Local Search for the Colouring Graph Problem. A Computational Study.*

Marco Chiarandini, Irina Dumitrescu, and Thomas Stützle

TU Darmstadt, FB Intellektik, FG Informatik Hochschulstr. 10, 64289 Darmstadt, Germany
{machud, irina, tom}@intellektik.informatik.tu-darmstadt.de

Abstract. The Graph Colouring Problem (GCP) is a well known $\mathcal{NP}$-hard problem with many theoretical and practical applications. In this paper we introduce a new local search algorithm based on a very large scale neighbourhood. We provide an extensive numerical comparison between this method and several other local search techniques considering also the embedding of the local search into more complex schemes like Iterated Local Search or Tabu Search.

1 Introduction

The Graph Colouring Problem (GCP) plays a central role in graph theory, has direct applications in real life [4, 5, 29], and is related to many other problems such as timetabling [28, 14, 34] and frequency assignment [21]. A K-colouring (assignment) of an undirected graph $G = (V, E)$, where V is the set of $|V| = n$ vertices and $E \subseteq V \times V$ the set of edges, is a mapping $\Psi : V \mapsto \{1, 2, \ldots, K\}$ that assigns a positive integer from $\{1, 2, \ldots, K\}$ (representing the colours) to each vertex. We say that a colouring is *feasible* if the end nodes of every edge in E have assigned a different colour, i.e. $\forall [u, v] \in E : \Psi(u) \neq \Psi(v)$. We call *conflict* the situation when two nodes between which an edge exists have the same colour associated to them. We say that a colouring is *infeasible* if at least one conflict occurs. Alternatively to the formulation as an assignment problem, the GCP can also be represented as a partitioning problem, in which a feasible K-colouring corresponds to a partition of the set of nodes into K sets $C_1, \ldots, C_K$ such that no edge exists between two nodes from the same colour class.

The decision version of the Graph Colouring Problem asks whether for a given graph G and a given K a feasible K-colouring can be found. In the optimisation version, the objective is to find the minimum number K such that a feasible K-colouring exists; this minimum number K is also known as the chromatic number χ_G of G. One approach to solving the optimisation GCP is to treat it as a sequence of decision problems. In this case, one can starts with an initial, possibly large number of colours K that is then repeatedly decreased by one, if the answer to the decision problem with K colours is found. An optimal solution to the optimisation problem is obtained when for a certain number of colours K no feasible assignment exists; this means that $\chi_G = K + 1$.

The GCP is $\mathcal{NP}$-hard [22]; exact algorithms are, for many instance classes, limited to rather small size problems [8, 15, 23, 32]. Therefore, large instances are often solved by approximate methods that are used to find sub-optimal colourings. Most of the successful approximate algorithms rely on effective local search strategies. In this paper we introduce a new local search algorithm that searches a large neighbourhood, based on ideas introduced by Thompson et al. [35, 36]; the neighbourhood searched is a cyclic exchange neighbourhood, a generalisation of the 2-exchange neighbourhood. We experimentally compare the use of this local search against other known local search techniques for the GCP. The computational results show an improved performance of our local search on several classes

* Technical Report AIDA-03-01 at FG Intellektik, FB Informatik, Technische Universität Darmstadt.

of instances when using the plain local search. However, the advantage of using our local search becomes less clear when incorporating the different local search schemes into Iterated Local Search and Tabu Search. In fact, in the latter case, only for specific graph classes our local search performs better than search methods that use simpler neighbourhoods.

Our paper is structured as follows. In Section 2 we shortly review approximate algorithms for the GCP, describe the known local search methods that we use in our computational study, and introduce our new local search algorithm. We present numerical results in Section 3 and conclude in Section 4.

2 Approximate algorithms for the GCP

Approximate algorithms for the GCP can broadly be classified into construction heuristics and improvement algorithms. Constructive algorithms typically build feasible solutions by iteratively colouring nodes of the graph and generate feasible colourings. Construction heuristics are typically used to generate an initial solution, from where improvement algorithms start their search. One well known example is the *greedy algorithm*. Given a set of empty colour classes $\{C_1, \ldots, C_K\}$ (with $K = |V|$) and a permutation π of $\{1, \ldots, n\}$ the greedy algorithm can be described as:

`greedy_algorithm`(G, π)

begin
  $K = |V|$.
  $C_1 = \{v_{\pi(1)}\}, C_2 = \emptyset, \ldots, C_K = \emptyset$.
  for $i = 2, \ldots, n$ **do**
    Let $h(i) = \min\{h : \forall v_j \in C_h, [v_i, v_j] \notin E\}$.
    $C_{h(i)} = C_{h(i)} \cup \{v_{\pi(i)}\}$.
  enddo
  Let $K = \max\{h : C_h \neq \emptyset\}$.
  Return the K-colouring, i.e. K and $C_1, \ldots, C_K$.
end

Clearly, the result of this greedy algorithm depends on the permutation π used [13]; it can further be improved by applying it iteratively using appropriately determined permutations [13]. Several other constructive algorithms exist that use more sophisticated selection criteria; these include DSATUR [7] and Recursive Largest First [28]. These algorithms may be further enhanced by using backtracking [6, 7, 27, 32].

The most commonly used improvement method is local search. There are two main ways of applying local search to the GCP.

1. *Solve the GCP as a sequence of decision problems with fixed number of colours.* At each iteration a decision GCP is solved. The evaluation function to be minimised is typically taken to be the number of conflicts of the assignments. When a feasible assignment (a solution with evaluation function value zero) is found, the next decision problem is defined by reducing number of colours by one and reapplying local search. This is done until for a certain number of colours no feasible solution can be found. The algorithm from this class that we use in our paper is given next:

Algorithm 1 *An algorithm that solves the GCP as a sequence of decision GCPs*

Step 1:Generate a random permutation π of the set of nodes V.
$assignment$=`greedy_algorithm`(G, π).
Step 2: **while** $assignment$ is feasible **do**
candidate=`reduce_colours`$(G, K, C_1, \ldots, C_K)$.
if *candidate* is not feasible **then**
$assignment$=`local_search`$(candidate)$.
enddo

The `reduce_colours`$(G, K, C_1, \ldots, C_K)$ procedure that we use starts by removing a random colour and continues by recolouring the colour free nodes according to the DSATUR constructive heuristic of Brélaz [7].

2. *Solve the GCP leaving the number of colours variable.* In this case the number of colours used may increase and decrease at run time. Typical approaches either include two components into the evaluation function, one that leads the search towards assignments with fewer colours and the other that drives in feasible assignments, or always maintain feasible colourings [27]. The general algorithm for GCP that we use is much simpler in this case: an initialisation step, identical to Step 1 of Algorithm 1, followed by local search with the initial colouring being the one output by the initialisation step.

Central to any local search algorithm is the neighbourhood, which defines the set of solutions that can be reached from a current solution by one *move*, and the evaluation function used to rate solutions. Concerning neighbourhoods, it is clear that the larger the neighbourhood size, the better the solutions that can be reached in one single move; however, this comes at an increased cost of searching for improved solutions.

Therefore, for the GCP the most common neighbourhood is the 1-exchange neighbourhood of Hertz and De Werra [25], where at each local search step the colour assigned to one vertex is changed. This neighbourhood can be searched efficiently using appropriate data structures even when using a best-improvement pivoting rule.

However, a recent trend in the design of local search algorithms is to look for large neighbourhoods that can be explored efficiently. For many problems like the traveling salesperson problem [24, 30], the generalised assignment problem [38], and many others [11, 3], local search algorithms exploiting large neighbourhoods are currently at the core of new state-of-the-art algorithms.

In this paper, we investigate a new neighbourhood search method that uses a very large scale neighbourhood, and we propose an algorithm that allows us to search the neighbourhood efficiently. This is one of the first attempts to attack the GCP using local search in larger neighbourhoods than the 1-exchange one. The only other large neighbourhood searches we are aware of are the Kempe chains neighbourhood of Morgenstern and Shapiro [33] and the ejection chain approach of González-Velarde and Laguna [37]. We will compare the performance of the new neighbourhood search against three other known local search methods that we briefly describe next.

2.1 Known local search approaches to the GCP

The first local search described falls in the category of the methods that solve the GCP as a sequence of decision problems. The last two local search algorithms solve the GCP directly, leaving the number of colours variable.

Local search with 1-exchange neighbourhood. Given a colouring, a 1-exchange move changes the colour of exactly one node. The evaluation function we use counts the number of conflicts in the assignment.

Given K and an infeasible assignment, at each iteration of the local search we follow a best-improvement strategy that examines all possible 1-exchange moves to discover the maximal reduction in the evaluation function. If several 1-exchange moves produce the same result, one of them is chosen uniformly at random [16, 18]. To reduce the size of the neighbourhood, we consider only those moves that affect vertices that are currently involved in a conflict. To speed up the evaluation of moves we use standard speed-up techniques [18] that allow to perform the first move in time $\mathcal{O}(n^2K)$, while each subsequent move can be done in $\mathcal{O}(nK)$ in the worst case (however, the neighbourhood evaluation is much faster for sparse graphs [18]).

Local search with penalty function. The penalty function local search uses a 1-exchange neighbourhood and the number of colours K is left variable. Formally, given a partition $\mathcal{C} = (C_1, \ldots, C_K)$, $1 \leq K \leq |V|$, a neighbouring partition is obtained by changing the colour of one node, i.e., a node v_i is removed from the class $C_{h(i)}$ to which it belongs, and it is moved into a class C_j, $1 \leq j \leq K+1$. If $j = K+1$, then a new colour was introduced. The evaluation function we consider is the one proposed by Johnson et al. [27]. The function has two components: the first one favours large colour classes (and thus empties the smaller ones and biases the search towards a low number of colours), while the second one reduces the number of conflicts. If $E(C_i)$ is the set of edges from E that have both ends in C_i, the evaluation function is:

$$f(S) = -\sum_{i=1}^{K} |C_i|^2 + \sum_{i=1}^{K} 2|C_i||E_i|. \tag{1}$$

Local minima for this function correspond to feasible colourings [27]. Based on preliminary experimental results, we decided to adopt a first improvement strategy.

Local search with Kempe chain neighbourhood. A *Kempe chain* is the set of nodes that form a connected component in the subgraph G' of G induced by the nodes that belong to two colour classes C_i and C_j, $i \neq j$. A Kempe chain *interchange* produces a new *feasible* colouring by swapping the colour class labels assigned to the vertices belonging to some specified Kempe chain. The neighbourhood of a given partition $\mathcal{C}$ is the set of feasible colourings that can be obtained from $\mathcal{C}$ by performing a Kempe chain interchange.

Let T be a Kempe chain in the subgraph G'. A Kempe chain interchange produces a colouring by replacing C_i with $(C_i \setminus T) \cup (C_j \cap G')$ and C_j with $(C_j \setminus T) \cup (C_i \cap G')$. We note that if $T = C_i \cup C_j$, the interchange would be simply a relabelling of the colour classes and therefore is to be avoided. The evaluation function that we use is:

$$f(A) = -\sum_{i=1}^{K} |C_i|^2. \tag{2}$$

Based on preliminary experiments, we chose to use a first improvement strategy. If more than one Kempe chain is available, we choose the best one, breaking ties with random selections.

2.2 Very large scale neighbourhood search

We consider the general neighbourhood proposed by Thompson et al. [35, 36] for partitioning problems. Since the graph colouring problem can be modelled as a partitioning problem, the neighbourhood that we introduce next can be used directly by our local search approach. In this approach, we attack the GCP as a series of decision problems.

The neighbourhood we introduce is the *cyclic exchange neighbourhood*, which can be seen as a generalisation of the well known two exchange neighbourhood. Instead of swapping only two elements from two different subsets of a current colouring, like in the case of a two exchange move, the cyclic exchange moves several elements, each belonging to a different subset. Formally, a *cyclic exchange* between l subsets (without loss of generality we can assume the sets to be $C_1, \ldots, C_l$ and the elements $v_1, \ldots, v_l$ with $v_i \in C_{h(i)}$) is represented by a cyclic permutation π of length l, $\pi \neq \mathbf{1}$, where $\pi(i) = j$ means that vertex v_i moves from subset $C_{h(i)}$ into subset C_j. The cyclic exchange modifies the sets of the partition and therefore their costs. The cost difference for each subset will be the difference between the cost of the subset after performing the cyclic exchange and the cost of the subset before the exchange. The *cost of the cyclic exchange* is defined as being the sum of all cost differences over all subsets in the partition. A colouring $\mathcal{C}'$ is said to be a neighbour of the colouring $\mathcal{C}$ if it is obtained from $\mathcal{C}$ after performing a cyclic exchange. The set of all neighbours of $\mathcal{C}$ defines the *cyclic exchange neighbourhood* of $\mathcal{C}$. Since this neighbourhood is of exponential size, much larger than the one or two exchange neighbourhoods, we can expect better quality solutions, provided that we can search the neighbourhood efficiently.

Ahuja et al. [2] showed that the problem of finding the best neighbour within a cyclic exchange neighbourhood can be modelled as the problem of finding the minimum cost cycle that uses at most one node from each subset in a new graph $G' = (V', A')$, called the *improvement graph*, induced by the graph $G = (V, E)$ and the current colouring (partition) $\mathcal{C} = \{C_1, \ldots, C_K\}$ considered. The set of nodes of G' is $V' = \{1, \ldots, n\}$, each node in V' corresponding to one node in V (i corresponds to v_i). The improvement graph contains the arc (i, j) if $v_i \in C_{h(i)}$, $v_j \in C_{h(j)}$, and $C_{h(i)} \neq C_{h(j)}$, i.e. v_i, v_j are coloured differently. The set of nodes V' is split into K subsets $T_1, \ldots, T_K$, induced by $\mathcal{C}$, i.e. the elements of T_h are in one-to-one correspondence with the elements of C_h, for each $h = 1, \ldots, K$.

If $i \in T_{h(i)}$ and $j \in T_{h(j)}$, we associate a cost $c_{(i,j)}$ with the arc $(i, j) \in A'$ equal to the difference between the cost of the set $T_{h(j)} \setminus \{j\} \cup \{i\}$ and the cost of the set $T_{h(j)}$. The cost of an arc (i, j) is defined to be the difference between the number of conflicts in $C_{h(j)} \setminus \{v_j\} \cup \{v_i\}$ and the number of conflicts in $C_{h(j)}$, that is, the cost of an arc is the difference in the number of conflicts when moving node i into colour class $C_{h(j)}$ and removing node j from $C_{h(j)}$. For the GCP, the cost associated with any arc (i, j) in G' can also be defined as the difference between the number of arcs adjacent to v_i in $C_{h(i)} \setminus \{v_j\}$ and the number of arcs adjacent to v_j in $C_{h(j)}$. Hence the construction of the improvement graph can be done in $\mathcal{O}(n^2 K)$, while its updating can be done in $\mathcal{O}(n^2)$.

Thompson and Orlin [35] showed that there is a one-to-one correspondence between the cyclic exchanges in G, with respect to $\mathcal{C}$, and the subset-disjoint cycles in the improvement graph G'. Clearly, an improving cyclic exchange will correspond to a subset disjoint cycle of negative cost.

Defined as above, a cyclic exchange would always maintain the cardinality of the subsets of the partition of V considered. This is avoided by Ahuja et al. [2] by introducing a subset of dummy nodes. Clearly, any cyclic exchange that would involve a dummy node would modify the cardinality of the subsets. Such an exchange can be seen as being based

on a *path*, instead of a cycle. In what follows we will make the distinction between such exchanges. We will therefore refer to cyclic exchanges and to path exchanges (see Figure 1). Finally, we note that in our approach we are only interested in finding *improving* cyclic or path exchanges. Therefore we try to find only subset disjoint *negative* cost cycles.

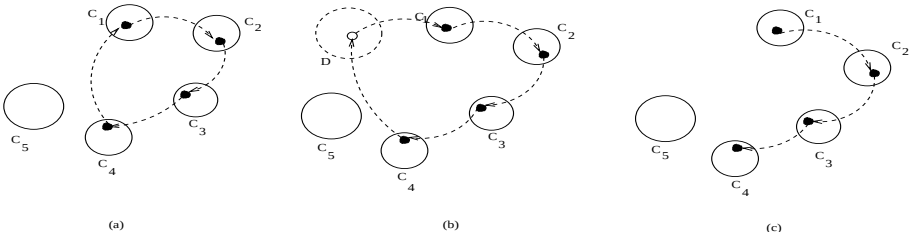

Fig. 1. (a) Cyclic exchange. (b) Cyclic exchange using a dummy node (from set D). (c) Path exchange.

We now present an exact algorithm based on dynamic programming ideas implemented via *labels* [1]. We then propose several ways of truncating it, such that the algorithm becomes an efficient heuristic. The algorithm makes use of the idea of subset disjoint *paths*, i.e. paths that visit every subset at most once. It is clear that if an arc exists between the last and the first node of such a path, then the path can be closed to form a a subset disjoint cycle.

In our algorithm, each subset disjoint path will be represented by a label. For such a path $p = (i_1, \ldots, i_r)$, we call i_1 the start node of p, denoted by $s(p)$ and i_r the end node of p, denoted by $e(p)$. We associate a binary vector $w(p) \in \{0,1\}^K$ with the path p, where $w_h(p) = 1$ if and only if p visits the subset T_h. The cost of p, denoted by $c(p)$, is the total cost of the arcs in the path, i.e. $c(p) = \sum_{j=1}^{r-1} c_{(i_j, i_{j+1})}$. The label associated with p is defined as the four-tuple $(s(p), e(p), c(p), w(p))$.

We say that (s^1, e^1, c^1, w^1) *dominates* (s^2, e^2, c^2, w^2) if $s^1 = s^2$, $e^1 = e^2$, $c^1 \leq c^2$, $w^1 \leq w^2$, and the labels are not equal; in extension of this definition we say that the path p^1 dominates the path p^2 if the label corresponding to p^1 dominates the label corresponding to p^2. We call *treatment* of a path the extension of that path along all outgoing arcs. After treatment, a path is marked as *treated*. At any time, only paths that have not previously been treated are selected for treatment. We now give in detail an algorithm for finding subset disjoint negative cost cycles, based on an all-pairs shortest path approach.

Algorithm 2 *An algorithm that finds subset disjoint negative cost cycles*

Step 1: Set $\mathcal{P} = \{(i,j) : (i,j) \in A', c_{(ij)} < 0\}$, negative cost paths of length 1.
Mark all paths in $\mathcal{P}$ as untreated.
Initialise the best cycle $q* = ()$ and $c^* = 0$.
Step 2: **for** each $p \in \mathcal{P}$ **do**
if $(e(p), s(p)) \in A'$ and $c(p) + c_{(e(p),s(p))} < c^*$ **then**
q^* = the cycle obtained by closing p and $c^* = c(q^*)$.
enddo
Step 3: **for** $l = 2, \ldots, K$ **do**
while there exists an untreated path in $\mathcal{P}$ of lengh l **do**
Select some untreated path $p \in \mathcal{P}$ of length l.

for each $(e(p), j) \in A'$ s.t. $w_{x(j)}(p) = 0$ and $c(p) + c_{(e(p),j)} < 0$ **do**
Add the extended path $(s(p), \ldots, e(p), j)$ to $\mathcal{P}$ as untreated.
enddo
if $(j, s(p)) \in A'$ and $c(p) + c_{(e(p),j)} + c_{(j,s(p))} < c^*$ **then**
q^* = the cycle obtained by closing the path $(s(p), \ldots, e(p), j)$.
$c^* = c(q^*)$.
if some criteria are satisfied **then** remove some dominated paths from $\mathcal{P}$.
enddo
enddo

Algorithm 2 is an exact algorithm. However, since the number of colours can be very large, the algorithm used in its exact form can become very inefficient. We therefore chose to use the algorithm as a heuristic, by limiting it in some ways. Firstly, we put a limit on the length of the cycle: at Step 3, l will not go all the way to K, but stop when a limit imposed by us is reached. Secondly, we reduce the state space of the algorithm by imposing a limit M on the number of untreated labels kept at any time. We order the untreated labels in increasing order of their cost. In our implementation the labels are kept on pairs of nodes, and only the cheapest M untreated labels on any pair will be recorded. Finally, when a new label is created, we do not check dominance along the whole list of labels, but only up to the position in the list where the new label is inserted.

For the case when we solve the GCP as a sequence of decision problems, we chose to consider not only the neighbourhood structures that we defined (1-exchange, cyclic and path exchange) but also combinations of them. Many such combinations can be considered; in this paper however we only use four of them:

- **C+1:** Union of cyclic and 1-exchange. At each iteration of the local search the best move in these neighbourhood is chosen.
- **C+P+1:** Union of cyclic exchange, path exchange, and 1-exchange. We note that when negative cost arcs (paths of length one) exist, then the best move in a path exchange neighbourhood coincides with the best move in an 1-exchange neighbourhood. A 1-exchange move however can be a move that does not necessarily improve the current colouring, equivalent to a path of length one of positive cost that, in our approach, would not be discovered by Algorithm 2.
- **(C+P)–1:** Systematic change of two neighbourhoods: the union of cyclic and path exchange and the 1-exchange. An improving move is searched in the first neighbourhood until none can be found. Then the second neighbourhood is examined.
- **1–(C+P):** Like the previous one, with the order of the two neighbourhood reversed.

In the following we will refer to these variants also as very large-scale neighbourhood searches (VLSN).

3 Experimental Results

We now present and discuss numerical results for techniques that use the local search methods described in the previous section. We also give results obtained when straightforward implementations of metaheuristics are used.

Each local search algorithm will be run many times on a single instance. For each instance we check whether the differences between the solutions found by the algorithms are statistically significant or not. To do so, we first use the Kruskal-Wallis one-way analysis

of variance by ranks on all the samples of one instance. If this test rejects the null hypothesis that all algorithms give the same results, we test all the possible pairs of samples by means of the pairwise Wilcoxon rank-sum test using the method of Holm to take the multiple comparisons into account [12]. In the presentation of the results we call an algorithm *better* than another on a single instance only if it is significantly better according to these tests, at a significance level of 5%.

Benchmark Problems We tested the algorithms on some of the benchmark instances proposed for COLOR02/03 [10]; the instances we used can be divided into the following classes.

- *Random graphs*. Instances from Johnson [27] in which for a given set of vertices the graph is obtained by including each possible edge with a probability p. The chromatic number for these instances is unknown.
- *Leighton graphs*. Structured graphs generated by a procedure that uses the number of vertices, the desired chromatic number, the average vertex degree and a random vector of integers to generate a certain number of cliques [28]. The chromatic number of these instances is known.
- *Queens graphs*. Given an $n \times n$ chess board, a queen graph is a graph with n^2 nodes, each corresponding to a square of the board. Two nodes are connected by an edge if the corresponding squares are in the same row, column or diagonal. The chromatic number is not known; however, it is known to be at least n.
- *Class scheduling graphs*. Graphs from course timetabling. The vertices corresponding to classes; an edge between two vertices exist if a student has to attend both classes or if both classes are taught by the same person [29].

In the following we present only computational results for a subset of the available instances; for the other instances of the specific classes the computational results show the same tendency.

Simple local search In the first experiment, we are interested in the distribution of the quality of the colourings returned by the different local search algorithms. Each local search is run 100 times for each instance of our test bed. The initial solutions were obtained using the greedy algorithm with different random permutations. Results for some instances are plotted in Figure 2 using box-plots.

The best performance is obtained by the very large scale neighbourhood variants or the Kempe chain algorithm. This is more evident on the random graphs with low density, the queens, and the Leighton graph. However, for higher density random graphs the Kempe Chains perform better, which is also the case for the scheduling instance. The worst performance overall is obtained by the 1-exchange local search algorithm. However, this is not necessarily due only to the neighbourhood definition, because the penalty function algorithm, which changes the colour of only one node for each move, performs much better on some few instances; we believe that the difference is rather due to the different evaluation functions used and the way of attacking the GCP (sequence of decision problems or variable number of colours).

Random restart local search The computation time was not taken into account when we were looking at the distribution of the colourings found by the various local searches. Therefore, we repeated the same type of experiments by allowing each algorithm to run for

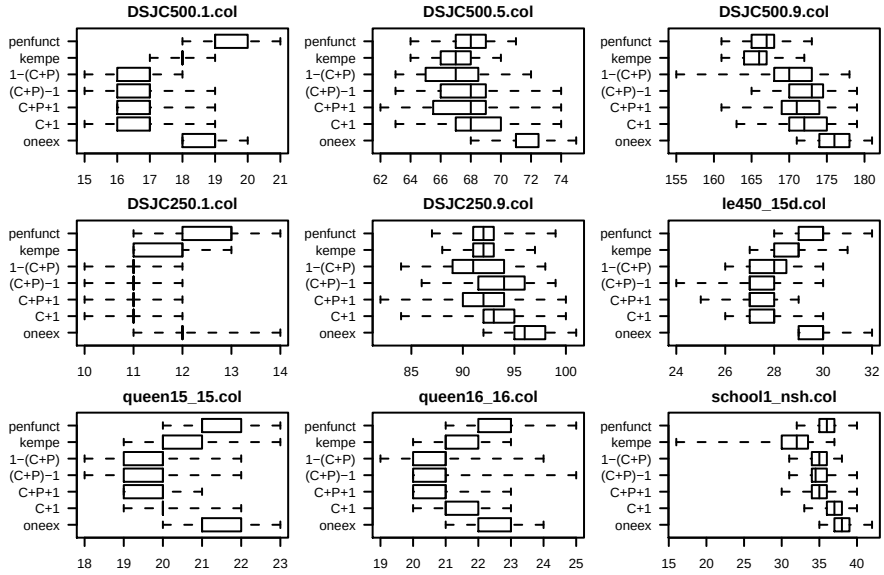

Fig. 2. For each instance we give the distribution of colours found by the local searches using box-plots. The central box shows the data between the quartiles, the median is represented by a line in the central box. "Whiskers" extend to the very extremes of the data. If there is only one line visible, this indicates that all trials with one algorithm returned the same number of colours. Each algorithm was run using 100 independent trials.

the same amount of time; this is equivalent to randomly restarting the local searches from random greedy solutions as often as possible in the allocated time and using as an output the best solution found within the time limit. The time limit is fixed to be the time required to run 20 random restart using the slowest local search, (C+P+1). We run 10 such trials per local search and instance and we record the number of colours found. Distributions of some of the results obtained with 10 trials for each algorithm are given in Figure 3.

In general, for all the algorithms the colourings returned are slightly better, however the relative ranking of the algorithms remains almost unchanged. It has to be said that for all the local searches, the results are in part quite far from the chromatic numbers or the best known solutions. We therefore try to improve our local searches by means of some straightforward implementations of metaheuristics.

Iterated Local Search Iterated local search (ILS) is a convenient way of iterating over a local search without incurring the known disadvantages of a random restart [31]. ILS does so by perturbing a locally optimal solution s, leading to some intermediate solution s', and then applying local search to s', which finally leads to a (hopefully) new local optimum s''. An acceptance criterion then says whether or not we apply the next perturbation to s or s''.

An application of ILS to the GCP is described in Chiarandini and Stützle [9]. Here we use the same perturbation scheme. Shortly, when a local optimum is reached a number of randomly chosen colour classes are emptied and the vertices reallocated by the greedy algorithm, avoiding the re-assignment to the same colour class. The number of colours to remove is given by the current number of colours multiplied by a factor γ, $\gamma < 1$. We fixed

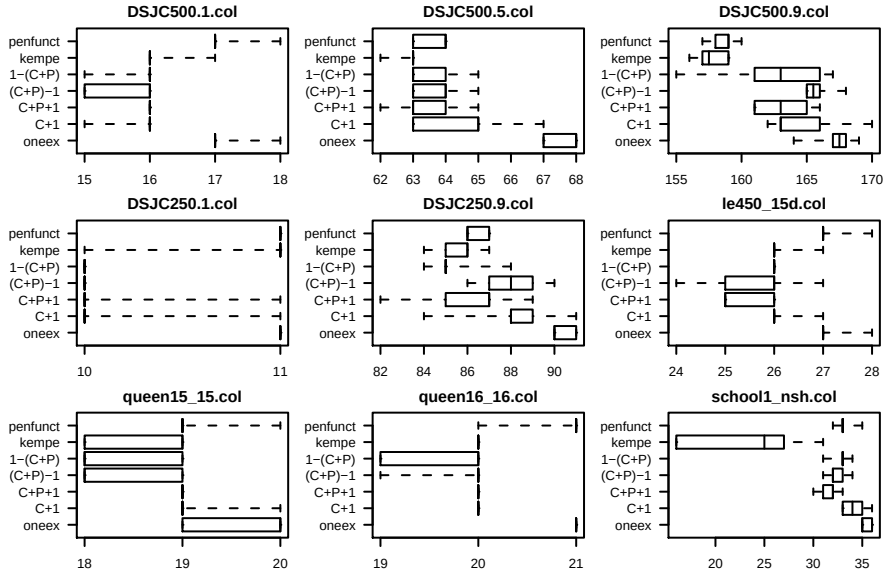

Fig. 3. Computational results of 10 independent trails for each algorithm with random restart. For an explanation of the box-plots we refer to the caption of Figure 2. See text for more details.

γ to 0.1. The perturbation is always applied to s'', that is, s'' is accepted independent of its quality.

We run seven variants of ILS that differ only in the local search chosen. Each variant was given the same maximum time as in the random restart case; the computational results are given in Figure 4. In general, embedding the local searches into an ILS algorithm improves the results over using a random restart in almost every case (compare the location of the box-plots for the single local search algorithm of Figure 4 to those of Figure 3). On many of the instances the relative order of the algorithms with respect to solution quality is maintained, with the following exceptions: (i) the penalty function local search is now much more competitive compared to the Kempe Chains and the VLSN local searches, (ii) also the 1-exchange local search appears more competitive on the non-random graphs than before (for example, it performs on par with VLSN local searches on the queen15 graph and obtains a better median colouring on the school graph), (iii) using the VLSN local searches still gives the best results on most of the instances with the main exception being the high density random graphs, where the penalty function local search is best, and the school graph. Hence, these results confirm that effective local search algorithms as obtained by searching large neighbourhoods can give an advantage over a simpler but faster one exchange neighbourhood, at least when incorporated into an ILS.

Tabu Search Currently, Tabu Search (TS) algorithms are at the core of the best performing local search approaches to the GCP [17, 16, 19, 20]. Therefore, it is an obvious next step to try to enhance the different local search algorithms tested through the use of basic tabu search components. In particular, we adopted the simple TS scheme proposed in [16, 25] for the 1-exchange and the k-exchange neighbourhood by forbidding the re-assignment of a vertex to a colour for a number of iterations (*tabu length*) given by $\mathsf{Random}(10) + \delta \times |n_c|$

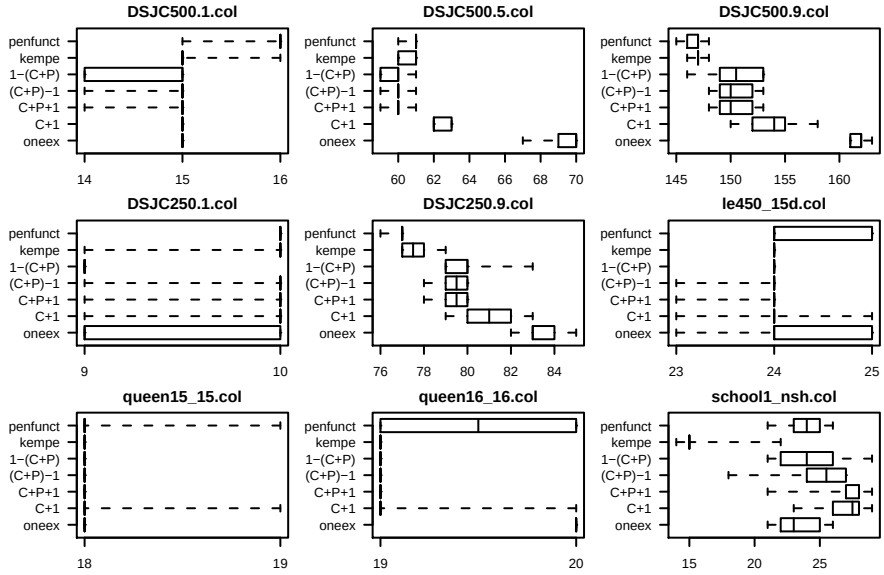

Fig. 4. Box-plots of 10 trials for each algorithm enhanced with Iterated Local Search. The stopping criterion was the same time limit used for obtaining the plots of Figure 3. For an explanation of the box-plots we refer to the caption of Figure 2.

with n_c being the number of conflicts in the assignment and Random(10) being a random number uniformly distributed between one and ten. The parameter δ is set to 0.6, a value which gave good results on average. For the penalty function approach we use the same prohibition criterion but with a tabu length equal to $Random(10) + \delta \times 2 \times K$. In the Kempe Chains neighbourhood we forbid a chain between two colour classes C_i and C_j if at least one vertex $v \in C_i$ of the chain was in colour class C_j; the tabu length is set to $Random(10) + \delta \times K$. We always keep δ fixed to 0.6.

These algorithms were again run for 10 trials using the same time as before. The computational results with this TS scheme, which are given in Figure 5, show a surprising result: Now, the 1-exchange neighbourhood actually gives the best performance on most of the instances, sometimes by a quite large margin. Similarly, the VLSN scheme that has the largest fraction of 1-exchanges now gives on most of the instances slightly better results than the other VLSN variants, different from what was observed before. There may be several reasons for this change in behaviour. One reason may be that a different TS features should be used for the local search that use large neighbourhoods. In fact, the TS scheme we used was already optimised by several researches since TS was first applied to the GCP [17, 16, 19, 20]; therefore, there may be a bias towards favouring the 1-exchange neighbourhood. Another reason may be that the computation times are too short for the large neighbourhood algorithms to catch up with the TS on the 1-exchange neighbourhood. In fact, some limited experiments with longer computation times (and a different starting heuristic based on DSATUR) have shown that using the C+1 neighbourhood in the TS can beat TS on the 1-exchange neighbourhood on the Queens graphs. However, more experiments are required to confirm this result also for other types of graphs.

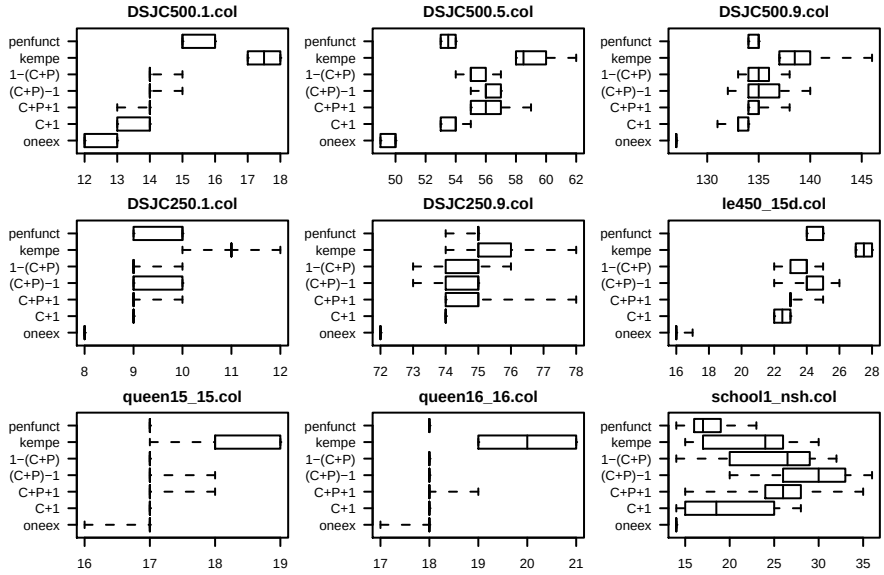

Fig. 5. Box-plots of 10 trials for each algorithm enhanced with Tabu Search. The stopping criterion used was the same as in Figure 3

4 Conclusions

In this work we studied a new local search algorithm using a very large-scale neighbourhood for the GCP. This algorithm, in addition to changes of the colour of one single vertex, allows to swap the colours of a set of vertices, where each vertex belongs to a different colour, in a cyclic exchange. We also presented an exact and a heuristic algorithm to search effectively through this new neighbourhood for improving exchanges. The algorithm solves a Subset Disjoint Negative Cost Cycle Problem with a dynamic programming approach [26]. Computational results comparing four VLSN variants to other existing local search algorithms showed that (i) the VLSN variants return effectively better quality solutions on a range of benchmark instances except of mainly large density random graphs, where a Kempe chains local search works better, (ii) when using VLSN local search as a black box local search in an ILS algorithm, significant improvements over a random restart algorithm can be obtained and VLSN local search is the method of choice for low density graphs, (iii) when opening the black box to include tabu search features, the smallest neighbourhood we tested, a 1-exchange neighbourhood, appears to be preferable over local search algorithms using large neighbourhoods at least when run only for short computation times. Preliminary results indicate that for some graph classes with increased run-times the VLSN local search with tabu search features improves over the 1-exchange neighbourhood in a similar tabu search algorithm. However, still further research efforts are required to make large-scale neighbourhood search techniques fully competitive. However, we strongly think that the results we report show that using large neighbourhoods offers a promising possibility to advance the state-of-the-art in GCP solving.

References

1. R.K. Ahuja, N.L. Boland, and I. Dumitrescu. Exact and heuristic algorithms for the subset disjoint minimum cost cycle problem. Working Paper.
2. R.K. Ahuja, T.L. Magnanti, and D. Sharma. Very large-scale neighbourhood search. *International Transactions in Operational Research*, 7:295–302, 2000.
3. R.K. Ahujaa, J.B. Orlin, and D. Sharma. Multi-exchange neighborhood search algorithms for the capacitated minimum spanning tree problem. *Mathematical Programming*, 91:71–97, 2001.
4. M. Allen, G. Kumaran, and T. Liu. A combined algorithm for graph-coloring in register allocation. In D. S. Johnson, A. Mehrotra, and M. Trick, editors, *Proceedings of the Computational Symposium on Graph Coloring and its Generalizations*, Ithaca, New York, USA, 2002.
5. N. Barnier and P. Brisset. Graph coloring for air traffi c fbw management. In *CPAIOR'02: Fourth Internation Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimisation Problems*, pages 133–147, Le Croisic, France, March 2002.
6. B. Bollob´as and A. Thomason. Random graphs of small order. *Annals of Discrete Math*, Random Graphs '83(28):47 – 97, 1985.
7. D. Br´elaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22(4):251–256, 1979.
8. M. Caramia and P. Dell'Olmo. Vertex coloring by multistage branch and bound. In D. S. Johnson, A. Mehrotra, and M. Trick, editors, *Proceedings of the Computational Symposium on Graph Coloring and its Generalizations*, Ithaca, New York, USA, 2002.
9. M. Chiarandini and T. Stützle. An application of iterated local search to graph coloring. In D. S. Johnson, A. Mehrotra, and M. Trick, editors, *Proceedings of the Computational Symposium on Graph Coloring and its Generalizations*, pages 112–125, Ithaca, New York, USA, 2002.
10. COLOR02/03: Graph Coloring and its Generalizations. http://mat.gsia.cmu.edu/COLOR02, January 2003.
11. R. K. Congram, C. N. Potts, and S. van de Velde. An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem. *INFORMS Journal on Computing*, 14(1):52–67, 2002.
12. W.J. Conover. *Practical Nonparametric Statistics*. John-Wiley and Sons, New York, USA, third edition edition, 1999.
13. J.C. Culberson. Iterated greedy graph coloring and the diffi culty landscape. Technical Report 92-07, Department of Computing Science, The University of Alberta, Edmonton, Alberta, Canada, June 1992.
14. D. de Werra. An introduction to timetabling. *European Journal of Operational Research*, 19:151–162, 1985.
15. I.M. Diaz and P. Zabala. A branch-and-cut algorithm for graph coloring. In D. S. Johnson, A. Mehrotra, and M. Trick, editors, *Proceedings of the Computational Symposium on Graph Coloring and its Generalizations*, Ithaca, New York, USA, 2002.
16. R. Dorne and J. Hao. A new genetic local search algorithm for graph coloring. In *Parallel Problem Solving from Nature - PPSN V, 5th International Conference*, volume 1498 of *LNCS*, pages 745–754. Springer-Verlag, Berlin, Germany, 1998.
17. R. Dorne and J.K. Hao. Tabu search for graph coloring, t-colorings and set t-colorings. In S. Voss, S. Martello, I.H. Osman, and C. Roucairol, editors, *Meta-heuristics: Advances and Trends in Local Search Paradigms for Optimization*, pages 77–92. Kluwer Academic Publishers, Boston, MA, USA, 1999.
18. C. Fleurent and J. Ferland. Genetic and hybrid algorithms for graph coloring. *Annals of Operations Research*, 63:437–464, 1996.
19. C. Fleurent and J. A. Ferland. Genetic and hybrid algorithms for graph coloring. In G. Laporte, I. H. Osman, and P. L. Hammer, editors, *Annals of Operations Research*, volume 63, pages 437–461. Baltzer Science Publishers, 1996.
20. P. Galinier and J. Hao. Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization*, 3(4):379–397, 1999.
21. A. Gamst. Some lower bounds for a class of frequency assignment problems. *IEEE Transactions of Vehicular Technology*, 35(1):8–14, 1986.

22. M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-Completeness*. Freeman, San Francisco, CA, USA, 1979.
23. A. Van Gelder. Another look at graph coloring via propositional satisfiability. In D. S. Johnson, A. Mehrotra, and M. Trick, editors, *Proceedings of the Computational Symposium on Graph Coloring and its Generalizations*, Ithaca, New York, USA, 2002.
24. K. Helsgaun. An effective implementation of the lin-kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126:106–130, 2000.
25. A. Hertz and D. de Werra. Using tabu search techniques for graph coloring. *Computing*, 39:345–351, 1987.
26. I.Dumitrescu. *Constrained path and cycle problems*. PhD thesis, The University of Melbourne, 2002.
27. 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.
28. F.T. Leighton. A graph coloring algorithm for large scheduling problems. *Journal of Research of the National Bureau of Standards*, 85:489–506, 1979.
29. G. Lewandowski and A. Condon. Experiments with parallel graph coloring heuristics and applications of graph coloring. In David S. Johnson and Michael A. Trick, editors, *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, 1993*, volume 26, pages 309–334. American Mathematical Society, 1996.
30. S. Lin and B.W. Kernighan. An effective heuristic algorithm for the travelling salesman problem. *Operations Research*, 21:498–516, 1973.
31. H. R. Lourenço, O. Martin, and T. Stützle. Iterated local search. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, volume 57, pages 321–353. Kluwer Academic Publishers, Norwell, MA, 2002.
32. A. Mehrotra and M. Trick. A column generation approach for graph coloring. *INFORMS Journal On Computing*, 8(4):344–354, 1996.
33. C. Morgenstern and H. Shapiro. Coloration neighborhood structures for general graph coloring. In *Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, pages 226–235. Society for Industrial and Applied Mathematics, 1990.
34. A. Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13:87–127, 1999.
35. P.M. Thompson and J.B. Orlin. The theory of cycle transfers. Technical report, Operations Research Centre, MIT, Cambridge, MA, 1989.
36. P.M. Thompson and H.N. Psaraftis. Cyclic transfer algorithms for multivehicle routing and scheduling problems. *Operations Research*, 41:70–79, 1993.
37. J. L. Gon´alez Velarde and M. Laguna. Tabu search with simple ejection chains for coloring graphs. *Annals of Operations Research*, 2003, to appear.
38. M. Yagiura, T. Ibaraki, and F. Glover. An ejection chain approach for the generalized assignment problem. Technical Report 99013, Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, 1999.

A The set of instances

In Table 1 we list the instances used in the experimental analysis. We add instances from three other classes not included in the text:

- *Optical network graphs*. Graphs from real-life optical network design problems. Each vertex corresponds to a light-path in the network and edges correspond to intersecting paths.
- *Quasigroups graphs*. Constraints on a multiplication table of a quasigroup defines a Latin Square. A Latin Square is a table such that no symbol occurs more than once in a row or in a column. Graphs model this table and the chromatic number is the *order* N of the quasigroup.
- *Latin Squares Graph*. Graph representing a problem from the design theory, relating to Latin squares. The graph has 900 vertex and independent sets no larger than 10 vertices. It is an open question whether or not this graph can be coloured with 90 colours. It is considered an hard problem [29].

inst	n	ρ	q	$sd(q)$	$\overline{q}$	$sd(\overline{q})$	DSATUR	$\chi(G)$	LB	best known
DSJC125.5.col	125	0.51	62.26	5.304	0.502	0.0428	22	-	12	17
DSJC250.1.col	250	0.1	25.74	5.119	0.103	0.0206	10	8	8	9
DSJC250.5.col	250	0.5	125.3	7.803	0.503	0.0313	37	-	13	22
DSJC250.9.col	250	0.9	223.2	4.604	0.896	0.0185	92		35	72
DSJC500.1.col	500	0.1	49.83	6.67	0.0999	0.0134	16	-	6	12
DSJC500.5.col	500	0.5	250.5	11.05	0.502	0.0221	65	-	16	48
DSJC500.9.col	500	0.9	449.7	6.366	0.901	0.0128	170	-	42	126
DSJC1000.1.col	1000	0.1	99.26	9.512	0.0994	0.00952	27	-	6	20
DSJR500.5.col	500	0.47	235.4	64.7	0.472	0.13	130		26	124
le450-15d.col	450	0.17	74.44	21.24	0.166	0.0473	24	15	15	15
queen10-10.col	100	0.59	29.4	2.344	0.297	0.0237	14	-	-	-
queen11-11.col	121	0.55	32.73	2.582	0.273	0.0215	15	11	11	12
queen12-12.col	144	0.5	36.06	2.818	0.252	0.0197	16	-	-	-
queen13-13.col	169	0.47	39.38	3.055	0.234	0.02	17	13	-	14
queen14-14.col	196	0.44	42.71	3.291	0.219	0.02	19	-	-	-
queen15-15.col	225	0.41	46.04	3.528	0.206	0.0157	21	-	-	17
queen16-16.col	256	0.39	49.38	3.764	0.194	0.0148	23	-	-	18
school1-nsh.col	352	0.24	83.02	35.17	0.237	0.1	27	14	14	14
wap01a.col	2368	0.04	93.64	48.21	0.0396	0.0204	47	-	-	42
wap06a.col	947	0.1	92.02	48.84	0.0973	0.0516	46	-	-	42
qg.order30.col	900	0.06	58	0	0.0645	0	36	30	30	30
qg.order40.col	1600	0.05	78	0	0.0488	0	45	40	40	40
qg.order60.col	3600	0.03	118	0	0.0328	0	69	60	60	60
latin-square-10.col	900	0.76	683	0	0.76	0	132	-	-	99

Table 1. In an instance with n vertices and density ρ, each vertex v_i, $i = 1, \ldots, n$ has a degree $q_i \in \{0, 1, \ldots, n-1\}$. We report the average vertex degree, i.e., $q = \frac{1}{n}\sum_{i=n}^{n} q_i$ and the standard deviation. Moreover, to make direct comparisons among instances with different number of vertices, we also report the normalisation of q: $\overline{q} = \frac{q}{n-1}$ and its standard deviation. Concerning the colouring number we report the value found by the constructive heuristic DSATUR of Br´elaz, the chromatic number, when known, a proved lower bound and the best approximate solution found in the literature.

B Further results

In the two following tables we report further results of two main experiments described in the text. In the third, we show instead peak performances reached by the algorithm studied when the Hybrid approach of Tabu Search and Iterated Local Search is applied.

	1-exchange			Kempe Chains			PenFunct		
inst	Min	succ	mean.iter	Min	succ	mean.iter	Min	succ	mean.iter
queen10-10.col	13	1	0	13	19	19	13	4	9
queen11-11.col	15	10	3	14	6	25	15	16	14
queen12-12.col	16	8	3	15	1	33	16	9	25
queen15-15.col	20	7	3	19	7	47	20	14	35
queen16-16.col	21	5	4	20	3	43	21	7	32
DSJC250.1.col	11	6	2	11	28	51	11	1	35
DSJC250.5.col	39	3	2	36	3	60	37	1	43
DSJC250.9.col	92	5	5	88	1	28	87	1	29
DSJC500.1.col	18	44	3	17	20	120	18	10	50
DSJC500.5.col	68	1	2	64	6	125	64	1	145
DSJC500.9.col	171	5	4	161	3	59	161	1	53
DSJC1000.1.col	29	9	6	27	1	293	29	6	167
DSJR500.1.col	13	6	2	13	30	173	13	2	3
DSJR500.5.col	141	1	0	141	2	56	141	3	34
le450-15d.col	29	35	2	27	2	122	28	4	67
qg.order30.col	32	6	3	30	26	94	32	17	51
school1-nsh.col	35	5	5	16	3	431	32	1	172
wap06a.col	51	3	4	49	3	447	51	5	119
latin-square-10.col	144	10	4	124	10	255	130	10	232

	C+1				C+P+1				(C+P)-1				1-(C+P)			
inst	Min	succ	mean.iter	rate	Min	succ	mean.iter	rate	Min	succ	mean.iter	rate	Min	succ	mean.iter	rate
queen10-10.col	12	1	17	1.13	12	2	15	0.53	12	2	12	0.21	13	52	10	1.85
queen11-11.col	14	23	9	0.52	14	33	8	0.37	14	34	8	0.07	14	38	13	1.99
queen12-12.col	15	10	11	0.52	15	20	11	0.27	15	13	11	0.10	15	23	17	1.99
queen15-15.col	19	22	12	0.59	19	38	11	0.35	18	5	21	0.06	18	3	32	1.94
queen16-16.col	20	20	14	0.50	20	36	14	0.38	20	26	14	0.12	19	1	43	3.30
DSJC250.1.col	10	11	14	0.34	10	21	15	0.18	10	21	13	0.01	10	18	26	2.37
DSJC250.5.col	36	1	19	0.36	35	5	28	0.26	35	1	25	0.19	35	1	49	2.27
DSJC250.9.col	84	1	20	1.00	82	1	32	0.60	86	1	25	0.47	84	1.09	22	1.44
DSJC500.1.col	15	1	42	0.27	16	39	21	0.13	15	1	44	0.00	15	1	79	2.29
DSJC500.5.col	63	1	61	0.61	62	1	45	0.10	63	3	44	0.14	63	3	65	1.93
DSJC500.9.col	163	2	38	1.00	161	1	53	1.04	165	3	20	0.34	155	1.08	82	1.48
DSJC1000.1.col	28	4	28	0.79	27	1	44	0.47	27	4	53	0.00	28	31	42	2.48
DSJR500.1.col	12	3	9	0.13	12	13	10	0.09	12	15	9	0.00	12	19	14	0.89
DSJR500.5.col	136	2	17	0.38	136	4	14	0.26	137	2	13	0.24	135	1	20	0.67
le450-15d.col	26	9	22	0.33	25	3	30	0.10	24	1	44	0.02	26	7	37	2.10
qg.order30.col	31	7	25	0.79	31	45	22	0.50	31	37	25	0.34	31	76	37	2.17
school1-nsh.col	33	2	22	0.63	30	1	31	0.55	31	1	32	0.00	31	1	50	2.13
wap06a.col	51	7	6	1.63	50	6	10	0.27	49	1	13	0.00	50	6	14	1.86
latin-square-10.col	143	10	33	1.75	136	6	57	0.33	138	20	48	0.04	135	10	94	2.62

Table 2. Statistics of results obtained by 100 runs of each local search. Shown are the best number of colours found, the times in which this colour was attained expressed in percentage, the average number of iterations needed for reaching that number of colours and the rate between 1-exchange and k-exchange moves performed.

	1-exchange		C+1		C+P+1		(C+P)-1		1-(C+P)		Kempe Chains		PenFunct	
inst	Min	succ	Min	succ	Min	succ	Min	succ	Min	succ	Min	succ	Min	succ
queen15-15.col	**16**	**70**	17	20	17	100	17	100	17	80	17	90	17	100
queen16-16.col	**17**	**40**	19	30	18	100	18	100	18	90	18	100	18	100
DSJC250.1.col	**8**	**100**	10	10	9	100	9	100	9	80	9	60	9	90
DSJC250.9.col	**72**	**100**	74	20	74	10	73	80	73	10	73	10	73	20
DSJC500.1.col	**12**	**10**	17	50	14	80	13	50	13	20	14	80	14	80
DSJC500.5.col	**49**	**100**	58	50	53	70	52	10	54	10	55	20	54	10
DSJC500.9.col	**126**	**10**	135	10	133	10	130	20	133	20	132	10	133	10
DSJC1000.1.col	**21**	**100**	27	20	24	10	22	10	22	10	23	40	23	80
le450-15d.col	**16**	**100**	27	50	23	40	22	50	22	10	22	10	22	10
qg.order30.col	30	100	30	100	30	100	30	100	30	100	30	100	30	100
school1-nsh.col	**14**	**100**	15	10	14	10	14	10	15	10	20	10	14	20
wap06a.col	**42**	**60**	47	40	45	10	45	30	44	10	45	10	45	10
latin-square-10.col	**101**	**100**	106	33	110	33	117	33	109	33	117	33	102	33

Table 3. Peak performance of the Hybrid Approach (TS+ILS). Given are the minimum number of colours found and the percentage of success in 10 trials per each algorithm on each instance. In bold the best results.