Heuristic rope team : a parallel algorithm for graph coloring

Laurent Moalic
Univ. Bourgogne Franche-Comté, UTBM, OPERA
F-90100 Belfort, France
laurent.moalic@utbm.fr

Alexandre Gondran
MAIAA, ENAC, French Civil Aviation University
Toulouse, France
alexandre.gondran@enac.fr

ABSTRACT

Optimization problems are often compared to mountain climbing. In particular, the classical Hill Climber is a so used local search. In this paper we propose to explore further the analogy with climbing a mountain, not alone as it is generally the case but in rope team. Indeed, for difficult problems a single climber (heuristic) can easily be trapped into a local optimum. This situation can be compared to a crevasse which the heuristic can not escape from. In this paper we propose the Rope Team heuristic dealing with several elementary memetic heuristics which corresponds to climbers. The leader of the rope team bring with him at least one other climber, linked by a rope, in order to help him to escape from a local optimum. This approach has been successfully applied to one of the most studied combinatorial problem: the Graph Coloring Problem. The advantage of this approach is twofold: on one hand, it is able to find the best known solution for most of the difficult graphs coming from the DIMACS benchmark; on the other hand, all climbers can be considered simultaneously, allowing parallelization of the algorithm. Among the significant results of this work we can notice 3 well-studied graphs, DSJC500.5 colored with 47 colors, DSJC1000.5 with 82 colors and flat1000_76_0 with 81 colors.

KEYWORDS

Metaheuristics, Memetic Algorithm, Hybridization, Parallelization

ACM Reference format:
Laurent Moalic and Alexandre Gondran. 2017. Heuristic rope team : a parallel algorithm for graph coloring. In *Proceedings of GECCO '17, Berlin, Germany, July 15-19, 2017,* 7 pages.
DOI: http://dx.doi.org/10.1145/3071178.3071291

1 INTRODUCTION

Graph coloring consists of assigning a color to each vertex of a given graph $G = (V, E)$, such that two adjacent vertices (linked by an edge) are assigned different colors. Such coloring is said legal. V is known as the vertices set and E edges set. The Graph Coloring Problem (GCP) is to find, for graph G, the minimum number of colors that provides a legal coloring. This minimum number is known as the *chromatic number* $\chi(G)$. GCP is very famous and important because lots of applied problems can be modeled as a GCP. As an example we can cite the frequency assignment problem [2], the timetabling problem [27], the scheduling problem [29], or the fly level allocation problem [4]. Since GCP is NP-hard [17], exact algorithms can be used only with easy (low or high edges density) or small graphs. But as soon as graphs become more complex, heuristic approaches are required.

Most of computational optimization techniques have been applied to solve GCP. We will present a short overview of these techniques grouped into five categories. The first approaches are constructive methods. One can find in this category greedy methods whose DSATUR [5] and RLF [18] are the most well-known, and can be used to quickly provide an upper bound. Then, many exacts methods have been developed to solve GCP for small graphs: branch and bound [11], backtracking [30], constraint programming [12] [6], linear programming [23], column generation [12, 21]. Third methods are local searches such as hill climbing method [19], simulated annealing [15], tabu search [13], which try to improve an initial solution by local moves. These approaches are the first ones able to find *good* coloring for large graphs. Some improvements have been brought successfully by using in the same algorithm different local strategies, such as in variable neighborhood search [3] or in variable space search [14] or [7]. Moreover, some population-based approaches have been developed with interesting results. Evolutionary algorithms [8, 9], ant colony optimization [22], particle swarm algorithm, are some of them. They work with several individuals that can interact together with a crossover operator, a shared memory, a repulsion or attraction operator, or others criteria such as sharing. All these methods provide interesting results with more or less efficiency. But the best known algorithms which are able to solve the most difficult GCP are hybridizations of previous algorithms (fifth approach). In many cases they use a powerful local search inside a population-based algorithm. The memetic algorithms [10, 20, 28] and quantum annealing [24–26] provide up to now the most interesting approaches. Note that for very large graphs, methods such as IE^2COL (Improving the Extraction and Expansion method for large graph COLoring [**?**]) use pre-processing techniques.

Today there is a common understanding that CPU speedup is limited. That is why parallelization is a promising research area, especially for heuristic algorithms known to be very CPU-time consuming. In this paper we present a new and very simple hybrid algorithm providing the best results on most of the DIMACS graphs. This parallel algorithm is based on the analogy with climbing: it is more secure to climb in rope team than alone, especially for the hardest mountains (or problems).

This article is organized as follows. The proposed algorithm is presented in Section 2. The experimental results are described in Section 3. An analysis of how the proposed algorithm runs is done in Section 4. And finally we draw the conclusions of this approach and the future works to do in Section 5.

2 ALGORITHM: CLIMIBING IN ROPE FOR CLIMBING HEIGHER

Many powerful algorithms are based on the hybridization of others more simple components. When a hybrid algorithm uses a local search inside a population based approach, it is called Memetic Algorithm (MA) or Genetic Local Search (GLS). In a MA the classical mutation is replaced by a local search for providing a better intensification. The proposed approach called *PRTCol* (for Parallel Rope Team Coloration) belong to this family. It shares common features with the Hybrid Evolutionary Algorithm (HEA) of Galinier and Hao [10] and Hybrid Evolutionary Algorithm in Duet (HEAD) from Moalic and Gondran [1] known to be among the best algorithms for solving GCP. Indeed HEA performs most of best results since 1999 on DIMACS benchmark [16], and HEAD has improved the results of HEA especially for very difficult graphs such as those detailed in this article (see table 1).

2.1 Climber definition

The main component of the proposed approach *PRTCol* is the local search, an improvement of the TabuCol of [13]. This is a powerful tabu local search able to provide good results, even if used alone (not combined with a population based approach). TabuCol can be seen as one foot of the climber, and is used to perform a local exploration of the search space. That means that for one climber two TabuCol are used, one per foot. Of course the two feet have to walk in the same direction. To ensure that, after the local exploration the two feet share some information. This sharing of information is done based on the crossover operator called Greedy Partition Crossover (GPX) from [10]. Its main characteristic is to build a child (next starting position for the foot) with the biggest color classes from each parent. Figure 1 illustrates how one climber walks. It is important to notice that both TabuCol and GPX work with a k-fixed penalty strategy that accepts non proper solutions (having edges in conflict; i.e. adjacent vertices colored with the same color). The aim of this heuristic is to find a free-conflict coloring, i.e. to minimize the number of conflicting edges until zero.

In this approach, a climber can be seen as a very simple memetic algorithm with a population of only two individuals. This idea was introduced in [1]. It has the great advantage of being very easy to treat. Indeed, there is no selection operator, neither replacement strategy to determine. At each step, the two individuals are crossed two time to create two children. Then each child is improved by the local search. It is to notice that such a process can only work with a stochastic crossover. Indeed, crossing two times the same parents must provide two different children.

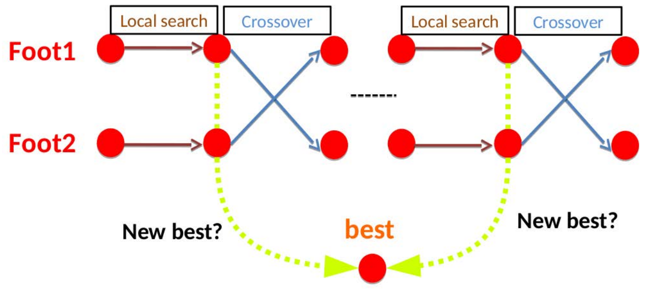

Figure 1: Climber heuristic: climber walk, alternating local exploration (LS) and information sharing between the two feet (Crossover). The best solution encountered during this heuristic is record.

Figure 2: Climbing in rope

2.2 From one climber to a rope team

Although using a population of only two individuals has the advantage of being simple, there is a risk of premature convergence. Indeed, the population involved in memetic algorithms brings the diversity while local search is used for intensifying the solutions. The greater the population size, the greater the diversification. That is why in memetic algorithm a population size of at least 10 is usually selected. Here we propose to introduce a new way for managing the diversity.

When the two feet of the climber become the same, then the crossover has no more effect and diversification disappears. That is to say the heuristic is trapped in a local optimum, or the climber fell into a crevasse. In this case, the other climbers of the rope take him back.

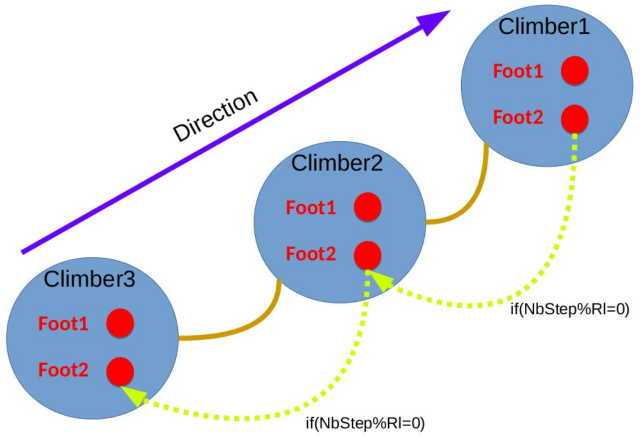

Figure 3: Climbing in rope: transmitting the direction to the team

Figure 2 illustrates how climbing in rope can help to go higher. Two mechanisms are introduced in *PRTCol* to reproduce this behavior.

(1) When all goes well, the leader of rope team *transmits the direction to the others*.
(2) When one climber falls, the others *retain him*.

2.2.1 Transmitting the direction to the team. The goal of climbing in team is to go, more or less, in the same direction, with a given distance between the climbers. In the proposed algorithm, this is done as follows. Every R_l steps (R_l representing the rope length between two individuals), climber i transmits the direction to follow to climber $i+1$. Thus, the climber i attracts to him the climber $i+1$ by transmitting him one solution (i.e. the position of one of his feet).

Figure 3 provides the main idea of climbing in rope. Each climber follows the process described in Figure 1 and every R_l steps the position of one foot of each climber $i = 2...n$ is replaced by the position of a foot of climber $i-1$. Of course, during the ascent climber #1, the leader of the team, receive no information from the others.

2.2.2 Retaining a fallen climber. One climber in *PRTCol* can be seen as two parallel TabuCol, with a sharing mechanism brought by the crossover. One of the main drawbacks of such a simple population is that sometimes it doesn't provide enough diversification. Then, after a given number of steps, one climber can have the two feet (solutions) quite similar. That is why diversification must be reintroduced, with the help of the rope. Then a very simple mechanism is introduced, similar to the one presented section 2.2.1 but in the opposite direction. When the two feet of a climber are equal, the previous and the next climbers retain him by transmitting their position. It can be seen as if foot #1 (solution #1) is replaced by the foot position of the previous climber, and foot #2 by a foot of the next climber. For the first and for the last in the rope team they are retain by their only neighbor.

2.3 Climbing algorithm

This section provides a detailed description of the proposed algorithm: Parallel Rope Team Coloration (algorithm 1). Some functions are used but not detailed in this paper. Especially one can find the crossover operator $GPX(p_1, p_2)$ (line 9), which build the next position of a foot (solution) from two positions p_1 and p_2. A detailed description of this algorithm is provided in [10]. In the same way, *PRTCol* refere to the function $TabuCol(c_1, Iter_{TC})$ (line 11). This is the local search coming from [13]. And finally, three easy functions are not detailed: $init()$ (line 3) which initializes randomly the position of the feet for every climbers, $saveBest()$ (line 15) which records the best solution found during the process and *nbConflicts()* (line 6) which returns the total number of conflicting edges for a given coloring.

In algorithm 1 one can notice that the main *while* loop is running until either a solution with no conflict is found, either a *stopCriterion* is reached (line 6). This ending criterion can be a time limit or others. In our case we stop the algorithm if it fall two times in the same "crevasse". Indeed, when the two feet of a climber become the same, this solution is recorded. If it is found again, we consider that the algorithm is trapped.

Notice that the transmission of the direction to the followers is done from the last to the first climber. Indeed, if the corresponding **for** loop went from #2 to #n, every climber would have the same value for foot #1 (line 28).

For heuristic algorithms, one of the key element is the control of diversity. It is probably the ability of introducing the right dose of diversity that will lead an algorithm to be effective or not. To much diversity risks missing an optimal solution; not enough diversity risks to lead to a local optimum. The algorithm presented here is likely a multi-level diversity management.

Level1: Diversity comes from the tabu tenure inside the LS itself.
Level2: A second level of diversity is introduced by the crossover. Information coming from the two feet are merged.
Level3: The third level of diversity is provided by the transmission of a direction from a climber to the next one.
Level4: Finally, the last level of diversity comes from the help of the team, when retaining a fallen climber.

An important point not yet discussed is the parallelization of the code. Notice that the speedup of the processors does not increase as fast as the number of available cores. This aspect must be taken into account for designing effective algorithms. This consideration led us to the idea of the proposed approach. Indeed, it is easy to see that every climber can walk together. And in fact, even each walker can move his two feet at the same time, since the interaction between the feet occurs only after the local exploration. This

Algorithm 1 Climbing in rope: *PRTCol*

```
1: Input: $Iter_{TC}$=number of LS iterations ; $R_l$=rope length between two climbers ; $n$=number of climbers.
2: Output: the best configuration found
3: $p_{1,1}, p_{1,2}, ..., p_{n,1}, p_{n,2} \leftarrow$ init() {initialize with random colorings}
4: $step \leftarrow 0$
5: $best \leftarrow init()$
6: while nbConflicts($best$) $> 0$ and not $stopCriterion()$ do
7:   {All the climbers advance one step}
8:   for i=1 to n do
9:     $c'_1 \leftarrow$ GPX($p_{i,1}$, $p_{i,2}$)
10:    $c'_2 \leftarrow$ GPX($p_{i,2}$, $p_{i,1}$)
11:    $c_1 \leftarrow$ TabuCol($c'_1, Iter_{TC}$)
12:    $c_2 \leftarrow$ TabuCol($c'_2, Iter_{TC}$)
13:    $p_{i,1} \leftarrow c_1$
14:    $p_{i,2} \leftarrow c_2$
15:    $best \leftarrow$ saveBest($c_1$, $c_2$, $best$)
16:  end for
     {Check if a climber has fallen and retain him}
17:  for i=1 to n do
18:    if $p_{i,1} = p_{i,2}$ then
19:      if $i > 1$ then
20:        $p_{i,1} \leftarrow p_{i-1,1}$
21:      end if
22:      if $i < n$ then
23:        $p_{i,2} \leftarrow p_{i+1,2}$
24:      end if
25:    end if
26:  end for
     {Each climber transmits the direction to its follower}
27:  if $step \% R_l = 0$ then
28:    for i=n to 2 do
29:      $p_{i,1} \leftarrow p_{i-1,1}$
30:    end for
31:  end if
32:  $step++$
33: end while
34: return $best$
```

aspect is very important insofar as almost all the calculation time is taken for the LS. It is therefor very easy with a multi-core processor to parallelize this algorithm. And even with a simple quad-core processor, two climbers can walk in the same time with the use of two cores per climber.

3 RESULTS

3.1 Instances and Benchmark

The results presented here were obtained for graphs coming from the second DIMACS challenge of 1992-1993 [16]. It is to date the most challenging and the most widely used benchmark for solving the graph coloring problem.

Two main types of graphs of DIMACS benchmark are taken into account: DSJC and FLAT, which are randomly or quasi-randomly generated. DSJC$n.d$ graphs are graphs with n vertices where each vertex is connected to an average of $n \times d$ vertices; d is the graph density. The chromatic number of these graphs is unknown. FLAT graphs have another structure. They are built for a known chromatic number. The flatn_χ graph has n vertices and χ is the chromatic number.

3.2 Computational Results

PRTCol was programmed in standard C++. The results presented in this section were obtained on a computer with an Intel Xeon 3.10GHz processor - 4 cores and 8GB of RAM. Almost all of the computation time is spent performing the tabu search (lines 11 and 12 of algorithm 1). The parallelization of this LS is done using the OpenMP API (Open Multi-Processing). The advantage of this API is its simplicity. Only one line is enough, a preprocessor macro, for using several cores. The execution times in the following table are *run time* and not *cpu time*. Thus, when we give an execution time of 30 minutes, for two climbers the CPU time is actually close to 2 hours using four processing cores.

Table 1 presents results of the principal methods known to date. For each graph, it indicates the lowest number of colors found by each algorithm.

One can notice that the proposed approach is able to find the best known results for most of the graphs: 16/17. When comparing the results of *PRTCol* to the simple TabuCol, it seams interesting to see that the improvement is due to the multi-level diversification presented section 2.3.

Table 2 presents in detail the results obtained with *PRTCol*. The first important remark is that *PRTCol* find solutions with a number of colors smaller or equal than all the best known methods except for C2000.5. The column $\mathbf{Iter_{LS}}$ gives the number of iterations of TabuCol algorithm (it is the stop criteria of TabuCol). The column $\mathbf{R_l}$ provides the rope length between two climbers. That is to say that every R_l steps, all the climbers except the first one get one solution (foot value) from the climber in front of him. The column **Steps** indicates the average number of steps before finding a legal coloring. As a reminder, one step corresponds to apply one TS at both feet of each climber plus one crossover, then 1 **Steps** $= 2 \times n \times Iter_{LS}$ iterations of the TS, with n the number of climbers. In tour case, with $n = 2$, one step is 4 parallel TS. The column **Falls** gives the average number of falls, triggering the mechanism of retaining a climber. Note that for the easiest graphs this number is near to 0. In these situations a simple climber without rope would have been enough. The column **Success** evaluates the robustness of the method, it gives the success rate: success_runs/total_runs; a success run is one which finds a legal k-coloring. Finally, the column **Time** gives the average run time in minutes for successful runs.

4 ANALYSIS OF THE ROLE PLAYED BY THE ROPE TEAM

When considering only one climber in *PRTCol*, it can be seen as a classical memetic algorithm with a population of size 2. In this case the algorithm alternates between periods of local search and periods of sharing information with the crossover. Although this simple version of *PRTCol* can be

		2008	1999	2015	2008	2010	2011	2012	2012
Graphs	*PRTCol*	TabuCol	HEA	HEAD	AmaCol	MACOL	EXTRACOL	IE^{2}COL	QA-col
dsjc250.5	**28**	28	28	28	28	28	-	-	28
dsjc500.1	**12**	13	-	12	12	12	-	-	-
dsjc500.5	**47**	49	48	47	48	48	-	-	47
dsjc500.9	**126**	127	-	126	126	126	-	-	126
dsjc1000.1	**20**	-	20	20	20	20	20	20	20
dsjc1000.5	**82**	89	83	82	84	83	83	83	82
dsjc1000.9	**222**	227	224	222	224	223	222	222	222
r250.5	**65**	-	-	65	-	65	-	-	65
r1000.1c	**98**	-	-	98	-	98	101	98	98
dsjr500.1c	**85**	85	-	85	86	85	-	-	85
le450_25c	**25**	26	26	25	26	25	-	-	25
le450_25d	**25**	26	-	25	26	25	-	-	25
flat300_28_0	**29**	31	31	31	31	29	-	-	31
flat1000_50_0	**50**	50	-	50	50	50	50	50	-
flat1000_60_0	**60**	60	-	60	60	60	60	60	-
flat1000_76_0	**81**	88	83	81	84	82	82	81	81
C2000.5	146	-	-	146	-	148	146	145	145

Table 1: Comparison between *PRTCol* and references algorithms

Instances (k*)	**k**	**Iter$_{\mathbf{LS}}$**	**R$_{\mathbf{l}}$**	**GPX**	**Steps**	**Falls**	**Success**	**Time (min)**
dsjc250.5 (28)	28	6000	30	Std	47	0.03	100/100	0.01
dsjc500.1 (12)	12	8000	10	Std	535	0.6	100/100	0.1
dsjc500.5 (47)	47	8000	30	Std	3703	13	7/10000	2.1
	48	8000	30	Std	402	0.6	100/100	0.2
dsjc500.9 (126)	126	25000	40	Std	1061	1.5	100/100	2.4
dsjc1000.1 (20)	20	4000	10	Std	1124	5.7	100/100	0.3
dsjc1000.5 (82)	82	30000	30	Std	2.1×10^4	21.6	20/100	81
dsjc1000.9 (222)	222	40000	10	Std	2.4×10^4	30	11/100	175
r250.5 (65)	65	10000	30	Std	114336	6.0	58/100	43
	65	4000	10	R(20)	147141	0.5	100/100	24
r1000.1c (98)	98	100000	30	Std	85	2.1	50/100	1.1
	98	20000	10	R(98)	96	0.0	100/100	0.2
dsjr500.1c (85)	85	400	30	R(85)	485	0.7	100/100	0.03
le450_25c (25)	25	500000	30	U(0.98)	300	0.44	96/100	3.4
le450_25d (25)	25	500000	30	U(0.98)	196	0.24	98/100	2.25
flat300_28_0 (28)	29	20000	30	Std	52706	3	2/5	63
	30	12000	30	Std	19896	18.8	30/100	11
flat1000_50_0 (50)	50	40000	30	Std	15	0.4	100/100	0.3
flat1000_60_0 (60)	60	40000	30	Std	28	0.3	100/100	0.5
flat1000_76_0 (76)	81	30000	30	Std	18620	12	36/100	87
C2000.5 (145)	146	40000	30	Std	36192	6.8	6/10	728

Table 2: Results of *PRTCol* algorithm

enough for some problems, for the most difficult graphs a premature convergence can occurs, due to the lack of diversity. In the following we present an analysis of when and how the introduction of the team will significantly improve the results.

4.1 Number of rescues carried out by the rope team

In this section we are interested in the relationship between the difficulty of coloring a graph and the number of falls. Figure 4 shows 4 typical situations for different graphs. All results are given for 100 runs of the algorithm. For each plot we provide the success rate of the algorithm (ordinate) for

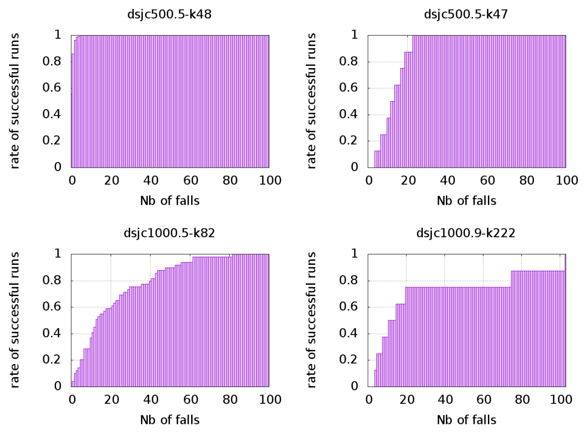

Figure 4: Rate of successful runs for a given number of falls

runs with a given maximum number of falls (in abscissa). Let us detail deeper what is shown. The top left plot describes the results for the graph *dsjc500.5* addressed with 48 colors. We can see that for this configuration, over 50% of $PRTCol$ runs finds a legal coloring without any fall. Moreover, all the solutions are found with less than 5 falls. That means that for this graph with 48 colors the rope team is not so important but it lets the search more robust than a climber alone.

The three other plots describe harder configurations and reveal the power brought by the rope team. Indeed, one can notice that no solution can be found without a fall: a climber alone could not find any legal coloring. For the two plots on the right at least 4 falls are recorded before finding a legal coloring. And even for the plot at the bottom left, at least one fall was encountered.

4.2 Relationship between the rope length and the ability to find solutions

As we have presented previously, diversification plays a key role in metaheuristics. In memetic algorithms, one of the major diversification operator is the crossover. The distance between the parents provides the level of diversity. That is why it is so important to control precisely this distance between the individuals: controlling the distance between individuals means controlling the impact of diversification. In $PRTCol$, every R_l steps each climber get a new position from the climber in front of him (a new parent for the crossover). It is thus easy to see that the distance between two individuals (feet) is very simple to manage and is controlled by the rope length. One can identify two limit situations:

- **Rope length is infinite.** In this case the team leader never transmits information to its follower. $PRTCol$ can be seen as a parallel multi-start algorithm. Each climber goes in its own direction until finding a local optimum solution. Once such a solution is found, the previous and next climbers retain him by transmitting one of their positions. These positions can be so different and the fallen climber goes in a totally new part of the search space.
- **Rope length is null.** Here every climbers share nearly the same "foot". When one climber falls in a crevasse, that likely means that every climbers fall with him: no new diversity can be introduced.

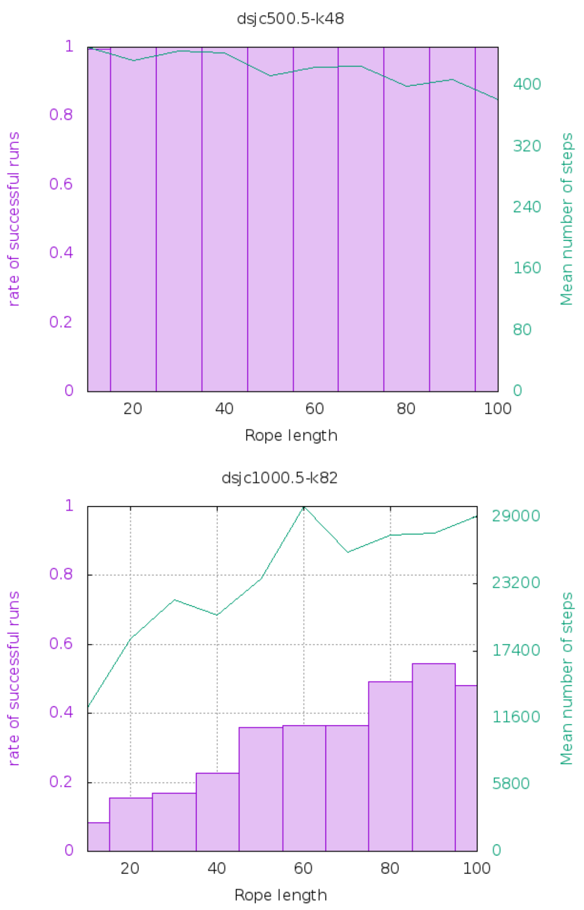

Figure 5: For a given number of rope length, rate of successful runs (left axis) and mean number of steps (right axis)

Figure 5 shows the role played by the rope length for two specific graphs. The purple bars give the successful rate (left y-axis) for increasing rope length (in abscissa). The green line gives the number of steps (right y-axis) for finding a legal coloring. Notice that for each rope length the results shown are obtained over 100 runs.

The first plot (top) describes what appends for a not so difficult graph, $dsjc500.5$ with $k = 48$ colors. In this case, for any rope length 100% of the runs are able to find the solution, except for $R_l = 10$ where it is 99%. Moreover, the number of steps seems to be more or less independent of the rope length.

The second plot (bottom) describes the situation for a very difficult graph, $dsjc1000.5$ with $k = 82$ colors. Up to now, only one other algorithm has been able to tackle this configuration, QA-Col [26]. We can see that for such a challenging graph, a rope length of $R_l = 10$ is not enough with less than 5% of successful runs. One can notice that a rope length of at least 80 is mandatory for a successful rate higher than 50%. On the same plot we can see the increase in the number of steps given the different rope length (green line). It gives an idea of the cost of robustness.

5 CONCLUSION

We have presented an effective parallel heuristic, called $PRTCol$, to solve the graph coloring problem. It is based on a hybridization of memetic algorithms, which reproduce the behavior of climbing in rope team. It showed its relevance when applied on a set of challenging DIMACS graphs. We proposed an analysis of the behavior of $PRTCol$ on specific graphs, especially where the proposed approach was able to significantly improve the results achieved by reference algorithms. In particular, we have shown how the rope length played an important role in the search of legal coloring for the most difficult graphs. Through these analyzes, $PRTCol$ has demonstrated the interest of this new way of managing diversity.

As future work, it seems interesting to apply this principle to other problems. We are confident about its ability to solve many problems, for which an asymmetric crossover can be defined. Another promising work could be to apply the main scheme with other elementary heuristics: crossover or local search.

REFERENCES

[1] 2015. In *Evolutionary Computation in Combinatorial Optimization*. Lecture Notes in Computer Science, Vol. 9026.

[2] KarenI. Aardal, StanP.M. Hoesel, ArieM.C.A. Koster, Carlo Mannino, and Antonio Sassano. 2003. Models and solution techniques for frequency assignment problems. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies* 1, 4 (2003), 261–317. DOI:http://dx.doi.org/10.1007/s10288-003-0022-6

[3] C. Avanthay, Alain Hertz, and Nicolas Zufferey. 2003. A variable neighborhood search for graph coloring. *European Journal of Operational Research* (2003).

[4] Nicolas Barnier and Pascal Brisset. 2004. Graph Coloring for Air Traffic Flow Management. *Annals of Operations Research* 130, 1-4 (2004), 163–178. DOI:http://dx.doi.org/10.1023/B:ANOR.0000032574.01332.98

[5] D. Brélaz. 1979. New Methods to Color the Vertices of a Graph. *Commun. ACM* 22, 4 (1979), 251–256.

[6] Massimiliano Caramia and Paolo Dell'Olmo. 2002. Constraint Propagation in Graph Coloring. *Journal of Heuristics* 8, 1 (2002), 83–107.

[7] Massimiliano Caramia, Paolo Dell'Olmo, and Giuseppe F. Italiano. 2006. CHECKCOL: Improved local search for graph coloringstar. *Journal of Discrete Algorithms* 4, 2 (2006), 277–298. DOI:http://dx.doi.org/doi:10.1016/j.jda.2005.03.006

[8] L. Davis. 1991. Order-based Genetic Algorithms and the Graph Coloring Problem. In *Handbook of Genetic Algorithms*. Van Nostrand Reinhold; New York, 72–90.

[9] C. Fleurent and J. Ferland. 1996. Genetic and Hybrid Algorithms for Graph Coloring. *Annals of Operations Research* 63 (1996), 437–464.

[10] Philippe Galinier and Jin-Kao Hao. 1999. Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization* 3, 4 (1999), 379–397. DOI:http://dx.doi.org/10.1023/A:1009823419804

[11] Fred Glover, Mark Parker, and Jennifer Ryan. 1996. Coloring by Tabu Branch and Bound. See [16], 285–307.

[12] Stefano Gualandi and Federico Malucelli. 2012. Exact Solution of Graph Coloring Problems via Constraint Programming and Column Generation. *INFORMS Journal on Computing* 24, 1 (2012), 81–100. DOI:http://dx.doi.org/10.1287/ijoc.1100.0436

[13] Alain Hertz and Dominique de Werra. 1987. Using Tabu Search Techniques for Graph Coloring. *Computing* 39, 4 (1987), 345–351.

[14] Alain Hertz, M. Plumettaz, and Nicolas Zufferey. 2008. Variable Space Search for Graph Coloring. *Discrete Applied Mathematics* 156, 13 (2008), 2551 – 2560. DOI:http://dx.doi.org/10.1016/j.dam.2008.11.008

[15] David S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. 1991. Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning. *Operations Research* 39, 3 (1991), 378–406.

[16] David S. Johnson and Michael Trick (Eds.). 1996. *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, 1993*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 26. American Mathematical Society, Providence, RI, USA.

[17] R.M. Karp. 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher (Eds.). Plenum Press, New York, USA, 85–103.

[18] F. T. Leighton. 1979. A Graph Coloring Algorithm for Large Scheduling Problems. *J. Res. Nat. Bur. Standards* 84, 6 (1979), 489–506.

[19] Rhyd Lewis. 2009. A general-purpose hill-climbing method for order independent minimum gr ouping problems: A case study in graph colouring and bin packing. *Computers & Operations Research* 36, 7 (2009), 2295–2310. DOI:http://dx.doi.org/10.1016/j.cor.2008.09.004

[20] Zhipeng Lü and Jin-Kao Hao. 2010. A memetic algorithm for graph coloring. *European Journal of Operational Research* 203, 1 (2010), 241–250. DOI:http://dx.doi.org/10.1016/j.ejor.2009.07.016

[21] A. Mehrotra and Michael A. Trick. 1996. A Column Generation Approach for Graph Coloring. *INFORMS Journal On Computing* 8, 4 (1996), 344–354.

[22] M. Plumettaz, D. Schindl, and Nicolas Zufferey. 2010. Ant Local Search and its efficient adaptation to graph colouring. *Journal of Operational Research Society* 61, 5 (2010), 819 – 826. DOI:http://dx.doi.org/10.1057/jors.2009.27

[23] David Schindl. 2003. Graph coloring and linear programming. (July 2003). http://roso.epfl.ch/ibm/jord03.html Presentation at First Joint Operations Research Days, Ecole Polytechnique Fédérale de Lausanne (EPFL), available on line (last visited June 2005).

[24] Olawale Titiloye and Alan Crispin. 2011. Graph Coloring with a Distributed Hybrid Quantum Annealing Algorithm. In *Agent and Multi-Agent Systems: Technologies and Applications*. Springer Berlin / Heidelberg.

[25] Olawale Titiloye and Alan Crispin. 2011. Quantum annealing of the graph coloring problem. *Discrete Optimization* 8, 2 (2011), 376–384. DOI:http://dx.doi.org/DOI:10.1016/j.disopt.2010.12.001

[26] Olawale Titiloye and Alan Crispin. 2012. Parameter Tuning Patterns for Random Graph Coloring with Quantum Annealing. *PLoS ONE* 7, 11 (11 2012), e50060. DOI:http://dx.doi.org/10.1371/journal.pone.0050060

[27] D. C. Wood. 1969. A Technique for Coloring a Graph Applicable to Large-Scale Timetabling Problems. *Computer Journal* 12 (1969), 317–322.

[28] Qinghua Wu and Jin-Kao Hao. 2012. Coloring large graphs based on independent set extraction. *Computers & Operations Research* 39, 2 (2012), 283–290. DOI:http://dx.doi.org/10.1016/j.cor.2011.04.002

[29] Nicolas Zufferey, P. Amstutz, and P. Giaccari. 2008. Graph Colouring Approaches for a Satellite Range Scheduling Problem. *Journal of Scheduling* 11, 4 (2008), 263 – 277.

[30] A. A. Zykov. 1949. On some properties of linear complexes. *Mat. Sb. (N.S.)* 24, 66(2) (1949), 163–188.