossover takes place on (Ind_1, Ind_2), the offsprings e_1 and e_2 replace Ind_1 and Ind_2 regardless of their fitness. Otherwise, Ind_1 and Ind_2 remain unchanged. Each individual of the population is then improved using a tabu algorithm (Section 4.3).

Different values of T_x lead to different application rates of the crossover operator. Setting T_x to 0 will give n independent executions of the tabu algorithm. Changing T_x allows one to determine the importance of the crossover.

The individuals in the initial population may be generated randomly, or using an existing coloring algorithms. We use the *RLF - Recursive Largest First* algorithm [12]. The *RLF* algorithm has poor performance, however, it gives naturally an initial value for k, the number of colors to be used in a coloring.

4.2 Encoding and fitness function

Given a graph G=(V, E) with n nodes and k the number of available colors (numbered from 1 to k), an individual (or a configuration in a local search terminology) $s = < c(v_1), c(v_2), ..., c(v_n) >$ corresponds to a complete assignment of the k colors to the nodes of the graph. The size of the search space S, is equal to $|S|$=k^n, and becomes very large as soon as the graph has more than 100 nodes.

For each individual s, the fitness $f(s)$ is simply the number of unsatisfied color constraints (Eq. (1)).

$$f(s) = \sum_{(v_i, v_j) \in E} q(v_i, v_j) \quad \text{where } q(v_i, v_j) \ = \begin{cases} 1 \text{ if } c(v_i) = c(v_j) \\ 0 \text{ elsewhere} \end{cases}$$

The goal of the optimization process here is to minimize $f(s)$ until $f(s)$=0 for the fixed k, which corresponds to a proper k-coloring.

4.3 Mutation by tabu search

The mutation used in this hybrid algorithm is ensured by a tabu search (TS) algorithm. For a complete presentation of TS, the reader is invited to consult the recent book by Glover and Laguna [9].

Tabu search is an advanced local search meta-heuristic. A TS algorithm visits iteratively a series of locally best configurations according to a neighborhood. In order to avoid a possible cycling. TS uses a special short term memory called *tabu list*, that maintains last l encountered configurations or more generally pertinent attributes of these configurations (l is called *tabu tenure*). At each iteration, TS chooses one best neighboring configuration among those not forbidden by tabu list, even if the chosen configuration is no better than the current one.

The TS algorithm used in our hybrid algorithm for the coloring problem is the one presented in [4] augmented with a random-walk option (see below). The main characteristics of this TS algorithm is the following.

Neighborhood function N: $S \to 2^S$, $\forall s$ and $s' \in S$, $s' \in N(s)$ if and only if s and s' are different at the value of a single "conflicting" node (i.e. it has the same color as some adjacent nodes.). Thus, a neighbor of s can be obtained by changing the color of a conflicting node. A move is then characterized by a couple $< i, c >$, i and c being respectively a node and a color.

Configuration evaluation: Special data structures and techniques are used to evaluate rapidly the fitness of each neighboring configuration. This is done by maintaining incrementally in a matrix δ the *move value* or *fitness variation* for each possible move from the current solution s.

Tabu list and tabu tenure: When a node v_i in a configuration s is assigned a new color (i.e. when a move is carried out), the pair $< v_i, old\_color >$ is classified tabu for l (tabu tenure) iterations. That is, the old color will not be allowed to be re-assigned to v_i during this period. The tabu tenure l is dynamically adjusted by a function defined over the number of conflicting nodes. More precisely, let CV be the set of conflicting nodes in s, then $l = \alpha * |CV| + random(g)$ where α and g are empirically determined.

Mixed move strategy: At each iteration, a neighboring configuration s' must be chosen to replace the current one s. The choice is realized with the following strategy: according to a probability p_{rw}, carry out a "random-walk", i.e. to pick randomly a neighbor among all the neighbors in $N(s)$; according to $1 - p_{rw}$, carry out a normal tabu iteration, i.e. to pick a best (non-tabu) neighbor in $N(s)$.

4.4 The UIS crossover

The UIS crossover follows the definition of the coloring problem with independent sets (cf. Section 2): a proper k-coloring is a collection of k independent sets. With this point of view, an individual corresponds to k color classes of n nodes: some classes are conflict-free (independent sets) and the others are not. According to this interpretation, if we try to maximize the size of each independent set by a combination mechanism, we will reduce the sizes of non-independent sets, which in turn helps to push these sets into independent sets. The UIS crossover is based on this idea.

UIS tries to unify pairs of independent sets (I_{p_1}, I_{p_2}) taken respectively from two parent-colorings p_1 and p_2. Two children-colorings e_1 and e_2 are the result of a set of unions. The pair of independent sets (I_{p_1}, I_{p_2}) to be unified are those which have the largest number of nodes in common. Let us see how to generate e_1 from p_1 and p_2.

Let $I_{p_1,c}$ be the largest conflict-free subset (an independent set) of the class having the color c in p_1, we seek, among the k color classes of p_2, the independent set $I_{p_2,c'}$ (a conflict-free subset of a color class) such that $I_{p_2,c'}$ has the largest number of nodes in common with $I_{p_1,c}$ ($\forall c' \in [1..k]$, $\mid I_{p_1,c} \cap I_{p_2,c'} \mid$ is maximal).

If there are more than one possible $I_{p_2,c'}$, one is taken randomly. This operation is repeated for each color of p_1 to obtain the set of unions U_{e_1} corresponding to the pairs of independent sets for e_1:

$$U_{e_1} \;=\; \{\; (I_{p_1,1}, I_{p_2,j_1}), ..., (I_{p_1,i}, I_{p_2,j_i}), ..., (I_{p_1,k}, I_{p_2,j_k}) \;\}$$

where $(I_{p_1,i}, I_{p_2,j_i})$ is the union of $I_{p_1,i}$ and I_{p_2,j_i} with j_i representing the color of the independent set of p_2 which is unified with the one of p_1 which has the color i (See Fig.1).

Once this set is obtained, we generate the child e_1 by assigning the color of $I_{p_1,i}$ to the nodes of $(I_{p_1,i}, I_{p_2,j_i})$. Note that U_{e_1} does not necessarily contains all the nodes of the graph. For each node $v \in V\text{-}U_{e_1}$, it keeps the color of p_1. To generate e_2, we carry out the same operation while exchanging p_1 with p_2 (*cf.* Fig. 1).

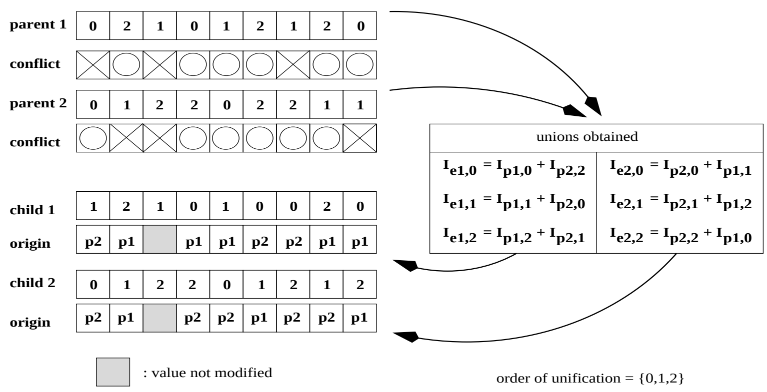

Fig. 1. UIS crossover

In terms of complexity, for each color of a parent p_1, we visit all the nodes of p_2 to find the best union. When all the unions are identified, we just have to assign the nodes of the children according to these unions. Thus the complexity of the UIS is in $O(k \times n)$ in the worst case.

Let us mention that using the notion of independent sets, other crossover operators are equally possible. For example, parallel to the current study, [8] investigates several such operators as well as other hybrid evolutionary algorithms.

Finally, note that the UIS crossover is completely different from conflict-based crossovers for the coloring problem [6]. There the coloring problem is considered as an assignment problem: an individual is an assignment of values to variables. Using this interpretation, an offspring will try to copy conflict-free colors from their parents. Previous work showed that combined with a local search algorithm, this crossover may give interesting results for some graphs. However, the search power of such a crossover is limited compared with the UIS operator.

5 Experiments and results

Our tests were carried out on an UltraSparc1 station with a 143 MHz processor and a memory of 128 Megabytes. The maximum number of moves (iterations) for our hybrid algorithm is fixed to be 10 000 000 moves for an attempt of finding a proper k-coloring. For very hard instances, this number is fixed to be 200 000 000 moves. Let us note that a crossover are also countered as a move.

Experiments are carried out on benchmarks described in Section 2. Results are compared with the best ones published in the literature:

1. *Fleurent and Ferland*, a tabu search algorithm (denoted by 1a in the tables) and a genetic tabu algorithm (denoted by 1b) [6]. These algorithms use the pre-processing technique of [10] for graphs larger than 300 nodes.
2. *Costa*, an evolutionary hybrid algorithm *EDM* (denoted by 2) with the above pre-processing technique [3].
3. *Morgenstern*, two local search algorithms S_0 and S_1 (denoted by 3a and 3b) based on a particular neighborhood. Two distributed algorithms based on a population of configurations called M_0 and M_1 (denoted by 3c and 3d). Finally a hybrid algorithm called $M_0\backslash dXRLF$ (denoted by 3e) using a population of individuals initialized by a parallelized version of *Johnson et al's XRLF* algorithm [14].

Our genetic local search algorithm is defined by four parameters (P_1, P_2, P_3, P_4):

P_1 the number of individuals of the population.
P_2 the crossover rate T_x for UIS.
P_3 the random-walk rate p_{rw}.
P_4 the value of α for the tabu tenure (g being always fixed to 10).

For example, the setting (10,0.2,0.1,2) corresponds to a hybrid algorithm with 10 individuals using the crossover UIS with a rate of 0.2 %, the random walk mutation with a rate of 0.1 % (a rate of tabu mutation equal to 99.9 %) and a tabu tenure equal to $2 \times |CV|$+random(10).

Tables 1 to 4 compare our results (denoted by CISM - Crossover by Independent Sets and Mutation) with those of the best existing algorithms described above. The best known results are summarized in columns 2-4: the smallest known number of colors, the algorithms which produced such a coloring and the best computing time necessary to obtain this result. When an algorithm does not appear for an instance, it means either the algorithm was not tested on this instance or it did not find the indicated coloring. For example, the first line of Table 1 indicates that two algorithms (1b and 2) find a coloring with 49 colors for the ggg1.col-ggg5.col graphs and the most effective algorithm (cited first) needs more than 6 hours to obtain this result.

The last five columns give the results obtained by our algorithm (denoted by CISM). We give the total number of runs with the number of failures between brackets (5th column), the smallest number of colors obtained (6th column), the number of moves and average time on the successful runs (7th and 8th columns), and the values of parameters used by the algorithm (last column). Improved results *w.r.t* the best known ones are highlighted in bold. Note that computing times are given only for indicative purpose because the best known results have been obtained on different machines and each algorithm uses very different data structures. The most important criterion for a comparison is the quality of colorings found.

problem	Best known			CISM				
	k^*	method	time	runs	k^*	moves	time	param.
ggg1-ggg5	49	1b,2	20 376	3(0)	**48**	10 656 000	16 410	10,0.05,0.5,2
gggg1-gggg2	84	1b	147 600	1(0)	**83**	143 580 000	304 654	20,0.03,0.3,2

Table 1. Random Graphs from *Hertz and De Werra*

Table 1 gives a comparison on random instances between our algorithm CISM and the algorithms (1a), (1b) and (2) on large instances from *Hertz and De Werra* (5 graphs of 500 nodes and 2 graphs of 1000 nodes). k^* corresponds to the smallest number of colors obtained for all the instances of a same class.

We notice that our algorithm CISM obtains respectively 48- and 83-colorings and outperforms thus the best known results for these two classes of graphs. Let us recall that without a pre-processing (described above), none of 1a, 1b, 2 and our tabu algorithm is able to find colorings respectively with a number of colors smaller than 50 and 88 colors. This implies that the UIS operator contributes largely to the search power of our hybrid algorithm.

To further confirm this result, Table 2 shows the results on the random instances of *Johnson et al.* with 10, 50, and 90 % edge densities.

problem	Best known			CISM				
	k	method	time	runs	k	moves	time	param.
DSJC500.1.col	12	3d	5 452	5(0)	12	5 970 000	1 250	10,0.2,1,2
DSJC500.5.col	48	3e	49 000	5(3)	48	19 693 000	12 283	10,0.2,1,2
DSJC500.9.col	126	3e	158 400	10(8)	126	3 812 000	9 110	10,0.1,1,5
DSJC1000.1.col	21	3c	210	5(0)	**20**	16 757 000	8 768	10,0.2,1,2
DSJC1000.5.col	84	3e,1b	118 000	2(1)	**83**	120 522 000	270 159	20,0.03,0.3,2
DSJC1000.9.col	226	3c	65 774	1(0)	**224**	13 282 000	67 907	10,0.03,0.3,4

Table 2. Random Graphs from *Johnson et al.*

On these graphs smaller than 500 nodes, there are several local search methods which can find good colorings. For larger instances, few methods are able to obtain interesting results even with a powerful pre-processing. The best results are obtained by *Morgenstern* using the most complex versions of his population-based algorithms (3c, 3d and 3e). We notice from Table 2 that for graphs of 500 nodes, our algorithm finds consistently the best known colorings with shorter solving times on average. For the graphs of 1000 nodes, our results improves on the best known colorings for each instance whatever the edge density. In particular, for the largest instance DSJC1000.9.col, the number of colors needed is reduced from 226 to 224.

Tables 3 and 4 give more detailed comparisons between our algorithm and the best algorithm $M_0\backslash dXRLF$ (3e) on the two largest instances: DSJC1000.5.col and DSJC1000.9.col.

For DSJC1000.5.col, *Morgenstern* noticed that without the help of *XRLF* his algorithms 3c and 3d occasionally find 88-colorings. Only an initial generation with *XRLF* allows his algorithm $M_0\backslash dXRLF$ (3e) to find 84-colorings. It is also important to note that *XRLF* finds already 86-colorings and consequently its contribution in the algorithm 3e seems considerable.

Let us note that 84-colorings were also obtained by *Fleurent and Ferland* with their hybrid algorithm (1b) and our tabu algorithm. However 84-colorings are only possible with the help of *Hertz and de Werra's* pre-processing technique. Without this pre-processing, neither *Fleurent and Ferland's* nor our tabu algorithm is able to find colorings better than 89-colorings.

	Best known			CISM			
k	method	runs	time	runs	moves	time	param.
85	3e	10(0)	$\simeq$ 40 000	4(0)	36 750 000	104 500	10,0.03,0.3,2
84	3e	10(0)	118 000	4(0)	54 210 000	144 650	10,0.03,0.3,2
83	-	-	-	2(1)	120 522 000	270 159	20,0.03,0.3,2

Table 3. Coloring DSJC1000.5.col

Our hybrid algorithm CISM finds not only consistently best known 84-colorings and provides also a 83-coloring never found before. Note that initial configurations generated by *RLF* require about 105 colors for this graph and that such colorings can be easily obtained by any good local search algorithm. For example, our tabu algorithm finds 91-colorings with random initial configurations. Therefore, it is the UIS crossover that contributes to the above results.

In terms of computing time, our algorithm is slower than $M_0\backslash dXRLF$ (3e) for finding 84 and 85-colorings. However, a fair comparison remains difficult due to the mixture of 3e. Let us simply mention that the method 3e without $XRLF$ (i.e. 3c) needs 91 000 seconds to find 89-colorings.

	Best known			CISM			
k	method	runs	time	runs	moves	time	param.
226	3c	10(0)	75 800	10(0)	11 660 000	62 673	10,0.03,0.3,4
225	-	-	-	3(0)	12 938 000	67 245	10,0.03,0.3,4
224	-	-	-	4(3)	13 266 000	67 907	10,0.03,0.3,4

Table 4. Coloring DSJC1000.9.col

For DSJC1000.9.col, our algorithm finds the best known result with 226 colors and goes on to find colorings with 225 and even a 224 coloring. For this graph, the previous comparative remarks about DSJC500.9.col remain valid.

6 Conclusion

In this paper, we introduced a new crossover called UIS for the graph coloring problem. This crossover is based on the notion of unions of independent sets. The UIS operator, combined with a tabu search algorithm leads to a simple, yet very powerful hybrid algorithm. Experiments of this algorithm on well-known benchmarks have produced excellent results. Indeed, it finds the best known results for random graphs of 500 nodes (densities 10%, 50% and 90%). More importantly, it improves on the best known results for random graphs of 1000 nodes (densities 10%, 50% and 90%). (Improved results have also been obtained on other classes of graphs, but not reported here).

The algorithm should be improved both in terms of search power and computing time. First, variants of UIS and other hybridizing schema are worthy of investigation. Second, the presented algorithm can be easily parallelized, thus the computing time can be expected to be largely reduced.

References

1. D. Brélaz. New methods to color vertices of a graph. *Communications of ACM*, 22: 251-256, 1979.
2. M. Chams, A. Hertz, and D. De Werra. Some experiments with simulated annealing for coloring graphs. *European Journal of Operational Research*, 32: 260-266, 1987.
3. D. Costa, A. Hertz, and O. Dubuis. Embedding of a sequential procedure within an evolutionary algorithms for coloring problems in graphs. *Journal of Heuristics*, 1(1): 105–128, 1995.
4. R. Dorne and J.K. Hao Tabu search for graph coloring, T-coloring and set T-colorings. Chapter 3 of *Metaheuristics: Theory and Applications*, I.H. Osman, C. Roucairol and S. Voss (Eds.), Kluwer Academic Publishers, 1998.
5. E. Falkenauer A hybrid grouping genetic algorithm for bin-packing. *Journal of Heuristics*, 2(1): 5-30, 1996.
6. C. Fleurent and J.A. Ferland. Genetic and hybrid algorithms for graph coloring. *Annals of Operations Research*, 63: 437–463, 1995.
7. B. Freisleben and P. Merz. New genetic local search operators for the traveling salesman problem. *Proc. of PPSN-96*, *Lecture Notes in Computer Science 1141*, pp890-899, Springer-Verlag, 1996.
8. P. Galinier and J.K. Hao. Hybrid evolutionay algorithms for graph coloring. Submitted to the *Journal of Combinatorial Optimization*, 1998.
9. F. Glover and M. Laguna. *Tabu Search.* Kluwer Academic Publishers, 1997.
10. A. Hertz and D. De Werra. Using tabu search techniques for graph coloring. *Computing*, 39: 345–351, 1987.
11. D.S. Johnson, C.R. Aragon, L.A. McGeoch, and C. Schevon. Optimization by simulated annealing: an experimental evaluation; part ii, graph coloring and number partitioning. *Operations Research*, 39(3): 378–406, 1991.
12. F.T. Leighton. A graph coloring algorithm for large scheduling problems. *Journal of Research of the National Bureau Standard*, 84: 79–100, 1979.
13. P. Merz and B. Freisleben. A genetic local search approach to the quadratic assignment problem. *In Proc. of ICGA-97*, pp 465-472, Morgan Kaufmann Publishers, 1997.
14. C. Morgenstern. Distributed coloration neighborhood search. *Discrete Mathematics and Theoretical Computer Science*, 26: 335-358. American Mathematical Society, 1996.
15. C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization - Algorithms and Complexity.* Prentice Hall, 1982.

This article was processed using the LaTeX macro package with LLNCS style