# A multi-agents approach for a graph colouring problem

B. Mermet, G. Simon, M. Flouret

April 24, 2002


## Abstract

This paper presents a multi-agent system for solving one of the graph colouring problems. It differs from others, agents or not, models in the way that our main goal is to optimize the existing colouring of a graph. First, we recall the graph colouring problem and then present a multi-agents approach. We end with the presentation of the automaton formalism of agents, and an application of this method.


**Keywords:** agent, automaton, behaviour, graph, colouring problem.

## 1 Introduction

Our aim in this paper is to show how to apply multi-agent principles to solve an NP-hard problem. We choose a quite general, by its applications, problem of graph colouring. Its problematic approach being easy, we can focus on the agent modelling. Indeed, the main goal is to develop a methodology which could be applied for other problems.

First we give a general presentation of the choosen graph colouring problem. It consists in minimizing the number of colors used to color the graph, that is the chromatic number of a graph. Then, we present our multi-agents approach, other points of view, and an automaton model used to specify agents behaviours.

Our solving starts from a correctly coloured graph (without any colour constraint) in which each node can be recoloured many times until having a satisfying colour. By this way, our agents number is set to the number of nodes. As far as the colouring problem is concerned, the main contribution of the proposed approach is to ensure a correctly coloured graph at any time.

## 2 The graph colouring problem

Many problems have been associated to graph colouring ones since Gurthrie and then Heawood [17] worked on maps, until it was updated with applications in the transmitting networks area.

Applications of this problem are typically situations where allocation of ressources must be performed, with critical constraints. This is for instance the case of time tabling and scheduling [1, 18]. This is also the case of register allocation in efficient compilers. Another usage of such algorithms consists in assigning frequencies for cellular radio networks [16].

The problem of graph colouring being NP-hard, algorithms looking for optimal solutions are numerous [7] but rarely usefull for real-size problems (there is no algorithm being able to solve such problems in polynomial time). Many heuristic-based approaches have been proposed to solve the colouring problem. Those algorithms do not guarantee to find the best solution, but try to propose an *acceptable* one. Many of them are based on an implicit enumeration. There are also solutions using genetic algorithms [8, 12, 13, 14]. Some models (constructive ones) start from an empty solution and build it by successive colourings of nodes. More recently, a few solutions using ants algorithms [5, 20] have been proposed.

This kind of algorithm has such a name for similarities between the model and ants colonies. Indeed, each ant has no notion of the global aim, but the

ants society behaviour allows to converge towards the goal. Here, ants have to move around nodes and change their colour according to local criteria. They move along edges with various constraints (parallel search, memorization, ants number for example).
Like ants algorithms, our approach is an agent-oriented solution. However, there is an essential difference between our proposal and others. Indeed, for most proposed solutions until now, the algorithms color a small part of the graph, and then try to progress to color a bigger part, and so on, until all the graph has been colored. So, to obtain a correct solution, we must wait until the algorithm ends, and this can be long if the graph has a great number of nodes. In our proposal, we start with a correct (but not optimized) solution in which each node of the graph has not the same color than its neighbours. As the time goes, our algorithm tries to suppress colors, keeping a correct coloration of the graph. So, at any time, we can stop our algorithm and obtain a correctly coloured graph. Of course, the more our algorithm will work, the more pertinent the proposed solution will be.

# 3 Definitions

Here are the main definitions and notions about graphs and colouration which will be used in the sequel. For further details it can be refered for example to [6]. They are followed by properties of coloured graphs.

## 3.1 Graphs

An oriented (resp. non oriented) *graph* is a pair $G = (N, E)$, with $N$ and $E$ two sets such that elements of $E$ are ordered (resp. unordered) couples $(u, v) \in N^2$, and $N \cap E = \emptyset$. The elements of $N$ are *nodes*, those of $E$ are the *edges*.
A graph is told *complete* if each couple $(u, v)$ of $N^2$ corresponds to an edge. It is *connected* if for all couple $(u, v)$ it exists a $u-v$ path. A *clique* of $G$ is a complete subgraph of $G$. Let $u, v$ two nodes of a graph $G = (N, E)$. They are *neighbours* if $(u, v) \in E$. We will denote $V(u)$ the set of all neighbours of $u$.

## 3.2 Coloured graph

Let $C(u)$ the color associated to a node $u$, and $C(V(u))$ the set of colours of $u$ neighbours. Now we will consider a $k$-coloured graph. The $k$-colouring of a graph $G = (N, E)$ is the attribution, to each node, of a colour among $k$ such that, for each edge $(u, v)$ of $E$, $C(u) \neq C(v)$.
A graph is *$k$-colourable* if a $k$-colouring can be applied[1]. The smallest $k$ such that $G$ is $k$-colourable is the *chromatic number* of $G$ and denoted $\chi(G)$.

## 3.3 Properties

As the chosen method is based on an efficient colouration for each node and its neighbours, we transpose the global problem to a set of local ones. To verify that it allows to obtain a better colouring, the following definitions and properties are needed.
The *local chromatic number* of a node $u$ is $lcn(u) = max\{|C|, \forall C$ clique of G $/u \in C\}$. The *current chromatic number* of $u$ is $ccn(u) = |\{c(u)\} \cup \{c(v)/v \in V(u)\}|$. Then, a node $u$ *satisfies its lcn* if and only if $lcn(u) = ccn(u)$.
The following properties are used to implement our solution.

**Theorem 3.1** *Let $G = (N, E)$ be a graph. For all node $u \in N$, if $G$ is correctly coloured, then $ccn(u) \leq lcn(u)$.*

To prove the theorem 3.1, the following results are needed.

**Lemma 3.1** *Let $G$ a correctly coloured graph.*
*For each clique $C = (N' = \{n_1, \cdots, n_p\}, E')$ of $G$, we have $C(n_i) \neq C(n_j), \forall i, j \in [1..p], i \neq j$.*

**Proof 3.1** *Let $n_i, n_j \in C$ such that $C(n_i) = C(n_j)$. As $C$ is a clique, it exists an edge $n_i - n_j$ and the graph is not correctly coloured.*

**Lemma 3.2** *Let $C = (N' = \{n_1, \cdots, n_p\}, E')$ a clique of $G$. For each node $n_i$ of $C$, $n_j \in V(n_i), j \in [1..i-1, i+1...p]$.*

[1]For $k \geq 3$, decide wether a graph is $k$-colourable or not is NP-hard.

**Proof 3.2** *A clique of $G$ is a complete subgraph of $G$, which gives the result.*

**Corollary 3.1** *Let $C = (N' = \{n_1, \cdots, n_p\}, E')$ a clique of $G$. For each node $n_i$ of $C$, $N' \subseteq V(n_i)$.*

**Proof 3.3** *The lemma 3.1 and the corollary 3.1 imply that if the graph $G$ is correctly coloured, we have for all clique $C = (N', E')$ of $G$,*

$$\forall u \in N' ccn(u) \geq |C|,$$

*that is*

$$\forall u \in N' ccn(u) \geq max(|C|) = lcn(u),$$

*which gives the result.*

**Theorem 3.2** *Let $G = (N, E)$ a graph, and let $\chi(G) = n$. For each node $u \in N$, we have $lcn(u) \leq n$.*

**Proof 3.4** *If $lcn(u) > n$, it would exist $u \in N$ such that more than $n$ colours would be needed to colour correctly the graph at this step. Then $\chi(G) \geq n$.*

**Remark 3.1** *Despite these two theorems, even if all the nodes of a graph satisfy their lcn, the chromatic number of the graph can not always be reached.*

**Remark 3.2** *Some graphs cannot be coloured such that each node satisfies its lcn.*

# 4 Agent, Multiagent System and Agentification

There are many definitions of the agent notion [3, 4, 9, 11, 15, 21]. An *agent* can be seen as an autonomous and reactive entity acting in a given environment that it takes into account through perceptions. Most of the time, an agent is situated in its environment, and so it perceives only a local part of the environment. Moreover, an agent being autonomous, no other agent can modify directly its properties. These two properties are called *the locality principles.*
A *Multi-Agent System* (MAS) is a program that allows to many agents to act together in a given environment.

The main principle of our solution is based on the fact that the lower bound of the current chromatic number of a node is its *lcn* (th.3.1). Thus, the aim of the algorithm is to turn down the *ccn* of a node while it does not satisfy its *lcn*.

Now, an explicit model of agents behaviour is given for the colouring problem. These agents have knowledge about their environment, can act on it and themselves. Their behaviour is induced by a goal they have to reach. This is a charateristic of eco-solving principle [10] where individual behaviours of a population of agents allow to obtain a solution for a problem. Each agent is told *eco-agent.* It has a set of elementary behaviours, and its goal is to reach a "satisfaction state". If it cannot be satisfied, it tries to flee.
For our model, we consider the agents as reactive and social ones, that is they can react to changes of their environment. The main agents are *node agents*, whose goal and actions are defined further. An agent is associated to each node of the considered graph. At the begining, the graph is correctly coloured, without special optimization.

The model given below comes from eco-resolution as the node agents try to satisfy themselves or flee.

Here, only node agents can change their own features, and they can only send information messages to the others, for clear conception.

## 4.1 Intuitive algorithm

The first idea was based on the properties and theorems presented earlier about *ccn* and *lcn*. We planed to give to "node agents" the goal of making their *ccn* decrease until it satisfies their *lcn*. However, to perform such a task, node agents should have to change the colors of their neighbours, and this would have broken the locality principles of agents.

## 4.2 Real algorithm

In order to respect the locality principle, we finally assigned to the node agents the goal of making decrease the ccn of their neighbours, changing their own color. This goal can be reached by solving a set of subgoals. This new algorithm leads to the achievement of the same goal than the first one. Indeed, as each node agent has at least one neighbour (we are only interested in connected graphs), its neighbours will make its *ccn* decrease.

#### 4.2.1 How a node agent changes its color

When a node agent acts (one of its neighbours, $n$, has a *ccn* greater than its *lcn*), it has to choose a color :

- existing in the graph ;
- making the *ccn* of the neighbour $n$ decrease ;
- being different from the colors of its neighbours.

Notice that the modifications of the color of nodes satisfy, at each step, the colouring rule given above.

In multi-agents systems, agents usually have just a local vision of the environment. Here, we consider that an agent can only see (and communicate with) its neighbours. So, to find a color verifying the two first items enumerated above, the acting agent asks to its neighbour $u$ the colors of its neighbours, and this gives a first set $C(V(u))$, the colors set of the neighbours of $u$.

To verify the third point, the acting agent $a$ first asks to its neighbours their colors and constructs the set $C(V(a))$ of these colors. Then it chooses a color among the new set $S = C(V(u)) \backslash \{C(V(a)) \cup C(a)\}$[2].

#### 4.2.2 When changing its color is not possible

Sometimes, the set $S$ described above is empty. This is because one of the colors of the set $C(V(u))$ is a member of the set $C(V(a))$. In such a case, the node agent $u$ attacks one of its neighbours whose color is in the set $C(V(u))$.

An agent under attack flees. An agent that has to flee tries to take another color, but without taking into account the *ccn* of its neighbours : it chooses a color among all the colors of the neighbours of its neighbours and which is not a color of its neighbours. If it is not possible, the fleeing agent do nothing special.

#### 4.2.3 Why nodes are agents

The behaviour of the node agents we described in the previous part is an eco-agent behaviour:

- the node agents are autonomous : they act without receiving orders from other entities ;
- the node agents are reactive : according to the perceptions they receive from their environment (and their neighbours), they choose different actions ;
- the node agents can attack other nodes, and nodes under attack have a fleeing strategy ;
- the node agents modify their own colours and perceive only the colours of their neighbours. So, the locality principles are verified.

#### 4.2.4 Limits of the algorithm

The algorithm presented here is based on neighbourhood, and so, it will not work well on graphs that are not connected (however, each subgraph will be correctly processed).

### 4.3 Other agents

Topological and drawer agents have been created too for coordination and implementation reasons. The topological agent creates the initial graph, node agents with their characteristics (e.g. list of neighbours, initial colors), and a drawer agent giving a graphic view of the graph updated when colors change. Notice that it exists two kinds of communications : asynchronous ones for sending informations on request (e.g. colour of neighbours, *ccn*), and synchronous ones (e.g. stoping signal).

### 4.4 Automata formalism for an agent

Automata with multiplicities allow to give a formalism for agents behaviour, preserving rational operations over automata while not ATN or Petri

[2] The new color must be different from the previous one, that is why $C(a)$ is removed from possible colors.

nets. This choice allows quite easily compositions between agents behaviour.
First of all, we have to present automata with multiplicities. Such an automaton is defined over a finite alphabet $\Sigma$ and a semiring $(K, \oplus, \odot)$, with union and concatenation operations. Let $\mathcal{A} = (\Sigma, Q, I, T, \delta)$, with $Q$ a finite set of states, $I$, $T$ respectively the sets of initial states and final ones, in fact functions such that $I, T : Q \to K$, and $\delta : Q \times \Sigma \times Q \to K$ the transition function.
Here the set $\Sigma$ corresponds to the perceptions and $K$ to the set of possible successions of elementary actions, transitions being labeled by couples (perception, reaction).
The behaviour of a node agent (responses to stimuli) can be given with an automaton [2], communications with other agents (node, topological or drawer) being possible in each one of the automaton states according to rules given in paragraph 4.2.
Let $\Sigma = \{Init, S, NS, A, IC, E\}$, with the perceptions

$Init$ : Synchronization signal,

$S$ : goal satisfied,

$NS$ : goal unsatisfied,

$A$ : attacked by an other node agent,

$IC$ : recolouring impossible,

$E$ : stop signal.

Now the set of elementary actions (behaviour primitives), performed when a state is reached, is defined by (denoting $N0$ the neutral action for $\oplus$, and $N1$ the neutral one for $\odot$) { In,L,C,Sa,At,St,F,N1,N0 Fa } with

$In$ : initialization of agent parameters,

$L$ : lauching agent functionalities,

$C$ : recolouring itself,

$Sa$ : satisfied message,

$At$ : agression,

$St$ : stop process,

$F$ : flee,

$Fa$ : final message.

The node agents behaviour is given by figure 3.

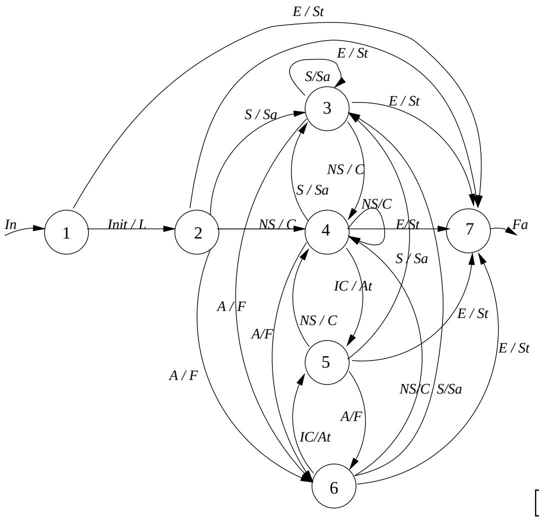


Figure 1: Node agent behaviour automaton

# 5 Conclusion

In this paper, we have described how to apply multi-agent principles to solve a coloring problem. To achieve that goal, we were guided by the locality principles that we have presented. This led us to define appropriate "node agents" which solve together the problem. Automata with multiplicities helped us to specify and implement the behaviour of these agents. These automata can be used to define many kinds of agents, even if here, we used them to specify eco-agents.

The multi-agent system described in this paper is currently developped using the MadKit platform [19] and Java language. In the near future, our research effort will focus on the methodology direction in order to generalize the agentification principles we used.

## References


[1] E. K. Burke, D. G. Elliman, and R. F. Weare. A university timetabling system based on graph colouring and constraint manipulation. *Journal of Research on Computing in Education*, 27(1):1–18, 1994. citeseer.nj.nec.com/burke93university.html.

[2] V. Jay D. Olivier C. Bertelle, M. Flouret and J.-L. Ponty. Automata with multiplicities as behaviour model in multi-agent simulations.

[3] A. Cardon. Adaptative multiagent systems. Number LIP6-2000-012, 2000.

[4] A. Cardon. *Conscience artificielle et systèmes adaptatifs*. Eyrolles, Paris, 2000.

[5] F. Comellas. An ant algorithm for the graph colouring problem. http://citeseer.nj.nec.com/112038.html.

[6] R. Diestel. *Graph Theory*. Springer-Verlag, New-York, 2000.

[7] dimacs92. Clique and coloring problems, a brief introduction, with project ideas, 1992. ftp://dimacs.rutgers.edu/pub/challenge.

[8] Raphaël Dorne and Jim-Kao Hao. A new genetic local search algorithm for graph coloring. In Agoston E. Eiben, Thomas Bäck, Marc Schoenauer, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature – PPSN V*, pages 745–754, Berlin, 1998. Springer. http://citeseer.nj.nec.com/dorne98new.html.

[9] A. Drogoul. *De la simulation multi-agents à la résolution collective de problèmes : une étude de l'émergence de structure d'organisation dans les systèmes multi-agents*. PhD thesis, Univ. Paris VI, 1993.

[10] J. Ferber. *Objets et agents : une étude des structures de représentation et de communications en intelligence artificielle*. 1989. PhD thesis, Univ. Paris VI.

[11] J. Ferber. *Les systèmes multi-agents*. InterEditions, Paris, 1995.

[12] G. Ribeiro Filho. Improvements on constructive genetic approaches to graph coloring. http://citeseer.nj.nec.com/242708.html.

[13] G. Ribeiro Filho and G. Lorena. A constructive genetic algorithm for graph coloring, 1997. http://citeseer.nj.nec.com/filho97constructive.html.

[14] G. Ribeiro Filho and G. Lorena. Constructive genetic algorithm and column generation: an application to graph coloring, 2000. http://citeseer.nj.nec.com/filho00constructive.html.

[15] S. Franklin and A. Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Agent Theories, Architectures, and Languages*, pages 21–35, 1996. http://citeseer.nj.nec.com/article/franklin96is.html.

[16] J. Hao and L. Perrier. Tabu search for the frequency assignment problem in cellelar radio networks, 1996. http://citeseer.nj.nec.com/hao96tabu.html.

[17] P.J. Heawood. Map colour theorems. volume 24, pages 332–338, 1890.

[18] S. Miner, S. Elmohamed, and H. Yau. Optimizing timetabling solutions using graph coloring, 1995. http://citeseer.nj.nec.com/miner95optimizing.html.

[19] J. Ferber O. Gutknecht and F. Michel. Madkit : une expérience d'architecture de plateforme multi-agent générique. 2000.

[20] A. Vesel and J. Zerovnik. How good can ants color graphs? *Journal of computing and Information Technology - CIT*, 8:131–136, 2000. http://citeseer.nj.nec.com/443529.html.

[21] M. Wooldridge and N. Jennings. Agent theories, architectures and language : a survey. volume LNAI 890, pages 1–32. in M. Wooldridge and N. Jennings, editors, Springer-Verlag, 1994.