A Distributed Hybrid Algorithm for the Graph Coloring Problem

Ines Sghir[1,3], Jin-Kao Hao[1,2⋆], Ines Ben Jaafar[3], and Khaled Ghédira[3]

[1] LERIA, Université d'Angers, 2 Bd Lavoisier, 49045 Angers Cedex 01, France
[2] Institut Universitaire de France, Paris, France
[3] SOIE, ISG, Université de Tunis, Cité Bouchoucha 2000 Le Bardo, Tunis, Tunisie

Abstract. We propose a multi-agent based Distributed Hybrid algorithm for the Graph Coloring Problem (DH-GCP). DH-GCP applies a tabu search procedure with two different neighborhood structures for its intensification. To diversify the search into unexplored promising regions, two crossover operators and two types of perturbation moves are performed. All these search components are managed by a multi-agent model which uses reinforcement learning for decision making. The performance of the proposed algorithm is evaluated on well-known DIMACS benchmark instances.

1 Introduction

Given an undirected graph $G = (V, E)$ with vertex set V and edge set E. A legal (or proper) k-coloring of G (k is an integer) is a partition of V, i.e., $S = \{V_1, V_2, ..., V_k\}$ where each subset $V_r \subset V$ is an independent set (also called a legal color class) such that no two vertices of V_r are linked by an edge. Given k colors, the k-coloring problem (k-COL) is to find a legal k-coloring. The graph coloring problem (GCP) is to determine the smallest integer k (i.e., the chromatic number χ_G of G) such that there exists a legal k-coloring of G.

GCP has numerous important applications in practice and is known to be computational difficult. Given its relevance, GCP is certainly among the most studied NP-hard problems [8]. Among the large number of GCP solution approaches (see e.g., [12, 20]), most of them are based on neighborhood search [1, 2, 5, 6, 15–17, 22, 27], hybrid population search [4, 7, 10, 9, 18, 19, 21, 23, 26] or other hybrid scheme [14, 24, 28]. More GCP methods can be found in [8, 12, 20].

In this paper, we study a distributed algorithm for GCP which is based on the principle of multi-agent systems. As our general solution strategy, we adopt the very popular k-fixed penalty approach [12] which was used in many previous algorithms like [5, 6, 9, 12, 18, 19, 23]. With this approach, we fix the number k of colors and seek a legal k-coloring among all possible (legal or illegal) k-colorings. Given a k-coloring S, the evaluation or fitness function $f(S)$ calculates the number of conflicts induced by S, i.e., the number of edges whose end-points are colored with the same color. Thus, $f(S) = 0$ indicates that S is a legal coloring.

⋆ Corresponding author: jin-kao.hao@univ-angers.fr

The algorithm tries to solve the k-coloring problem by minimizing the fitness function f. Finally, to approximate the chromatic number of G, we try to solve a series of k-coloring problems with decreasing values of k.

The rest of the paper is organized as follows. Section 2 describes the proposed algorithm. Section 3 presents the experimental results achieved on DIMACS benchmark instances. Finally, section 4 concludes the paper.

2 A distributed hybrid algorithm for GCP

The proposed distributed hybrid algorithm for GCP (DH-GCP) explores a set of interacting agents which are local optimization procedures, crossover operators and perturbation techniques. The coordination of these agents is realized in an informed way using reinforcement learning. The learning mechanism modifies and adapts the search strategy according to the experiences obtained during the search process. The agents are learners and players that ensure the role of intensification and diversification to explore the given search space. This study constitutes a continuation of our recent work on multi-agent based optimization applied to the quadratic assignment problem [25].

2.1 Weight matrix with Reinforcement learning

Reinforcement learning aims to learn what to do and how to plan situations to actions, in order to maximize a numerical reward signal. In most forms of learning, the learner is told which actions to take, but for reinforcement learning, the learner needs to discover the action that leads to the best reward based on previous experiences. A learner must be able to learn from its own experiences to make decisions. In the proposed DH-GCP algorithm, decisions or actions correspond to techniques of diversification or intensification to apply and experiences are acquired during the search progress. Following [13], we use decision rules represented by a couple $(Condition, Action)$. Let C be the set of conditions describing the search progress and A the set of actions or decisions to perform. For a condition C_i, a weight W_{ij} (initialized to 0) is associated to each action A_j. We use the following equation [13] to calculate the probability $P(C_i, A_j)$ for applying an action A_j based on a condition C_i:

$$P(C_i, A_j) = \frac{W_{ij}}{\sum_{j \in A} W_{ij}} \tag{1}$$

At the beginning of the algorithm (i.e., first iterations of Algorithm 2), the improvement situation is assigned to a default condition. According to the weight matrix W, the most appropriate action for this condition is selected based on the probability given in Eq. (1). Then, at the end of each generation, the performed action is evaluated and the concerned weight value is increased if there is an improvement in solution quality. A credit assignment is used to perform reinforcement learning in order to select the beneficial experiences and determine a reward for them. Here, an experience is represented as a triplet (*condition* C_i,

action A_j, *improvement* V). When a new best local or global solution is found, the weight value W_{ij} which is related to the action of this generation is reinforced by adding a reward rate σ to W_{ij}. Before adding the reinforcement value, the weight values W_{ij} in the decision matrix is decreased with an evaporation value μ, in order to enlarge the influence of the new reward obtained in the current generation. The reinforcement with reward σ is then performed using the following equations [13]:

$$W'_{ij} = \mu \times W''_{ij} \tag{2}$$

$$W_{ij} = \mu \times W'_{ij} + \sigma \tag{3}$$

where W'_{ij} is the weight value before the reinforcement, W''_{ij} is the weight value before the evaporation, μ is the evaporation value and σ is the learning factor.

Figure 1 shows an illustrative example of this reinforcement learning process (See section 2.3 for more details). In the proposed algorithm, this matrix is used by the mediator agent (Section 2.3) and the tabu search agents (Section 2.4).

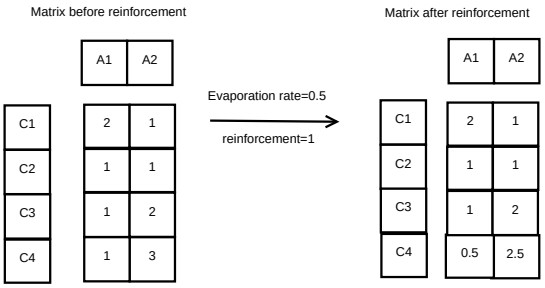

Fig. 1. An example of the reinforcement learning procedure with weight matrix: We suppose that the current condition is C_4 (e.g., the local best solution has not been improved in recent 10 generations). Under this condition, action A_2 (e.g., activate crossover agents) is performed for the current generation (this action has the highest value in the matrix) and obtained a further improvement. Then, reinforcement is applied by adjusting the weight W_{42} to augment the chance of selecting again the applied action under this condition (e.g. $W_{42} = 3 \times 0.5 + 1 = 2.5$). The weight W_{41} is decreased by μ (e.g. $W_{41} = 1 \times 0.5 = 0.5$)

2.2 Agent interaction in DH-GCP

The proposed DH-GCP is a distributed approach composed of interacting agents. Each agent has a local view of the problem, but the collaboration of these agents can help find good solutions for GCP. We consider the following agents: the mediator agent, the tabu search agents, the perturbation agent and the

crossover agents. Figure 2 describes the architecture of DH-GCP while Algorithm 1 presents the general procedure of DH-GCP. Algorithms 2 and 3 describe the behaviors of the mediator agent and the tabu search agents.

The DH-GCP algorithm explores several search cycles (generations, see the 'while' structure of Algorithm 2). In each cycle, the mediator agent is responsible to decide which agents will be activated using its weight matrix according to the state of search process. The activated agents can be tabu search agents or crossover agents (Algorithm 2). During the process of tabu search agents, they can trigger the perturbation agent (to diversify the search). Note that agents are not activated in a pre-specified order. Instead, their activation depends on the past learning experiences and is dynamically adjusted. In the following subsections, we explain the behaviors of each type of agents.

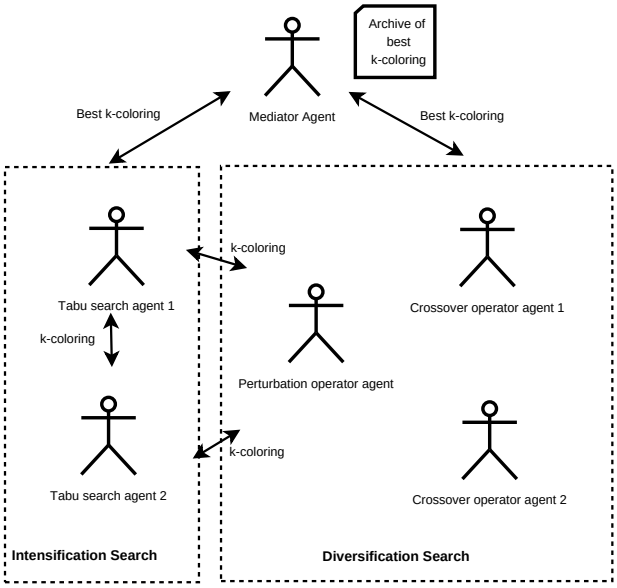

Fig. 2. Agent communication in DH-GCP: Mediator agent is the agent who manages the search according to the improvement realized, tabu search agents ensure search intensification while crossover agents and perturbation agent are responsible for diversification

2.3 Mediator agent

The mediator agent selects other agents to trigger based on its weight matrix (Section 2.1). When other agents (tabu search agents or crossover agents) are triggered, the mediator agent waits for (improved) solutions received from these

Algorithm 1 DH-GCP general procedure

Require: Graph G, number of colors k, four types of agents: mediator agent, two tabu search agents, perturbation agent and two crossover agents
Ensure: The best k-coloring S_{best}

while A legal k-coloring S_{best} not reached **do**
  The mediator agent starts the algorithm by initialing the search and then decides to trigger tabu search agents or crossover agents based on its weight matrix (Algorithm 2)
  if The mediator agent decides to activate tabu search agents **then**
    The tabu search agents are activated and the mediator agent waits for a k-coloring from the activated tabu agents (Algorithm 3)
    if An activated tabu search agent needs to trigger a perturbation agent **then**
      The tabu search agent actives the required perturbation agent and waits for solution from the perturbation agent (Section 2.5)
      The perturbation agent is killed after sending the k-coloring found to the corresponding tabu search agent
    end if
    if An tabu search agent wants to cooperate with other tabu search agent **then**
      The requiring tabu search agent waits for a new k-coloring from other tabu search agent (Algorithm 3)
    end if
    The tabu search agents are killed after sending the best solutions generated during their search to the mediator agent
  end if
  if The mediator agent decides to activate crossover agents **then**
    The crossover agents are activated and the mediator agent waits for the best k-coloring from the crossover agents (Section 2.6)
    The crossover agents are killed after sending new solutions to the mediator agent
  end if
end while
Return The best legal k-coloring found S_{best} from the mediator agent

Algorithm 2 Mediator agent behavior

Require: Graph G, number of class k, parameters: improvement threshold interval *interval*, consecutive non-improving iterations *max_opt*.
Ensure: A best legal k-coloring S_{best} found so far

$S \leftarrow Generate\_initial\_k-coloring\_S_0$ {Section 2.3}
$S_{best} \leftarrow S_0$ {S_{best} records the best k-coloring found so far}
$f_{best} \leftarrow f_0$ {f_{best} records the best objective value of the best k-coloring reached so far}
opt $\leftarrow 0$ {*opt* is the counter for consecutive non-improving local optimum}
$W \leftarrow 0$ {W is the weight matrix of the mediator agent}
pop $\leftarrow \emptyset$ {*pop* is the archive of elite solutions found during the search}
while A legal k-coloring S_{best} not reached ($f_{best} \neq 0$) **do**
  Update W using *interval*, *opt* and *max_opt* {Sections 2.1 and 2.3}
  Action_type $\leftarrow$ Select an action to activate based on W {Section 2.3}
  if *Action_type* = tabu search agents **then**
    Trigger tabu search agents and send S_{best} to them
  else
    Trigger crossover agents and send S_{best} to them
    opt $\leftarrow 0$
  end if
  $S_1 \leftarrow \emptyset$, $S_2 \leftarrow \emptyset$ {S_1 and S_2 are k-colorings received from the activated agents}
  if $S_1 \neq \emptyset$ AND $S_2 \neq \emptyset$ **then**
    if $f(S_1) \leq f(S_2)$ **then**
      $S \leftarrow S_1$
    else
      $S \leftarrow S_2$
    end if
    $tr \leftarrow Exist(S_1, S_2, pop)$ {Check if S_1 and/or S_2 are in the archive *pop*}
    if tr = false **then**
      add S_1 and/or S_2 to *pop* {add both k-colorings or one of them in *pop*}
    end if
    if $f(S) \leq f_{best}$ **then**
      $S_{best} \leftarrow S$
      $f_{best} \leftarrow f(S)$
    else
      opt $\leftarrow$ *opt* $+ 1$
    end if
  else
    block this agent {The mediator agent waits for k-coloring from other activated agents}
  end if
end while
Return S_{best}

agents and to record the received solutions in the shared memory (archive). The behavior of the mediator agent is described in Algorithm 2.

The initial solution The mediator agent creates an initial legal coloring using the greedy largest saturation degree heuristic (DSATUR) [3]. Then, starting with this initial coloring, it randomly displaces the vertices whose color number is higher than the given color number k to a color class between $[1, k]$. This procedure usually leads to an illegal k-coloring which will be repaired by the DH-GCP algorithm.

Conditions and actions of weight matrix The weight matrix of the mediator agent is composed of two types of actions: A_1 corresponds to activating the tabu search agents, and A_2 corresponds to activating the crossover agents. The conditions, which cover significant situations that may occur in the search process, are:

- C_1 = The algorithm does not reach m_0 generations (cycles);
- C_2 = The local or global best solution is improved in recent m_1 generations and this improvement is a small improvement in the fitness function value f;
- C_3 = The local or global best solution is improved in recent m_1 generations and this improvement is a large improvement in the fitness function value f;
- C_4 = The global best solution has not been improved in recent m_2 generations. This solution is a deep local optimum or an optimum solution.

where m_0, m_1 and m_2 are parameters set by the user according to the total generation number. When there is a large improvement obtained by the application of an action between two successive generations (this corresponds to the situations C_1 and C_3), it is better to apply an intensification process by triggering the tabu search agents. If the mediator agent observes no improvement or an insignificant improvement (this corresponds to the situations C_2 and C_4), the search needs to be diversified by activating crossover agents. After each generation (i.e., when the activated agents return their found solution), the mediator agent updates its weight matrix (see Section 2.1).

Archive of elite solutions The mediator agent saves the best k-coloring, received from tabu search agents and crossover agents, in an archive. The archive represents a shared memory between all agents. It is updated by the mediator agent with new solutions of good quality.

2.4 Tabu search agents

The mediator agent can activate two tabu search agents, when it observes that the search process needs to be intensified (lines 4 – 11 of Algorithm 2). Each tabu search agent applies a specific strategy based on a particular neighborhood

to seek new solutions (line 7 to line 10 of Algorithm 3). During the search, a tabu search agent can exchange its solutions with another alive tabu search agent or with a perturbation agent (line 14 to line 28 of Algorithm 3). These communications depend on a weight matrix(lines 16 and 17 of Algorithm 3). At the end of each tabu search agent run, the best k-coloring found by each agent is sent to the mediator agent (line 36 of Algorithm 3). The behavior of the tabu search agent is described in Algorithm 3. Below, we define the used neighborhood structures for each tabu search agent. Then, we explain the conditions and actions employed by them.

Neighborhoods A candidate solution for GCP can be generated by changing the color class of vertices. Different modifications lead to different neighborhood structures. In this work, we explore 3 neighborhoods: the vertex neighborhood which changes the color of some conflicting vertices, the class neighborhood which changes the color of some or all vertices of a conflicting color class, and the non-increasing neighborhood which changes the color of some vertices without increasing the total number of conflicting edges.

Neighborhood exploration strategies In DH-GCP, we use two complementary neighborhood strategies due to the cooperation act realized by each tabu search agent. One of these strategies, performed by our first tabu search agent, changes the colors of conflicting vertices to produce new k-colorings. This is done by moving a conflicting vertex x from its original color class V_i to the best possible other color class V_j ($i \neq j$) (this change or move is denoted by (x, i, j)). The new color class for each conflicting vertex x is chosen among those which are not assigned to vertices adjacent to x. Among these color classes found, the best possible color class (in terms of fitness minimization) is selected for the considered conflicting vertex. Our second tabu search age nt uses the same mechanism of selecting the best color class to be assigned to vertices as the first tabu search agent. The difference is that these vertices are not the set of conflicting vertices, but the adjacent of conflicting vertices. The tabu search agent chooses the best color class for each vertex belonging to the set of adjacent vertices of conflict vertices. The best color affected must not belong to the color classes affected to conflicting vertices.

For these two neighborhood strategies, tabu search agents evaluate each move using an incremental evaluation technique [6, 9, 10]. This technique consists of maintaining a special data structure that records the move values for each candidate neighborhood move.

Tabu list Each tabu search agent uses a tabu list to forbid the reverse moves. When a move (x,i,j) is generated, vertex x is forbidden to move back to color class V_i for the next h iterations (called tabu tenure). The tabu tenure is dynamically determined by $h = f(S) + r(10)$ where $r(10)$ is a random number between 1 and 10 [10]. The stop condition of each tabu search is a fixed number of iterations.

Algorithm 3 Tabu search agents behavior

Require: Graph G, number of colors k, A k-coloring S_{best} received from mediator agent, parameters: maximum iterations $iteration\_max$, improvement threshold interval $interval$, consecutive non-improving iterations max_opt_TS.

Ensure: A k-coloring S_{best_TS}

$S \leftarrow S_0$ {S is the current k-coloring found by each tabu search agent}
$Tabu\_list \leftarrow 0$ {$Tabu\_list$ is the tabu list, Section 2.4}
$Q \leftarrow 0$ {Q is the weight matrix of each tabu search agent, Section 2.4}
$opt = 0$ {opt is the counter for consecutive non-improving local optima for each tabu search agent}
$S_1 \leftarrow S_0$ {S_1 is the k-coloring obtained in generation $iteration - 1$}
while $iteration \leq iteration\_max$ **do**
  $S \leftarrow$ Generate the best neighboring k-coloring {Sections 2.4 and 2.4}
  Update $Tabu\_list$
  if $f(S) \leq f(S_{best\_TS})$ **then**
    $S_{best\_TS} \leftarrow S$
  else
    $opt = opt + 1$
  end if
  if $(f(S) - f(S_1) < interval)$or $(opt = max\_opt\_TS)$ **then**
    $S_{perturbed} \leftarrow \emptyset$ {$S_{perturbed}$ is a k-coloring received from other agents (tabu search agent or perturbation agent}
    Update Q {Update the weight matrix based on the improvement of the current solution, opt, $interval$ and max_opt_TS, Section 2}
    $Action\_exchange \leftarrow$ Select the agent to trigger based on Q
    if $Action\_exchange$=Activating perturbation agent **then**
      Activate the perturbation agent and send it the current k-coloring S
      opt=0 {opt is reset to 0, only when strong perturbation behaviour is trigged, Section 2.5}
    end if
    if $Action\_exchange$=Activating other tabu search agent **then**
      Request the current k-coloring of other tabu search agent
    end if
    Let $S_{perturbed}$ be the perturbed solution received from the perturbation agent or other tabu search agent
    if $S_{perturbed} \neq \emptyset$ **then**
      $S \leftarrow S_{perturbed}$
    else
      block this agent {Tabu search agent waits for a solution from other agents}
    end if
  else
    $S_1 \leftarrow S$ {Tabu search agent applies tabu search without exchanging solutions with other agents}
  end if
  $iteration = iteration + 1$
end while
Return S_{best_TS} to mediator agent

Conditions and actions of weight matrix The actions considered by the tabu search agents are as follows:

- A_1 = activating other tabu search agents;
- A_2 = activating the strong perturbation behavior in the perturbation agent;
- A_3 = activating the reduced perturbation behavior in the perturbation agent.

The set of the conditions are:

- C_1 = the local best k-coloring is improved in recent q_1 generations and this improvement is a small improvement in the fitness function value f;
- C_2 = the local best k-coloring is improved in recent q_2 generations;
- C_3 = the local best k-coloring is improved in recent q_3 generations and $q_3 > q_2$.

where q_1, q_2 and q_3 are parameters set by the user according to the total generation number.

Each of these conditions promotes a certain action. Thus, C_1 increases the chance of activating other tabu search agent, to reinforce intensification. C_2 and C_3 reinforce the action of triggering the perturbation agent, in order to increase diversification. The selection of the most suitable action is controlled by the corresponding weight matrix of each tabu search agent.

2.5 Perturbation operator agent

The perturbation agent, triggered by tabu search agents (lines 18-20 of Algorithm 3), creates a disturbed k-coloring solution by exploring two types of perturbations. The new k-coloring is then sent to the tabu search agent (line 25 of Algorithm 3) for further improvement.

Reduced perturbation technique The reduced perturbation technique can be triggered when a tabu search agent observes a slight search stagnation (condition C_2 of Section 2.4). From the k-coloring received from the tabu search agent, the perturbation agent makes t moves to create a new solution, where each move changes randomly the color of a conflicting vertex of the incumbent solution. The number t of moves is chosen randomly between 1 and $conf$ (where $conf$ is the number of conflicting vertices).

Strong perturbation technique The strong perturbation technique is performed when a tabu search agent tabu search agent observes deep search stagnation. The perturbation agent uses the shared archive of elite k-colorings to create a new solution. It extracts the number of occurrence of each vertex x colored by each color class V_i. Starting with an uncolored graph, each vertex x is colored with a color class V_i which has the smallest occurrence number. Dedicated data structures are employed to avoid the creation of the same solution for future calls to the perturbation agent.

2.6 Crossover agents

When the mediator agent decides to activate the crossover agents (line 13 of Algorithm 2), two crossover agents are created based on two different operators from the literature: the AMPaX operator [18] and the GPX operator [10]. The new k-coloring solutions from the two crossover agents are sent to the mediator agent to continue the search process. Experimental results showed that the joint use of these two crossover operators performs better than any of them used alone.

3 Experimentation

3.1 Experimental results

In this section, we present experimental results of our DH-GCP algorithm on a set of well-known DIMACS coloring benchmarks and compare the results with other state-of-the-art coloring algorithms from the literature. Our DH-GCP algorithm was programmed in Java using the multi-agent platform Jade. The program was run on a computer with a Core I5 2.5 GHz, 8GB of RAM.

Table 1. Computational results of DH-GCP on the set of difficult DIMACS challenge benchmarks

						DH-GCP		
Instances	$\|V\|$	$\|E\|$	dens	k^*	references	k	hit	time(m)
DSJC250.5	250	15,668	0.50	28	[10, 11, 16, 19, 22, 23, 26, 14, 28]	28	10/10	5
DSJC500.1	500	12,458	0.10	12	[2, 11, 16, 19, 22, 23, 26, 14, 28, 21]	12	10/10	6
DSJC500.5	500	62,624	0.50	47	[21]	-	-	-
				48	[2, 10, 11, 16, 19, 22, 26, 14, 28]	48	10/10	85
DSJC500.9	500	112,437	0.90	126	[2, 11, 16, 19, 22, 23, 26, 14, 28, 21]	126	10/10	320
DSJC1000.1	1000	49,629	0.10	20	[10, 2, 11, 16, 19, 22, 23, 26, 14, 28, 21]	20	10/10	441
DSJC1000.5	1000	249,826	0.5	82	[21]	-	-	-
				83	[10, 19, 22, 23, 14, 28]	83	10/10	205
DSJC1000.9	1000	449,449	0.90	222	[10, 2, 16, 22, 23, 26, 14, 28, 21]	222	4/10	801
DSJR500.1c	500	121,275	0.97	85	[16, 19, 23, 26, 14, 28]	85	10/10	60
DSJR500.5	500	58,862	0.47	122	[16, 24, 23, 26, 14, 28]	122	3/10	480
R250.5	250	14,849	0.48	65	[2, 19, 23, 26, 14, 28]	65	10/10	42
R1000.1c	1000	485,090	0.97	98	[2, 19, 23, 26, 14, 28]	98	10/10	55
R1000.5	1000	238,267	0.48	234	[16]	-	-	-
				238	[26]	240	2/10	1120
le450_15c	450	16,680	0.17	15	[11, 16, 19, 22, 26, 14, 28]	15	10/10	40
le450_15d	450	16,750	0.17	15	[11, 16, 19, 22, 26, 14, 28]	15	10/10	50
le450_25c	450	17,343	0.17	25	[2, 19, 23, 26, 14, 28]	25	10/10	120
le450_ 25d	450	17,425	0.17	25	[2, 19, 23, 26, 14, 28]	25	10/10	42
flat300_26_0	300	21,633	0.48	26	[2, 19, 26, 14, 28]	26	10/10	40
flat300_28_0	300	21,695	0.48	29	[22, 18]	30	5/10	500
flat1000_50_0	1000	245,000	0.49	50	[11, 16, 19, 22, 26, 14, 28, 21]	50	10/10	40
flat1000_60_0	1000	245,830	0.49	60	[11, 16, 19, 22, 26, 14, 28, 21]	60	10/10	45
flat1000_76_0	1000	246,708	0.49	81	[14, 21]	-	-	-
				82	[19, 23, 26, 28]	82	10/10	280
C2000.5	2000	999,836	0.50	145	[14]	-	-	-
				146	[28]	147	1/5	8000
latin_sqr_10	900	307,350	0.76	97	[26]	98	2/10	600

Each instance was solved 10 times independently (5 times for very large graphs). The algorithm was stopped when a legal k-coloring was found or the fixed execution timeout was reached. For all instances, a timeout limit of 10 days was used except for the large graph C2000.5 where a limit of 20 days (note that large computing times were usually allowed in the literature on GCP). We adjusted the parameters of the proposed algorithms by an experimental study. The number of iterations for each tabu search agent ($iter\_max$) was fixed to 1000. The parameters max_opt (for mediator agent) and max_opt_TS (for tabu search agent), that evaluate the improvement of solutions between generations, were fixed to 20 and 2 respectively. For $interval$, we considered the same value 10 for the same agents. The rate μ used in updating the weight matrices was fixed to 0.9.

Table 1 summarizes the computational results of our DH-GCP algorithm. Columns 2-4 show the features of the tested instance: the number of vertices ($|V|$), the number of edges ($|E|$) and the density of the graph ($dens$). Columns 5 and 6 corresponds to the best known results k^* ever reported in the literature and the corresponding references. The remaining columns give the computational results of our DH-GCP algorithm: the smallest number of colors needed to obtain a legal k-coloring, the success rate ($\#hit$) and the average time for reaching the best legal k-coloring ($time$ in minutes).

Table 1 shows that the results obtained by our DH-GCP algorithm are competitive with respect to many state of the art algorithms in terms of solution quality (i.e., the number of colors used). It can reach previous best known results except for 7 very difficult cases (DSJC500.5, DSJC1000.5, flat300_28_0, flat1000_76_0, latin_sqr_10, C2000.5 and R1000.5) for which few algorithms are able to attain the best known results. For these 7 instances, the deviation between our results and the best-known results is respectively 0.021 (for DSJC500.5), 0.012 (for DSJC1000.5), 0.034 (for flat300_28_0), 0.012 (for flat1000_76_0), 0.002 (for R1000.5), 0.013 (for C2000.5) and 0.01 (for latin_sqr_10) respectively. Even if we do not show detailed comparisons with individual algorithms due to space limit, we mention that the results achieved by DH-GCP remain competitive compared with many reference coloring algorithms in terms of solution quality.

4 Conclusion

The proposed distributed hybrid algorithm for the Graph Coloring Problem (DH-GCP) relies on the principles of multi-agent systems to explore a search space with the help of an ensemble of working agents (tabu search agents, crossover agents, perturbation agents). These agents are coordinated by a mediator agent using a reinforcement learning mechanism in order to make right search decisions. Decisions are influenced by a learning-based probabilistic strategy which dynamically adjusts the application probability of a particular action under a specific condition. According to whether the search process needs to be intensified or diversified, the mediator agent triggers, based on a weight matrix,

either an intensification agent (tabu search agents) or a diversification agent (perturbation agents, crossover agents).

The proposed algorithm was assessed on a set of 23 difficult DIMACS coloring benchmarks. The computational results showed that DH-GCP was able to reach the previous best known results except for 7 very difficult cases and remains competitive compared to many coloring algorithms. On the other hand, the current version of the algorithm, which is a proof-of-concept prototype, is rather time consuming, partially due to the multi-agent platform Jade used for its implementation. One possible way to improve the computational efficiency of the algorithm would be to envisage a dedicated distributed implementation. Finally, both this work and the previous study on the quadratic assignment problem [25] demonstrate that the proposed framework is general enough to be adapted to solve other combinatorial search problems. It would be worthy of investigating this multi-agent based optimization framework within other settings.

Acknowledgments

We are grateful to the referees for valuable suggestions and comments which helped us improve the paper. The work is partially supported by the PGMO project (2013-2015, Jacques Hadamard Mathematical Foundation, Paris).

References

1. Avanthay, C., Hertz, A., Zufferey, N.: A variable neighborhood search for Graph coloring. European Journal of Operational Research 151(2): 379–388 (2003)
2. Blochliger, I., Zufferey, N.: A graph coloring heuristic using partial solutions and a reactive tabu scheme. Computers and Operations Research 35(3): 960–975 (2008)
3. Brélaz, D.: New methods to color the vertices of a graph. Communications of the ACM. 22(4): 251–256 (1979)
4. Chalupa, D.: Population-based and learning-based metaheuristic algorithms for the graph coloring problem. In: Krasnogor, N., Lanzi, P.L. (eds.) GECCO, pp. 465–472. ACM (2011)
5. Chiarandini, M., Stützle, T.: An application of iterated local search to graph coloring. In: Johnson, D.S., Mehrotra, A., Trick, M. (eds.) Proc. of the Computational Symposium on Graph Coloring and its Generalizations, Ithaca, New York, USA, pp. 112–125 (2002)
6. Dorne, R., Hao, J.K.: Tabu Search for graph coloring, T-coloring and Set T-colorings. In: Osman, I.H., et al. (eds.) Metaheuristics 1998: Theory and Applications. ch. 3. Kluver Academic Publishers (1998)
7. Dorne, R., Hao, J.K.: A new genetic local search algorithm for graph coloring. PPSN-98, vol 1498 of LNCS, pp. 745–754, Springer-Verlag, (1998)
8. Johnson D.S., Trick M. (ed.): Cliques, coloring, and satisfiability: Second DIMACS implementation challenge, DIMACS Series in Discrete Math. and Theor. Comput. Sci. vol. 26 (1996)
9. Fleurent, C. and Ferland, J.A.: Genetic and hybrid algorithms for graph coloring. Annals of Operations Research 63: 437–461 (1996)

10. Galinier, P., Hao, J.K.: Hybrid evolutionary algorithms for graph coloring. Journal of Combinatorial Optimization 3(4): 379–397 (1999)
11. Galinier, P., Hertz, A. and Zufferey, N.: An adaptive memory algorithm for the K-colouring problem. Discrete Applied Mathematics 156(2): 267–279 (2008)
12. Galinier, P., Hamiez, J.P., Hao, J.K. and Porumbel D.: Recent advances in graph vertex coloring. In I. Zelinka, A. Abraham, V. Snasel (Eds.) Handbook of Optimization. Springer. pp 505–528 (2013)
13. Guo, Y., Goncalves, Y., and Hsu, T.: A multi-agent based self-adaptive genetic algorithm for the long-term car pooling problem. Journal of Mathematical Modelling and Algorithms in Operations Research 12(1): 45–66 (2013)
14. Hao, J.K., Wu Q.: Improving the extraction and expansion method for large graph coloring. Discrete Applied Mathematics 160(16-17): 2397–2407 (2012)
15. Hertz, A., de Werra, D.: Using tabu search techniques for graph coloring. Computing 39(4): 345–351 (1987)
16. Hertz, A., Plumettaz, M., Zufferey, N.: Variable space search for graph coloring. Discrete Applied Mathematics 156(13): 2551–2560 (2008)
17. Johnson, D., Aragon, C., McGeoch, L., Schevon, C.: Optimization by simulated annealing: An experimental evaluation; Part II, Graph coloring and number partitioning. Operations Research 39(3): 378–406 (1991)
18. Lü, Z., Hao, J.K.: A memetic algorithm for graph coloring. European Journal of Operational Research 203(1): 241–250 (2010)
19. Malaguti, E., Monaci, M. and Toth, P.: A metaheuristic approach for the vertex coloring problem. INFORMS Journal on Computing 20(2): 302–316 (2008)
20. Malaguti, E., Toth P.: A survey on vertex coloring problems. Intl. Transactions in Operational Research 17(1): 1–34. (2010)
21. Moalic, L., Gondran A.: The new memetic algorithm HEAD for graph coloring: an easy way for managing diversity. In Ochoa and Chicano (Eds.) EvoCOP2015, vol 9026 of LNCS, pp. 173–183 (2015)
22. Porumbel, D., Hao, J.K. and Kuntz, P.: A search space "cartography" for guiding graph coloring heuristics. Computers and Operations Research 37(4): 769–778 (2010)
23. Porumbel, D., Hao, J.K. and Kuntz, P.: An evolutionary approach with diversity guarantee and well-informed grouping recombination for graph coloring. Computers and Operations Research 37(10): 1822–1832 (2010)
24. Prestwich, S.: Coloration neighbourhood search with forward checking. Annals of Mathematics and Artificial Intelligence 34(4): 327–340 (2002)
25. Sghir, I., Hao, J.K., Ben Jaafar, I., and Ghédira, K.: A multi-agent based optimization method applied to the quadratic assignment problem, Expert Systems with Applications 42(23): 9252–9263 (2015)
26. Titiloye, O. and Crispin, A.: Graph coloring with a distributed hybrid quantum annealing algorithm. In J. O'Shea et al. (eds.), Agent and Multi-Agent Systems: Technologies and Applications, vol. 6682 of LNCS, pp. 553–562 (2011)
27. Trick, M.A., Yildiz, H.: A large neighborhood search heuristic for graph coloring. In: Van Hentenryck, P., Wolsey, L.A. (eds.) CPAIOR 2007. vol. 4510 of LNCS, pp. 346–360 (2007)
28. Wu, Q., Hao, J.K.: Coloring large graphs based on independent set extraction. Computers and Operations Research 39: 283–290 (2012)