Further results of Gravitational Swarm Intelligence for Graph Coloring

Israel Rebollo-Ruiz and Manuel Graña
Computational Intelligence Group
University of the Basque Country (UPV/EHU)
Spain
email: beca98@gmail.com, ccpgrrom@si.ehu.es

***Abstract*—We provide enhanced results of an innovative nature inspired algorithm to solve the Graph Coloring Problem (GCP): the Gravitational Swarm Intelligence (GSI). Swarm Intelligence solves complex problems by extracting information from the spatial configurations of of agents that decide their actions/motions based on local available information, without any central control system. In the GSI application to GCP, agents correspond to graph's nodes, moving as particles in the gravitatory field defined by some target objects corresponding to graph node colors. Knowledge of the graph's topology is available to the agents' local navigation control as friend-or-foe information. We discuss the convergence of the algorithm and test it over well-known benchmarking graphs, achieving good results in a reasonable time.**

***Keywords*-Gravitational Swarm Intelligence; Graph Coloring;**

I. Introduction

The Graph Coloring Problem (GCP) is a classical combinatorial optimization problem which is of NP-complete complexity [9], [15], [16], [18], [19]. The GCP consist in assigning a color to the nodes of a graph with the restriction that any pair of nodes that are linked can't have the same color. The chromatic number K is the minimum number of colors needed to color the graph. Classical algorithms to solve GCP are deterministic search algorithms [2], [7], [6]. Heuristics and random search allow to obtain approximations to the optimal solutions in bounded time. Some recent approches have applied Ant Colony Optimization (ACO) [10], Particle Swarm Optimization (PSO) [14], and Reynolds Boyd swarms [4], [11].

We consider a nature inspired strategy to solve this problem following a Swarm Intelligence (SI) [25] approach. The bee hives [1], ant colonies [12] and flocking birds [8], [24], [23] are examples of such swarms. In SI models, the emergent collective behavior is the outcome of a process of self-organization, where the agents evolve autonomously following a set of internal rules for its motion and interaction with the environment and the other agents. Intelligent complex behavior appears from simple individual behaviors. An important feature of SI is that there is no leader agent or central control. One of its biggest advantage is that it allows a high level of scalability, because the problem to be solved is naturally divided into small problems, one for each agent. In real life, when some ants of a colony (also valid for bees, birds or other swarms) fail in its task, it won't alter too much the behavior of the overall system, and in some problems occurs the same, so the algorithm based on SI can be robust against individual failure.

In this paper we enhance our proposition of a Gravitational SI (GSI) for the approximate solution of GCP [22]. GSI agents correspondend to graph nodes. We place the GSI agents in a torus shaped space, moving towards the color goals. Agents are attracted to specific space places (color goals) where the corresponding graph node acquires a color. Such attraction is modeled as a gravitatory field extended to the entire space. When the GSI agent reaches a color goal, it remains there unless pushed out by repulsive forces of antagonistic agents. The friend/foe relation between SI agents is determined by the graph to be colored. Nodes connected by an arc correspond to antagonistic GSI agents exerting mutually repulsive forces. When an agent is impeded to reach any color goal because of these repulsive forces, its "discomfort" grows increasing unidirectionally the force it can exert on foe agents to push them out of the color goals. This discomfort reaction allows the system to escape local minima that are not solutions of the GCP. The SI dynamics reaches a termination stable state when all the agents are in a color goal and there is no conflict among them.

The rest of the paper is organized as follow: section II presents our Gravitational Swarm Intelligence algorithm. In Section III we discuss the convergence of the algorithm. In Section IV we show experimental results comparing our algorithm performance with other methods using well-known graphs. Finally, section V gives some conclusions and lines for future work.

II. Gravitational Swarm Intelligence

The natural inspiration of our algorithm does not come from living beings, such as ants, bees or birds, but from a basic physics law: the gravitational atraction between objects. We construct a world where agents nav-

igate through the space attracted by the gravitational pull of specific objects, the color goals, and may suffer specific repulsion forces, activated by the friend-or-foe nature of the relation between agents.

Let be $G = (N, E)$ a graph with a set of nodes $N = \{1, \ldots, n\}$ and edges $E \subseteq N \times N$. We define B as a group of GSI agents $B = \{b_1, b_2, ..., b_n\}$ each corresponding to a graph node. Each agent navigates inside a square planar toric world. Each GSI agent moves through this space according to a speed vector $\overrightarrow{v_i}$. In any moment of time we know the position attribute of each agent $p_i(t) = (x_i, y_i)$ where x_i and y_i are the cartesian coordenades in the space. When $t = 0$ we have the initial position of the agents $p_i(0) = (x_{0i}, y_{0i})$. Supposse that we want to color the graph with K colors, denoting as $C = \{1, 2, ..., K\}$ the set of colors. If K is the minimum number of colors that allow to color the graph, then K is the chromatic number of the graph. We assign to these colors, K fixed points in space, the color goals $CG = \{g_1, \ldots, g_K\}$, endowed with a gravitational attraction resulting in a velocity component $\overrightarrow{v_{gc}}$ afecting the agents. The attraction force decreases with the distance, but affects all the agents in the space.

When the euclidean distance between an agent and the color goal is below a threshold *nearenough*, the agent stops moving and the corresponding node is assigned to this color. We denote the set of agents whose position is in the region of the space near enough to a color neighbourhood of the color as $N(g_k) = \{b_i \text{ s.t. } \|p_i - g_k\| < nearenough\}$. We denote the fact that the node has been assigned to the corresponding color assigning value to a the agent color attribute $b_i \in N(g_k) \Rightarrow c_i = k$. Initial value of the agent color attribute is zero or null. Inside the neighbourhood of a color there is no further gravitational atraction. However, there may be a repulsion force between agents that are conected with an edge in the graph G. In the current implementation of the algorihtm, this repulsion is only effective for agents in the same color goal neighbourhood. The function *enemy* has value 1 if a pair of GSI agents have an edge between them, and 0 otherwise. The repulsive forces experimented by agent b_i from the agents in the color goal g_k are computed as follows: $R(b_i, g_k) = \sum_{N(g_k)} enemy(b_i, b_j)$.

We can model the problem as a tuple $F = (B, CG, \{\overrightarrow{v_i}\}, K, \{\overrightarrow{a_{i,k}}\}, R)$ where B is the group of GSI agents, $\{\overrightarrow{v_i}\}$ the set of agent velocity vectors at time instant t, K the hypothesized chromatic number of the graph and $\{\overrightarrow{a_{i,k}}\}$ are the attraction forces of the color goals exerted on the agents. R denotes the repulsion forces in the neighbourhoord of color goals.

The cost function defined on the global system spatial configuration is:

$$f(B, CG) = |\{b_i \text{ s.t. } c_i \in C \;\&\; R(b_i, g_{c_i}) = 0\}| . \quad (1)$$

This cost funciton is the count of number of graph nodes which have a color assigned and no conflict inside the color goal. The agents outside the neighbourhood of any color goal can't be evaluated, so it can be a part of the solution of the problem. The dimension of the world and the definition of the *nearenough* threshold allows controlling the speed of convergence of the algorithm. If the world is big and the near enough variable is small the algorithm converges slowly but monotonically to the solution, if the world is small and the *nearenough* variable is big the algorithm is faster but convergence is jumpy because the algorithm falls in local minima and needs transitory energy increases to escape them. The reason of this behaviour is that the world is not normalizad and the magnitude of the velocity vector can be bigger than the color goal spatial influence and can cross a goal without falling in it. The dynamics of each GSI agent in the world is specified by the iteration:

$$\overrightarrow{v_i}(t+1) = \begin{cases} 0 & c_i \in C \& (\lambda_i = 1) \\ d \cdot \overrightarrow{a_{i,k^*}} & c_i \notin C \\ \overrightarrow{v_r} \cdot (p_r - p_i) & (c_i \in C) \,\&\, (\lambda_i = 0) \end{cases} , \quad (2)$$

where d is the vector difference of the agent's position p_i and the position of the nearest color goal g_{k^*}, $\overrightarrow{a_{i,k^*}}$ represents the attraction force to approach the nearest goal, and $\overrightarrow{v_r}$ is a random vector to avoid being stuck in spurious unstable equilibrium, towards a random position p_r. Parameter λ_i represents the effect of the degree of *Comfort* of the GSI agent. When a GSI agent b_i reaches to a goal in an instant t, its velocity becomes 0. Every time step that the GSI agent stays in that goal without been disturbed, its *Comfort* increases, until reaching a maximum value $maxconfort$. When an GSI agent b_i outside the color goal g_{k^*} tries to go inside the neighborhood of that color goal, the repulsion force $R(b_i, g_{k^*})$ is evaluated. If the repulsion force is greater than zero then the incoming agent is challenging the stability of the color neighbourhood and at least one agent must leave the goal, which can be the incoming agent itself. The repulsion force is only applied between conected agents. If the *Confort* values of the challenged agents are bigger than 0 then their *Comfort* decreases. If the *Confort* reaches 0, then one conected agent is expelled from the color goal to a random position in the space p_r with velocity v_r. In equation 2 when Comfort is positive the parameter has value $\lambda_i = 0$. If the Repulsion force is greater than zero and the *Comfort* of a GSI agent b_i inside that goal is equal to 0 then$\lambda_i = 1$ and b_i is expelled from the goal. When all the GSI agents stop,

i.e. $\forall i, \overrightarrow{v_i} = 0$; therefore $f(B, CG) = n$ and the GCP of assigning K colors to graph G is solved.

Each color goal has an attraction well spanning the entire space, therefore the gravitational analogy. But in our approach the magnitude of the attraction drops proportionally with the Euclidean distance d between the goal and the GSI agent, but it never disappears. If $\|d\| < nearenough$ then we make $d = 0$, and the agent's velocity becomes 0 stopping it. The flowchart of figure 1 shows the internal logic works of each GSI agent .

III. Convergence issues

We discuss in this section the convergence of the algorithm from an intuitive point of view. The GSI agents start in a position $p_{0i} = \{x_{0i}, y_{0i}\}$ and with an initial speed $\overrightarrow{v_{0i}}$. The direction and value of the speed vector changes with the dynamics of the system. The gravitational attraction of each color goal is the same for all, but the attraction force applied to a GSI agent from the nearest color goal does not add or interfere with the other color goals. The speed is directly proportional to the distance between the GSI agent and its target color goal, when the distance is below a threshold the GSI agent stop. Initially each GSI agent tries to get to the nearest goal. The attraction of the goals is strong when the GSI agent is far away and weak when the GSI agent is near the goal. The algorithm assigns the GSI agent a speed equal to zero when the GSI agent is inside the goal radius. If antagonistic GSI agents are already inside the goal, it tries to expel them out of the goal. An expelled GSI agent is attracted by his new nearest goal and so on, until it can enter in a goal without enemies.

The system reaches an stationary state only if all the GSI agents' speed becomes zero. Then, the algorithm has converged to some fixed state where all of them are inside a color goal and there is no conflict inside the color goal neighborhoods. If the chromatic number is the hypothesized K or lower, then such an state exists. If there are any conflict, the system is no stable, because at least one agent will be expelled from the color goal neighborhood and continue moving searching for an appropriate color goal. An GSI agent speed only becomes zero when is inside a goal without enemies.

When a GSI agent tries to enter inside a goal, if there is one or more enemies in the goal it will try to find another goal empty of enemies. If there is no one, then it will proceed to expel the enemies from one of the goals. The GSI agent selects a random foe and evaluates its λ_i parameter. If it is zero, then that enemy is expelled from the goal to a random point p_r with a velocity magnitude v_r. And all its enemies' *Comfort* inside that goal decreases. It doesn't matter if other enemies $\lambda_j = 0$, the GSI agent can only expel one GSI agent at one step. The GSI agent doesn't stop because it can be still more enemies in that goal and must wait until the next step to get inside. With this behavior, when a GSI agent is inside a goal, it's sure that there are no enemies in the goal. So when our algorithm stops because all the agents have stopped, it has reached a GCP solution, because no two adjacent graph nodes have the same color assignment and all the agents are "inside" a color goal. If the agents never stop, it means that the hypothetical number of colors is lower than the true chromatic number.

IV. Experimental results

We have made experiments with a group of well-known graphs that appear often in the literature. We have implemented our Gravitational Swarm Intelligence (GSI) algorithm and the following four state-of-the-art algorithms to compare results:

1) A greedy backtracking algorithm: this algorithm explores all the search space and always return the optimal solution if exists. As the GCP is a NP-complete problem we can use backtraking only in small problems or especial graphs like the mycielsky graphs.
2) DSATUR (Degree of Saturation): this algorithm developed by Brèlaz [2] is a greedy backtraking algorithm but does not explore exhaustively all the search space. It looks for the biggest clique in the graph, we have used the Bron-Kerbosch [3] algorithm in our implementation, setting its size as the initial number of colors. Then starts the search to determine the color of the remaining nodes of the graph.
3) Tabu Seach: it is a random local search with some memory of the previous steps, so the best solution is always retained while exploring the environment.
4) Simulated Annealing: this random algorithm has a big problem in the graph coloring problem, because there are a lot of neighboring states that have the same energy value. Despite this handycap Simulated Annealing algorithm provides state-of-the-art results for this problem[21].

We have implement all these algorithm because it's easier to compare with our GCP, instead of using result published in the literature. The programing languaje, the computer used or even the estructures used in the implementation can made a big diference between different works. We have implement all the algorithms using Visual Basic .Net, and all the experiments have been run in the same computer.

A. GSI implementation

Even though, our algorithm is about GSI agents moving around the search space, we haven't use any parallel implementation. At each time step all the GSI agents motion is evaluated. After each time step, the

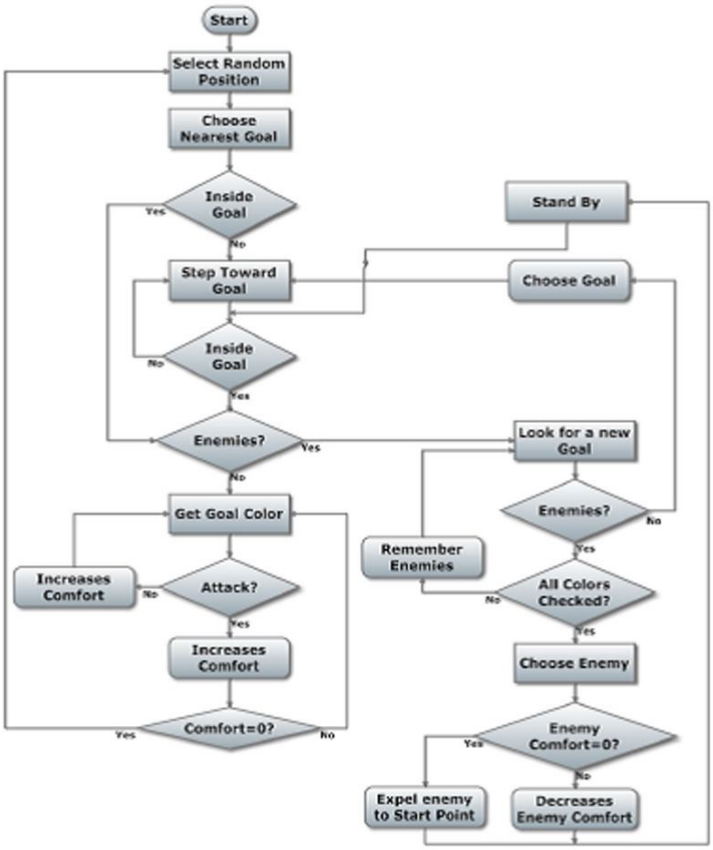

Figure 1. GSI agent behavior flowchart for GCP

Table I
Experimental graph test suite

Graph name	#nodes	#Edges	Density	K
anna	138	986	0.10	11
david	87	812	0.21	11
hommer	561	3258	0.25	13
huck	74	662	0.22	11
jean	80	508	0.16	10
myciel3	11	20	0.36	4
myciel4	23	71	0.28	5
myciel5	47	236	0.21	6
myciel6	95	755	0.17	7
myciel7	191	2360	0.13	8

cost function must be evaluated to see if the problem is solved or not. We have two time reference units, the standard hours, minutes and seconds to compare with other algorithms and the iteration steps to compare experiments over the same graph. The real computing time can change from one computer to another, but the steps will be always the same. When we are evaluating the next position of a GSI agent in the step t, we take into account the position of the other GSI agents in the step $t-1$.

B. Graph test suite

For validation, it's a good idea to use well-known benchmarking graphs, whose chromatic number is known. The Mycielsky graphs [20] are a family of graphs whose chromatic number is equal to the degree of the graph plus one $K = m + 1$ where m is the Mycielsky number. There are also collections of benchmark graphs, such as the DIMACS graphs [16], [17]. For graphs whose chromatic number is unknown the algorithm validation comes from the comparison to other graph coloring algorithms [26], [5].

We include in our test suite the Mycielsky graph family up to order 6 [13]. They serve to tune the algorithms' implementation because of their regular structure and inmediate knowledge of their chromatic number. We include in our test suite a special family of graphs, the Book Graphs, proposed by Knuth. Given a literary work, a graph is created where each node represents a character. Two nodes are connected by an edge if the corresponding characters encounter each other in the book. Knuth creates the graphs for five classic works:

Table II
Graph coloring results over the test graphs of the Backtracking (BT), DSATUR, Tabu Search (TS), Simulated Annealing (SA) and Gravitational Swarm Intelligence (GSI) results. Asterisk (*) means no solution was obtained.

		BT	DSATUR	TS		SA		GSI	
Graph name	K	#back	#back	#iter	%success	#iter	%success	#iter	%success
Myciel3	4	1	1	13	100	21	100	25	100
Myciel4	5	1	1	51	100	716	100	46	100
Myciel5	6	1	1	393	96	407074	28	241	100
Myciel6	7	1	1	970	94	*	0	630	100
Myciel7	8	1	1	1575	92	*	0	1103	98
anna	11	*	1	4921	2	483859	6	718	98
david	11	*	1	*	0	478207	10	1428	92
homer	13	*	*	*	0	*	0	2583	76
huck	11	1	1	3363	54	180975	64	251	98
jean	10	1	1	2471	68	281418	44	439	98

Tolstoy's Anna Karenina (anna), Dicken's David Copperfield (david), Homer's Iliad (homer), Twain's Huckleberry Finn (huck), and Hugo's Les Miserables (jean). Table 1 contains the features of the test suite. The name, number of nodes, number of edges, the density of the graph (calculated as the ratio between the number of edges in the actual graph and the complete graph) and, most important, the best chromatic number found in the literature and in our tests.

In table 2 we show the results of applying the deterministic greedy algorithms backtraking and DSATUR, the Tabu Search and Simulated annealing heuristics, and our GSI algorithm. Backtracking and DSATUR are deterministic algorithm so the result is always the same, we give the number of backtracks needed to solve the problem. We also stop the greedy algorithms after 10.000.000 backtracks due to their big computational time. The other three algorithms are random searches and have been tested 50 times with each graph, we give the average number of steps and the % of success. In all the situations we stop the experiments after 5000 iterations (backtrack or steps), except for the SA that, which we extend up to 10.000.000 steps.

Our Algorithm has the best success ratio of the approximation heuristic algorithms. It manages to find a solution at least once for each problem. In the Myciel family the greedy algorithm are the best due to their special of the graph structure. In the Book Graphs the greedy algorithms have failed where our GSI has succeded.

Although SA has poor results, it has an especial feature that make it very interesting for unknown graphs. The initial number of color is not necesary for this algorithm. We can launch it with a number of colors equal to the number of nodes and the algorithm itself reduces the number of colors on the fly. This is useful to find and upper-bound of the chromatic number of unknown graphs.

Finally, in table 3 we show the experiment time for each graph and method. We take the time of the GSI as a referencial time. The number of steps of the greedy algorithm and the SA have been evaluated using this referencia. We can observate that our GSI algorithm is faster than other random search algorithms and also is faster than the greedy algorithm in an scenario where the greedy algorithm have problems finding the solution.

Graph Name	BT	DSATUR	TS	SA	GSI
Myciel3	1	1	1	1	1
Myciel4	1	1	1	1	1
Myciel5	1	1	11	1067	9
Myciel6	1	1	69	*	55
Myciel7	1	1	307	*	210
anna	*	2	959	596	137
david	*	1	*	319	177
homer	*	*	*	*	2456
huck	1	1	276	134	26
jean	1	1	206	239	48

Table III
Experiment computational time in seconds. (*) means that no solution was found until reaching the limit number of iterations.

V. Conclusions

We proposed a new algorithm for the Graph Coloring Problem using Swarm Intelligence. We have modeled the problem as a collection of agents trying to reach some of a set of goals. Goals represent node colorings, agents represent graph's nodes. The color goals exert a kind of gravitational attraction over the entire virtual world space. With these assumptions, we have solved the GCP using a parallel evolution of the agents in the space. We have argued the convergence of the system, and we have demonstrated empirically that it provides effective solutions in terms of precision and computational time. We will continue to test our algorithm on an extensive collection of graphs, comparing its results with state of the art heuristic algorithms. We are working on a formal convergence proof of the algorithm dynamics.

References

[1] Bahriye Akay and Dervis Karaboga. A modified artificial bee colony algorithm for real-parameter optimization. *Information Sciences*, In Press, 2011.

[2] Daniel Brelaz. New methods to color the vertices of a graph. *Commun. ACM*, 22:251–256, April 1979.

[3] Coen Bron and Joep Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM*, 16:575–577, September 1973.

[4] B. Cases, C. Hernandez, M. Graña, and A. D'anjou. On the ability of swarms to compute the 3-coloring of graphs. In S. Bullock, J. Noble, R. Watson, and M. A. Bedau, editors, *Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*, pages 102–109. MIT Press, Cambridge, MA, 2008.

[5] V. Chvatal. Coloring the queen graphs, 2004. http://www.cs.concordia.ca/ chvatal/queengraphs.html, Web repository (last visited July 2005).

[6] Derek G. Corneil and Bruce Graham. An algorithm for determining the chromatic number of a graph. *SIAM J. Comput.*, 2(4):311–318, 1973.

[7] R. D. Dutton and R. C. Brigham. A new graph colouring algorithm. *The Computer Journal*, 24(1):85–86, 1981.

[8] Gianluigi Folino, Agostino Forestiero, and Giandomenico Spezzano. An adaptive flocking algorithm for performing approximate clustering. *Information Sciences*, 179(18):3059 – 3078, 2009.

[9] Philippe Galinier and Alain Hertz. A survey of local search methods for graph coloring. *Comput. Oper. Res.*, 33(9):2547–2562, 2006.

[10] Fangzhen Ge, Zhen Wei, Yiming Tian, and Zhenjin Huang. Chaotic ant swarm for graph coloring. In *Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on*, volume 1, pages 512 –516, 2010.

[11] M. Graña, B. Cases, C. Hernandez, and A. D'Anjou. Further results on swarms solving graph coloring. In D. Taniar et al., editor, *ICCSA 2010 Part III*, number 6018 in LNCS, pages 541–551. Springer, 2010.

[12] Julia Handl and Bernd Meyer. Ant-based and swarm-based clustering. *Swarm Intelligence*, 1:95–113, 2007.

[13] Francine Herrmann and Alain Hertz. Finding the chromatic number by means of critical graphs. *J. Exp. Algorithmics*, 7:10, December 2002.

[14] Ling-Yuan Hsu, Shi-Jinn Horng, Pingzhi Fan, Muhammad Khurram Khan, Yuh-Rau Wang, Ray-Shine Run, Jui-Lin Lai, and Rong-Jian Chen. Mtpso algorithm for solving planar graph coloring problem. *Expert Syst. Appl.*, 38:5525–5531, May 2011.

[15] 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.

[16] David S. Johnson and Michael A. Trick. *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, volume 26. American Mathematical Society, 1993.

[17] D.S. Johnson and M.A. Trick, editors. *Proceedings of the 2nd DIMACS Implementation Challenge*, volume 26. American Mathematical Society, 1996. DIMACS Series in Discrete Mathematics and Theoretical Computer Science.

[18] A. Mehrotra and M. Trick. A column generation approach for graph coloring. *INFORMS Journal On Computing*, 8(4):344–354, 1996.

[19] Kazunori Mizuno and Seiichi Nishihara. Constructive generation of very hard 3-colorability instances. *Discrete Appl. Math.*, 156(2):218–229, 2008.

[20] Jan Mycielski. Sur le coloureage des graphes. *Colloquium Mathematicum*, 3:161–162, 1955.

[21] Andreas Nolte and Rainer Schrader. Simulated annealing and graph colouring. *Comb. Probab. Comput.*, 10:29–40, January 2001.

[22] Israel Rebollo, Manuel Gra?a, and Carmen Hernandez. Aplicacion de algoritmos estocosticos de optimizacion al problema de la disposicion de objetos no-convexos. *Revista Investigacion Operacional*, 22(2):184–191, 2001.

[23] Craig Reynolds. Steering behaviors for autonomous characters, 1999.

[24] Craig W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. In *Computer Graphics*, pages 25–34, 1987.

[25] Shyam Sundar and Alok Singh. A swarm intelligence approach to the quadratic minimum spanning tree problem. *Information Sciences*, 180(17):3182 – 3191, 2010. Including Special Section on Virtual Agent and Organization Modeling: Theory and Applications.

[26] Jonathan S. Turner. Almost all k-colorable graphs are easy to color. *Journal of Algorithms*, 9(1):63 – 82, 1988.