

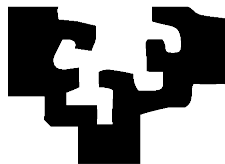
Gravitational Swarm for Graph Coloring

By

Israel Carlos Rebollo Ruiz

<http://www.ehu.es/ccwintco>

Dissertation presented to the Department of Computer Science
and Artificial Intelligence in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy



PhD Advisor:

Prof. Manuel Graña Romay

At

The University of the Basque Country

Donostia - San Sebastian

2012

**AUTORIZACION DEL/LA DIRECTOR/A DE TESIS
PARA SU PRESENTACION**

Dr/a. _____ con N.I.F. _____

como Director/a de la Tesis Doctoral: _____

realizada en el Departamento _____

por el Doctorando Don/ña. _____,

autorizo la presentación de la citada Tesis Doctoral, dado que reúne las condiciones necesarias para su defensa.

En _____ a _____ de _____ de _____

EL/LA DIRECTOR/A DE LA TESIS

Fdo.: _____

CONFORMIDAD DEL DEPARTAMENTO

El Consejo del Departamento de _____

en reunión celebrada el día ____ de _____ de ____ ha acordado dar la conformidad a la admisión a trámite de presentación de la Tesis Doctoral titulada: _____

dirigida por el/la Dr/a. _____

y presentada por Don/ña. _____
ante este Departamento.

En _____ a _____ de _____ de _____

Vº Bº DIRECTOR/A DEL DEPARTAMENTO SECRETARIO/A DEL DEPARTAMENTO

Fdo.: _____

Fdo.: _____

ACTA DE GRADO DE DOCTOR
ACTA DE DEFENSA DE TESIS DOCTORAL

DOCTORANDO DON/ÑA. _____

TITULO DE LA TESIS: _____

El Tribunal designado por la Subcomisión de Doctorado de la UPV/EHU para calificar la Tesis Doctoral arriba indicada y reunido en el día de la fecha, una vez efectuada la defensa por el doctorando y contestadas las objeciones y/o sugerencias que se le han formulado, ha otorgado por _____ la calificación de:
unanimidad ó mayoría

En _____ a _____ de _____ de _____

EL/LA PRESIDENTE/A,

EL/LA SECRETARIO/A,

Fdo.:

Fdo.:

Dr/a: _____

Dr/a: _____

VOCAL 1º,

VOCAL 2º,

VOCAL 3º,

Fdo.:

Fdo.:

Fdo.:

Dr/a: _____ Dr/a: _____ Dr/a: _____

EL/LA DOCTORANDO/A,

Fdo.: __

Acknowledgments

I would like to thank my advisor, Prof. Manuel Graña, who caught me some years ago and led me to this exciting research life. Colleagues at the GIC have been very supporting, specially Carmen Hernandez and Blanca Cases which started work on the application of Reynolds' boids to the GCP.

I also would like to thank to my companies "Informatica 68 S.A." and "Informatica 68 Investigación y Desarrollo S.L." for the support and help received from them.

I would thank at last and not at least, to my parents Jesus and Africa for always being beside me and encouraging me to never give up. And finally to my wife Isabel, the light that guides my way.

Thanks so much to everybody!

Israel Carlos Rebollo Ruiz

Gravitational Swarm Intelligence for Graph coloring by

Israel Carlos Rebollo Ruiz

Submitted to the Department of Computer Science and Artificial Intelligence on June 21, 2012, in partial fulfillment of the requirements for the degree of Doctor of Philosophy

abstract

This Thesis deals with the development of a Swarm Intelligence algorithm to solve the classical problem of Graph Coloring. The Gravitational Swarm Intelligence (GSI) algorithm maps the GCP problem into a collection of autonomous agents that move in a space following a global gravitational attraction to the color goals and attraction-repulsion local forces corresponding to the graph topology. The Thesis provides formal asymptotic convergence proofs showing that the GSI stationary states correspond to GCP solutions. The Thesis provides also extensive empirical support of the GSI comparing it with state of the art algorithms.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Objectives	3
1.2.1	Fundamental objectives	3
1.2.2	Operational objectives	4
1.3	Contributions of the Thesis	4
1.3.1	Publications	5
1.4	Structure of the dissertation	6
1.5	Selected notation	7
2	State of the art	11
2.1	Graph Coloring Problem	11
2.1.1	Properties	12
2.2	Swarm Intelligence	13
2.2.1	Flocking behaviors	13
2.2.2	Ant Colonies	15
2.2.3	Particle Swarms	16
2.2.4	Gravitational swarm	17
2.3	Graph Coloring Algorithms	17
2.3.1	Classical algorithms	17
2.3.2	Approximate algorithms	19
2.3.2.1	Graph Families	21
2.3.3	Swarm Intelligence for Graph Coloring	23
2.4	GCP Applications	23
3	GCP Algorithms	25
3.1	Graph Coloring Problem methods	25

3.2	Backtracking (BT)	28
3.3	DSATUR	28
3.4	Tabu Search (TS)	31
3.5	Simulated Annealing (SA)	32
3.6	Ant Colony Optimization (ACO)	33
3.7	Particle Swarm Optimization (PSO)	34
3.8	Gravitational Swarm Intelligence (GSI)	35
4	Gravitational Swarm Intelligence	37
4.1	Gravitational Swarm for	38
4.2	Graph coloring problem	42
4.3	Gravitational Swarm	42
4.4	Gravitational Swarm for GCP	46
5	Parameter tuning	51
5.1	GSI model parameters	51
5.1.1	Chromatic number	53
5.2	Goal Radius	53
5.3	Comfort	56
5.4	Nom Parametric Tests	59
5.4.1	Friedman test	60
5.4.2	Friedman test to GSI	61
5.4.2.1	Goal Radius	61
5.4.2.2	Comfort	62
5.4.3	Post-Hoc test: Nemenyi's test	63
5.5	Concluding remarks	67
6	Graph Coloring Results	69
6.1	Experimental design	69
6.2	Trees and bipartite graphs	71
6.3	Kuratowski based planar graphs	72
6.4	Mizuno's 3-colorable	75
6.5	KRG graphs	79
6.6	DIMACS graphs	85
6.6.1	Test	92
6.7	Sequential chromatic number determination	94
6.7.1	Random Graphs	95
6.7.2	Leighton graphs	97

6.7.3	Real Graphs	97
6.8	Conclusions	99
7	Conclusions and further work	101
7.1	Future works	102
A	Graph experimental instances and graph generators	105
A.1	Introduction	105
A.2	Trees and bipartite graphs	106
A.2.1	Trees	106
A.2.2	Bipartite	106
A.3	DIMACS	106
A.3.1	Mycielski graphs	107
A.3.2	Queens Graphs	108
A.3.3	Miles graphs	108
A.3.4	Fullins graphs	109
A.3.5	Books graphs	109
A.3.6	Leighton Graphs	110
A.4	Random Graphs	111
A.5	Mizuno's Graphs	112
A.6	Planar Graphs	112
A.7	KRG graphs	114
A.8	Real Graphs	114
B	Graph Coloring Suite	117
	Bibliography	121

List of Figures

2.1	The Grötzsch Graph	14
2.2	Flocking Birds	15
2.3	Ants using pheromones to find the shortest path	16
2.4	An instance of a Graph Coloring Problem solution.	18
4.1	GSI agent behavior flowchart for GCP	41
4.2	Simplified Flowchart	43
5.1	Average success ratio vs Goal Radius	54
5.2	Average success ratio vs Goal Radius in 3D	55
5.3	Average Steps ratio vs Goal Radius	56
5.4	Average success ratio vs Comfort	57
5.5	Average success ratio vs Comfort in 3D	58
5.6	Average steps ratio vs Comfort	59
5.7	Nemenyi's diagram for 9 goal radius and 90% of acceptance . . .	64
5.8	Nemenyi's diagram for 9 goal radius and 95% of acceptance . . .	64
5.9	Nemenyi's diagram for 9 goal radius and 99% of acceptance . . .	64
5.10	Nemenyi's diagram for 5 goal radius and 95% of acceptance . . .	65
5.11	Nemenyi's diagram for 5 goal radius and 99% of acceptance . . .	65
5.12	Nemenyi's diagram for 6 comfort values and 95% of acceptance .	66
5.13	Nemenyi's diagram for 6 comfort values and 99% of acceptance .	66
5.14	Nemenyi's diagram for 5 comfort values and 90% of acceptance .	67
6.1	Trees and bipartite success ratio	71
6.2	Trees and bipartite average time in seconds	72
6.3	Trees and bipartite average number of steps	73
6.4	Kuratowski based graphs success ratio	74

6.5	Kuratowski based graphs average time in seconds	76
6.6	Kuratowski based graphs average number of steps	77
6.7	Mizuno's graphs success ratio	78
6.8	Mizuno's graphs average time in seconds	79
6.9	Mizuno's graphs average number of steps	80
6.10	KRG success ratio	82
6.11	KRG average time in seconds	83
6.12	KRG average number of steps	84
6.13	Big KRG success ratio	86
6.14	Big KRG average time in seconds	87
6.15	Big KRG Average number of steps	88
A.1	Mycielski graph transition from M_3 to M_4	108
A.2	8 queens problem solution	109
A.3	United States Cites Graph	110
A.4	Books where the Books Graphs come from	111
A.5	Mizuno's MUGs for 3-colorable graphs generation [1]	113
A.6	Aleatory graphs, Kuratowski's planar graphs, Mizuno's 3-colorable graphs and KRG new developed graph type generator.	115
B.1	Graph Coloring Suite	118
B.2	Snapshot of a result file	119

List of Tables

5.1	Average results of KRG graph of 30 nodes, 50 edges and 6 colors	52
5.2	Friedman ranking for Goal Radius	61
5.3	Friedman ranking for Comfort	61
6.1	Layout of the experimental graphs	90
6.2	Results of SGB graphs	91
6.3	Results of CAR graphs	93
6.4	Algorithm Rankings for Friedman Test	93
6.5	Graphs of 100 nodes and 1000 edges. Between parentheses the minimum solution found.	95
6.6	Graphs of 100 nodes and 2000 edges. Between parentheses the minimum solution found.	96
6.7	Graphs of 100 nodes and 3000 edges. Between parentheses the minimum solution found.	96
6.8	Graphs of 100 nodes and 4000 edges. Between parentheses the minimum solution found.	96
6.9	Leighton Graphs results	98
6.10	Exam timetabling of Lewis [2] plus GSI	98

Chapter 1

Introduction

This introductory chapter provides the motivation for this Thesis in Section 1.1. Section 1.2 enumerates the objectives of the work. Section 1.3 highlights the contributions achieved by this Thesis, including the relevant publications in Section 1.3.1. Section 1.4 comments the structure of this dissertation. Finally, Section 1.5 summarizes notation used in the dissertation.

1.1 Motivation

The works of the candidate towards this thesis have followed some meanderings until reaching an stationary focus in the topics covered in this memory. Works have covered some combinatorial optimization problems, as well as RFID applications in industry. The actual topic of the thesis comes from the initial works by members of the research group towards the application of Reynolds' boids to a combinatorial optimization problem. Soon it was realized that no general abstract application was possible, so that a specific problem was to be attacked. The Graph Coloring Problem (GCP) was selected by its long history, current approaches and potential applications. One basic approach was to map problem solutions to boids, so that the boid dynamics would provide some solution, much like the Particle Swarm Optimization approach. However, a different view, that of mapping graph nodes to boids and study the aggregation and separation dynamics of boids in order to obtain some information on the problem solution, proved to be fruitful. The basic problem of defining the color compatibility problem was modeled as some kind of attraction and repulsion between boids.

Initial works in the group were addressed to show that the approach effectively provided some kind of solution to the GCP. The main questions then were:

- It is possible to map the GCP to the state of some kind of boid swarm system?
- It is possible to define some dynamics that lead the boids to reach global configurations corresponding to feasible solutions of the GCP?
- It is possible to obtain some optimal solution, thus determining the chromatic number?

The published results showed that including some attraction/repulsion terms in the boids equations depending on the adjacency of the corresponding nodes in the underlying graph it was possible to approach coloring solutions. Improvements were introduced by the definition of specific positions in space corresponding to color goals. The works of this thesis started in the participation in computational and formal studies directed to assess the role of the boid perceptual field in system's convergence. Some percolation results were used as background trying to establish some bounds. However the results were inconclusive and counterintuitive. Large perceptual fields lead to poor convergence, and the theoretical results had poor correspondence with the empirical results obtained from extensive simulations.

At this point the thesis has a paradigm shift introducing the gravitational metaphor, which allows for local interactions corresponding to graph adjacency and global attraction corresponding to color goal seeking. Pursuing this has proven fruitful in theoretical and (computational) empirical results. We have been able to prove some important results, namely that the system will always converge to a solution if it is feasible (the hypothesis on the number of colors is not lower than the graph's chromatic number). An open question is to ensure that the system always converges to a stationary state, i.e. that there are no limit cycles or chaotic (bounded random) behaviors.

Why swarm approaches?: Swarm approaches may lead to problem solving methods that are knowledge distributed in the sense that the knowledge of the actual state of the solution is distributed over the collection of individuals. No single individual possess the knowledge of a complete solution. Observation of the system as a whole gives the sought answer. It is expected that such kind of approaches would give benefits in terms of

- computational economy: the computational cost is distributed among the agents, no single agent bears the cost of computing the complete solution of the problem
- robustness against failure and noise: the global system will perform even if some agents fail or have uncertain/noisy information
- adaptation to non-stationary environments: solutions emerge as a result of a collective unsupervised interaction, therefore, the system may adapt to changing circumstances in an unsupervised way, no need for a master to activate the adaptation mechanism.

Why the GCP?: The GCP is a classical combinatorial problem, extensively studied, easy to understand and to implement competitive approaches for validation/evaluation purposes. Therefore is a magnificent test ground for innovative computational approaches. Mapping other problems into the GCP may provide practical solutions to other (real-life) problems.

1.2 Objectives

The main objective of the Thesis is the development of innovative nature inspired algorithms for the approximate solution of combinatorial problems, specifically the Graph Coloring Problem (GCP).

1.2.1 Fundamental objectives

Fundamental objectives are the main driving research questions that our research tries to answer.

- It is possible to map a combinatorial problem into a swarm, so that its dynamics provide a solution to the posed problem? Specifically, can the GCP be mapped into such a system?
- The swarm can evolve to a feasible solution even if all the swarm members are ignorant of the problem that is being solved? In other words, can the problem solution be posed as an emergent collective behavior?
- It is possible to give a formal proof of the convergence of the system to such kinds of solutions?

- The proposed method is competitive regarding the current state-of-the-art?
- How sensitive is the proposed method to the setting of its computational parameters?

1.2.2 Operational objectives

In order to attain the stated fundamental objectives, we need to develop some instruments:

- Creation of a collection of benchmark graphs for the replicability of the computational experiments. Such collection must show some specific features that are important for the evaluation of the algorithms
- Implementation of the competing algorithms. Most algorithms reported in the literature have no public implementations provided by the authors.
- Defining methodological steps for sound comparison of algorithms.
- Managing, analysing and plotting the big quantities of results obtained from the computational experimentes. Performing the maintenance of the experiment execution which can span several days.
- Performing the review of the state of the art, searching for competing algorithms and reference results.

1.3 Contributions of the Thesis

According to the objectives set in the previous section, the achievements of the Thesis reported in this dissertation, are summarized as follows:

1. A new algorithm for Graph Coloring Problem.
2. An application of the nature inspired Swarm Intelligence to a mathematical problem.
3. Add the Gravitational theory of Newton the the Swarm approach.
4. A new Graph class called KRG that generates graphs with a known chromatic number.

5. A Graph coloring suite that includes seven algorithm implementation, including our Gravitational Swarm Intelligence algorithm and also a toolbox for generating graphs:
 - (a) Aleatory Graphs
 - (b) Hard 3-coloreable Graphs
 - (c) Planar graphs
 - (d) KRG non planar graphs
6. A demonstration of the convergence of the algorithm.

1.3.1 Publications

Journal publications:

- Israel Rebollo, Manuel Graña, Carmen Hernández, "Aplicación de algoritmos estocásticos de optimización al problema de la disposición de objetos no-convexos" in *Investigación Operacional* editada por la Dirección de Información Científico Técnica de la universidad de La Habana, volumen 22, número 2 de 2001. pags 184-192.
- Israel Rebollo, Manuel Graña, Blanca Cases, "On the effect of spatial percolation on the convergence of Graph Coloring Boid Swarm" in *International Journal on Artificial Intelligence Tools*, DOI No: 10.1142/S0218213012500157 Accepted 2012-01-23.
- Blanca Cases, Israel Rebollo, Manuel Graña, "A Spatial-social-logical model explaining human behavior in emergency situations" in *Logic Journal of the IGPL (2011)* (published on line) DOI No: 10.1093/jigpal/jzr006.
- Manuel Graña, Israel Rebollo, "Gravitational Swarm finds Graph Colorings", 2012, submitted.

Conference publications:

- Israel Rebollo, Manuel Graña, "An empirical comparison of some approximate methods for Graph Coloring", in *7th International Conference Hybrid Artificial Intelligent Systems*, Part 2, pp. 600-609. ISBN 978-3-642-28930-9

- Israel Rebollo, Manuel Graña. "Gravitational Swarm Approach for Graph Coloring" in *Studies in Computational Intelligence*, 2011, Volume 387, 159-168, DOI: 10.1007/978-3-642-24094-2 D.A. Pelta, N. Krasnogor, D. Dumitrescu, Camelia Chira and R. Lung (eds) Publisher: Springer-Verlag Berlin / Heidelberg ISBN 978-3-642-24093-5
- Israel Rebollo, Manuel Graña, "Further results of Gravitational Swarm Intelligence for Graph Coloring" in *Nature and Biologically Inspired Computing (NaBIC), 2011 Third World Congress on*, pp. 183 - 188. DOI No: 10.1109/NaBIC.2011.6089456 ISBN 978-1-4577-1122-0.
- Israel Rebollo, Manuel Graña, "Dynamic Tabu Search for Non Stationary Social Network identification based on Graph Coloring" in *7th International Conference on Soft Computing Models in Industrial and Environmental Applications*, (submitted)

1.4 Structure of the dissertation

The PhD dissertation report has the following structure.

Chapter 2 contains a detailed description of the state of the art for the Graph Coloring Problem and the Reynolds' boids approach to Swarm Intelligence.

Chapter 3 show the algorithms for GCP implemented in the suite except ours that will be more exhaustive explained in the next section. This implementation are not exactly the same that appear in the literature so we need to explain the special features.

Chapter 4 provides a deep description of our Gravitational Swarm Intelligence algorithm, with the flowchart that describes the model. In this section we show formally the convergence of our algorithm to a stable state.

Chapter 5 discusses the parameter that need the model. If there are necessary or not, and we use non parametrical test to extract the best parameter for testing our new method.

Chapter 6 discusses the computational experiment results obtained over all the graph families shown in chapter 5. We plot the accuracy of our method against the other implemented methods and the result that appear in the bibliography. We also plot the computational time in steps and seconds.

Chapter 7 gives the conclusions of the Thesis and some ideas for future research.

Appendix A A describes the graph instances used for test. We also describe the graph generator program embeded in the suite of coloring. The new graph class KRG that has been developed in this thesis is explained.

1.5 Selected notation

V Set of nodes of a graph

$E \subseteq V \times V$ Set of edges of a graph

$G = \{V, E\}$ Graph with V nodes and E edges

B A group of agents $B = \{b_1, \dots, b_n\}$

\vec{v}_i The speed vector of the agent i

$p_i(t) = (x_i, y_i)$ The position in Cartesian coordinates x, y of the agent i in the instant t

k A color

C A group of colors $C = \{1, \dots, k\}$

CG The colors of each goal $CG = \{g_1, \dots, g_K\}$

nearenough The goal radius, inside which the agents get the goal color

$\mathcal{N}(g_k)$ The neighborhood in the goal k , the number of agents inside the goal influence

enemy A pair of agents that have a link and the same color

$R(b_i, g_k)$ Repulsive forced exerted between the agent b_i and it's enemies in the neighborhood of the goal g_k

$\{\overrightarrow{a_{i,k}}\}$	Are the attraction forces of the color goals exerted on the agents
d	The Euclidean distance between the position of an agent and the nearest color goal
<i>Comfort</i>	is the number of cycles an agent has a color without having enemies in its neighborhood
λ	Represent the probability of an agent to be expelled from a goal
<i>maxcomfort</i>	Is the maximum comfort that an agent can reach
M	The chromatic number of a graph
S	The search space
μ_i	Is the charge of the agents, represent repulsion for linked nodes
δ_{ij}^A	Is the attraction force between the objects i and j
δ_{ij}^R	Is the repulsion force between the objects i and j
θ_A	A threshold where the attraction force has effect
θ_R	A threshold where the repulsion force has effect

Chapter 2

State of the art

This chapter provides some background information on the Graph Coloring Problem (GCP) and the Swarm Intelligence (SI) approach. We give formal definitions of the GCP to be solved and a intuitive description of the essential SI. The chapter contains a review of current approaches to solve the GCP, including some SI methods.

The content of the chapter is as follows: Section 2.1 introduces the GCP. Section 2.2 introduces the Swarm Intelligence direct precedent to the algorithm proposed in this thesis. Section 2.3 provides an state of the art of current approaches to solve GCP. Section 2.4 comments on some applications of GCP to real life problems found in the literature.

2.1 Graph Coloring Problem

An undirected graph is a collection of nodes linked by edges $G = (V, E)$, such that $V = \{v_1, \dots, v_N\}$ and $E \subseteq V \times V$, and $(v, w) \in E \Rightarrow (w, v) \in E$. The neighborhood of a node in the graph is the set of nodes linked to it: $\mathcal{N}(v) = \{w \in V \mid (v, w) \in E\}$.

The Graph Coloring Problem (GCP), aka Graph Labeling and Node Coloring in graph theory, is an assignment of colors to nodes of a graph subject to certain constraints:

1. Two adjacent nodes connected by an edge don't share the same color.
2. The number of colors is greater than the chromatic number $\chi(G)$ which is the minimum number of colors that can be used to color the graph.

Similarly, an edge coloring assigns a color to each edge so that no two adjacent edges¹ share the same color, and a face coloring of a planar graph assigns a color to each face or region so that no two faces that share a boundary have the same color. But all these problems can be summarized in the node coloring. The coloring of a graph can have multiple solutions.

Definition 1. Graph coloring. Let $C = \{c_1, \dots, c_M\}$ denote a set of colors. Given a graph $G = (V, E)$, a graph coloring is a mapping of graph nodes to colors $\mathcal{C} : V \rightarrow C$ such that no two neighboring nodes have the same color, i.e. $w \in \mathcal{N}(v) \Rightarrow \mathcal{C}(v) \neq \mathcal{C}(w)$.

Definition 2. Minimal graph coloring. A set of colors C^* is minimal relative to graph $G = (V, E)$ if (1) there is a graph coloring $\mathcal{C}^* : V \rightarrow C^*$, and (2) for any smaller set of colors there is no graph coloring using it: $|C| < |C^*| \Rightarrow \neg \exists \mathcal{C} : V \rightarrow C$. Alternative definition: any graph coloring on this graph has a greater or equal set of colors $\mathcal{C} : V \rightarrow C \Rightarrow |C| \geq |C^*|$.

Definition 3. Chromatic number: The chromatic number $\chi(G)$ is the number of colors of the minimal graph coloring C^* .

Definition 4. Chromatic polynomial is the number of possible coloring solutions using a given number of colors. $P(G, k)$ being G the graph and k the number of colors.

Definition 5. Edge Coloring is a proper coloring of the edges, meaning an assignment of colors to edges so that no node is incident to two edges of the same color. Changing edges by nodes and nodes by edges the problem can be transformed into node coloring.

Definition 6. Total Coloring is coloring edges and nodes at the same time, keeping the constraints of the GCP. Two adjacent nodes can't have the same color but neither can an edge and an end-node of the edge.

2.1.1 Properties

We present some properties related to the GCP.

Definition 7. A complete graph is a graph that has edges between all its nodes.

¹Two edges are adjacent if they share one end node.

Remark 8. The bounds on the chromatic number are $1 \leq \chi(G) \leq n$. All nodes can be colored with a single color, i.e. $\chi(G) = 1$ only if the graph has no edges. If the graph is complete then we need a different color for each edge $\chi(G) = n$.

Definition 9. A Clique is a sub-graph of a graph that is complete.

Remark 10. The chromatic number of a graph is at least equal to the size of the graph's biggest clique.

Remark 11. Graphs with large cliques will have high chromatic number, but the opposite is not true.

Theorem 12. *Mycielski:* There are triangle-free graphs with arbitrarily high chromatic number.

The Mycielski graphs are constructed following a precise procedure giving a constructive proof of this theorem. The Grötzsch graph shown in figure 2.1 is the Mycielski M4 graph. We have used this kind of graphs for testing the GCP solving algorithms because the chromatic number is known and it can be made as large as needed.

2.2 Swarm Intelligence

Swarm Intelligence (SI) is the emergence of meaningful configurations from the collective behavior of decentralized and self organized systems, whose dynamics are inspired in the nature. Therefore, SI proposes a distributed computational model to solve combinatorial problems by multi-agent systems. A definition extracted from [3] reads "SI is a model where the emergent collective behavior is the outcome of a process of self-organization, in which agents are engaged through their repeated actions and interaction with their evolving environment".

2.2.1 Flocking behaviors

The inspiration in flocking birds appears in [4], where we first find the application of the behavior of a group of animals, in this case birds, to the solution of mathematical problems. The application of flocking behavior appears in [5, 6] to guide a flock of self organized mobile robots. El-abd [7] created a Particle Swarm Optimization (that we explain latter) based in flocking behavior. We will recall the

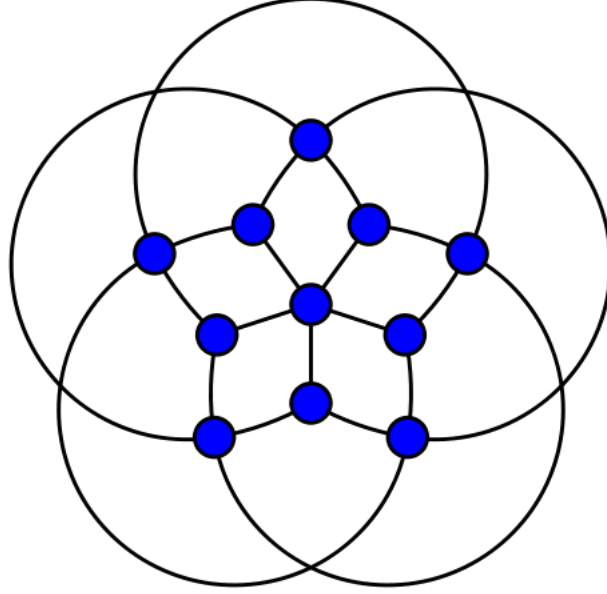


Figure 2.1: The Grötzsch Graph

Reynolds studied the behavior of the birds and define three rules, that can be translate into mathematical formulas, and then applied to mathematical problems. The formalization of the problem starts with a group of bird $B = \{b_1, \dots, b_n\}$ placed in the position p_i . Let define ∂_i as the group of birds in the neighborhood of radius z of the bird b_i . Each bird moves through the space with a speed v_i . The Reynolds rules for the behavior of flocking birds are:

1. Separation: avoid crowding neighbors (short range repulsion). Steer to avoid crowding local flock-mates inside a private zone of radius z .

$$v_{s_i} = - \sum_{b_j \in \partial_i: d(b_j, b_i) < z} (p_j - p_i) \quad (2.1)$$

2. Alignment: steer in the direction of the average heading of local flock-mates.

$$v_{a_i} = \frac{1}{|\partial_i|} \sum_{b_j \in \partial_i} v_j - v_i \quad (2.2)$$

3. Cohesion: steer to move toward the average position of local flock-mates



Figure 2.2: Flocking Birds

(long range attraction).

$$v_{c_i} = c_i - p_i \text{ where } c_i = \frac{1}{|\partial_i|} \sum_{b_j \in \partial_i} p_j \quad (2.3)$$

With these three rules we can compute the movement speed of each bird at the time $t + 1$ like

$$v(t + 1) = \text{fmaxN}(\alpha_0 v(t) + \alpha_s v_s(t) + \alpha_a v_a(t) + \alpha_c v_c(t) + \alpha_n v_n). \quad (2.4)$$

Where the α_x are modification parameters. The v_n is noise and fmaxN is a normalization value.

2.2.2 Ant Colonies

Besides birds, other living creatures have been considered for computational inspiration. Works like [8, 9] have found inspiration in the ant colonies. Briefly, ants move from the ant colony towards food sources leaving a trail of pheromones behind them. Using this pheromone trail, ants manage to find the optimal path between the ant colony and the food source. A more detailed explanation of the trails of the ants is shown in [10]. The first problem where the ant colony optimization (ACO) has been shown to provide feasible solutions was the classical Travel Salesman Problem (TSP) [9]. There are a lot of versions of ACO solving this problem, and ensuing applications, i.e. tracing the route of a vehicle [11]. Balaprakash [12] presents an ACO probabilistic TSP.

ACO can solve more problems, not only the TSP. Franks in [13] show ants deciding how to get to moving targets. The ligand of proteins using the PLANTS

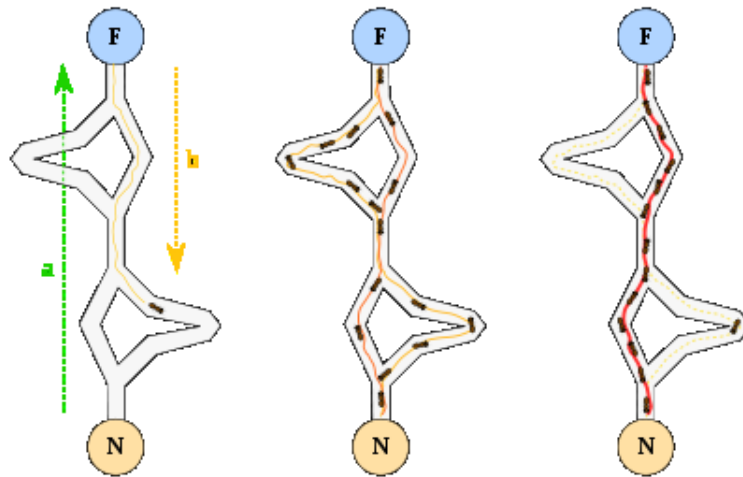


Figure 2.3: Ants using pheromones to find the shortest path

(Protein Ligand ANT System) algorithm of Korb [14]. More about robots with ants in [15]. Ant clustering with locally weighted where the ants has some memory in Peterson [16]. A reinforcement learning algorithm based algorithm can be found in [17]. More about trail formation and TSP in Shah [18]. The list is endless, and continue growing as this nature inspired approach remains in fashion nowadays.

2.2.3 Particle Swarms

The other big area of research in the SI field is the Particle Swarm Optimization (PSO), with increasing applications and results. Here the agents are particles moving in the solution space searching for the optimal solution to the problem. But agents have a particular feature: memory. This is the main difference with other forms of SI. The PSO particles know the global objective function of the system, keeping in their memory the best global solution found by the Swarm and also their own best local solution found so far. Particles move in the surroundings of their global and local best position found. This way the systems moves towards the global optimum, but this heuristic doesn't guarantee finding the best solution. Moreover, this algorithm can solve ill-posed problems, noisy and changing along time, even those modeled discontinuous functions because

PSO does not use the objective function's gradient.

Convergence of the standard PSO algorithm is proven in [19], additional convergence is discussed in [20]. The essential PSO is introduced in [21], but there are a lot of variants with specific convergence properties, such as [22], or the ensemble PSO [23]. Applications are not restricted to combinatorial problems, for instance [24] applies PSO in steganographic JPEG images .

2.2.4 Gravitational swarm

The Swarm computing natural inspiration is not necessarily coming from living beings. Rashedi [25] present a work where the nature inspiration comes from the gravitational Law of Newton. In this work, the algorithm is built using the gravitational law in a very strict way, using masses, velocities and distances. We have also been inspired in the gravitational theory in our main contribution, but without going into the detail of the physical laws of the real world. We have assumed that the masses are not relevant. That only the goals has an attraction force. And the speed of the agent is inversely proportional to the distance to the goals. Near the goals the attraction disappear, breaking Newton's laws.

2.3 Graph Coloring Algorithms

We give a brief survey of different GCP solving algorithms, from classical algorithm up to recent Swarm Intelligence approaches. There are a lot of reference books like [26] where we can find information about this problem, and more about graph theory. In figure 2.4 we can see an example of a colored graph.

We are going to introduce the most famous algorithms starting from classical algorithm, recalling some relevant theorems from graph coloring theory. We are going also to present some graph families with special features, that will help to better understand the different approaches for this problem. Then we are going to show algorithm based in SI approach, although we are going to explain the SI more detailed. Finally we are going to show that that GCP is applied in real life with a bunch of application using the GCP.

2.3.1 Classical algorithms

The GCP is a classical problem in mathematics. A well known instance of the problem appears when trying to distinguish states or regions in England

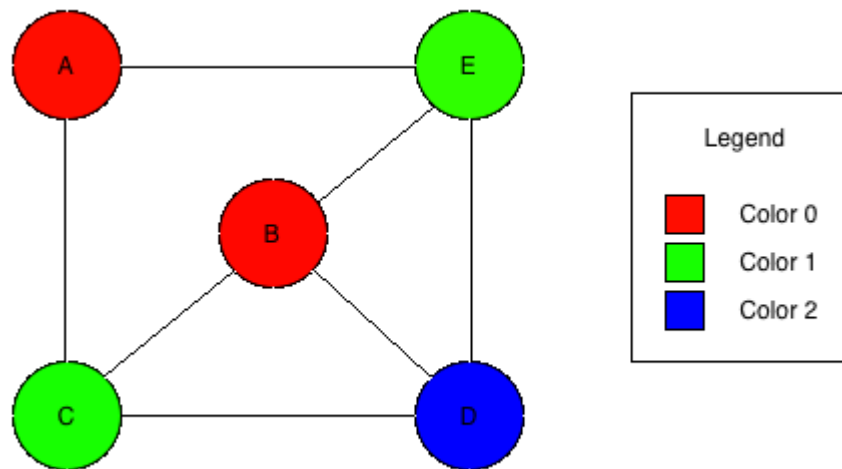


Figure 2.4: An instance of a Graph Coloring Problem solution.

by colors in political maps, thus the Francis Guthrie four color conjecture in the 17th century. At the time, all the graphs considered were planar graphs representing pieces of land, and there wasn't a restriction of the number of colors to use, even though four was enough. But in the 20th century, more complex instances of GCP were considered. We are going to lay aside the history and go to the mathematical problem. In the year 1949 the Russian scientific Alexander Zykov stated a theorem which gives the bases for a lot of GCP solving algorithm. Specifically, the contraction algorithm is based on Zykov's theorem of contraction [27], aims to reduce the complexity of a graph reducing the computational time need to solve it.

Theorem 13. *Zykov: $x(G) = \min \{x(G/x, y), x(G + xy)\}$ for non adjacent nodes x and y .*

We can see how this algorithm works in recent Odaira's paper [28]. Before, Corneil in [29] developed an algorithm that searches through the Zykov tree in a depth-first manner. Dutton [30] searches non-adjacent nodes with a maximal number of common neighbors to contracted until a complete graph is achieved. The algorithm Recursive-Largest-First (RLF) of Leighton is presented in [31] where the contraction affects the largest path between nodes, Palubeckis present a more actual version of this algorithm in [32]. A modified LF-algorithm

(Largest First) presented by Hansen in [33] is an optimization of RLF where the recursive technique is changed to a sequential way.

The most famous algorithm was developed by Brelaz [34] in 1979. This algorithm was called DSATUR because it is based in the degree of saturation of a graph. It is an approximate algorithm, that does not perform an exhaustive search. We will explain it in detail in chapter 3. Although, this algorithm sometimes fails finding the optimum [35], it is still a reference algorithm. Turner [36] said that almost all k -colorable graphs are easy to color with his heuristic, and also proposed a new implementation of Brelaz algorithm to enhance its draw falls. Wood in 1997 [37] and more recently Mendez-Diaz [38] presented another optimized version of the DSATUR improving that of Turned.

2.3.2 Approximate algorithms

Sometimes the computational effort to find the exact minimal coloring of a graph is too huge. So that there are some algorithms that don't report the exact solution to the problem, as it has been said in zykov's based algorithms. These algorithms basically give a upper bound on the chromatic number providing a quick solution to the problem. Then, other techniques must be applied to find the exact solution of the problem. The theorems of Vizing [39] and Shanon [40] are used to achieve this.

Theorem 14. *Vizings: In a graph or multigraph G , let denote $\Gamma(v)$ the valency of node V , and let denote $\Gamma(G)$ the largest valency in G . Let the multiplicity $\mu(v, w)$ of nodes V and W be the number of parallel edges that link them. Let $\mu(G)$ be the largest multiplicity in G . A graph is a multigraph for which $\mu(G) = 1$. An edge coloring of a (multi)graph G is a mapping from the set of its edges E to a set of items K called colors, in such a way that at any node V , the $\Gamma(v)$ edges there all have a different color. An edge- κ -coloring is an edge-coloring where $|K| = \kappa$. The chromatic index $\chi(G)$ is the smallest number κ for which an edge- κ -coloring of exists.*

Theorem 15. *Shanon: For any multigraph G , $\chi(G) \leq \lfloor \frac{3}{2}\Gamma(G) \rfloor$.*

The COSINE algorithm by Hertz [41] and the Clique Covers (CC) algorithm of Klotz [42] are two examples of this kind of approaches. The authors of COSINE have published an updated work in [43]. The COSINE algorithm first tries to find an upped bound of the chromatic number in a quick manner, and second uses a more sophisticated coloring procedure based on Tabu search

techniques. This algorithm gets a good relation between accuracy and time. The CC uses the Vizing and Shanon theorems to find an upper bound of the chromatic number and eventually find it.

The Backtracking Sequential Coloring was discovered by Brown [44], whose work was improved by Brelaz to make his DSATUR, Wilf [45] showed that if the number of color approaches infinity the order of Backtracking is $O(1)$, and Park [46] presented a new version and an application to network security.

The simulated annealing has been applied to GCP recently by Titiloye [47] using the Monte-Carlo path-integral. Simulated annealing method has been also used by Johnson [48] showing empirical result to the GCP. Nolte [49] focus the use of Simulated Annealing to 3-colouring problem. Bonomo [50] make a mapping of the Bounded coloring problem to the classical Travel Salesman Problem TSP and solve it using the simulated annealing.. Other probabilistic methods have been used such as hill-climbing used by Rhyd [51] for order independent minimum grouping problems, and applied to GCP.

Tabu search algorithm has been widely used for GCP solving. The Tabucol algorithm presented by Galinier and Hertz [52] proposed in 1987 that today is still present in a lot of evolutionary and hybrid algorithm by it's performance in local search. Blochliger [53] used a partial solutions method based in a Tabu Scheme for local search. The hybridization mention before can be seen in Mabrouk [54] presenting a parallel Genetic-Tabu algorithm for the GCP. Qu [55] present a hyper-heuristic based in heuristic for coloring graphs, where the Tabu Search is also presented.

It is possible to built optimized heuristics for graphs with special features. Nakayama [56] proposed an heuristic for interval graphs, permutation graphs and trapezoid graphs, the interval graph is also presented in Yu [57] who develops a parallel algorithm. Gaun [58] works over weighted graphs a particular problem that needs different approaches to solve it like the first-fit algorithm. Vredeveld [59] also works over weighted graphs focused in local search. Bouchard [60] is interested in a mix of interval graphs as we have seen in Yu and Nakayama and weighted graphs studied by Gaun and Vredeveld, but solved this more complex problem. (p,k) -coloring problem is studied by Demange [61] that generalizes the GCP by replacing stable set by cliques and stable sets. Complexity of two coloring problems in cubic planar bipartite mixed graphs presented by Ries [62]. The P_4 graphs where a P_4 is an induced path with four nodes and P_4 is any P_4 -free graph is studied by Campos [63]. Daniel [64] works in chordal graphs where a graph is chordal if each of its cycles of four or more nodes has a chord, which is

an edge joining two nodes that are not adjacent in the cycle. Confessore worked previously in more explicit chordal graphs [65]. The Paley graph P_q which is the graph with nodes the elements of the finite field F_q and an edge between x and y if and only if $x-y$ is a non-zero square in F_q is solved by Maistrelli in [66]. Yadav reduced the problem to acyclic node coloring of graphs of maximum degree 5[67]. Galinier and Hertz [68] present a more general algorithm for GCP using a memory based algorithm different from their famous Tabucol algorithm. Costa [69], that we will see later, here study graphs with cardinality constraints on the neighborhoods, other variant in the GCP problem.

The evolutionary strategy have been use in a large number of works. Marino [70] made a mapping into a graphs and then solve the GCP to show a theoretical framework to break the symmetry of the search space in a partitioning problem using a Genetic Algorithm. Shen [71] present a Genetic Algorithm for GCP. Biman develop a genetic algorithm with a new operator called Multipoint Guided Mutation and Biman's work [72] is a brief example. Porumbel [73] present a search space analysis for improving local search is GCP and solve it with a Genetic algorithm that appears in [74]. A parallel technique in genetic algorithm is also common like Sivanandam [75] who present a hybrid parallel genetic algorithm approach, Kokosiski [76] present a parallel genetic approach for the sum coloring problem which asks to find a node coloring of a given graph G , using natural numbers, such that the total sum of the colors is minimized and Yu [77] who applies the Parallel genetic approach in VLSI Channel Routing. Finally there are hybridization of Genetic Algorithm with other methods like artificial neural networks ANN presented by Maitra[78, 79].

There are different strategies like Abasian's[80] that uses a non-systematic method based on a cultural algorithm to solve the GCP. Dukanovic[81] indicates a way to find the lower bound of a chromatic number. Mehrotra [82] propose a column generation method for implicit optimization of the linear program at each node of the branch-and-bound tree to solve the GCP. The fuzzy approach can't be missed in the survey and Gomez[83] present a algorithm for fuzzy graphs.

2.3.2.1 Graph Families

Not all the graphs have the same complexity to be solved. Some times we can observe special features that helps in the process of finding the chromatic number, as we have saw in the previous section. The was a challenge in 1993,

the second DIMACS challenge [84, 85] where a lot of graph were shown to test them. The researches were focused in these particular graphs because it becomes very easy to compare with the rest of the scientific community. More detailed explanations are given in Appendix A.

A big problem is to know *a priori* the chromatic number of a graph. Some theorems can give precious information in this regard, and also can be used to build graphs in a way that fill the requirements of the theorems and be more easy to solve. A example is the planar graphs, that are 4 colorable [86, 87]. In Appendix A we recall the Kuratowski's theorem. Luzar [88] works over planar graphs. More complex planar graphs appears in Werra [89].

A famous family of graphs is Mycielski graphs [90], already mentioned before. The Mycielski graphs M_i are triangle-free, with chromatic number i , $N_i = 3 * 2^{i-2} - 1$ nodes and $3 * N_{i-1} + N_i$ edges. The M_4 is also called Grötzsch graph. These graphs are been solved in [91, 92]. But even though are a good starting point in the GCP, they are very easy graphs. Caramia [93] shows that sometimes is not feasible to solve the GCP with any heuristic. More works based in Mycielski graphs can be found in Lam and Larsen[94, 95].

Other classic problem is the queens' graphs, based in the chessboard and chess rules. A more exhaustive explanation is given in appendix A. We find a solution of these graphs in [96].

Mizuno [1, 97] have generated 3 colorable graphs that are hard to solve. Using Mizuno's method we can build graphs that although their chromatic number is only 3 are very hard to solve.

Other graph type can be found in Chang [98] describing outer-planar graphs and trees and in Petrosyan [99, 100]. A tree is an undirected graph in which any two nodes are connected by exactly one simple path. More graphs with the geometrical special feature in Kang and Klotz [101, 42]. Less tested graph families found in the literature are also important as the one reported in [102] inspired in the DNA. The GCP is usually focused in non directed graphs, but directed graph can be colored as well [103].

Permutation graphs in [104] and [105]. Graphs with long paths are solved in [106], but it can be problems solving these graphs as noted in [107]. Furmanczyk [108] used mixed graphs with directed and non directed edges. Herrmann in [109] speak about critical graphs.

2.3.3 Swarm Intelligence for Graph Coloring

As we have mentioned, the SI has been used for a lot of mathematical problems, and also for GCP solution. The ACO and PSO approaches are the most used, relegating other SI algorithms, such as the flocking behaviors, to the background.

The first work that we find using ants to GCP solving is [110] where the ants are used to perform the coloring of a graph. We can see a survey of ACO applications in [111]. An algorithm using a chaotic ant colony is proposed in [112]. Borkar[17] introduces an incremental learning component. Lu[113] introduces a memetic algorithm for graph coloring also using ants. Dowsland [114] have improve the classical ACO algorithm. Another ant based algorithm appears in Bui [115].

The PSO has been used in Cui [116] where a modified PSO using disturbances is used, obtaining better results than standard PSO in planar graphs. Hsu [117] adding a modified turbulence to previous PSO, obtained better results in 4 colorable graphs solving them efficiently and accurately.

The use of basic SI have been forgotten, but SI is enough to solve the GCP. We demonstrate in Graña [118] that SI can color graphs. Adding the gravitational law and a method for escaping from local optimum we build a competitive algorithm based in SI [119, 120].

Into the SI category we can present agent based algorithm like Xie[121], but there are no more references.

2.4 GCP Applications

The GCP can be applied to a wide number of areas. In Demange [122] is applied in robotics. Clustering dynamics of nonlinear oscillator network using graph coloring in Wu [123]. A monthly crew scheduling problem with preferential bidding in the airline industry is solved in Gamache [124]. Communication protocols in Buck [125].

The GCP is a particular case of an optimization problem with quadratic constraints. The mapping procedure and an appropriate parameter-setting procedure are detailed by Talavan [126] to solve it.

Experiments with graphs are shown in Lewandowski [127] applied to scheduling. Lewis [128] applied GCP solution in round-robin sports scheduling.

Chapter 3

GCP Algorithms

This chapter gives a description of the algorithms that have been applied to solve the GCP, focusing on the state-of-the-art algorithms used as competitors to the Gravitational Swarm which will be explained in detail in forthcoming chapters.

The structure of the chapter is as follows: Section 3.1 gives an introductory view of the algorithms. Algorithms are described in detail as follows: Backtracking in Section 3.2, DSATUR in Section 3.3, Tabu Search in Section 3.4, Simulated Annealing in Section 3.5, Ant Colony Optimization in Section 3.6, Particle Swarm Optimization in Section 3.7, and finally Gravitational Swarm in Section 3.8.

3.1 Graph Coloring Problem methods

We have implemented 6 GCP solving methods as described in the literature: Backtracking, DSATUR, Tabu Search, Simulated Annealing, Ant Colony Optimization and Particle Swarm Optimization. These methods have been proved individually to solve the GCP, but there is no reported direct comparison between them. We have developed a new algorithm called Gravitational Swarm Intelligence that is included in this comparison, after proving that our algorithm works with the GCP. A description of each algorithm follows:

1. Backtracking is an exhaustive deterministic algorithm that explores all the search space and always returns the optimal solution if it exists. As the GCP is a NP-complete problem we can use backtracking only in small size problems or special types of graphs like the Mycielsky graphs. This

algorithm always return the same solution for the same graph instance. Backtracking is no useful with medium size or big graphs, because it needs a huge computational time.

2. DSATUR (Degree of Saturation): this algorithm developed by Br elaz [34] is a greedy backtracking algorithm which does not explore exhaustively all the search space. It looks for the biggest clique in the graph fixing the initial number of colors needed to color it. Then starts the search to determine the coloring of the remaining nodes of the graph. The clique of a graph [129] is a subset of its nodes such that every two nodes in the subset are connected by an edge. It will be necessary at least the same number of colors k as the clique degree to color the graph, which is the reason for the algorithm’s name “degree of saturation”.
3. Tabu Search (TS): it is a random local search with some memory of the previous steps, so the best solution is always retained while exploring the environment [73]. TS needs a great amount of memory to keep the solutions visited, and if the Tabu list is big, it will need so much time to search in the Tabu list indeed.
4. Simulated Annealing [130]: inspired in the annealing performed in metallurgy, this probabilistic algorithm finds solutions randomly. If a solution is worse than the previous solution it can nevertheless be accepted as the new solution with a certain probability that decreases with a global parameter called temperature. Initially, the temperature is big and almost all the solutions are accepted, but as the temperature cools down, only the best solutions are selected. This process allows the algorithm to avoid being trapped in local optima. This algorithm has a big handicap when applied to solve the GCP, because there are a lot of neighboring states that have the same energy value. Despite this handicap, Simulated Annealing algorithm provides state-of-the-art results for this problem[49].
5. Ant Colony Optimization (ACO): we have build an implementation following [112] where we have $n \times n$ ants making clusters around the colors. We have n ants in each of the n nodes. Each ant is labeled with a randomly selected color, and the color of a node is equal to the color of the maximum size group of ants of the same color in this node, i.e. is decided by majority voting. In each step, the ants that have a color different from the node’s color move through the edges to the neighbor nodes. With the

exiting ants and the new incoming ants, the color of each node is again evaluated until the problem is solved.

6. Particle Swarm Optimization (PSO): we have built a PSO version of [117] for graph coloring. The PSO algorithm uses the knowledge of the agents on the problem solution. After each step the agents know if they improve their last state and also if the overall system has found an improved solution. With a given probability value the bad colored agents try first to go back to their own best known position. Then with other probability value the bad colored agents try to go back to the system's best position. We have added a parameter that alter the probabilities, making easy to escape from poor local minima in the first stages of the algorithm, but more difficult as the time goes on. We also have added a random probability to change good agents color using the local and global probabilities.
7. Gravitational Swarm for Graph Coloring (GSGC): this algorithm is inspired in the Gravitation physic law of Newton, and the Boids swarm of Reynolds [4]. The gravitation law has been previously used in Swarm Intelligence for function minimization [25], unrelated to the GSGC formulated as GSI in [119]. This algorithm does not try to mimic exactly a physical system obeying Newton's law. The GSGC consist in a group of agents representing the graph nodes navigating in a world where the colors are represented as goal locations that exert an attraction to the agents. When an agent arrives to a goal it can get corresponding color and stop moving if there are no other agents that can't have the same color for the GCP solution, called enemies. Initially a random position is selected for each agent. Depending on its position relative to the color goals, the agent moves toward the nearest color goal until reaching it. If there are enemies settled in that goal, been a enemy a node that can't have the same color, then the agent tries to expel the enemies outside the goal to a random position before going itself inside. If it is no able to expel the enemies then the agent is expelled to a random position and starts again looking for a stable color goal. Otherwise the agent holds the goal color position and stops moving. If all the agents are stopped then the algorithm has solved the GCP.

All the algorithm implementations allow for a sequential search of the graph's chromatic number. Starting from an upper bound, the GCP is solved for de-

creasing numbers of colors, until reaching a number when the algorithm does not reach a feasible solution. The previous number is assumed as the chromatic number.

3.2 Backtracking (BT)

The BT algorithm orders the nodes of the graph, assigning a color to the first nodes and then starts coloring the next nodes. The colors of the nodes linked with it can't be used so that the algorithm uses the remaining colors. If the group of available colors is empty, at least one of the previous colored nodes is wrong. The algorithm doesn't know which node is wrong. It only knows the order of the nodes. The Algorithm goes back and change the color of the node. The algorithm go on with this new configuration. If the problem appears again, (the group of colors is empty) then it goes back again, but each previous colored node has a smaller color list to avoid using a color that has been proven to be wrong.

Moving forward and backward, the BT algorithm explores the whole search space and it is guaranteed to find the optimal solution if the number of colors given as input is greater or equal to the graph's chromatic number.

If the problem is solved, the algorithm stops. If we allow the algorithm a bounded computational time, and the algorithm can't solve the problem in such time then the algorithm doesn't return any kind of solution.

If the order of the nodes don't change, the algorithm always return the same coloring solution. We can see the algorithm in Algorithm 3.1. Where V is the set of nodes, $Vlist$ the list of used color for each node, and $k \in \{1, 2, \dots, C\}$ the set of colors.

3.3 DSATUR

The DSATUR algorithm can be seen as a improvement of the BT. The base of this algorithm is the degree of each node, that is the number of edges that has the node. The algorithm order the nodes by the degree and start coloring from these nodes. As this nodes has more links with other nodes, if they are well colored, the probability of backtrack is small. But this probability exists and even this algorithm is faster than the BT, can be still too slow if the degree of the nodes is small, the graph is very big or the degree of the nodes is very

Algorithm 3.1 Backtracking

1. Arrange the V nodes randomly.
 2. Color the first node with color 1.
 3. Go to next node
 - if last node
 - then** stop
 4. Color node V_i with color $k \notin Vlist_i$.
 - if $k = \emptyset$
 - then** clear $Vlist_i$
 - go** back to previous node
 - go** to 4
 - else** keep k color is $VList_i$
 4. If there is a conflict
 - then** go to 4
 - else** go to 3
-

Algorithm 3.2 DSATUR

1. Find the biggest clique BC and arrange the nodes of the clique first V_{BC}
 - then** the $V - V_{BC}$ nodes randomly.
 2. Color the V_{BC} with the first $card(V_{BC})$ colors.
 3. Go to next node
 - if** last node
 - then** stop
 4. Color node V_i with color $k \notin Vlist_i$.
 - if** $k = \emptyset$
 - then** clear $Vlist_i$
 - go** back to previous node
 - go** to 4
 - else** keep k color is $VList_i$
 4. If there is a conflict
 - then** go to 4
 - else** go to 3
-

similar, because in this cases it is equivalent to a BT.

Our implementation of the DSATUR uses cliques instead of degrees. Finding the biggest clique, we can assign colors to the clique so the algorithm needs less backtracks than the BT. It is important to order the nodes according to the clique nodes. This way the algorithm start with fixed colored nodes. Finding the biggest clique of a graph can be a hard job, so we look for the first biggest clique and then we apply the standard BT algorithm as specified in Algorithm 3.2, where BC is the biggest clique, V the number of nodes, V_{BC} the nodes of the clique, $card(V_{BC})$ the cardinality of V_{BC} and $k \in \{1, 2, \dots, C\}$ set of colors, where C must be $C \geq card(V_{BC})$.

Theorem 16. *The Dsatur algorithm is exact for bipartite graphs.*

3.4 Tabu Search (TS)

We have implemented a basic version of the Tabu Search. In the literature we found the TabuCol algorithm [53, 52] where each node has a Tabu list of color transitions. This implementation of the Tabu Search looks quite similar to other implementations, so we have used a standard Tabu Search instead of TabuCol, using a Tabu tenure for a full solution. The algorithm flow is as follows:

- In the Initialization phase, we assign colors aleatory to the node. We set a iteration counter $t = 0$.
- Increase the time counter $t + 1$ and generate a new solution C' between all the possible solution in the previous iteration t . This new solution can be inside the tabu list TL and then we generate a new solution until the new solution is not in TL.
- If the cost function of the new solution C' is better than previous
- If we have success assigning colors then we have finished, if not we execute the *Tabu_check()* procedure until we find a solution of the problem or we have try more than a given number of iteration (*maxiter*).

The *TabuCheck()* procedure saves the new configuration in the Tabu tenure. Choose the node with bad colors and change their colors in a way that the new configuration is not in the Tabu tenure. If the new configuration is not in the Tabu tenure then we go on with the algorithm, if not, we change the colors assigned to the bad until the new configuration is not in the Tabu tenure.

If the new configuration solves the problem then we finish. If not we add the configuration into the Tabu list and we call again the *TabuCheck()* procedure. The Tabu list has a limit. If we exceed the Tabu tenure limit, then we delete the old instances of the Tabu tenure. The size of the Tabu tenure is directly related with the number of iterations (*maxiter*) so it's quite difficult to exceed the size of the list, but is possible.

We don't assign values to the register in the Tabu list, because we only change the bad colored node. We don't fall in local minimum because we can't repeat a color that has been previously used and also we change dynamically the bad colored node. The description of the algorithm is given in Algorithm 3.3.

We must say that this algorithm can be slow if the Tabu tenure is big, and also the number of edges is also big, because we will need a lot of memory to

Algorithm 3.3 Tabu Search

1. Choose an initial solution $C(0)$. Set $t = 0$.
 2. Set $t = t + 1$
 - if** $t = \text{maxiter}$
 - then** stop
 3. Generate C' a new solution from $C(t)$
 - if** $C' \in TL$
 - then** go to 3
 4. If $f(C) < f(C')$ then
 - set** $C \leftarrow C'$.
 5. Update tabu and aspiration conditions.
 6. If a stopping condition is met then stop.
 - Else** go to 2.
-

keep the list, and a lot of time to travel through the list in each step of the algorithm.

3.5 Simulated Annealing (SA)

We have developed a version of the Simulated Annealing algorithm [130] adapted to use in the GCP. The temperature parameter is the number of steps, and the algorithm stops when the temperature reaches to zero if it hasn't find a solution before. We can see this algorithm in Algorithm 3.4, but the reader can find a better explanation of the algorithm in our previous work [130].

$E(t)$ is the energy function calculated in the instant t . This function evaluates the number of bad colored nodes in the graph. $p(t)$ is a random probability between 0 and 1.

Algorithm 3.4 Simulated Annealing

1. Choose an initial solution C and $E(0)$
 2. $C' \leftarrow C$
 $C'' \leftarrow C'$
 3. calculate $p(t+1)$ and $E(t+1)$
 4. if $e^{-\frac{E(t)-E(t+1)}{\Delta temp}} > p(t)$
then $C'' \leftarrow C'$
decrease $temp$
 5. $C \leftarrow C''$
 6. If a stopping condition is met or $temp = 0$
then stop
else go to Step 2.
-

3.6 Ant Colony Optimization (ACO)

We have implemented the ACO algorithm in a particular way. Ants have an assigned color. The ants move if their color is no acceptable for the actual node where they are, staying if the color is acceptable. The ACO algorithm start with N ants for each one of the N nodes. Color assignment to ants is random. We order the ants of each vertice by their colors and the color with the biggest number of ants becomes the color of the vertice. We evaluate if the configuration solves the problem or not. If not, then the ants that hasn't the color of their vertice moves towards other nodes that are connected with their vertice. The movement of the ants follows an order. If there is any vertice connected with their vertice that has their color, then the ants move towards that vertice, if not they move randomly. All the ties are solved randomly.

If a node gets empty of ants, then the algorithm generate new N ants with random colors for that vertice. The algorithm stops after *maxiter* iterations. The detailed specification of the algorithm can be seen in Algorithm 3.5.

- t is the number of iterations.
- " a " is an ant. $C(x)$ is the color $C = \{1, \dots, k\}$ of x , ant of node.

Algorithm 3.5 Ant Colony Optimization

1. set $t = 0$ and set random colors to $n * n$ ants.
 2. place n ants in each node V_n
 3. set $t = t + 1$
 3. Evaluate the color k of each node
 4. for each ant a in a node V_i
 - if $C(a) = C(V_i)$
 - then stay
 - else $Move(a) \rightarrow V_j \mid \exists E(i, j)$
 5. if problem solved or $t = maxiter$
 - then stop
 - else go to 3
-

- $Move(a)$ is a function to traslade ant from current node to a new node.
- $E(i, j)$ represente an edge between the nodes V_i and V_j .

This algorithm has some problems. First of all is that the number of agents or ants that must be checked is n times bigger than other algorithms, because we have n ants for each vertice, and this number can grow during the execution of the algorithm. The amount of memory and the computational time of this algorithm can be very big, and the parellelization wouldn't help because we will need too many processes, and the communication between process will be too much expensive.

3.7 Particle Swarm Optimization (PSO)

The PSO is a fashionable algorithm that is been used for a lot of problems. The GCP doesn't escape to this trend and there are a lot of algorithm based in Swarm Intelligence for this problem.

We have implemented a basic version of a PSO algorithm for GCP. The algorithm keeps the best solution found for each agent of the swarm and also the best solution that all that agent have achieved for the GCP. If a node has a bad coloring, it means that breaks the rule of the coloring of a graph, then with

a certain probability the agent gets the color of their best local position. If the agent fails getting it's bet local position, then with another certain probability try to get to color of the global best position. If the agent fails again then a random color is assigned to the agent. The new color assigned can be the local best, or the global best but must be different from the current color.

Then with another certain probability, much lower than the local and global probabilities we try to change the color of good colored nodes. The good nodes try to get their best local position or best global position in the same way as the bad nodes. Here the current color can be the local or the global position so we can choose it. This is to escape from local minimum.

After each iteration of the algorithm the probability of bad nodes to get theirs local or global best increases, and the probability of the good nodes of changing their color's decreases. The algorithm is presented in Algorithm 3.6. Lb is the Local best solution. Gb is the global best solution. pl is the acceptance probability of the local best solution for a bad colored node V_i . pg is the acceptance probability of the global best solution for a bad colored node V_i . pn is the probability to change the color of the a good colored node V_i . $U(t)$ is the acceptance threshold of the probabilities in the instant t . This threshold increases each time until reaches to 1 and only local best solutions are selected.

3.8 Gravitational Swarm Intelligence (GSI)

For completeness, a pseudo-code of the Gravitational Swarm Intelligence (GSI) algorithm for GCP is presented in Algorithm 3.7. Chapter 4 is devoted to a detailed description of the algorithm, as well as some convergence theoretical results.

Algorithm 3.6 Particle Swarm Optimization

1. Set C a random coloring and $t=0$
 2. Set $Lb = Gb = f(C)$
 3. set $t = t + 1$, choose random $pl(t)$, $pg(t)$ and $pn(t)$ and a threshold $U(t+1) \geq U(t)$
 4. Choose a new solution C' where
 - 4.1 for each bad colored node V_{bad}
 - if $pl(t) > U$
 - then $C'(V_i) = Lb_i$
 - go to 4.1
 - if $pg(t) > U$
 - then $C'(V_i) = Gb_i$
 - go to 4.1
 - $C'(V_i) = Random(C)$
 - for each good colored node V_{good}
 - if $pn(t) > U$
 - then $C'(V_i) = Random(C)$
 5. if problem solved of $t=\maxiter$
 - then stop
 - else go to 3
-

Algorithm 3.7 Gravitational Swarm for Graph Coloring

1. deploy goal colors GC and agents V randomly in the space.
 2. assign a random position to enemies V_e
 2. $Move(V_i) \rightarrow GC$
 3. if $V_i \subset GC_k$
 - then $C(V_i) = k$
 4. if $\forall V \exists E(i, j) \mid C(V_i) = C(V_j)$
 - then randomly $V_e \leftarrow i \vee j$
 - go to 2
-

Chapter 4

Gravitational Swarm Intelligence

This chapter contains the central contribution of the thesis from the formal and theoretical point of view. Later chapters report the empirical support for the GCP solving approach. We give an intuitive description of the Gravitational Swarm Intelligence (GSI) algorithm, and some theoretical results in the limit case under some simplifications. 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 attraction between objects. We construct a world where agents navigate 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 induced by the adjacency relation in the underlying graph.

The chapter is organized as follows: Section 4.1 gives an intuitive description of the algorithm as it is currently implemented. Section 4.2 recalls the definition of the GCP. Section 4.3 formalizes the most general Gravitational Swarm giving some basic asymptotic convergence results. Section 4.4 specifies the Gravitational Swarm for the GCP solving, giving asymptotic convergence results.

4.1 Gravitational Swarm for

Initial definitions: Let be $G = (V, E)$ a graph defined on a set of nodes $V = \{v_1, \dots, v_N\}$ and edges $E \subseteq V \times V$. We define a group of GSI agents $B = \{b_1, b_2, \dots, b_N\}$ each corresponding to a graph node. Each agent navigates inside a square planar toric world according to a speed vector \vec{v}_i . At any moment in 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 $C = \{1, 2, \dots, K\}$ the set of colors, where K must not be lower than the chromatic number of the graph for the GSI to converge. We assign to these colors, K fixed points in space, the color goals $CG = \{g_1, \dots, g_K\}$, endowed with a gravitational attraction resulting in a velocity component \vec{v}_{g_c} affecting the agents. The attraction force decreases with the distance, but affects all the agents in the space.

GSI definition: We can model the system as a tuple $F = (B, CG, \{\vec{v}_i\}, K, \{\vec{a}_{i,k}\}, R)$ where B is the set of GSI agents, $\{\vec{v}_i\}$ the set of agent velocity vectors at time instant t , K the hypothesized chromatic number of the graph, and $\{\vec{a}_{i,k}\}$ are the attraction forces of the color goals exerted on the agents. R denotes the repulsion forces in the neighbourhood of color goals.

Dynamics: When the euclidean distance between an agent and the color goal is below a threshold *nearenough*, the agent stops moving and the corresponding graph 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

$$\mathcal{N}(g_k) = \{b_i \text{ s.t. } \|p_i - g_k\| < \textit{nearenough}\}. \quad (4.1)$$

We denote the fact that the node has been assigned to the corresponding color assigning value to a the agent color attribute

$$b_i \in \mathcal{N}(g_k) \Rightarrow c_i = k. \quad (4.2)$$

The initial value of the agent color attribute c_i is zero or null. Inside the spatial neighbourhood of a color goal there is no further gravitational attraction. However, there may be a repulsion force between agents that are conected with an edge in the graph G . This repulsion is only effective for agents inside the

same color goal neighbourhood. To model this effect, we define function *enemy* which 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:

$$R(b_i, g_k) = \sum_{N(g_k)} \text{enemy}(b_i, b_j). \quad (4.3)$$

Agent velocity: The dynamics of each GSI agent in the world is specified by the iteration:

$$\vec{v}_i(t+1) = \begin{cases} 0 & c_i \in C \& (\lambda_i = 1) \\ d \cdot \overrightarrow{a_{i,k^*}} & c_i \notin C \\ \vec{v}_r \cdot (p_r - p_i) & (c_i \in C) \& (\lambda_i = 0) \end{cases}, \quad (4.4)$$

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 \vec{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.

Comfort dynamics: Every time step that the GSI agent stays in that goal without been disturbed, its *Comfort* increases, until reaching a maximum value *maxcomfort*. 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 *Comfort* values of the challenged agents are bigger than 0 then their *Comfort* decreases. If the *Comfort* 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 (4.4) 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. $\forall i, \vec{v}_i = 0$; therefore $f(B, CG) = n$ and the GCP of assigning K colors to graph G is solved.

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

$$f(B, CG) = |\{b_i \text{ s.t. } c_i \in C \ \& \ R(b_i, g_{c_i}) = 0\}|. \quad (4.5)$$

This cost function 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 normalized 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.

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 $\|d\| < \textit{nearenough}$ then we make $d = 0$, and the agent's velocity becomes 0 stopping it.

Flow diagram specification: The flowchart of figure 4.1 shows the internal logic working of each GSI agent. This flowchart is defined for each agent, and can happen that two agents arrive at the same state at the same time. To break such ties, we decided to choose an aleatory order for the agents in order to avoid cycles in the behaviour of the algorithm.

When all the agents are in a color goal without enemies then they move to Finish state and the problem solution is reported. If there are agents still without a proper color then the proper colored agents must wait in the "Stand By" state.

If an agent's confort reaches *maxconfort* value, then the "Increases Confort" state is only a transition to the "Stand By" state, without increasing the confort and without affecting the overall behavior of the algorithm.

A simpler version of this flowchart is given in figure 4.2. The simplified flowchart starts in the green transitory state. Then select a random position for

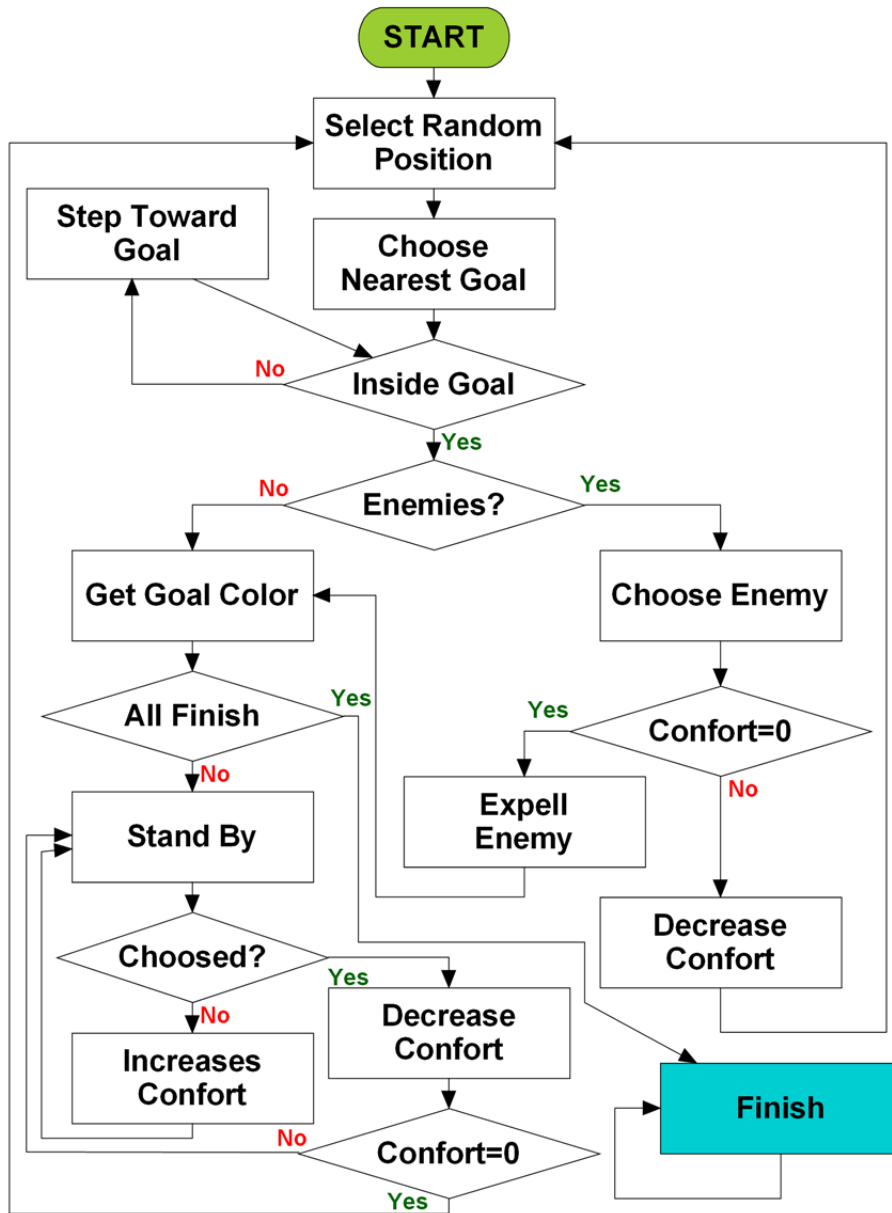


Figure 4.1: GSI agent behavior flowchart for GCP

the agent. The agent goes toward the nearest goal attracted by the gravity of the goal, until get inside a goal. In that moment the agent get the color of the goal and check for enemies. If there are no enemies the agent stop, if not the agent or it's enemies is selected for expulsion of the goal and the system start again. In this simplification, we haven't mention the friend-or-foe relation, nor the optimization to avoid goal with enemies.

4.2 Graph coloring problem

An undirected graph is a collection of vertices linked by edges $G = (V, E)$, such that $V = \{v_1, \dots, v_N\}$ and $E \subseteq V \times V$, and $(v, w) \in E \Rightarrow (w, v) \in E$. The neighborhood of a vertex in the graph is the set of vertices linked to it: $\mathcal{N}(v) = \{w \in V \mid (v, w) \in E\}$.

Definition 17. Graph coloring. Let $C = \{c_1, \dots, c_M\}$ denote a set of colors. Given a graph $G = (V, E)$, a graph coloring is a mapping of graph vertices to colors $\mathcal{C} : V \rightarrow C$ such that no two neighboring vertices have the same color, i.e. $w \in \mathcal{N}(v) \Rightarrow \mathcal{C}(v) \neq \mathcal{C}(w)$.

Definition 18. Minimal graph coloring. A set of colors C^* is minimal relative to graph $G = (V, E)$ if (1) there is a graph coloring $\mathcal{C}^* : V \rightarrow C^*$, and (2) for any smaller set of colors there is no graph coloring using it: $|C| < |C^*| \Rightarrow \neg \exists \mathcal{C} : V \rightarrow C$. Alternative definition: any graph coloring on this graph has a greater or equal set of colors $\mathcal{C} : V \rightarrow C \Rightarrow |C| \geq |C^*|$.

Definition 19. Chromatic number: The chromatic number M^* is the number of colors of the minimal graph coloring C^* .

4.3 Gravitational Swarm

Definition 20. A Gravitational Swarm (GS) is a collection of particles $P = \{p_1, \dots, p_L\}$ moving in an space \mathcal{S} subjected to attraction and repulsion forces. Attraction correspond to long range gravitational interactions. Repulsions correspond to short range electrical interactions. Particle attributes are: spatial localization $s_i \in \mathcal{S}$, mass $m_i \in \mathbb{R}$, charge $\mu_i \in \mathbb{R}$, set of repelled particles $r_i \subseteq P$. The motion of the particle in the space is governed by equation:

$$\dot{s}_i(t) = -m_i(t) A_i(t) + \mu_i(t) R_i(t) + \eta(t), \quad (4.6)$$

Flowchart

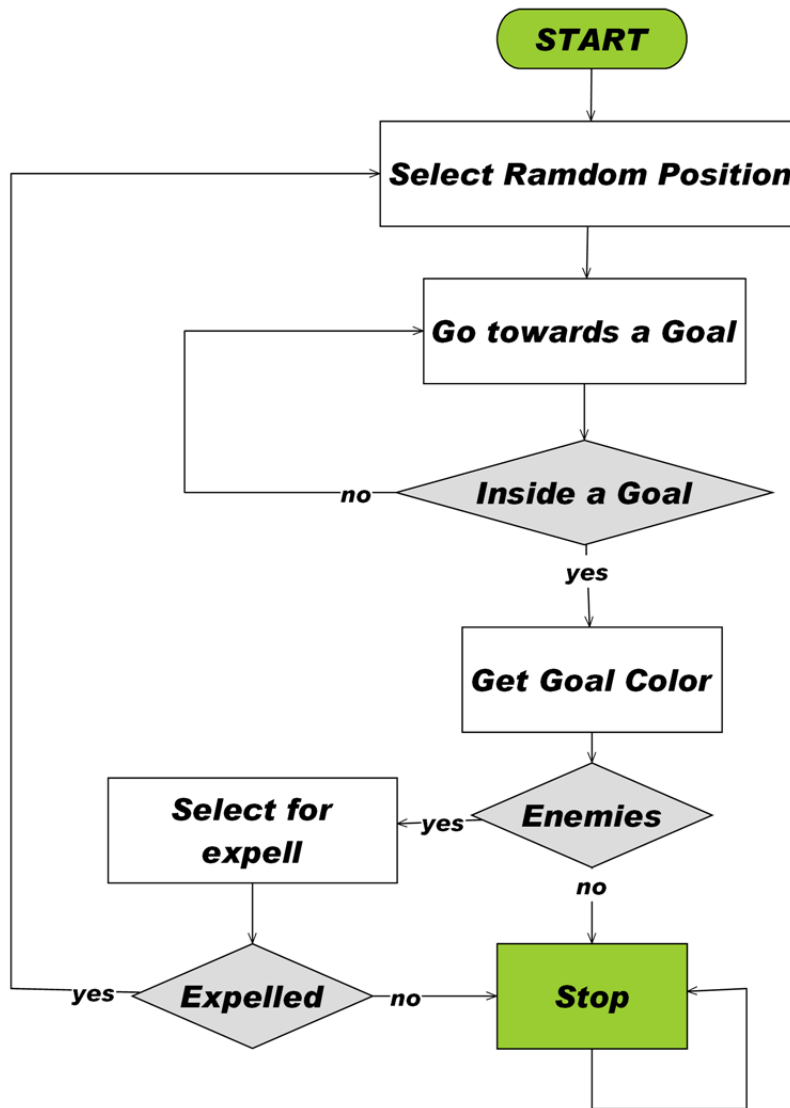


Figure 4.2: Simplified Flowchart

where $A_i(t)$ and $R_i(t)$ are the result of the attractive and repulsive forces, and $\eta(t)$ is a random (small) noise term. The attractive motion term is of the form:

$$A_i(t) = \sum_{p_j \in P - r_i} m_j(t) (s_i - s_j) \delta_{ij}^A, \quad (4.7)$$

where

$$\delta_{ij}^A = \begin{cases} \|s_i - s_j\|^{-2} & \|s_i - s_j\|^2 > \theta_A \\ 0 & \|s_i - s_j\|^2 \leq \theta_A \end{cases}. \quad (4.8)$$

The repulsive term is of the form

$$R_i(t) = \sum_{p_j \in r_i} \mu_j(t) (s_i - s_j) \delta_{ij}^R.$$

where

$$\delta_{ij}^R = \begin{cases} \|s_i - s_j\|^{-2} & \|s_i - s_j\|^2 \geq \theta_R \\ 0 & \|s_i - s_j\|^2 < \theta_R \end{cases}. \quad (4.9)$$

Remark 21. The two delta functions have different roles in the definition of the GS. The attractive δ_{ij}^A corresponds to the inverse to the distance strength of attraction. To avoid singular values when two particles are close to zero distance we set a threshold θ_A which determines the region around the particles where the motion due to attraction forces disappear. The repulsive δ_{ij}^R defines the maximum extension of the repulsive forces, which are short range forces. The threshold θ_R determines the region around the particles where the repulsive forces are active.

Remark 22. We allow both mass and charge to be time varying. In exploratory computational experimental works [119, 120] we have found that manipulating them can be useful to enhance convergence, however we will not need them to be time varying in the ensuing formal proofs.

Lemma 23. *A particle p_i reaches zero velocity when it is clustered with all non repulsive particles and all repulsive particles are at distance greater than the specified threshold. Formally, when $\|s_i - s_j\|^2 \leq \theta_A$ for all $p_j \in P - r_i$, and $\|s_i - s_j\|^2 > \theta_R$ for all $p_j \in r_i$.*

Proof. From the definition of particle velocity. □

Lemma 24. *A necessary and sufficient condition for all particles to reach zero velocity, thus GS reaching an stationary state, is that for all p_i, p_l, p_k if $p_l \in P - r_i$ and $p_k \in P - r_l$, then $p_k \in P - r_i$. Equivalently, if $p_l \in r_i$ and $p_k \in r_l$,*

then $p_k \in r_i$. In other words, the GS can reach an stationary state if only if the attractive relation between particles forms an equivalence relation .

Proof. We prove necessary and sufficient conditions □

- If: For each particle p_i , the distance between p_i and all particles in $P - r_i$ will converge to zero. Particle positions will converge to an average position $\bar{s}_i = \bar{s}_j$ for al p_j in $P - r_i$. Particles in r_i will be pushed to a distance θ_R from \bar{s}_i . Thus both attractive and repulsive terms of the particle speed will converge to zero.
 - Only if: proven by contradiction. Assume that $p_k \notin P - r_i$ and still we have stationary states. We have $p_k \in r_i$. Then p_k will be attracted to \bar{s}_i because $p_l \in P - r_i$ and $p_k \in P - r_l$, i.e. p_l is attracted to p_i and p_k follows p_l . However, when p_k is below θ_R distance of \bar{s}_i then the repulsive forces will be in effect. Therefore, when attractive forces become zero because particles are inside a θ_A distance, repulsive forces will be non zero for at least one of the particles.

Lemma 25. *Global convergence of GS. If the conditions of Lemma 24 hold, any non stationary state of a GS leads to a stationary state.*

Proof. For all particles in $P - r_i$ will be attracted to p_i whatever the distance, while all particles in r_i will be moving away from p_i until both distance terms will be zero. For any $P(t)$ and $P(t+1)$, we have that the following holds:

$$\|s_i(t) - s_j(t)\| > \|s_i(t+1) - s_j(t+1)\|; p_j \in P - r_i,$$

until $\|s_i(t) - s_j(t)\| \leq \theta_A$, and

$$\|s_i(t) - s_k(t)\| < \|s_i(t+1) - s_k(t+1)\|; p_k \in r_i,$$

until $\|s_i(t) - s_k(t)\| \geq \theta_R$. □

Remark 26. The condition of Lemma 24 implies that the GS can only reach an stationary state if the graph defined by the repulsive relations consists of a collection of disjoint cliques. For this the reason the GS applied to GCP needs some additional stationary particles, and the attractive factor of equation 4.8 is changed to equation 4.10.

Remark 27. The Lemma 25 means that the GS has robust global convergence, any initial state will lead to a stationary state, if there is any one. This is a highly desirable result, but limited even in the case of the basic GS.

4.4 Gravitational Swarm for GCP

Definition 28. A GS P for the coloring of graph $G = (V, E)$ with M colors is constructed as follows. The set of particles consists of two subsets $P = P_C \cup P_V$: the vertex particles corresponding to the graph vertices $P_V = \{p_1, \dots, p_N\}$ and static color particles $P_C = \{p_{N+1}, \dots, p_{N+M}\}$. There is a bijective mapping of graph vertices to particles $\phi : V \rightarrow P_V$. The repulsive particles for each particle are determined by the neighboring vertices in the graph:

$$r_i = \{p \in P_V \mid \phi^{-1}(p) \in \mathcal{N}(\phi^{-1}(p_i))\}.$$

There is similar bijective map $\phi_C : C \rightarrow P_C$ from colors to color particles. The mass of color particles may be much greater than the charge of vertex particles $m_i \gg \mu_j$ for $p_i \in P_C, p_j \in P_V$. Moreover, they are considered as static particles:

$$\dot{s}_i = 0; p_i \in P_C.$$

Besides, the velocity attraction term of the particles specified by equation 4.7 is changed to

$$A_i(t) = \sum_{p_j \in P_C} \{m_j (s_i - s_j) \delta_{ij}^A\}. \quad (4.10)$$

Remark 29. Each vertex particle is attracted to its closest color particle according to the different color particle masses. The noise term in equation 4.6 has the effect of breaking any compensation between forces that would cancel them. It might be required to show that the configuration of the particle positions that lead to such cancelations are in a manifold of measure zero, so that the system will never be stuck in an instable stationary state, but we believe that such mathematical depth is beyond the scope of the letter.

Lemma 30. *A vertex particle of a GS-GC reaches zero velocity if and only if it is at distance below θ_A of a color particle and no repulsive particle is in θ_R range.*

Proof. We prove the necessary and sufficient conditions.

- If: by definition of particle velocity in equations 4.6 and 4.10 all terms of the equation will be zero.
- Only if: by contradiction. Assume that the particle has zero velocity and it is either out of range of a color particle or within range of a repulsive particle. Then, either the attractive or the repulsive terms will be different from zero. Therefore the particle velocity will be non-zero unless there is some cancelation effect. The mass of the color particles can be made big enough to avoid any cancelation between attractive and repulsive forces. Cancelation of attractive forces has an arbitrarily small probability and the noise term in equation 4.6 moves the GS-GC from such unstable stationary states.

□

Remark 31. The noise term in equation 4.6 has to be small enough not to push a vertex particle outside of a color particle region of influence.

Corollary 32. *Distances between color particles must be above the repulsive range $\|s_i - s_j\|^2 > \theta_R$ for $p_i, p_j \in P_C$, $p_i \neq p_j$ to ensure that colored particles can reach zero velocity, avoiding repulsive interaction between colored particles.*

Remark 33. When a vertex particle reaches a zero velocity it has attained a locally correct coloring of its corresponding vertex in the graph.

Definition 34. The neighborhood of a color particle $p_i \in P_C$ is the set of vertex particles inside its threshold of attraction $\mathcal{N}(p_i) = \left\{ p_j \in P_V \mid \|s_i - s_j\|^2 \leq \theta_A \right\}$. Color particle neighborhoods are disjoint $\mathcal{N}(p_i) \cap \mathcal{N}(p_{i'}) = \emptyset$ for any $p_i \neq p_{i'}$, because a particle can not be in two places simultaneously.

Definition 35. A global state of the GS-GC is the vector composed of all vertex particles positions $\mathbf{s} = \{s_i; p_i \in P_V\}$.

Definition 36. A global state of the GS-GC is stationary if all the particle velocities are simultaneously zero: $\forall p_i \in P; \dot{s}_i = 0$. Color particles are stationary by definition, therefore the stationary is a property required of the vertex particles.

Theorem 37. *A global state of the GS-GC is stationary if and only if all vertex particles are placed in the neighborhood of some color particle without*

any repulsive particles located at the same color particle neighborhood:

$$\bigcup_{p_j} \mathcal{N}(p_j) = P_V, \quad (4.11)$$

$$p_i \in \mathcal{N}(p_j) \Rightarrow \mathcal{N}(p_j) \cap r_i = \emptyset. \quad (4.12)$$

Proof. We prove the necessary and sufficient conditions:

- **If:** Let $p_j, p_{j'} \in P_C, p_j \neq p_{j'}$. Each vertex particle in a color particle neighborhood has a zero attraction velocity term $p_i \in \mathcal{N}(p_j) \Rightarrow A_i(t) = \mathbf{0}$. Moreover, all particles are in some color particle neighborhood, therefore all attraction terms will be zero. Furthermore, all mutually repulsive particles are in different color particle neighborhoods: $p_k \in r_i \Rightarrow p_k \in \mathcal{N}(p_{j'})$ being outside repulsive range $\|s_i - s_k\|^2 > \theta_R$, therefore the vertex particle repulsive term is also zero $R_i(t) = \mathbf{0}$.
- **Only if:** by contradiction. Assume that the GS-GC is in a stationary state, but the theorem conditions do not hold.
 - If equation 4.11 does not hold, then there is at least one particle which is outside all color particles whose attraction term is non-zero $\forall p_j \in P_C; p_i \notin \mathcal{N}(p_j) \Rightarrow A_i(t) \neq \mathbf{0}$. Therefore, the GS-GC is not in a stationary state.
 - If equation 4.12 does not hold, then two mutually repulsive particles are in the same color particle, therefore their repulsive term is non-zero,

$$p_i \in \mathcal{N}(p_j) \wedge \mathcal{N}(p_j) \cap r_i \neq \emptyset \Rightarrow R_i(t) \neq \mathbf{0},$$

consequently the GS-GC state is not stationary.

□

Remark 38. Theorem 37 implies that we need that the number of color particles has to be in relation with the graph chromatic number. Next theorems establish this relation.

Theorem 39. *If the graph's chromatic number M^* is smaller than or equal to the number of color particles $M^* \leq M$, there will be a non-empty set of stationary states of the GS-GC.*

Proof. By construction. There is at least one optimal graph coloring $\mathcal{C}^* : V \rightarrow C^*$, from which we can construct one stationary state of the GS-GC as follows:

- Assign M^* color particles to the colors in C^* , i.e. $\phi_C(c_i) = p_{N+i}$, the last $M - M^*$ color particles will remain without color assignment.
- Translate the coloring map into a partition of vertex particles in color particle neighborhoods:

$$\mathcal{C}^*(v) = c \Rightarrow \phi(v) \in \mathcal{N}(\phi_C(c)).$$

□

Remark 40. Theorem 39 builds the stationary state corresponding to the optimal graph coloring. However, a graph coloring obtained by the GS-GC may be sub-optimal if the number of color particles is greater than the chromatic number. Nevertheless, we need to be sure that any stationary state corresponds to a graph coloring, i.e. there are no spurious stationary states that can not be translated into a graph coloring.

Theorem 41. *Any stationary state of the GS-GC corresponds to a graph coloring.*

Proof. Given an stationary state we can build a graph coloring as follows:

- Each graph vertex is colored. By Theorem 37 in a stationary state each vertex particle $p_i \in P_V$ belongs to a color particle neighborhood $p_i \in \mathcal{N}(p_j)$, $p_j \in P_C$, therefore it is colored accordingly:

$$p_i \in \mathcal{N}(p_j) \Rightarrow \mathcal{C}(\phi^{-1}(p_i)) = \phi_C^{-1}(p_j).$$

- The coloring is correct: By Theorem 37 in a stationary state $p_i \in \mathcal{N}(p_j) \Rightarrow \mathcal{N}(p_j) \cap r_i = \emptyset$, therefore neighboring graph vertices will have different colors:

$$v_k \in \mathcal{N}(v_i) \Rightarrow p_k \in r_i \Rightarrow \mathcal{C}(\phi^{-1}(p_i)) \neq \mathcal{C}(\phi^{-1}(p_k)) \Rightarrow \mathcal{C}(v_i) \neq \mathcal{C}(v_k)$$

□

Remark 42. Any stationary state of the GS-GC corresponds to a graph coloring. Any graph coloring corresponds to a GS-GC stationary state. There are

no spurious stationary states. Finally, what happens if we underestimate the number of color particles needed to represent the graph coloring?

Theorem 43. *If the graph's chromatic number is greater than the number of color particles, there are no stationary states in the GS-GC.*

Proof. By contradiction. M^* is the chromatic number. Assume that an GS-GC with $M < M^*$ has any stationary state. This stationary state can be translated into a graph coloring with M colors by Theorem 41, therefore the chromatic number is M contrary to the initial assumption. \square

Remark 44. Theorem 43 implies that the GS-GC will not converge to an stationary state if the number of color particles is lower than the chromatic number. However, lack of convergence does not allow us to give any conclusion about the chromatic number of the graph because it may be due to the dynamics of the GS-GC. We need to establish the existence of global convergence conditions, and the relation of the GS-GC parameters to the speed of convergence. In our preliminary results we have introduced mechanisms in the GS-GC dynamics which equivalent to manipulation of the charge of the vertex particles. We are working on the formalization of such process for its analysis.

Remark 45. The problem of determining the chromatic number can be related to the GS-GC dynamics. We have been considering bottom-up and top-down approaches. In the bottom-up approach, the system is initialized with a low number of color particles. Lack of convergence is interpreted as the need to add some color particle to reach the chromatic number. Top down approaches start with a large number of color particles. After finding a stationary state, the number of color particles is reduced and the search restarted, until lack of convergence. A third line of research is to establish some conditions on the color particles masses that would induce some order on the convergence of vertex particles to color particle neighborhoods, so that the chromatic number might be obtained as a by-product of the GS-GC system dynamics.

Chapter 5

Parameter tuning

In this chapter we deal with the sensitivity of the GSI to the fine tuning of its parameters. We try to determine both its robustness against poor settings of parameters, and the optimal range of values for sensitive parameters.

The chapter structure is as follows: Section 5.1 gives the introductory description of the parameters. Section 5.2 reports sensitivity results on the color goal radius. Section 5.3 reports sensitivity results on the comfort parameter. Section 5.4 reports the results of non-parametric statistical tests assessing the statistical significance of the results. Section 5.5 gathers the concluding remarks explaining the parameter settings for following computational experiments.

5.1 GSI model parameters

In the proposed GSI algorithm we have three parameters that must be tuned to get the best result. These parameters are:

- The chromatic number,
- the goal radius or influence region and
- the Comfort or Stress.

Also we have the world size as an additional parameter, but this parameter is irrelevant for our model, because the agents' speed adapts to the world size. If the world is bigger, the agents go faster. The goal radius also limits the influence of the world size. We have experimented over a toric world of 100x100 units of length to simplify the arithmetical calculus. Finally we have a number of steps

Comfort	1	10	20	30	40	50	60	70	80
0	0	0	8.35	118.15	178.05	147.7	74.9	36.1	8.1
1	0.05	248.4	712.1	880.35	837	738.8	607.6	357.8	95.8
5	0	197.05	688	816.15	787	725.1	640	379.55	99
10	0	153.45	686.8	787.5	778.1	721.6	646.25	386.1	104.4
15	0.1	141.6	684.25	787.75	772.55	720.15	644.55	383.2	100.05
20	0.05	135.9	689.15	782.7	773.8	721.7	644.35	389.3	100.35

Table 5.1: Average results of KRG graph of 30 nodes, 50 edges and 6 colors

limit to avoid enormous execution times of days, weeks or months. Obviously this parameter doesn't affect the accuracy of the algorithm, it is only a stopping criterion. We have run computational experiments testing our model for tuning the goal radius and the comfort. We have built ten families of 20 instances each families of KRG graphs.

1. 6-colorable graphs of 30 nodes and 50 edges.
2. 4-colorable graphs of 45 nodes and 90 edges.
3. 5-colorable graphs of 60 nodes and 150 edges.
4. 7-colorable graphs of 75 nodes and 180 edges.
5. 8-colorable graphs of 90 nodes and 200 edges.
6. 9-colorable graphs of 105 nodes and 230 edges.
7. 10-colorable graphs of 120 nodes and 250 edges.
8. 11-colorable graphs of 135 nodes and 270 edges.
9. 12-colorable graphs of 150 nodes and 330 edges.
10. 13-colorable graphs of 165 nodes and 360 edges.

We have a total of 200 graph instances. We have launched our algorithm changing the parameters as we show in the table 5.1, where we also show average results for the first graph family.

We have repeated each experiment 1,000 times, and we limit the number of steps in each execution to 100. The stop condition is small because we wanted to make a large number of experiments to extract conclusions on the effect of the parameter, not about the accuracy of the algorithm. We have run the algorithm 10,800,000 times.

5.1.1 Chromatic number

This parameter doesn't need to be tuned because, like the graph, is an input parameter related with the graph. If we know the chromatic number this is true but if not we must guess the chromatic number. Some theorems help to know a priori this number but not always we can apply these theorems.

We have implement a waterfall approach, where an upper bound and a lower bound are given to the algorithm. The upper bound must be big enough to solve the problem. If the algorithm can solve the GCP with this number then decreases the upper bound one unit and try to solve the problem again with this new chromatic number. The algorithm repeats this behavior until the chromatic number reaches the lower bound or after a fixed amount of time the algorithm is unable to find a solution, been the chromatic number the previous tested number.

In this work we always have used graphs with a known chromatic number so we never have need to used the waterfall approach. We let the upper bound equal to lower bound equal to the chromatic number.

5.2 Goal Radius

The goal radius is a very important parameter because this parameter determines the color of the agents. In our approach there is an attraction over the search space center in the goals. When an agent in near enough to a goal then we assume that the agents color must be the color of the goal. But when we can say that an agent is near enough?. The goal radius determine this distance. When the euclidean distance of an agent to the nearest goal is less than this radius then we assign that color to the agent.

$$dist(a, b) = \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2}. \quad (5.1)$$

If an agent goes towards a goal and when the agent enter inside the goal radius it is going to take the goal color, why don't we assign that color to the agent?. The agents trajectory doesn't change until get inside a goal. The reason is that that agent get the color goal if there are no enemies inside the goal, and that information change along the time. If two enemies moves towards the same goal and we assign that color to the agents, as they are enemies one must change it's trajectory towards another color, but immediately the system assign the new

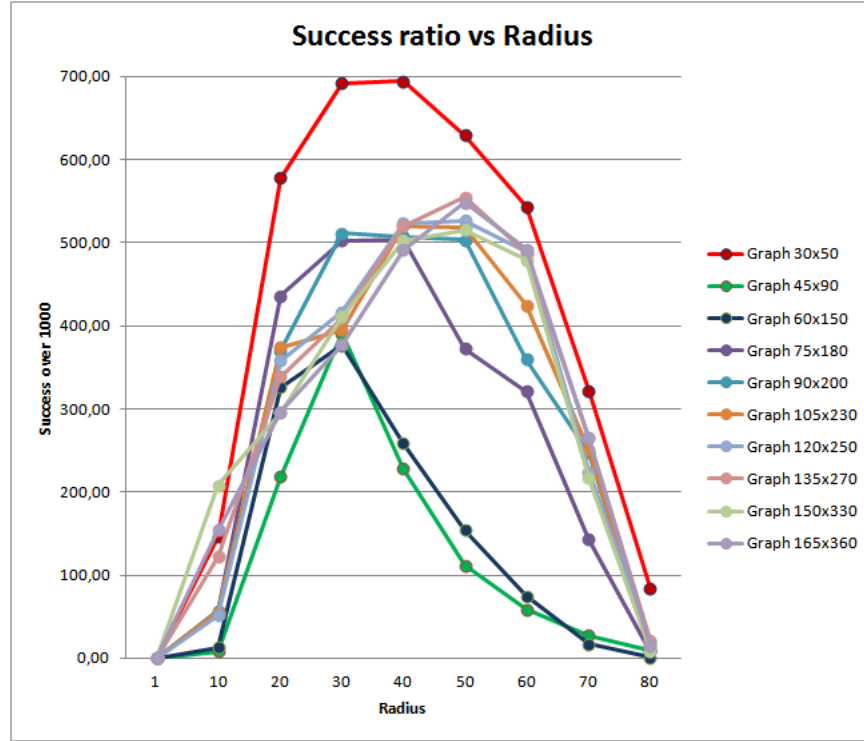


Figure 5.1: Average success ratio vs Goal Radius

color to the agent and in the same time that agent can find enemies with it's color. The result is that the system is jumpy because there is no transition between states. The system doesn't converge.

The experimental results show that the goal radius is very important in the accuracy of the algorithm. As we can see in figure 5.1 when the goal radius is small (1 or 10) the algorithm performance is very low. The same happen with big radius (70 or 80). When the goal radius is between 30 and 50, we get the best results. The families with small number of nodes get the best result in the surroundings of radius 30. When the number of nodes grow, the best radius moves towards a bigger radius. The biggest graph families achieve the best result in radius of about 50 units. The Goal radius change with the number of nodes until a critical point where the performance of the algorithm falls. We have plot the results in 3D to have another point of view in figure 5.2.

In figure 5.3 we have plot the average number of steps need to find a solution. If the algorithm don't manage to find a solution then the number of steps is

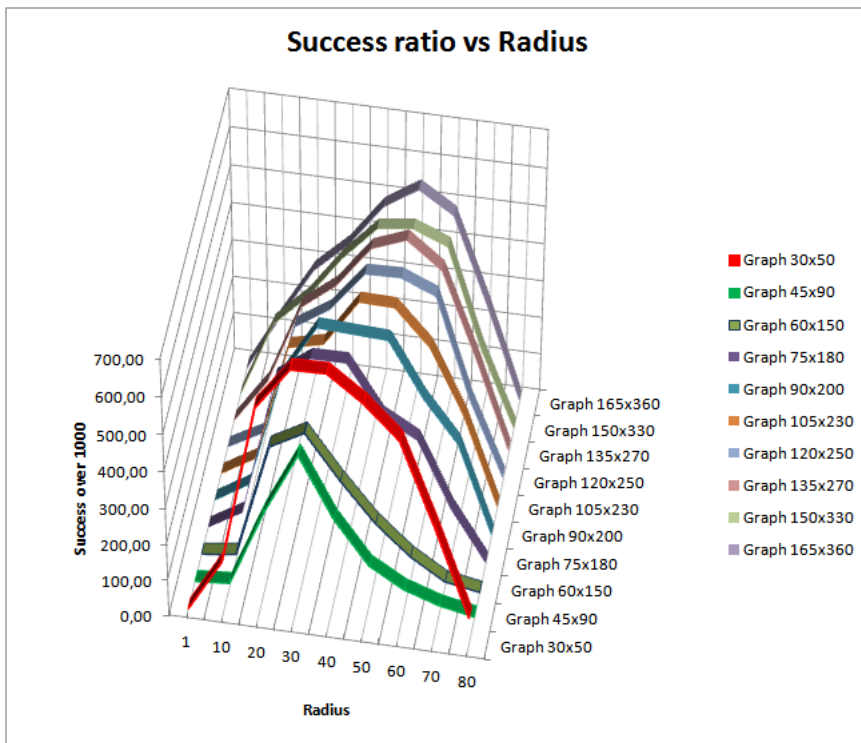


Figure 5.2: Average success ratio vs Goal Radius in 3D

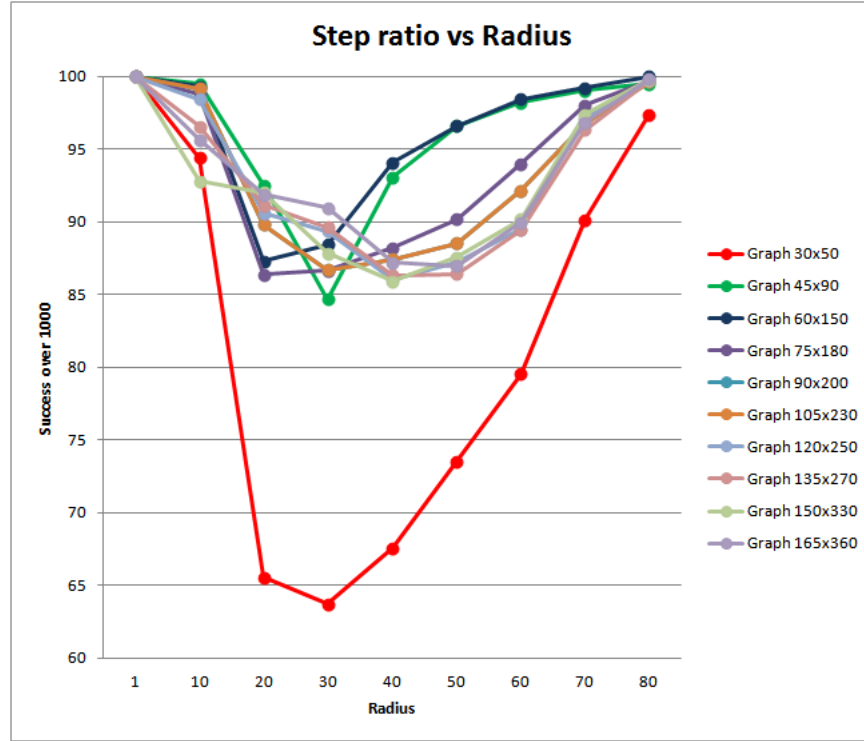


Figure 5.3: Average Steps ratio vs Goal Radius

100. The graphic is very similar to the average success graphic. This is logical because failing finding a solution implies 100 steps, so if in the goal radius tested are a lot of fails, then the number of steps must be big. We can see that not always this is true. The graph family 60x150 and 75x180 are very fast with small radius even though they didn't get the best result with that radius. So small radius looks faster than big radius. We can observe that the tuning of the parameters can be made very quickly, although we have make a loot of experiment. We also have seen that average radius is the best choice. We have use this result for the accuracy experiments in the next chapter.

5.3 Comfort

The comfort is a special parameter that allow the algorithm escape from local minimum, and also contributes to the stability of the system. When an agent gets inside a goal it stops moving and according to it's comfort wait until the

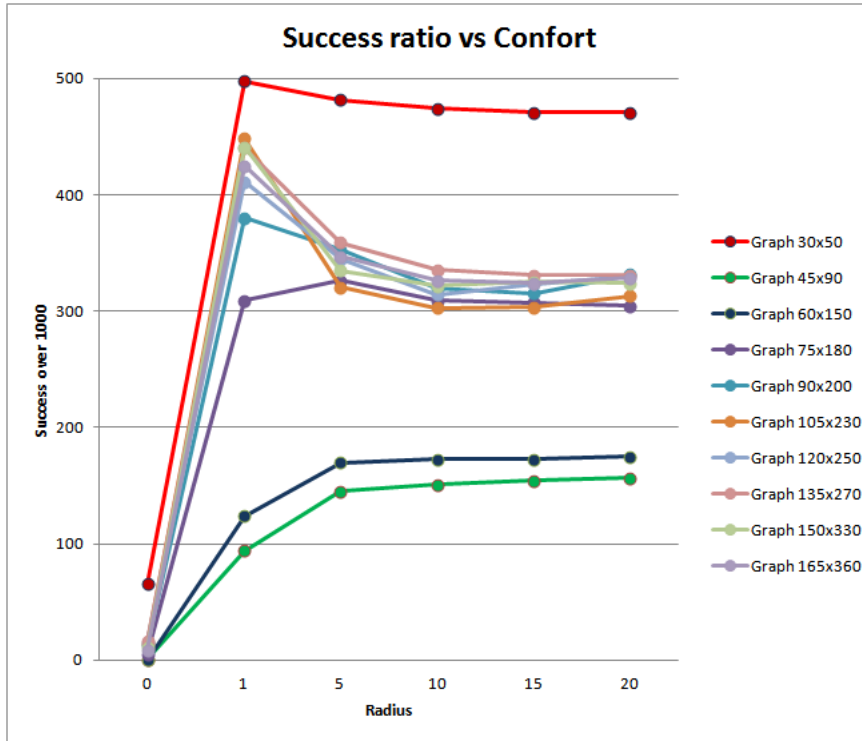


Figure 5.4: Average success ratio vs Comfort

system stops or other agent try to expel it from it's color. Without this parameter, the system would have the same problem as assigning the color before reaching to a goal, the system behavior would be unstable because the agents will be jumping form one goal to another. The problem of the local minimum is that an agent reaching a colorgoal stops inside it. If this color is incorrect for the agent, we need a mechanism to expel the agent towards another goal. That mechanism is Comfort.

In figure 5.4 we can see that the comfort is necessary for the system to get good results. Without if the algorithm fails. But which values is the best?. The system behavior is more or less stable from comfort value equal to 5 to comfort value equal to 20. That means that the comfort value is necessary to be bigger than zero, but is independent to the exact value. We have plot again the results in 3D to have another point of view in figure 5.5.

In figure 5.6 we can see average number of steps need to find a solution. As in the goal radius case, when the algorithm don't manage to find a solution the

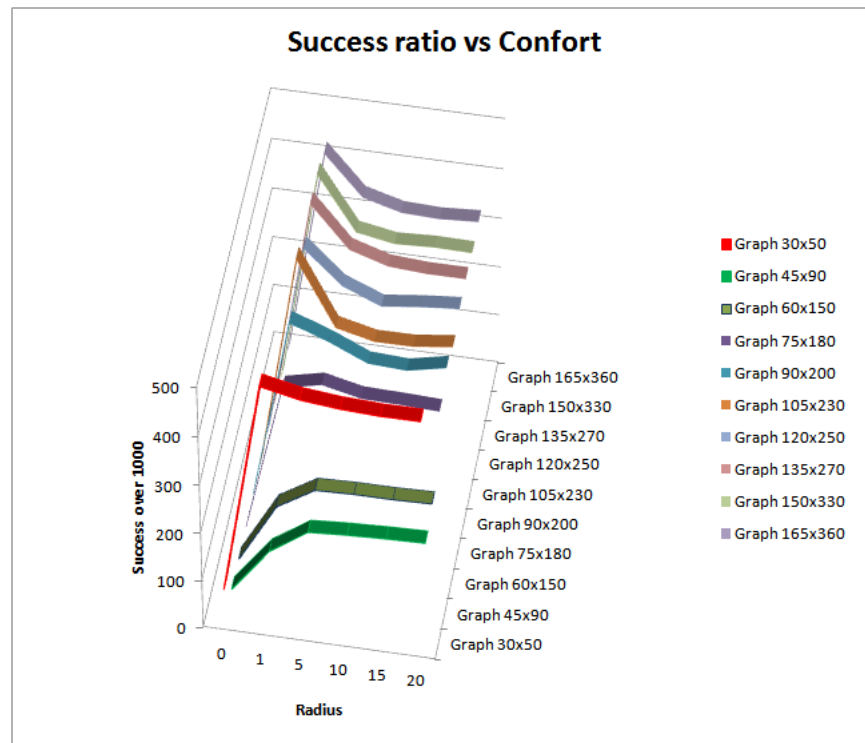


Figure 5.5: Average success ratio vs Comfort in 3D

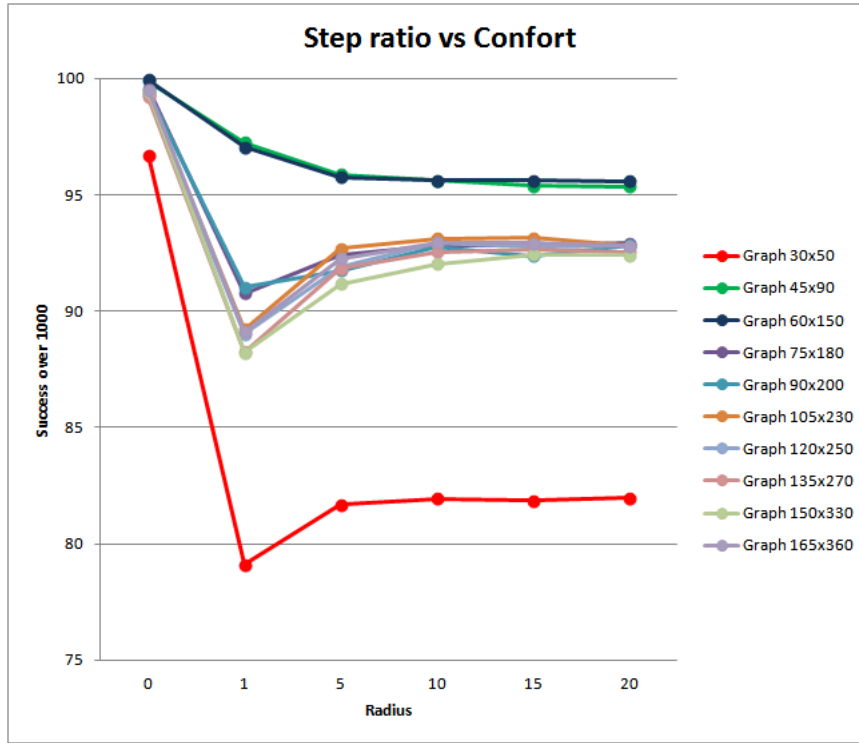


Figure 5.6: Average steps ratio vs Comfort

number of steps is 100. It is clear that if the comfort is zero, then the number of steps is almost 100. We can see that the number of steps with comfort equal to one are best, but it is not clear, because sometime, this is not true as we can see in graph families 45x90 and 60x150. When the comfort is five of above five, then the systems behavior is more stable. We have assume that the best comfort is five, and we have use it in our experiments. As we have said before, setting this parameter is done very quickly, and with this results, it worthless to worry about it.

5.4 Nom Parametric Tests

We want to show in a formal way that the qualitative conclusions obtained on visual inspection of the results plots are statistically significant. For that reason we apply some non parametric test to the result obtained. We are going to use the Friedman test [131, 132]. If the null hypothesis of this test is not comply

then we can use a post-hoc test like Nemenyi's [133].

5.4.1 Friedman test

The Friedman test is a non parametric test that was originally developed by the economist Milton Friedman. This test can be applied when we have n groups and k treatments to these groups. We order the results of applying the k treatment to each group in a table of n rows and k columns. Then we assign ranking to the k treatments to each row $r_{n,k}$ where the best result is assigned 1 and the worst result is assigned k . If there are ties then we assign an average value.

Then we have to calculate the average rankings of each treatment as:

$$R_k = \frac{\sum_{i=1}^n r_{i,k}}{n} \quad (5.2)$$

The null hypothesis indicates that all the treatments behavior. Under the null-hypothesis, which states that all the algorithms are equivalent and so their ranks R_k should be equal, the Friedman statistic is:

$$\chi_F = \frac{12n}{k(k+1)} \left[\sum_k R_k - \frac{k(k+1)^2}{4} \right] \quad (5.3)$$

This statistic follows a χ^2 stochastic distribution of Pearson [134] of $k - 1$ degrees of freedom. If the Friedman values is bigger than the null hypothesis then we can say that the treatment are statistically different so now we can make a post-hoc test of the treatments. If the Friedman values if smaller than the null hypothesis then we can't say that the treatments are statistically different, so all the treatments behavior are similar.

As the Friedman test sometimes if quite conservative, Iman and Davenport [135] introduce an improvement to the Frieman stochastic value.

$$\chi_{ID} = \frac{(n-1)\chi_F}{n(k-1) - \chi_F} \quad (5.4)$$

That follows a F of Fisher-Snedecor [136, 137] stochastic distribution with $k - 1$ and $(k - 1)(n - 1)$ degrees of freedom.

Graph\Radius	1	10	20	30	40	50	60	70	80
Graph 30x50	9	7	4	2	1	3	5	6	8
Graph 45x90	9	8	3	1	2	4	5	6	7
Graph 60x150	9	7	2	1	3	4	5	6	8
Graph 75x180	9	7	3	2	1	4	5	6	8
Graph 90x200	9	7	4	1	2	3	5	6	8
Graph 105x230	9	7	5	4	1	2	3	6	8
Graph 120x250	9	7	5	4	2	1	3	6	8
Graph 135x270	9	7	5	4	2	1	3	6	8
Graph 150x330	9	7	5	4	2	1	3	6	8
Graph 165x360	9	7	5	4	3	1	2	6	8
R_k	9	7.1	4.1	2.7	1.9	2.4	3.9	6	7.9

Table 5.2: Friedman ranking for Goal Radius

Graph\Radius	0	1	5	10	15	20
Graph 30x50	6	1	2	3	5	4
Graph 45x90	6	5	4	3	2	1
Graph 60x150	6	5	4	3	2	1
Graph 75x180	6	3	1	2	4	5
Graph 90x200	6	1	2	4	5	3
Graph 105x230	6	1	2	5	4	3
Graph 120x250	6	1	2	5	4	3
Graph 135x270	6	1	2	3	5	4
Graph 150x330	6	1	2	5	3	4
Graph 165x360	6	5	4	3	1	4
R_k	6	2.4	2.5	3.6	3.5	3.2

Table 5.3: Friedman ranking for Comfort

5.4.2 Friedman test to GSI

We have applied the Friedman test over the two parameters that we are testing. The Goal Radius and Comfort. In tables 5.2 and 5.3 we can see the ranking for each graph and value o goal radius and comfort and the average rankings R_k

5.4.2.1 Goal Radius

The Friedman value for goal radius is $\chi_F = 71.333$. The value of the square-chi with eight degrees of freedom and a probability of accepting the null hypothesis with 0.9 is $\chi^2 = 13.4$, with 0.95 $\chi^2 = 15.5$ is and with 0.99 is $\chi^2 = 20.1$. We can see that in the tree cases, the Friedman value is bigger than the null hypothesis so we can say that the goal radius values are statistically different and now we

can practice a post-hoc test.

But we are going to use the Iman-Davenport improvement. So new value is $\chi_{ID} = 74.077$. The new null hypothesis with 8 and 72 degrees of freedom and an accepting probability of 0.9, 0.95 and 0.99 are $F(8, 72) = 1.757$, $F(8, 72) = 2.07$ and $F(8, 72) = 2.769$. We can see that in the tree cases, the Iman-Davenport value is bigger than the null hypothesis so we can say that the goal radius values are statistically different

We can see that the goal radius equal to one is always get the worst behavior. The goal radius 80 is the second worst, and also the goal radius 10 and 70. These four rows have a stable behavior with all the data sets, so we don't need a parametric test to state that these values don't affect to the overall behavior of the system, so we have repeated the Friedman test without taking into account these columns. The new values for Friedman and Iman-Davenport are $\chi_F = 14.72$ and $\chi_{ID} = 5.241$. The null hypothesis for Friedman with 5 degrees of freedom and acceptance probability of 0.9, 0.95 and 0.99 are $\chi^2 = 9.244$, $\chi^2 = 11.8$, and $\chi^2 = 12.8$, and for Iman-Davenport with 5 and 45 degrees of freedom are $F(5, 45) = 1.98$, $F(5, 45) = 2.422$, and $F(5, 45) = 3.454$. In this case, the Friedman value and the Iman-Davenport value are next to chi square and F values but are still bigger so the goal radius parameter is statistically different even in this special scene.

5.4.2.2 Comfort

The Friedman value for comfort is $\chi_F = 28.457$. The value of the square-chi with five degrees of freedom and a probability of accepting the null hypothesis with 0.9 is $\chi^2 = 9.24$, with 0.95 $\chi^2 = 11.1$ is and with 0.99 is $\chi^2 = 12.8$. We can see that in the tree cases, the Friedman value is bigger than the null hypothesis so we can say that the comfort values are statistically different and now we can practice a post-hoc test.

Again, we are going to use the Iman-Davenport improvement. So new value is $\chi_{ID} = 11.889$. The new null hypothesis with 5 and 45 degrees of freedom and an accepting probability of 0.9, 0.95 and 0.99 are $F(5, 45) = 1.98$, $F(5, 45) = 2.422$ and $F(5, 45) = 3.454$. We can see that in the tree cases, the Iman-Davenport value is bigger than the null hypothesis so we can say that the comfort values are statistically different

We can see that the comfort equal to zero is always get the worst behavior. This row has a stable behavior with all the data sets, so we don't need a

parametric test to state that this value don't affect to the overall behavior of the system, so we have repeated the Friedman test without taking into account this column. The new values for Friedman and Iman-Davenport are $\chi_F = 9.84$ and $\chi_{ID} = 2.936$. The null hypothesis for Friedman with 4 degrees of freedom and acceptance probability of 0.9, 0.95 and 0.99 are $\chi^2 = 7.78, \chi^2 = 9.49$, and $\chi^2 = 11.1$, and for Iman-Davenport with 4 and 36 degrees of freedom are $F(4, 36) = 2.12$, $F(4, 36) = 2.65$, and $F(4, 36) = 3.95$. In this case, the Friedman with a probability of 0.9 states that this values are statistically independent, but with a probability bigger than 0.95, the null hypothesis is accepted so the comfort values are not statistically independent, so being different from zero is enough for this parameter. Applying the Iman-Davenport test, only when the probability of acceptance is bigger than 0.99 the null hypothesis is accepted and can conclude that the parameters are not statistically independent.

5.4.3 Post-Hoc test: Nemenyi's test

As we can proof that that goal radius and comfort values are statistically independent, we are going to pass a Post-hoc test. This test consist of looking at the data when all the experiments have concluded, and try to find patterns that were not specified a priory. We use the Nemenyi's test [133]. This test is similar to the Tukey's test [138] and is used when all classifiers are compared to each other. The performance of two classifiers is significantly different if the corresponding average ranks differ by at least the critical difference CD:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6n}} \quad (5.5)$$

where critical values q_α are based on the Studentized range statistic divided by $\sqrt{2}$.

In figure 5.7 we can see the result of applying the Nemenyi's test to the goal radius, using all the nine values and with an acceptance of the 90%. The CD values is 3.497. There are four groups that join different values of the goal radius. In figure 5.8 using all the nine values and an acceptance of 95%, the CD value is 3.802, bigger. We have now five groups, and the diagram is a big mess. The information that we can extract from this diagram is very small. In figure 5.9 using all nine values and an acceptance of 99%, the CD value is 4.399, the biggest. Here we come back to the four groups, but still this result doesn't help to much.

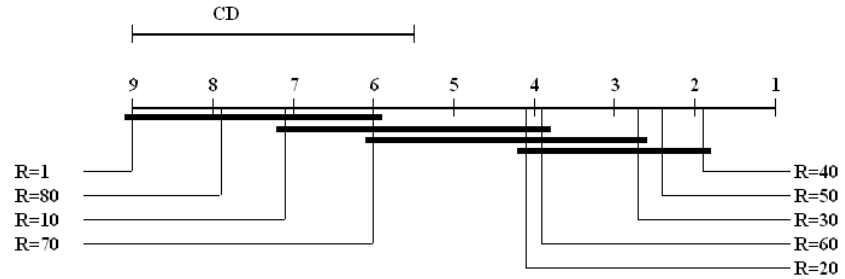


Figure 5.7: Nemenyi's diagram for 9 goal radius and 90% of acceptance

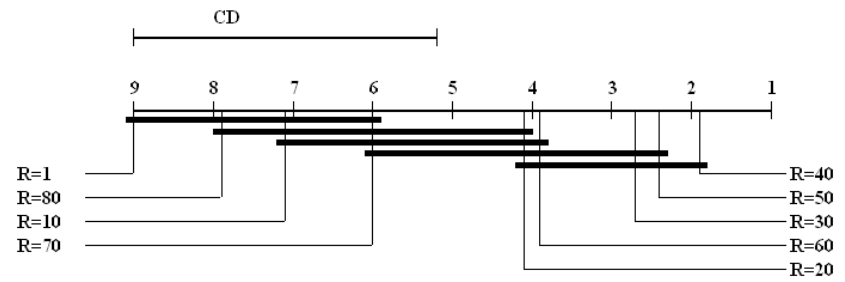


Figure 5.8: Nemenyi's diagram for 9 goal radius and 95% of acceptance

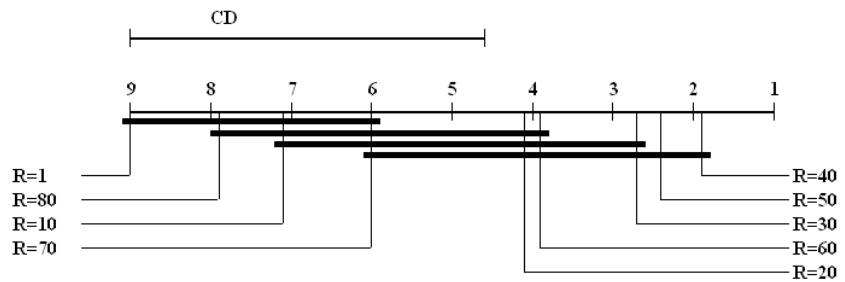


Figure 5.9: Nemenyi's diagram for 9 goal radius and 99% of acceptance

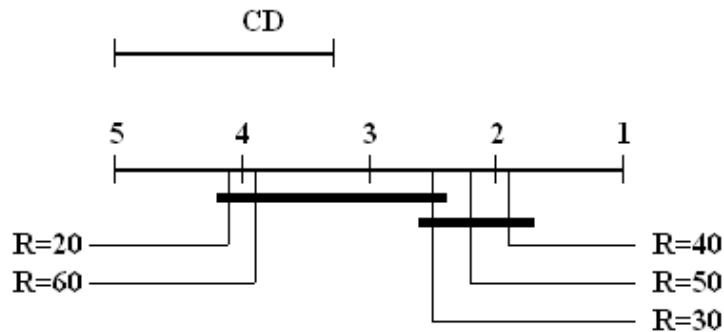


Figure 5.10: Nemenyi's diagram for 5 goal radius and 95% of acceptance

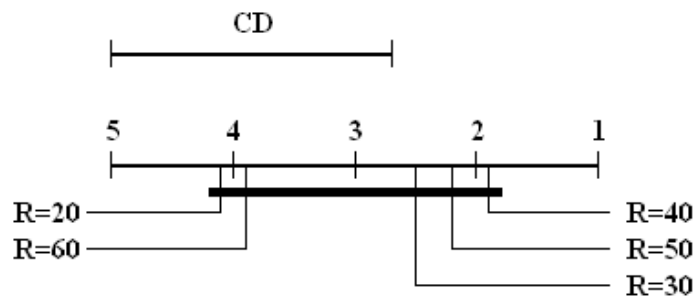


Figure 5.11: Nemenyi's diagram for 5 goal radius and 99% of acceptance

In figure 5.10 we can see the Nemenyi's test result on the goal radius, but now without taking into account the goal radius equal to 1, 80, 70 and 10, in the same way that we have made with the Friedman test. We have now only five radius and with an acceptance of 95% the CD values is 1.93. There are two groups, but all the values of the goal radius are more or less linked. In figure 5.11 using five values and an acceptance of 99%, with a CD values equal to 2.3, it is more clear that the goal radius between the values 20 and 60 don't affect too much to the behavior of the algorithm. We have only one group. Although the Friedman test tell us that the goal radius values are statistically independent, the Nemenyi's test shows that it's not true, at least for the central values, but even including the outliers, the Nemenyi's test is different respect to Friedman.

In figure 5.12 we can see the Nemenyi's test result on the Comfort values,

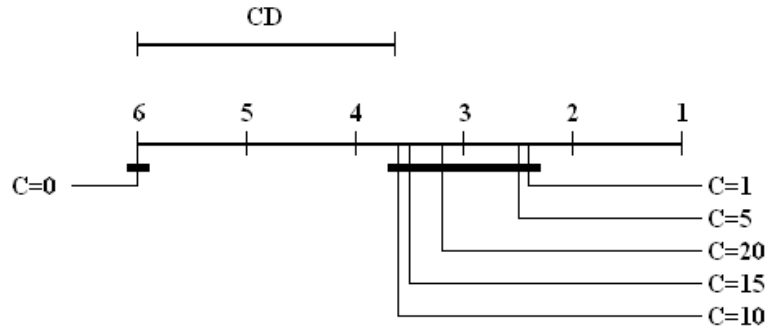


Figure 5.12: Nemenyi's diagram for 6 comfort values and 95% of acceptance

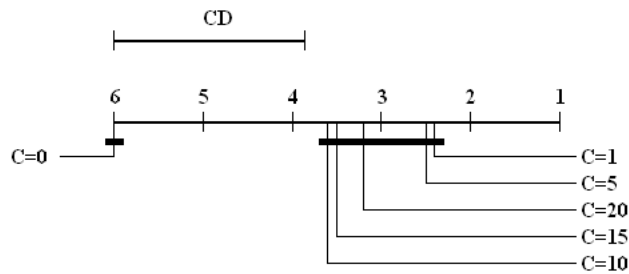


Figure 5.13: Nemenyi's diagram for 6 comfort values and 99% of acceptance

using the six values and with an acceptance of 95%, the CD values is 2.384. It is very clear that the value of comfort zero, and all the values above zero, are disconnected in the diagram. We can say that the comfort must be different to zero, but it doesn't matter the exact value. In figure 5.13 using six values and an acceptance of 99%, the CD value is 2.816. The CD values is bigger but the result don't change respect to the 95% acceptance.

In figure 5.14 we have made an quick experiment. As long as the Friedman test has said that the comfort value, without taking into account the zero value, are statistically dependent. We don't need to use a Post-hoc test as Nemenyi's, but we have applied the test to the five values and an acceptance of only 90%. The CD values is 1.739. The diagram shows that there is only one group for all the values, so the Friedman test was right, the comfort values bigger than zero are statistically dependent.

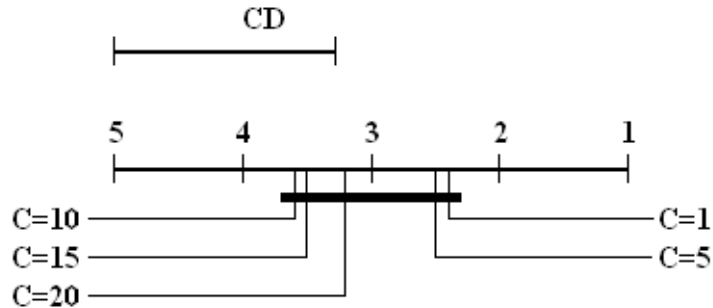


Figure 5.14: Nemenyi's diagram for 5 comfort values and 90% of acceptance

5.5 Concluding remarks

In conclusion, we can use a color goal radius between 30 and 60, because the non parametric test results said that the results after setting goal radius in this range are statistically independent, we are going to fix the goal radius equal to 30 for the additional computational experiments. The value 30 is because the algorithm works very fast with this value in small graphs, and we are going to work over small graphs to compare our results with slow methods as Tabu Search or ACO methods. The comfort value must be bigger than zero. We are going to fix the comfort value equal to 5, because when the comfort is 5, the behavior of our algorithm looks more stable.

Chapter 6

Graph Coloring Results

In this chapter we report an exhaustive comparison of performance results between the proposed algorithm and state of the art algorithms. We run the algorithms on the experimental benchmark graph families described in Appendix A following two basic strategies. (a) the chromatic number is set to the already known chromatic number, and (b) applying a sequential strategy to find the chromatic number.

The structure of the chapter is as follows: Section 6.1 describes the basic elements of the experimental design. Afterwards the results on the specific graph families are reported: trees and bipartite graphs in Section 6.2, Kuratowski graphs in Section 6.3, Mizuno graphs in Section 6.4, KRG graphs in section 6.6, DIMACS graphs in section 6.6. Section 6.7 reports experiments on the sequential determination of the chromatic number. Finally, section 6.8 summarizes the results giving some conclusions.

6.1 Experimental design

We have prepared a big bank test to assess that our GSI algorithm get good results is a wide range of problems. We start testing on the Mycielski graphs [90]. These graphs are easy to solve, but they are a good start point. With Mycielski we have accurately tuned the most critical parameters of our algorithms, the goal radius and comfort maximum value. With the parameters obtained with previous test we have applied seven GCP solving algorithms:

1. A Backtracking greedy algorithm (BT).

2. Breaz [34] famous DSATUR algorithm (DS)
3. An stochastic simulated annealing (SA) [130].
4. A Tabu search (TS) [54].
5. And three Swarm Intelligence based algorithms:
 - (a) Ant Colony Optimization (ACO) [139]
 - (b) Particle Swarm Optimization (PSO) [20]
 - (c) Gravitational Swarm Intelligence (GSI) [119]

The first two algorithms are deterministic so they run only once on each graph. For the rest five heuristics we have repeated the algorithm execution 30 times for each method and graph. As the GCP is NP-Complete, for some graphs we would need a lot of time (perhaps years) to solve them so we have limited them to an amount of steps depending on the complexity of each algorithm.

- The BT and DS single steps are computationally light, therefore we have allowed them 10.000.000 steps.
- The SA with a medium complexity step, we have allowed 100.000 steps.
- The TS, PSO and our GSI have a maximum of 10.000 steps, because each iteration of PSO and GSI are complex algorithms that need a lot of time for each step and TS because it needs a lot of memory to allocate the Tabu list and it is also quite slow.
- For the ACO we only allow it 1.000 steps because, even though it has the same complexity of Swarm algorithms, the number of agents (ants) grows very fast with the size of the graphs making this algorithm the slowest because it needs more mathematical calculus than any other.

We have obtained results on the benchmark graph families described in Appendix A: tree and bipartite graphs, kuratowski based planar graphs, Mizuno's method 3-colorable graphs, new developed graphs called KRG and finally well-known DIMACS graphs. Success is measured as the number of times that an algorithm obtains a valid coloration of the graph.

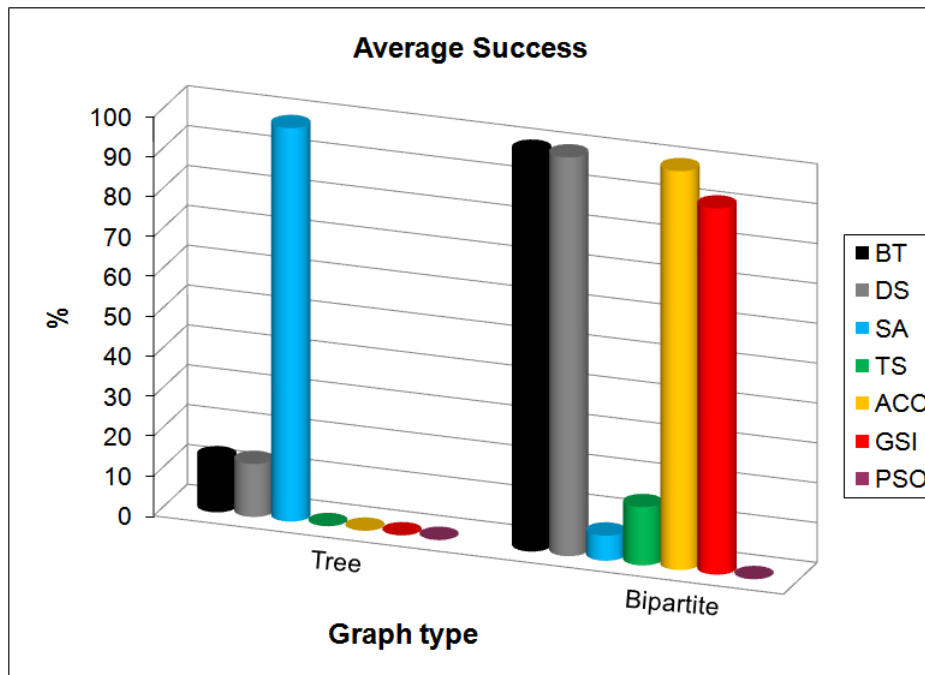


Figure 6.1: Trees and bipartite success ratio

6.2 Trees and bipartite graphs

A tree is a connected graph without cycles. A bipartite graph is a graph that does not contain any odd-length cycles. We have test over these graphs because their chromatic number is 2 and we assume that this graphs are easy to color. We have use 30 different trees and 30 different bipartite graphs of 100 nodes.

As we can see in figure 6.1 the trees are more difficult as we expected. The deterministic algorithms rarely can find the solution and the stochastic algorithm except the Simulated Annealing, all of them fail solving these graphs. The SA reached a 100% success ratio for these graphs.

The bipartite graphs meet better our assumptions. The PSO is the only algorithm that fails in all the experiments. The behavior of the SA is strange for its poor performance compared with previous results. The TS also gets poor results. The other four algorithms including our GSI give 100% of success (GSI only 90%). These results show that even the simplest graphs can be hard to solve, and that our algorithm, as we will show later, is not always the best.

In figures 6.2 and 6.3 we can see the average time and number of steps for

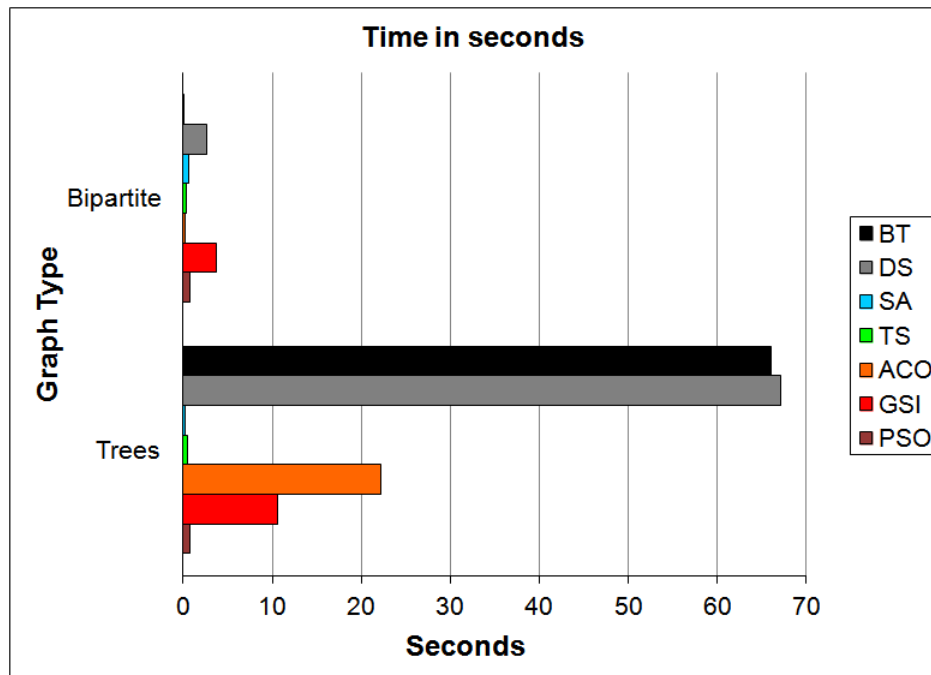


Figure 6.2: Trees and bipartite average time in seconds

each algorithm. The number of steps is normalized to 1.000. We can see the big difference between the time and the steps. The DSATUR time for trees is almost 70 seconds and for the PSO is only a pair of seconds, but both algorithm used almost the same number of steps. This is one of the reason of using steps instead of seconds, besides abstracting from the computer architecture.

6.3 Kuratowski based planar graphs

We have generated planar graphs using the Kuratowski theorem because we have an upper bound of the chromatic number, that is 4. We have built our own graph generator. We have generated 25 families of 10 graph each family. This families are grouped by the number of nodes, starting from 10 and increasing the number of nodes adding 10 more nodes until reaching to a family with 250 nodes. The number of edges have been calculated $E = n * 2$.

In figures 6.4 we can see that instances with few nodes are easy to solve and all the algorithms can cope with them, but when the number of nodes grows, only our GSI algorithm is robust against the size of the graph. Again

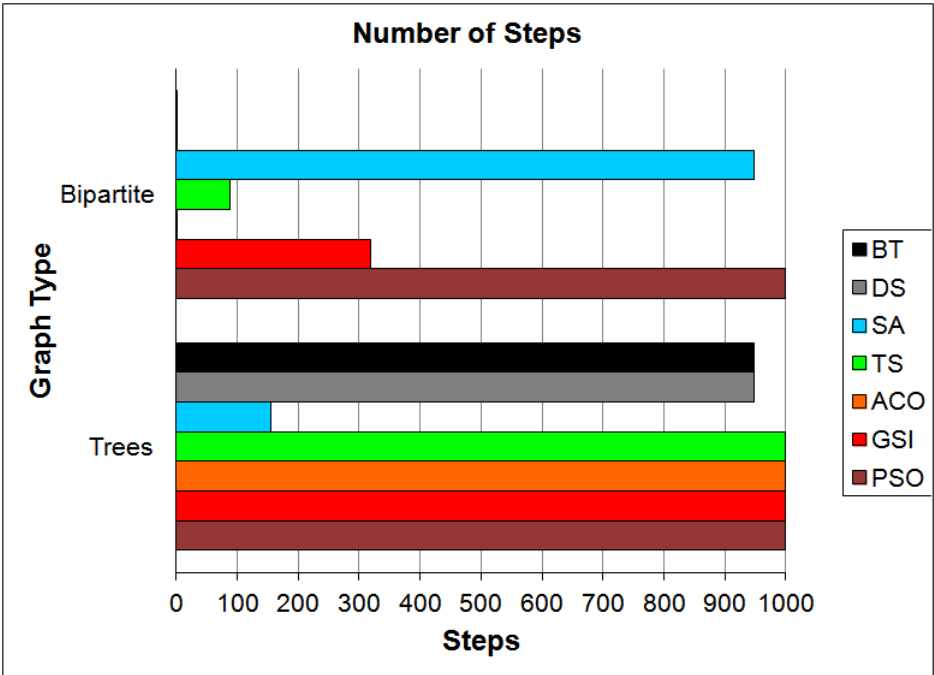


Figure 6.3: Trees and bipartite average number of steps

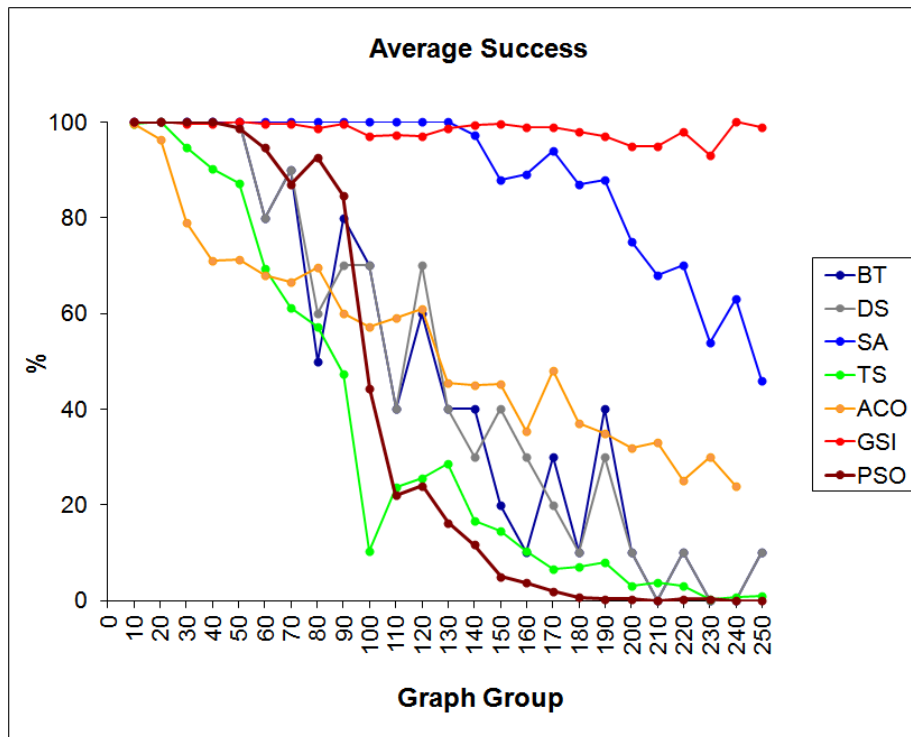


Figure 6.4: Kuratowski based graphs success ratio

the SA gets good results but it also fails when the number of nodes goes over 180 nodes. A very important feature of the Swarm based algorithms is their scalability property, because we divide the problem into small tasks that the agents can carry on with a low effort. The most wealthy part of these algorithm is to join the information of all the agents and decide if the problem is solved or not. In the results we can see that our GSI algorithm find the solution of the problem near 100% in all the cases, but the PSO fails below 20% with graphs of only 100 nodes. This means that PSO fall into local minimum easily because the agents have memory about their local best and the global best and with only four colors there are little variants. In the GSI algorithm the agents only worried about themselves so is more difficult to fall in local minimum. The ACO algorithm reduces it's accuracy near linearly, without big jumps or falls. As we can say the number of ant or agents grows faster than GSI or PSO so the size of the problems affects directly to it's performance, but with a parallel programing this algorithm can get better results.

In figure 6.5 it can be seen that the time grows directly proportional to the size of the graphs. The values of the TS in green illustrate this very clear. But not all the algorithms performance grow the same. GSI, PSO and SA time grow slowly compared with the other algorithms. For the GSI algorithm it has a simple explanation, with a big success ratio the time is small but this is not true for PSO or even SA. The relation between success and steps is evident in figure 6.6. This figure is like the figure 4 invested.

6.4 Mizuno's 3-colorable

Mizuno had developed a method to built hard 3 colorable instances of graphs. These instances are a good test because you can try over very difficult graphs with a known chromatic number. A small chromatic number can made some algorithm run faster and find the solution in a short time. The process of building a graphs consist of merging two special graph instances call MUGS (there are 12 MUGS). And then apply some actions to ensure that the new graph is 3 colorable. The process can be replied as many times as you wanted adding different MUGS to the results.

The figure 6.7 show the results over 25 families of 10 graph instances. The first family is the simples merging two MUGS. The next family is two iterations of the methods, until applying 25 iterations of the methods. As the MUGS num-

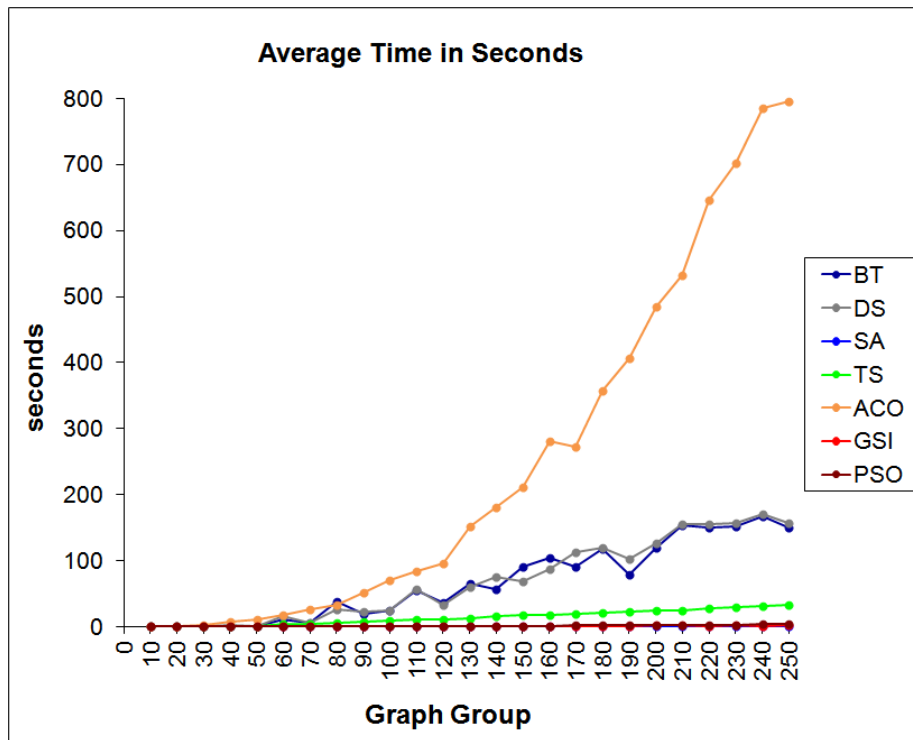


Figure 6.5: Kuratowski based graphs average time in seconds

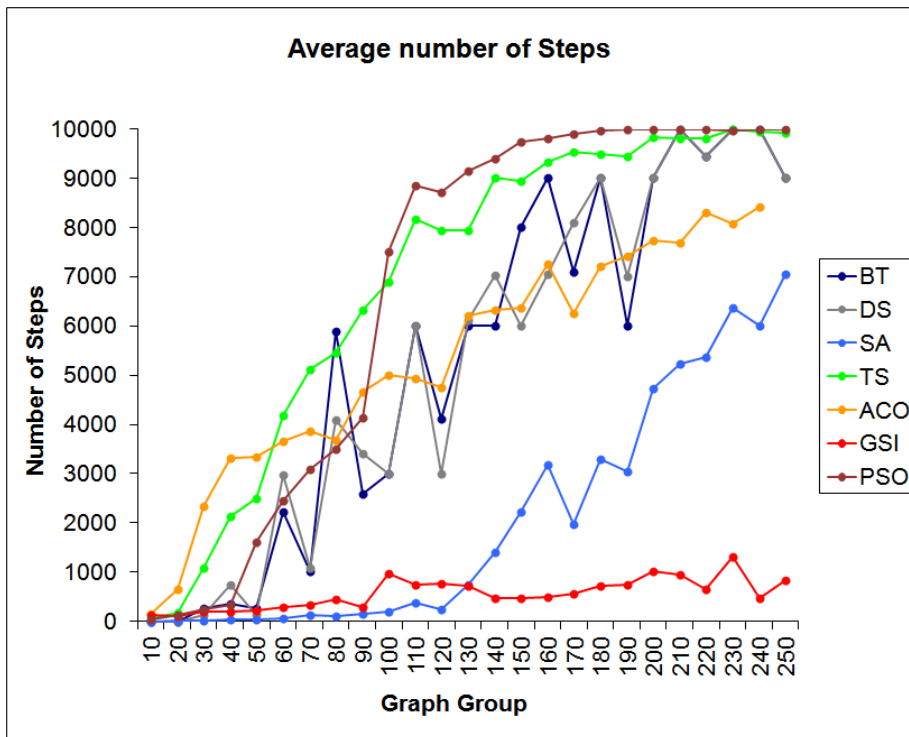


Figure 6.6: Kuratowski based graphs average number of steps

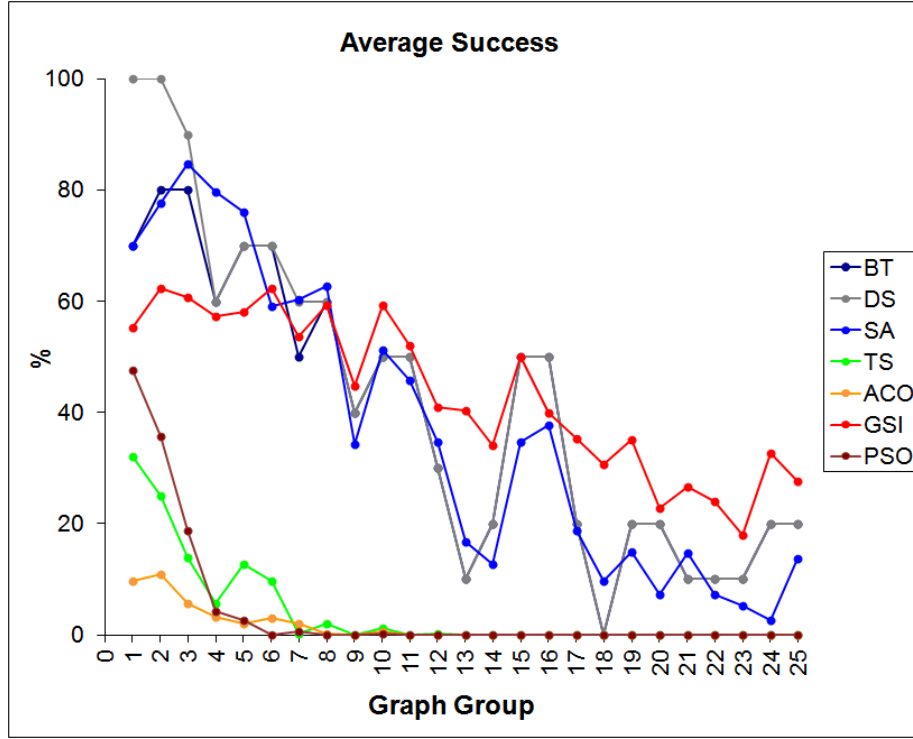


Figure 6.7: Mizuno's graphs success ratio

ber of nodes and nodes is not the same for each MUGS, the resulting instances are no homogeneous, like Kuratowsky's graphs.

In the figure we can different two groups of behaviors. PSO, TS and ACO start with very poor results and very quickly the success ratio fall near zero. The BT and DS non deterministic algorithm start as the best method for small instances and fall slowly towards zero. These two algorithms only have differences is small graphs, with large graph the have the same results. SA has similar results as BT and DS, and like them it has big jumps with different families.

Our GSI algorithm start with average result under the 60 % of accuracy, but it's performance decreases more slowly than all the others, achieving the best result in medium and large instances. This behavior comes again by the Swarm approach.

In figure 6.8 we can see the results in seconds. The BT and DS even though obtain good results, the computational time is very big. The ACO time is also big as we expected. PSO and TS are very quick and stable, because the perform

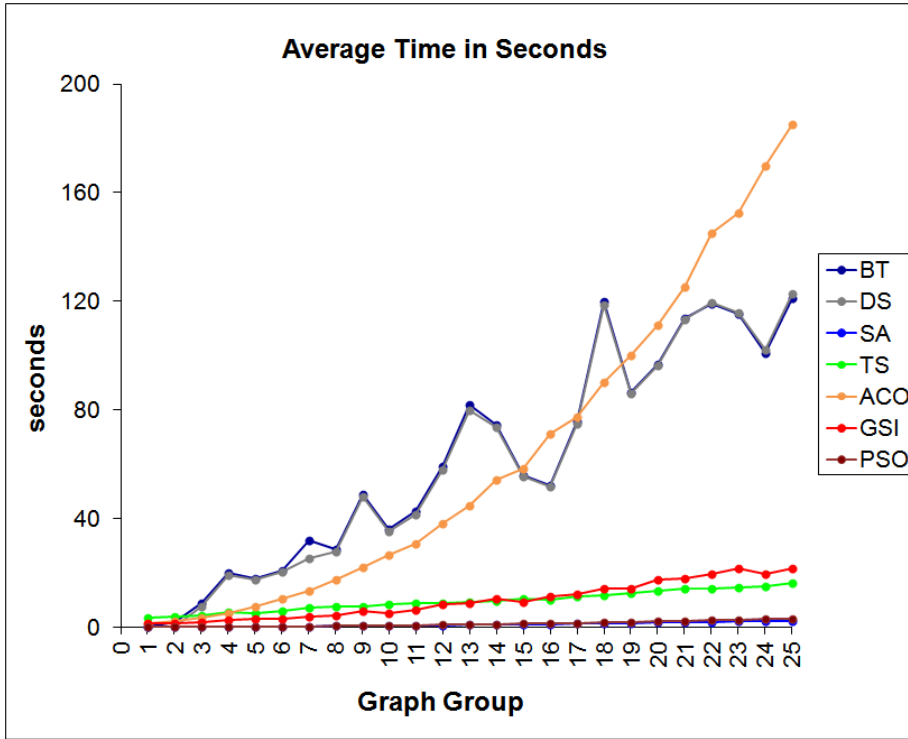


Figure 6.8: Mizuno's graphs average time in seconds

bad results from the beginning and the success ratio don't affect them. SA is again the fastest and GSI this time penalized the dimension of the instances. As we have said the GSI algorithm has been generated to be a parallel algorithm and a parallel implementation will show better results in time.

Figure 6.9 show the results in steps. The figure as similar as the success ratio but changing the y axis. The only remarkable thin is that The number of steps of the deterministic algorithms is very similar as GSI's although the GSI obtain better results.

6.5 KRG graphs

The KRG graphs are a special family of graphs inspired in Kuratowsky's theorem and developed ad-hoc for this research. The idea is to built planar graphs in a first step and then add a clique of cardinality N been $N \geq 4$. With this assumption, we can ensure that the chromatic number of the graph is N . This is

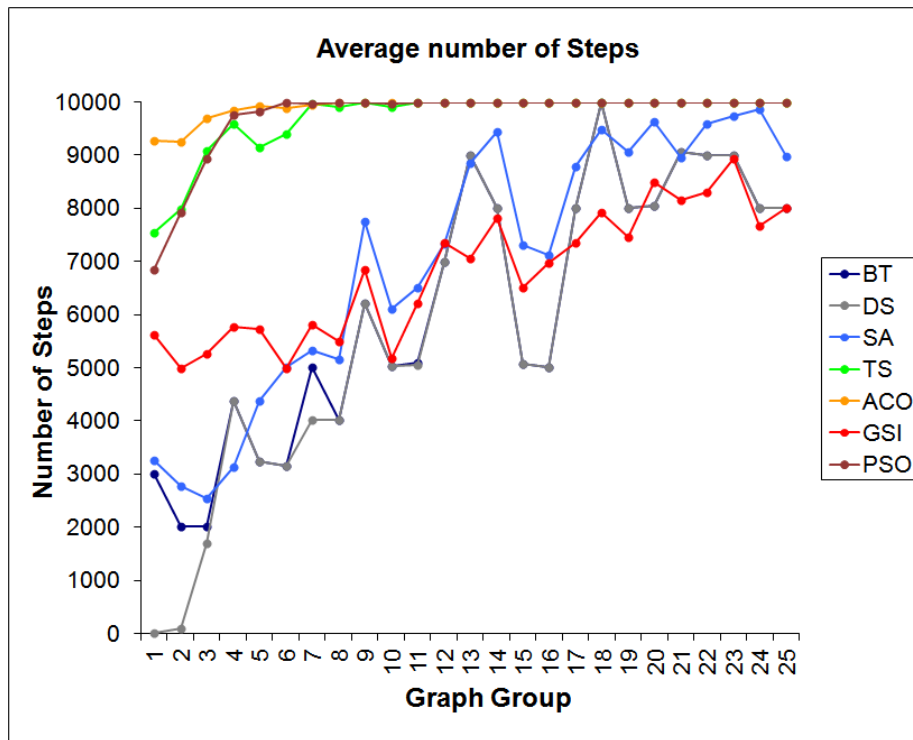


Figure 6.9: Mizuno's graphs average number of steps

true because through the Graph Theory we know that planar graph's chromatic number is 4, and also that the lower bound of the chromatic number of a graph is the graph's biggest clique. Then our KRG graphs lower bound is N , and the upper bound of the graph G is 4 so the chromatic number of the graph G is N .

We have prepared two sets of experiments. The first one starts with three families of graphs with 100 nodes and 300 edges, 400 edges and 500 edges. The maximum number of edges E of a planar graph $G=\{V,E\}$ is $E=V*3-3$. The KRG graphs don't need to fulfill this rule, so we can have a 100 node graph with more than 297 edges. For each graph family we have applied $N=4$, $N=5$, $N=6$ and $N=7$ so finally we have 12 families of 10 graphs. We have used these families to show empirically that the KRG graphs are good for testing.

In figure 6.10 we can see three graphics. The first one shows the four chromatic number graphs of 300 edges, second with 400 edges and finally with 500 edges. The first thing we can see is that the complexity of KRG graphs decreases inversely with the chromatic number. The success ratio of $N=7$ graphs is bigger than $N=4$ graphs. The second thing is that the KRG graphs chromatic number is equal to N .

The results are quite different from previous experiments. The GSI is almost always the best algorithm and in most cases near 100% of success. The PSO algorithm results are the second best results even winning out GSI algorithm in KRG_100_300_7 family. The ACO results are pretty good in the 300 and 400 edge families, but the Swarm algorithms are the best accurate in this experiment. The results in the 500 edge family are quite different, the ACO and SA can't find a solution in any instance. The SA is usually a good algorithm, here it is the worst. The TS, BT and DS behavior is similar as other tests.

In figure 6.11 we can see that the time in seconds needed for TS and ACO is very big compared with the other methods. The BT, DS and PSO are average in computational time. SA as we expected is the fastest in almost all the instances, but our GSI algorithm is very slow with complex graphs with small chromatic number but near as fast as SA in graphs with a big chromatic number, but even better in some experiments.

The steps graphics behavior is similar as success graphics, changing the y axis. The results are in figure 6.12.

With these experiments we have shown that the KRG graphs really comply with our assumption. Now we are going to test with bigger graphs to extend the results obtained. We have built 4 new families of 500 nodes and 1400 edges,

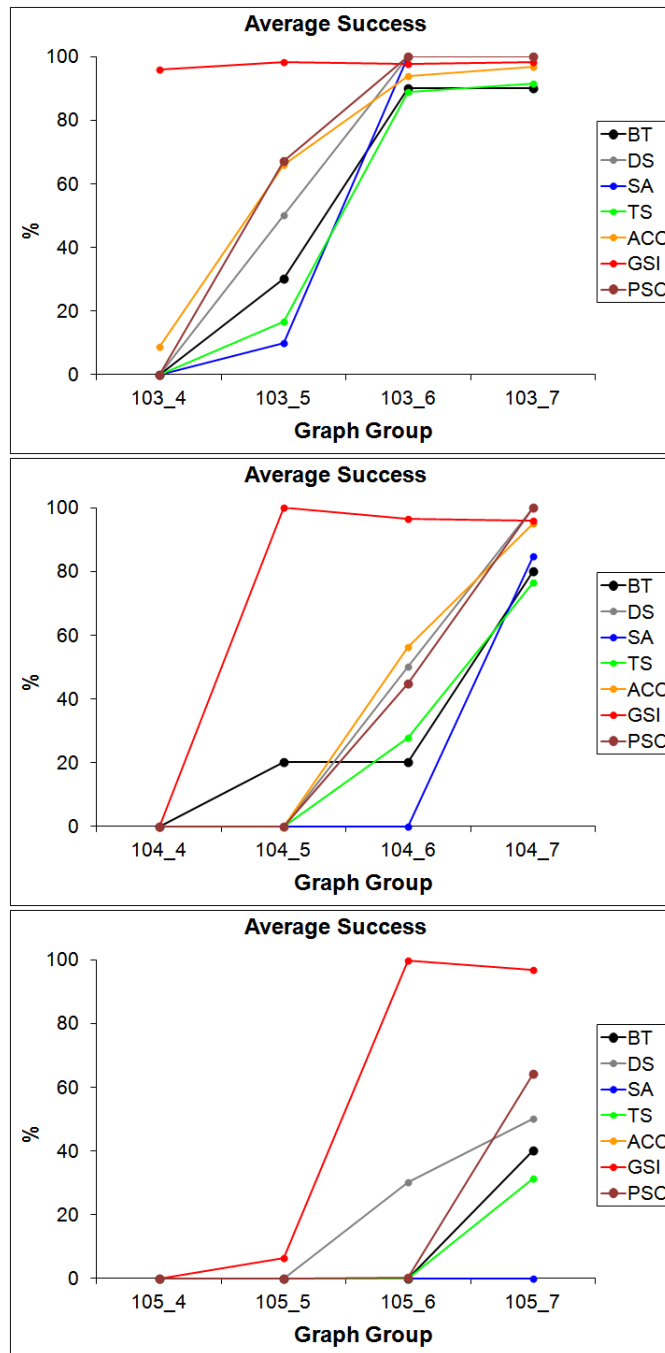


Figure 6.10: KRG success ratio

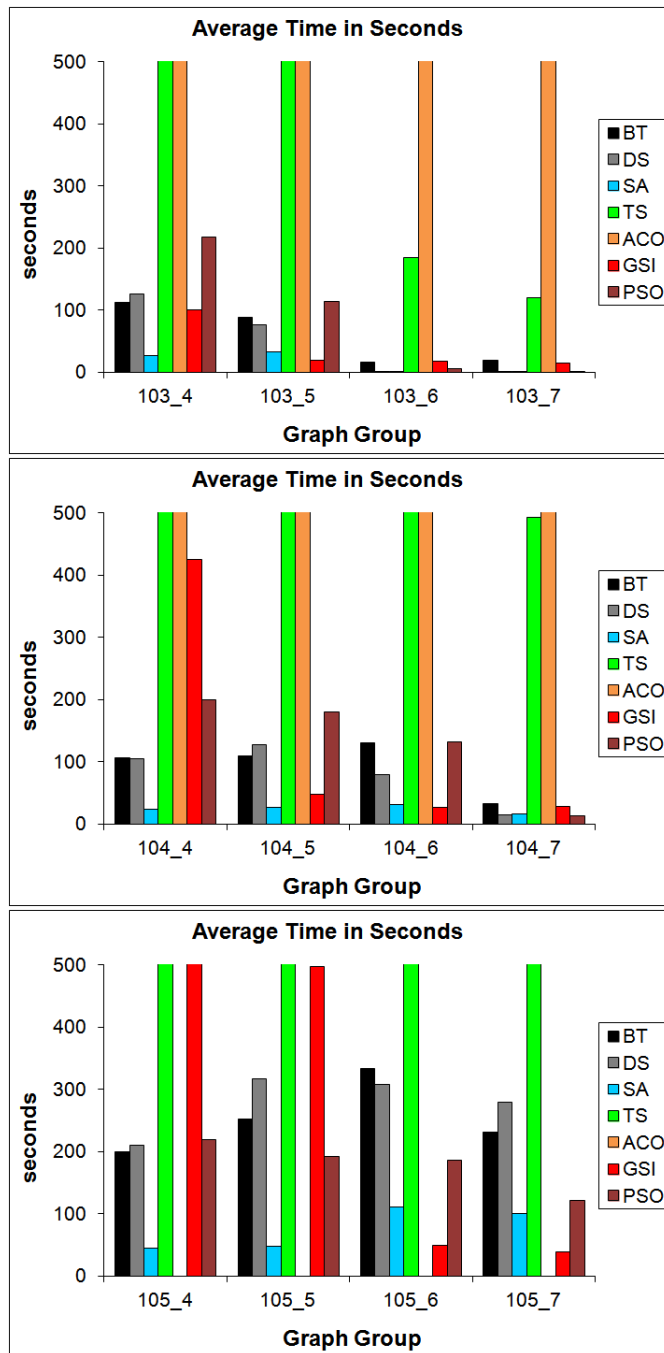


Figure 6.11: KRG average time in seconds

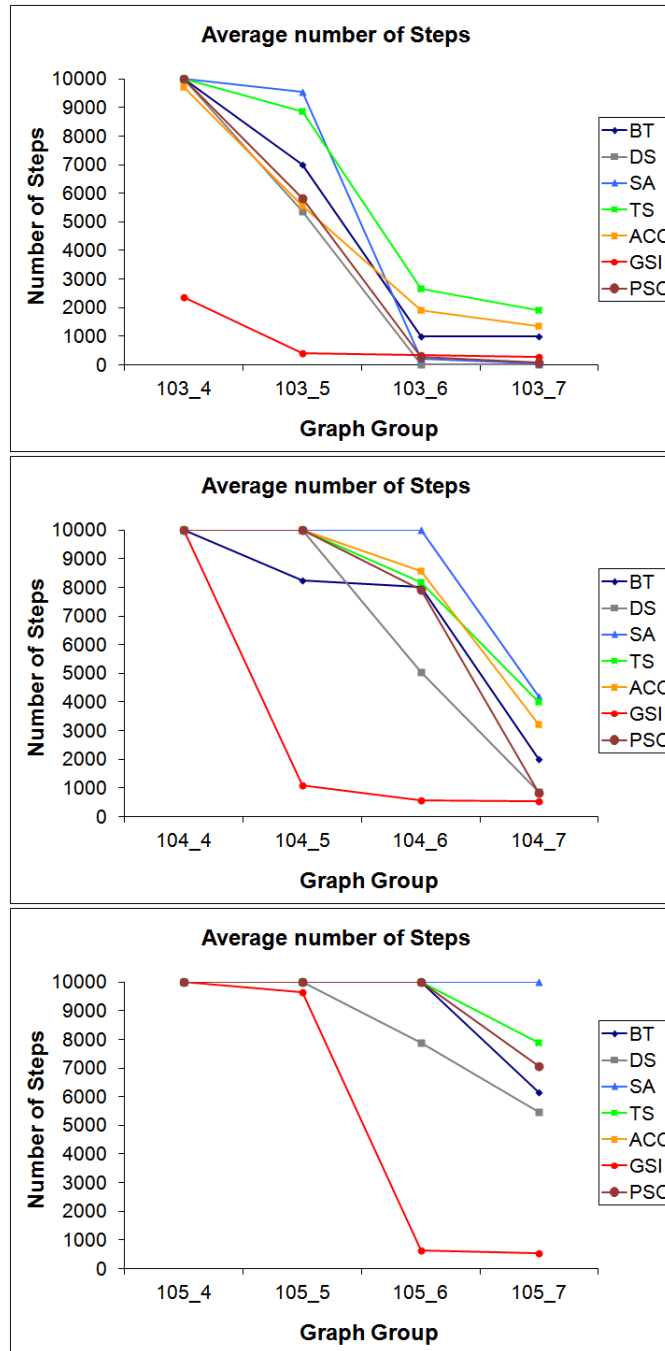


Figure 6.12: KRG average number of steps

100 nodes and 2800 edges, 1500 nodes and 4200 edges and 2000 nodes and 5600 edges. We have again applied $N=4$, $N=5$, $N=6$ and $N=7$ so we have test over 16 graphs families.

The results are significantly different. The results of the ACO and TS are worst, because now we have big graphs, exactly the scenery where these algorithm have problems. All the algorithms fails solving the $N=4$ instances, except our GSI, and even with problems. This is because these instances are very hard.

There is a disappointing result with graphs with $N=6$ and $N=7$. Our algorithm is the best with $N=4$ and $N=5$, gets good results with $N=6$ and $N=7$ but it is beaten by DS, BT and ACO, and also by the PSO algorithm in the 500 and 100 node's families. The results are in figure 6.13

In figure 6.14 we have the computational time. Here again we get a surprise. Our method, that we expected to be very fast appears more slowly. Even the TS is faster. We have to say that the result obtained is not bad compared with ACO and DS, but the other four methods have managed to beat it in different tests.

In figure 6.15 we see more differences with previous experiments. Now the success graphic and steps graphs are no as similar as before. The results of the GSI algorithm are better having into account only the steps, with a difference with the best algorithms in $N=6$ and $N=7$. The PSO algorithm needs always more steps, even thought get better results in some tests.

6.6 DIMACS graphs

The DIMACS challenge in 1993 [84] was a great moment in the GCP community because:

1. It was presented a format of representing graphs that has become to a standard.
2. In that challenge appears a group of graphs for testing that cover a wide range of graph types, that become a point of union in the research of graph coloring, because that graphs appears as a reference in any paper since then.

We have test over our own graphs and it is difficult to compare with the literature, so we are going to go on testing with these graphs.

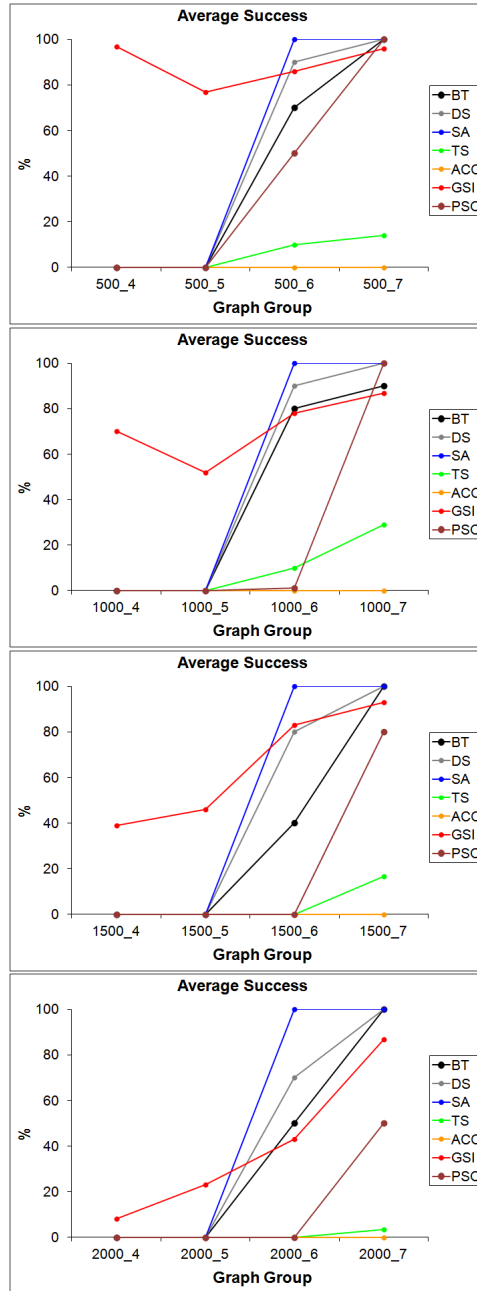


Figure 6.13: Big KRG success ratio

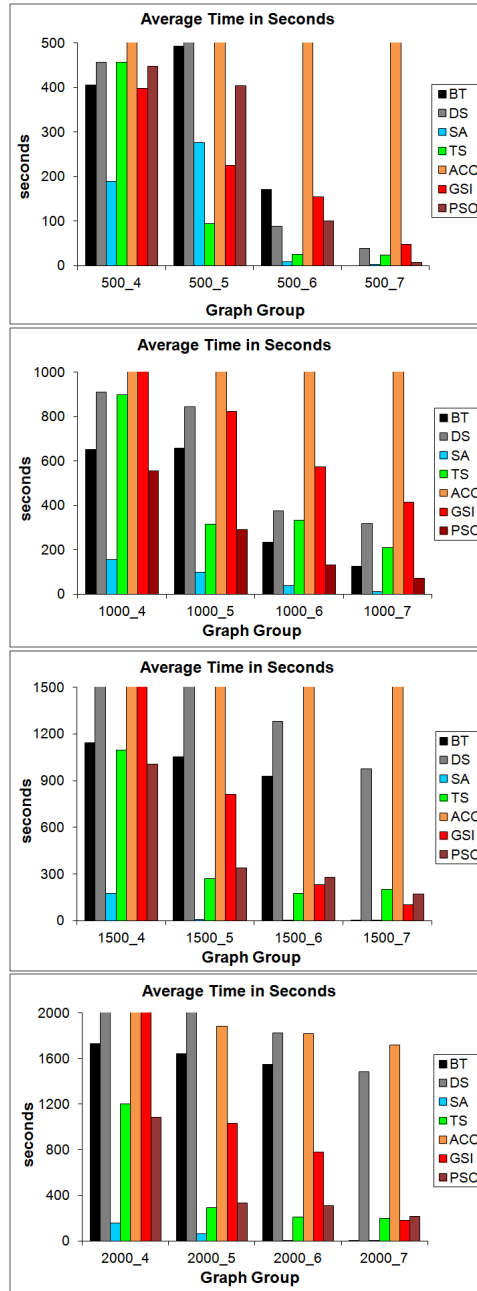


Figure 6.14: Big KRG average time in seconds

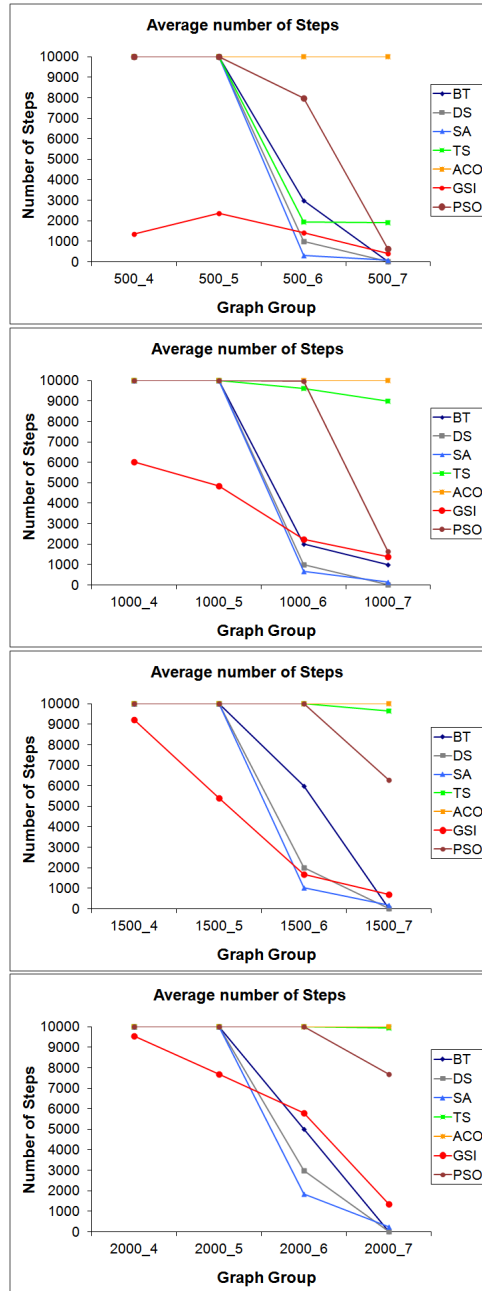


Figure 6.15: Big KRG Average number of steps

The first test has been over the Mycielski graphs because there are small and easy to test.

Then we have choose the books graphs. These graphs are a group of graphs from the Stanford GraphBase (SGB). The construction of this graphs is:

Given a work of literature, 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: 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).

We have also test the miles graphs, members of the SGB family. These graphs are similar to geometric graphs in that nodes are placed in space with two nodes connected if they are close enough. These graphs, however, are not random. The nodes represent a set of United States cities and the distance between them is given by by road mileage from 1947.

Another member of SGB family are the Queen Graphs. Given an n by n chessboard, a queen graph is a graph on n^2 nodes, each corresponding to a square of the board. Two nodes are connected by an edge if the corresponding squares are in the same row, column, or diagonal. Unlike some of the other graphs, the coloring problem on this graph has a natural interpretation: Given such a chessboard, it is possible to place n sets of n queens on the board so that no two queens of the same set are in the same row, column, or diagonal if and only if the graph has coloring number n . Martin Gardner states without proof that this is the case if and only if n is not divisible by either 2 or 3. In all cases, the maximum clique in the graph is no more than n , and the coloring value is no less than n .

Finally, we have test some graphs from CAR family, the fullins graphs. These graphs are a generalization of myciel graphs with inserted nodes to increase graph size but not density.

We have test five BOOKS graphs, five miles graph and 6 queens graphs, all of them of the SGB graph family and fourteen graphs of the CAR graph family. In the next table we show the basic layout of the test graphs, the number of nodes, number of edges, the density that is $D = \frac{2*|E|}{|V|*(|V|-1)}$, and also the chromatic number. We have shown the results in table 6.1.

The two first columns are the graph name and the chromatic number given by the authors in DIMACS [84].

Table 6.1: Layout of the experimental graphs

Graph	#Nodes	#Edges	Density	#K
myciel3	11	20	0.363636364	4
myciel4	23	71	0.280632411	5
myciel5	47	236	0.218316374	6
myciel6	95	755	0.169092945	7
myciel7	191	2360	0.130063378	8
anna	138	493	0.052152756	11
david	87	406	0.108527132	11
homer	561	1629	0.010370512	13
huck	74	301	0.111440207	11
jean	80	254	0.080379747	10
miles250	128	387	0.047613189	8
miles500	128	1170	0.14394685	20
miles750	128	2113	0.259965551	31
miles1000	128	3216	0.395669291	42
miles1500	128	5198	0.639517717	73
queen5_5	25	160	0.533333333	5
queen6_6	36	290	0.46031746	7
queen7_7	49	476	0.404761905	7
queen8_8	64	728	0.361111111	9
queen8_12	96	1368	0.3	12
queen9_9	81	2112	0.651851852	10
1-FullIns_3	30	100	0.229885057	3
1-FullIns_4	93	593	0.138616176	4
1-FullIns_5	282	3247	0.08195149	5
2-FullIns_3	52	201	0.15158371	4
2-FullIns_4	212	1621	0.07247608	5
2-FullIns_5	852	12201	0.033655517	6
3-FullIns_3	80	346	0.109493671	5
3-FullIns_4	405	3524	0.043075419	6
3-FullIns_5	2030	33751	0.016388475	7
4-FullIns_3	114	541	0.083993169	6
4-FullIns_4	690	6650	0.027975852	7
4-FullIns_5	4146	77305	0.008996711	8
5-FullIns_3	154	792	0.067226891	7
5-FullIns_4	1085	11395	0.019376945	8

Table 6.2: Results of SGB graphs

Graph	#K	BT	DS	SA	TS	ACO	PSO	GSI
anna	11	0	0	3	17	0	0	96
david	11	0	100	3	0	0	0	93
homer	13	0	0	0	0	0	0	100
huck	11	100	100	100	80	0	100	90
jean	10	100	100	100	80	27	96	93
miles250	8	0	0	17	0	0	7	100
miles500	20	0	0	0	0	0	0	66
miles750	31	0	0	0	0	0	0	37
miles1000	42	0	0	0	0	0	0	100
miles1500	73	0	0	0	0	0	0	100
queen5_5	5	100	100	0	97	0	100	100
queen6_6	7	100	100	0	3	0	3	13
queen7_7	7	100	100	0	0	0	0	3
queen8_8	9	0	0	0	0	0	0	0
queen8_12	12	100	100	0	0	0	0	10
queen9_9	10	0	0	0	0	0	0	0

BackTrack and DSATUR results are 0 or 100, because they success of fail solving the problem.

In table 6.3 we have put the Mycielski graphs, with the CAR graphs, as they as quite similar. The Myciels result confirm our supposition that these graphs are very simple to solve. But the fullins graph appear much more difficult. In almost all the experiments we get very poor result even 0 in a lot of cases, except our GSI approach that gets good results even in the more complex graphs. These graphs are particularly difficult, because even some author claim that the chromatic number is unknown.

The results of SGB graphs are printed in table 6.2. Books graphs are not particularly difficult except the homer graph. Our algorithm get results over the 90% but it is not the best algorithm. SA appears again as a good approach for this kind of graphs, as so the PSO, the TS usually a bad approach here get good results.. The miles graphs are very hard and only our GSI manage the solve them. Finally the queens graphs show a very hard face. The greedy algorithm are the best. The results of the GSI, better than the other, are very poor. The instance queens8_8 and queens9_9 failed to find the optimal chromatic number. A coloring with 10 and 11 for this graphs have been found instead.

In [70] try to solve anna and homer graphs using a Genetic Algorithm. They can solve anna graph but no homer graph. As we have said our GSI can solve all

the books graphs. In [78] and [79] solve the SGB graphs using another genetic algorithm, but fail finding the optimum solution in the CAR family. The results are shown in table 6.3. Here again made a good job in CAR family, and get good results in the SGB, perhaps they get better results, but the comparison is very difficult because we don't know if a special tuning for that graph family has been made. As we have mentioned, our GSI haven't suffer any modification for any experiment. In [115] use an ant-based algorithm for coloring graphs. They perform well in these graphs, finding the best known solution in all the instances, but here we have to compare the computational time. Our ACO implementation is a simple approach to the ant based algorithm, because the aim of this work is the Swarm Intelligence. More result using Genetic Algorithms appears in a Parallel Genetic approach can be seen in [76] and [75] but the number of experiments is quite poor compare with ours. They perform well but in a reduced group of graphs.

In [80] a cultural algorithm is implemented to solve the GCP. The results are similar to ours but again with a small group of algorithms. In [140] using grouping genetic algorithms, works good but fails in some graphs. We also fails in some graphs but test different families.

Result over queens graphs can be seen in [72], where a Genetic algorithm is used again, but here introducing a new mutation operator. The results are not optimum. Other example where the basic genetic strategy fails compared with ours is [71]. Our algorithm has show that the queens graphs are very difficult for it, but in the other hand we have good result in other graphs.

6.6.1 Test

The Friedman test is a non-parametric statistical test developed by Milton Friedman [131] as we have said in the previous chapter. We are going to use it to proof that the results of our algorithm are statistically independent from the other methods. For the SGB graphs we have the ranking values shown in table 6.4.

With this table, we obtained the Friedman value $\chi_F = 21.6$. The value of the square-chi with six degrees of freedom and a probability of accepting the null hypothesis with 0.99 $\chi^2 = 16.8$. The Friedman values is bigger than the null hypothesis, so we can say that the results of these algorithms are statistically independent. Using the Iman-Davenport improvement we have a $\chi_{ID} = 4.5$. The new null hypothesis with 6 and 78 degrees of freedom and an accepting

Table 6.3: Results of CAR graphs

Graph	#K	BT	DS	SA	TS	ACO	PSO	GSI
myciel3	4	100	100	100	100	100	100	100
myciel4	5	100	100	100	100	100	100	100
myciel5	6	100	100	100	100	100	100	100
myciel6	7	100	100	100	100	100	100	100
myciel7	8	100	100	100	100	100	100	100
1-FullIns_3	3	100	100	100	78	93	100	100
1-FullIns_4	4	0	0	0	10	0	20	93
1-FullIns_5	5	0	0	0	0	0	0	90
2-FullIns_3	4	0	100	20	43	73	100	93
2-FullIns_4	5	0	0	0	3	0	0	93
2-FullIns_5	6	0	0	0	0	0	0	90
3-FullIns_3	5	0	0	3	37	0	100	87
3-FullIns_4	6	0	0	0	3	0	0	63
3-FullIns_5	7	0	0	0	0	0	0	87
4-FullIns_3	6	0	0	3	40	0	97	66
4-FullIns_4	7	0	0	0	0	0	0	66
4-FullIns_5	8	0	0	0	0	0	0	77
5-FullIns_3	7	0	0	0	20	0	57	97
5-FullIns_4	8	0	0	0	0	0	0	73

Method	BT	DS	SA	TS	ACO	PSO	GSI
anna	5.5	5.5	3	2	5.5	5.5	1
david	5.5	1	3	5.5	5.5	5.5	2
home	4.5	4.5	4.5	4.5	4.5	4.5	1
huck	2.5	2.5	2.5	6	7	2.5	5
jean	2	2	2	6	7	4	5
miles250	5.5	5.5	2	5.5	5.5	3	1
miles500	4.5	4.5	4.5	4.5	4.5	4.5	1
miles750	4.5	4.5	4.5	4.5	4.5	4.5	1
miles1000	4.5	4.5	4.5	4.5	4.5	4.5	1
miles1500	4.5	4.5	4.5	4.5	4.5	4.5	1
queen_5_5	2.5	2.5	6.5	5	6.5	2.5	2.5
queen_6_6	1.5	1.5	6.5	4.5	6.	4.5	3
queen_7_7	1.5	1.5	5.5	5.5	5.5	5.5	3
queen_8_12	1.5	1.5	5.5	5.5	5.5	5.5	3
\bar{R}_j	3.61	3.29	4.21	4.86	5.5	4.36	2.18

Table 6.4: Algorithm Rankings for Friedman Test

probability of 0.99 is $F(6, 72) = 3.29$. The Iman-Davenport value is bigger than the null hypothesis so we can say that the results of these algorithms values are statistically different. Finally, we could make a Post-hoc test, but it is not necessary, because we wanted to show that our algorithm is independent from the others, but it doesn't matter if the other algorithm has or not a dependence.

6.7 Sequential chromatic number determination

All the test explained in previous sections have been made knowing the chromatic number of the graphs. What what happened if the Chromatic number is unknown or the algorithm can solve the graph in a proper time. In real life problem we will find that the chromatic number is unknown or that is not necessary to get the best solution (a solution under a certain threshold is enough).

We have added to our algorithm an improvement that we have call Sequential approximation (SeccApp). The SeccApp consist of starting from an upper bound to the chromatic number of a graphs, we solve the GCP with the given number and if we succeed, we reduce the number of colors and try to solve the problem again until we reach to a lower bound, given to the algorithm, or we arrive to the chromatic number and the problem can go further, stopping until a number of iterations.

We can't say that the result of the SeccApp is the chromatic number, but we can determine an upper bound to the graph. The accuracy of this improvement is directly related with the number of iteration to stop the algorithm after finding the chromatic number and the lower bound given to the algorithm. If the number of iteration is small, we will get a result far away for the optimal solution, but in the other hand we will have to wait a long time after finding the optimal.

This experiment has two problems:

1. As we don't know the chromatic number, we can't say how near we are to the optimal. We can extract the chromatic number from small graphs with the greedy exacts algorithms like backtracking and DSATUR, but with big graphs it is impossible to use this method.
2. If the chromatic number is unknown, or the the graphs have been generated aleatory, there is any reference in the literature to compare with.

We have prepare three test using the SeccApp. Test over random graphs, DI-

Graphs100x1000	BT	SA	TS	PSO	ACO	GSI
Random1	9	15(14)	11	11	17.1(17)	8.6(8)
Random2	10	15.1(15)	11	11	17(17)	9.2(9)
Random3	10	15(14)	14	11	17.2(17)	9.3(9)
Random4	9	15.1(14)	17	11	17.1(17)	9.1(9)
Random5	10	15(14)	11	11	17(17)	9.1(9)
Random6	11	14.9(14)	11	11	17(17)	9.2(9)
Random7	10	15(14)	11	10	17(17)	9.2(9)
Random8	10	15.1(14)	13	11	17.2(17)	9.1(9)
Random9	10	15(14)	15	10	17.3(17)	9.3(9)
Random10	11	15(14)	11	10	17.1(17)	9.2(9)

Table 6.5: Graphs of 100 nodes and 1000 edges. Between parentheses the minimum solution found.

MACS Leighton’s graphs and real graphs from “Exam timetabling problems” [141].

6.7.1 Random Graphs

We have generated four families of absolutely random graphs. We have add any feature to these graphs. We have generated 10 graphs per family with 100 nodes and 1000 edges in the first family, 2000 edges in the second family, 3000 edges in the third family and 4000 edges in the last family. We have started from an upper bound of 50 colors and try to reduce the number of colors to a lower bound of 5 colors. We have use the same experimental metric as used in previous section.

The results are plotted in tables 6.5,6.6,6.7 and 6.8

The analysis of these results come very easily from the tables. Our GSI algorithm gets the best result for all the families. We can’t say that the result of the GSI if the chromatic number but it is very near. The reason for this experiment is to test the SeccApp improvement and not to find the chromatic number of these random graphs.

The ACO algorithm is the worst as we expected, but the SA works in a estrange manner getting very bad results. The behavior of the SA algorithm is very estrange and would be interesting to study it, but is out of the scope of this work.

The TS and PSO obtain very competitive results, very similar for these two algorithms. The performance of these algorithm is reduced when the density of

Graphs100x2000	BT	SA	TS	PSO	ACO	GSI
Random1	16	29.5(28)	18	18	34	15
Random2	16	29.5(28)	22	18	35	15
Random3	17	29.5(28)	19	18	34	15
Random4	18	29.4(28)	23	19	35	15
Random5	15	29.5(28)	19	19	34	15
Random6	18	29.4(28)	28	18	34	15
Random7	17	29.6(28)	17	18	34	15
Random8	16	29.5(28)	21	19	34	15
Random9	15	29.4(29)	23	19	34	15
Random10	18	29.6(29)	19	18	35	15

Table 6.6: Graphs of 100 nodes and 2000 edges. Between parentheses the minimum solution found.

Graphs100x3000	BT	SA	TS	PSO	ACO	GSI
Random1	23	45(44)	27	28	50	23
Random2	24	45.5(44)	30	28	50	22
Random3	23	45.6(44)	34	27	50	22
Random4	26	45.8(44)	36	27	50	23
Random5	25	45.7(44)	31	27	50	23
Random6	24	45.4(44)	32	28	50	23
Random7	25	45.4(14)	32	29	50	22
Random8	25	45.5(44)	29	28	50	23
Random9	25	45.4(44)	31	27	50	23
Random10	25	45.5(44)	35	28	50	22

Table 6.7: Graphs of 100 nodes and 3000 edges. Between parentheses the minimum solution found.

Graphs100x4000	BT	SA	TS	PSO	ACO	GSI
Random1	36	50	47	41	50	32
Random2	36	50	44	40	50	33
Random3	36	50	15	40	50	34
Random4	37	50	39	42	50	34
Random5	35	50	47	39	50	33
Random6	36	50	40	40	50	33
Random7	36	50	42	42	50	33
Random8	34	50	40	40	50	33
Random9	39	50	43	41	50	33
Random10	35	50	43	40	50	33

Table 6.8: Graphs of 100 nodes and 4000 edges. Between parentheses the minimum solution found.

the graphs grows.

Finally, the deterministic BT algorithm get the second best results in the given number of steps (the BT will find the chromatic number always). This behavior is also unexpected because beats more sophisticated algorithms.

The implementation of the DS has a limitation of a chromatic number of 8 so we haven't used it in this experiment.

6.7.2 Leighton graphs

Theorem 46. *Two finite graphs which have a common covering have a common finite covering [142].*

Based in this theorem Leighton show a way to built graphs [31] where the chromatic number is known.

After testing the SeccApp with random graphs we have make a test with DIMACS Leighton's graphs, because although we know the chromatic number of these graphs, we have problems solving them with out graph coloring suite so we decide, at least, return the most approximate solution with our algorithm. We can see them in table 6.9.

In Fleurent [143] a Evolutionary Tabu Search Approach has been used with good results, finding the best solution for all the 5-colorable and 15-colorable graphs, and two of the 25 colorable graphs. The time needed to solve these problems has been high but similar to the time used in our experiments, but the methods presented in this paper has been tune for a particular kind of graphs, and ours is a more general graphs, as we can see the different types of experiments.

6.7.3 Real Graphs

Finally we have use graphs from the Exam timetabling problem of several universities of the United States. In the papers of Lewis [2, 128], we can find a great comparison of some methods for graph coloring. We have add a new column with our result. The results appear in table 6.10.

Been TabuCol a Tabu Search Based Algorithm. PartialCol a particularization of a Tabu Search algorithm searching only feasible solutions. AntCol an Ant Colony Optimization, similar to the ACO algorithm implemented in our suite but more accurate than ours. HEA is an Hybrid Evolutionary Algorithm. HC a standard Hill Climbing algorithm. BT the Backtracking algorithm but different

Graph	#node	#Edges	Density	#K	GSI
le450_5a	450	5714	0.056	5	8
le450_5b	450	5734	0.057	5	8
le450_5c	450	9803	0.097	5	9
le450_5d	450	9757	0.096	5	9
le450_15a	450	8168	0.081	15	18
le450_15b	450	8169	0.081	15	18
le450_15c	450	16680	0.165	15	20
le450_15d	450	16750	0.166	15	20
le450_25a	450	8260	0.082	25	26
le450_25b	450	8263	0.082	25	27
le450_25c	450	17343	0.171	25	30
le450_25d	450	17425	0.172	25	30

Table 6.9: Leighton Graphs results

Graph	best #K	TabuCol	PartCol	AntCol	HEA	HC	BT	GSI
sta-f-83	13	13.35	13	13.13	13	13	13	13
hec-s-92	17	17.22	17	17.04	17	17	19	18
kfu-s-93	19	20.76	19	19	19	19	19	20
yor-f-83	19	19.74	19	19.87	19.06	19	20	21
tre-s-92	20	20.58	20	20.04	20	20	23	23
car_f-92	27	39.92	32.48	30.04	28.5	27.96	27	36
car-s-91	28	39.1	30.2	29.23	29.04	29.1	28	37
pur-s-93	33	50.7	45.48	33.47	33.7	33.87	33	44

Table 6.10: Exam timetabling of Lewis [2] plus GSI

from ours because this version of BT uses an especial ordering to improve the results.

The results of our GSI algorithm is equal to the best in the sta-f-83 graphs, and not the best but not the worst in the rest of the examples. This is because the amount of time used in both experiments, GSI iterations and the appearing in [2] is different. They used a measure that they call checks and the experiments stops until $5e^{11}$ checks. We used a measure called steps that is bigger than checks obviously, but we stop in only $1e^5$ steps. Other problem is that we haven't use the parallel advantage of our algorithm in the implementation that is critical in large problems like this.

6.8 Conclusions

We have test our GSI algorithm with a wide range of problems. We have compared the results obtained of our GSI with the other six algorithms implemented in our Suite, and also results extracted from the bibliography. Our algorithm is clearly better than other algorithms implemented in our Suite in almost all the situations. When our algorithms didn't get the best results, it is nevertheless ranked among the best. The comparison with result that appear in the literature is more difficult, because the metric used to carry out the test in not always the same. Our algorithm is most cases gets at lest the same results or even better than appear in the literature, finding the chromatic number reported by other researchers.

Sometimes our algorithm don't find the chromatic number, or get worse results than the appearing in a publication, but this is a minority. We must take into account that we explore a big number of different classes, where published works, usually focuses in one kind of graph.

Chapter 7

Conclusions and further work

Graph Coloring is a classical combinatorial problem, which has been studied from many diverse points of view, and still benefits from fresh approaches in the approximate solution in acceptable computational time. The work reported in this Thesis aims to contribute an innovative approach with general good convergence results. Graph Coloring importance from a practical point of view lies in the definition of mappings from other combinatorial problems, so that an efficient solution of the Graph Coloring provides efficient solutions to the original practical problem.

The approach followed in this Thesis is nature inspired in two ways:

- Uses the swarm intelligence metaphor that has produced innovative computational solutions such as the Ant Colony Optimization (ACO) and the Particle Swarm Optimization (PSO).
- Uses the gravitational and electromagnetic interaction (loosely speaking) metaphors to map the GCP into the swarm dynamics.

In this regard, we can state that we have succeed in proposing an efficient nature inspired algorithm for the solution of the GCP. The proof of the value of algorithm follows from two separate works.

1. First, we have been able to state formally some desired convergence properties giving formal proof in the asymptotic case. Specifically, we have shown that upon convergence to a stationary state the GSI always reaches a solution of GCP if the number of color goals is no lower than the graph chromatic number.

2. Second, we have been able to perform computational experiments over an extensive collection of benchmark graphs, comparing with other state-of-the-art algorithms. The performances obtained were competitive or even improved the state of the art in several respects.

The extensive sensitivity analysis experiments have allowed to assess the general insensitivity of the GSI to fine parameter tuning, amounting to a high degree of robustness.

7.1 Future works

Theoretical works: Future works in the theoretical domain must address the dynamical convergence of the GSI: does the GSI always converge to a stationary state? Are there cyclic behaviors? For such research more sophisticated mathematical tools dealing with dynamic stochastic processes may be needed. In any case, the GSI definition must be extended to include some of the intuitive elements, such as the comfort, which play a role in this dynamic convergence.

Computational verification: Future works in the computational domain always ask for more extensive computational experimentation, with more accurate implementations of the competing algorithms, finer tuning of all algorithms.

Non-stationary: Another interesting issue is that of coping with non-stationary environments. It will be highly interesting to study the theoretical convergence issues in the non-stationary case, which would surely imply innovative definitions of the processes involved. For computational evaluations, the definition of appropriate benchmarking and even the adaptation of other approaches to the non-stationary setting will be non trivial

Applications: Future works in the application domain require the identification of appropriate practical problems which can be mapped into a GCP. For instance, actually the candidate is working in social network community detection and tracking, which is a dual problem to the GCP. The GSI may be very proficient in the adaptation to changes in the social structure by its own nature. The GSI does not stop its computation when reaching an stationary state, and if some external condition changes the relation between nodes and

thus GSI agents, the GSI is automatically activated to restore the equilibrium state, meaning computing the closest GCP solution.

Appendix A

Graph experimental instances and graph generators

In this appendix we present the collection of graphs that have been used in the computational experiments for sensitivity exploration and/or comparison among GCP solving algorithms. After a brief introduction in Section A.1, we visit each of the kinds of graphs used: the trees in Section A.2, the DIMACS graphs in Section A.3, random graphs in Section A.4, Mizuno's graphs in Section A.5, planar graphs in Section A.6, KRG graphs in Section A.7, and real graphs in Section A.8

A.1 Introduction

Testing algorithms for Graph Coloring Problem (GCP), including our Gravitational Swarm algorithm, requires a controlled collection of graphs to allow for verification of results and reproduction by independent testers. We have prepared eight groups of graphs to use in the experimental works:

1. 30 Tree graphs.
2. 30 bipartite graphs.
3. 20 DIMACS graphs [84].
4. 250 Kuratowski's theorem based graphs [86].
5. 250 Mizuno's method graphs [1].

6. 280 particular graphs of a new graph family of our own design that we call KRG.
7. 40 completely random graphs of a given number of nodes and edges.
8. 8 real graphs extracted from [2] modeling a scheduling problem in some universities in the USA.

It's impossible to test all kinds of graphs, and even in the selected groups, there are infinite subfamilies with small differences between them, but the graphs in the list above show many special features that are of interest in the GCP.

A.2 Trees and bipartite graphs

A.2.1 Trees

The trees are special graphs with the specific feature that their chromatic number is 2. A Tree is an undirected graph such that any two nodes are connected by exactly one single path. In other words, a tree is a connected graph without cycles.

Trees can be drawn in such a way that we have a root node with two or more offspring. Each of these offspring can have more offspring until all the nodes are drawn forming a structure similar to a tree.

Coloring trees is quite easy, nevertheless trying to color them may uncover unexpected problems for some algorithms. Therefore, they are a kind of bottom benchmark for algorithm performance.

A.2.2 Bipartite

A bipartite graph is a graph whose nodes can be divided into two disjoint sets U and V such that every edge connects a node in U to one in V ; that is, U and V are sets of independent nodes. By definition, bipartite graphs are 2-colorable. And are very easy to color. But as happened with the trees, the way that the algorithm tries to find the coloring can make it fail.

A.3 DIMACS

The Center for Discrete Mathematics and Theoretical Computer Science at Rutgers University, DIMACS, celebrated a challenge in 1993 to probe the state of

the art of algorithms solving the GCP. Gathering material from different sources, they offered a lot of graphs to test and show the goodness of the algorithms.

These graphs are very important because the scientific community used them to test different GCP algorithms allowing easy and fair comparison of different works. This is only partially true, because even though a lot of researchers use these graphs for testing new algorithm, it is also true that the specific parametrization, stopping conditions, and computational resources used for the experiments, make really difficult to compare one work with another. In the DIMACS challenge, a graph file format was introduced as a way to share graphs between different research groups. The format has three tags. The lines that start with a letter «c» are comments, the line starting with the letter «p edge» followed by two numbers contain the number of nodes and the number of edges respectively. The line starting with «e» followed by two natural numbers represent an edge between two nodes identified by numbers. An example graph codification follows:

```
c Test graph
p edge 5 6
e 1 2
e 1 3
e 2 3
e 2 4
e 2 5
e 4 5
```

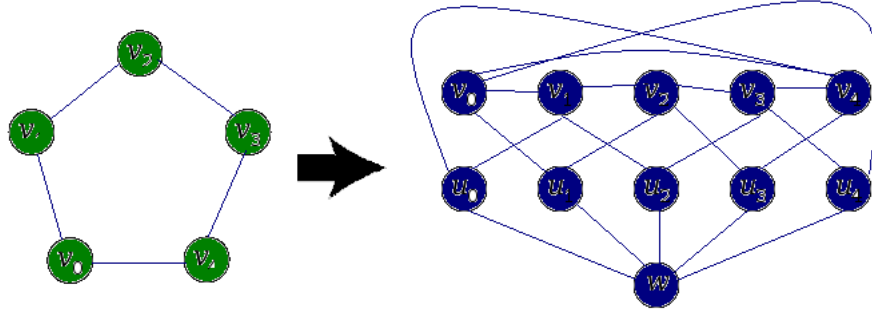
The DIMACS graphs used in our experiments are the Mycielski graphs, queens graphs, miles graphs, fullins graphs and the books graph.

A.3.1 Mycielski graphs

Among all the DIMACS graphs, the Mycielski graphs [90] have been very important in our experiments. The Mycielski graphs are constructed following the Mycielski theorem, and are a family of small and simple graphs useful to tune the parameters of optimization algorithms. The Mycielski theorem reads as follows:

Theorem 47. *There are triangle-free graphs with arbitrarily high chromatic number.*

The construction of the Mycielski graph follow a mathematical formula as

Figure A.1: Mycielski graph transition from M_3 to M_4

follows. Let us denote the n nodes of a given graph G as v_0, v_1 , etc. The Mycielski graph of G contains G itself as an isomorphic sub-graph, together with $n + 1$ additional nodes: a node u_i corresponding to each node v_i of G , and another node w . Each node u_i is connected by an edge to w , so that these nodes form a star-shaped sub-graph $k_{1,n}$. In addition, for each edge (v_i, v_j) of G , the Mycielski graph includes two edges, (u_i, v_j) and (v_i, u_j) .

Thus, if G has n nodes and m edges, $m(G)$ has $2n + 1$ nodes and $3m + n$ edges. In figure A.1 we can observe the graph transition from M_3 to M_4 . (source wikipedia).

A.3.2 Queens Graphs

The $n \times n$ queen graph has the squares of a $n \times n$ chessboard as its nodes and two nodes are adjacent if and only if queens placed on the two squares attack each other. These graphs are very dense, because as the queen can move in all directions and all the squares that she wanted, it is very difficult to place n queens in a $n \times n$ chessboard. This graph family is inspired in the classical problem of placing 8 queens in a standard 8×8 chessboard, see figure A.2. Martin Gardner states [144] without proof that the $n \times n$ queen graph is n -colorable whenever $n \bmod 6$ is 1 or 5.

A.3.3 Miles graphs

Miles graphs are similar to geometric graphs in that nodes are placed in space with two nodes connected if they are close enough. These graphs, however, are not random. The nodes represent a set of United States cities and the distance

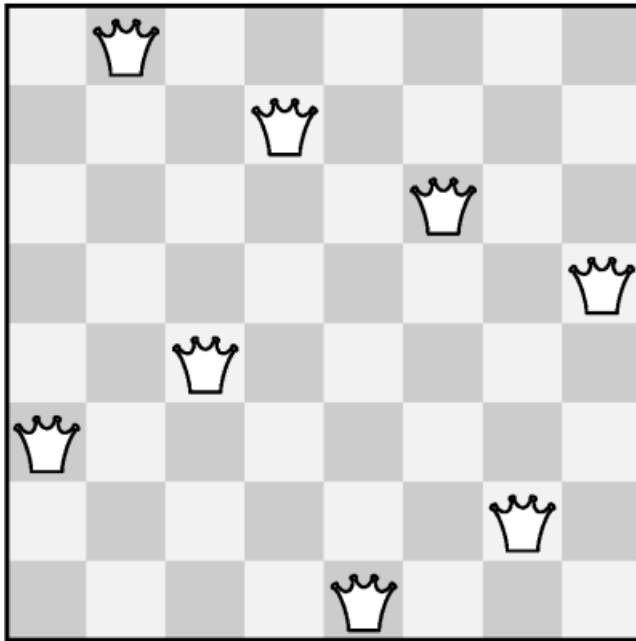


Figure A.2: 8 queens problem solution

between them is given by by road mileage from 1947. These graphs are also due to Knuth. There are different graphs according to the number of cities used for its construction. In the figure A.3 we can see a map of the USA where some cities of different states are linked by a wire.

A.3.4 Fullins graphs

Full Insertion aka Fullins graphs are a generalization of Mycielski graphs with inserted nodes to increase graph size but not density. The result is a graph more complex and difficult to color. The problem of Mycielski graphs is that they are very easy to solve by deterministic algorithms and don't offer a handicap. This generalization allow to use the Mycielski theorem to built more difficult graphs.

A.3.5 Books graphs

These graphs are extracted from books. The construction is as follows: Given a work of literature, a graph is created where each node represents a character. Two nodes are connected by an edge if the corresponding characters encounter

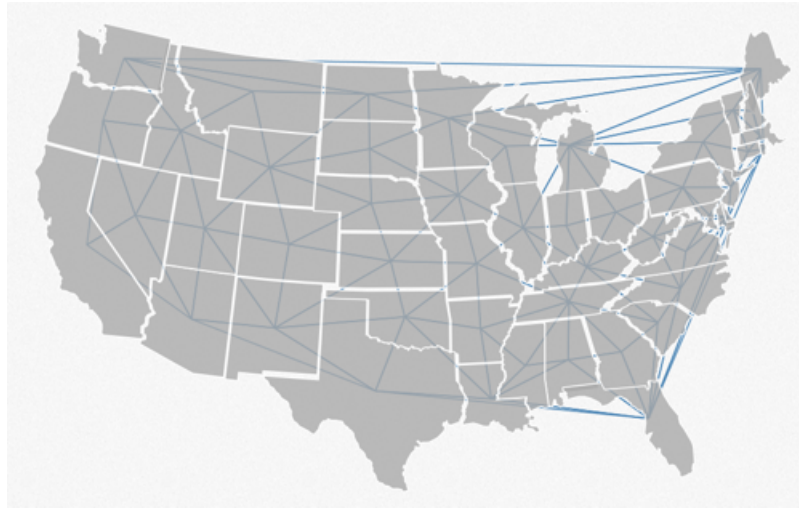


Figure A.3: United States Cites Graph

each other in the book. Donald E. Knuth from the Stanford University created the graphs for five classic works:

1. Leon Tolstoy's *Anna Karenina* called `anna.col` with 138 node, 493 edges and 11 colors.
2. Charles Dicken's *David Copperfield* called `david.col` with 87 node, 406 edges and 11 colors.
3. Homer's *Iliad* called `homer.col` with 561 node, 1629 edges and 13 colors.
4. Mark Twain's *Huckleberry Finn* called `huck.col` with 74 node, 301 edges and 11 colors.
5. Victor Hugo's *Les Miserables* called `jean.col` with 80 node, 254 edges and 10 colors.

A.3.6 Leighton Graphs

This graphs are based on Leighton's graph covering theorem [142] that says:

Theorem 48. *Two finite graphs which have a common covering have a common finite covering.*

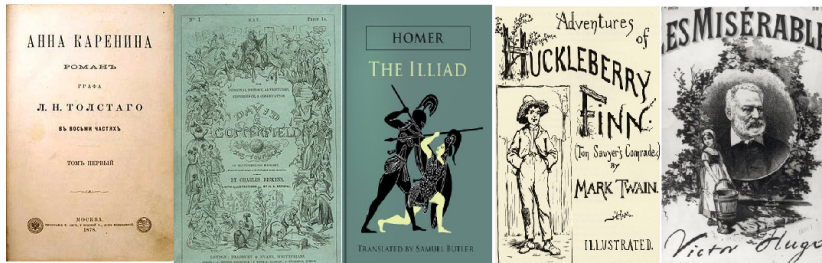


Figure A.4: Books where the Books Graphs come from

This graph family has been widely studied and represent a particular group of graphs called covering graphs.

Let $G = (V, E)$ and $C = (V_2, E_2)$ be two graphs, and let $f : V_2 \rightarrow V$ be a surjection. Then f is a covering map from C to G if for each $v \in V_2$, the restriction of f to the neighborhood of v is a bijection onto the neighborhood of $f(v) \in V$ in G . In other words, f maps edges incident to v one-to-one onto edges incident to $f(v)$. If there exists a covering map from C to G , then C is a covering graph (or a lift) of G .

A.4 Random Graphs

We have implemented a graph generator that given a number of nodes V and a range of edges E_1 and E_2 , the application can generate a graph $G = \{V, E\}$. The test using this kind of graphs is useful to challenge different algorithms starting without any information. The chromatic number is unknown, so starting from an upper-bound, the algorithms must find the chromatic number, or try to get near as fast as possible.

The random graphs are used in advanced test, when the algorithms have been previously tuned. Not all the methods allow a sequential decreasing search, but is essential in graphs of unknown chromatic number if we want the search of the chromatic number to be an automatic task. The other way implies a manual search changing the number of color after each success try. All implementations in our Graph Coloring Suite came with this feature.

A.5 Mizuno's Graphs

We have implemented the graph generation method developed by Mizuno in [97]. Mizuno's Graphs are a family of 3-colorable graphs. Mizuno claims that the graphs generated by his method are very hard to color. We have confirmed empirically that it's true.

The constructions of these graphs is based on 12 different graphs called MUGs that are 4 colorable. We can see these MUGs in figure A.5. The method takes two of these MUGS and join them according to a rule to form a new graph that is exactly 3-colorable. The resulting graph can join with another MUG, following the same rule, to create bigger graphs. This can be repeated until the user wanted increasing the size of the graphs. All these graphs have the same complexity but obviously, bigger graphs are even more difficult to solve than small graphs.

The Mizuno's graphs appear in the DIMACS challenge but we have built our own generator because the chromatic number is known so we can make extensive and exhaustive test on difficult graphs. The number of Mizuno's graphs that appears in DIMACS is very short.

A.6 Planar Graphs

We have prepared an especial family of Graphs: Planar Graphs. A planar graph is a graph that can be embedded in the plane: all the edges can be draw in the plane without any crossing. Planar graphs are 4-colorable according to the four color theorem. This theorem states that, given any separation of a plane into contiguous regions, producing a figure called a map, no more than four colors are required to color the regions of the map so that no two adjacent regions have the same color. In other words the chromatic number of planar graphs is 4.

We have prepared an application that generates this graphs using the Kuratowski's theorem that reads as follows:

Theorem 49. *A finite graph is planar if and only if it does not contain a sub-graph that is a subdivision of k_5 or $k_{3,3}$.*

Where k_5 is the complete graph on five nodes, and $k_{3,3}$ is the complete bipartite graph of six nodes, three of which connect to each of the other three.

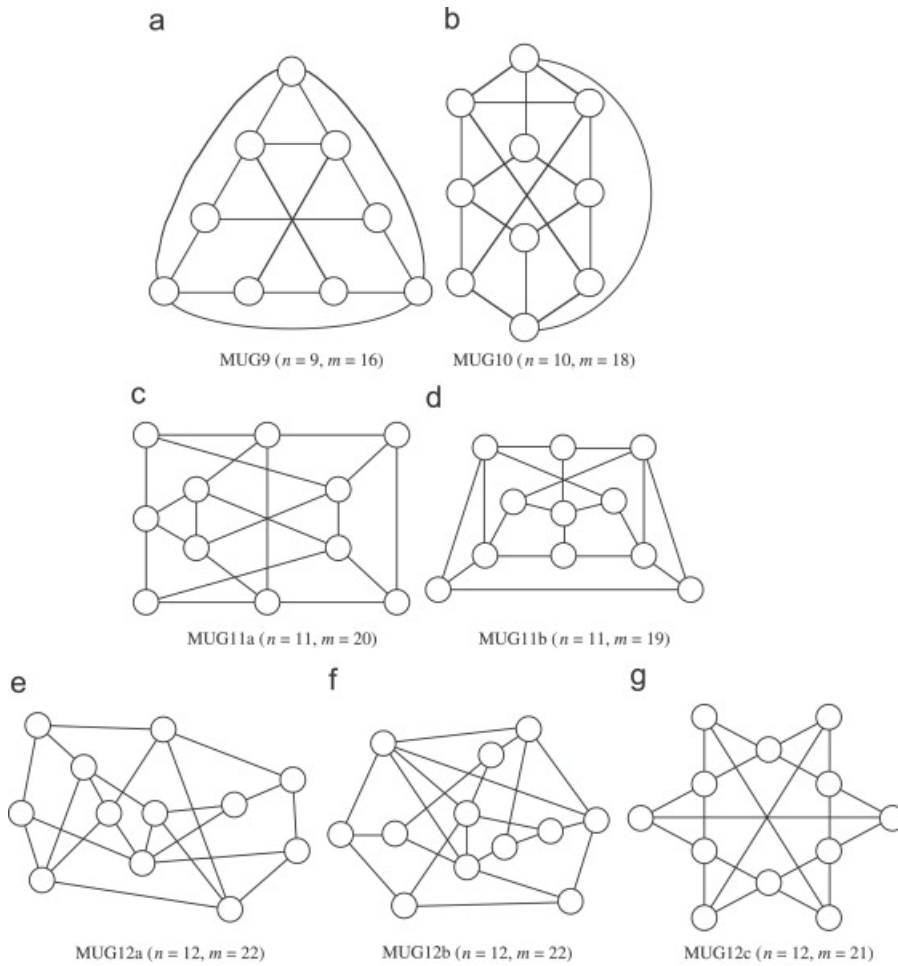


Figure A.5: Mizuno's MUGs for 3-colorable graphs generation [1]

Our implementation, like for random graphs, asks for the number of nodes V and the number of edges E . The maximum number of edges comes from the formula $M = (V * 3) - 6$. If $E > M$ then it is impossible to generate a planar graph.

A.7 KRG graphs

The KRG graphs are a special family of graphs discovered in this thesis whose chromatic number is given when we built them. We need three parameters, the nodes V , the edges E and the chromatic number C . Taking into account the Kuratowski theorem, there are any k_5 or $k_{3,3}$ sub-graphs. Then we choose C nodes and we connect all the nodes forming a K_c sub-graph of n nodes, $n = \frac{V*(V-1)}{2}$. Then we have a graph $G = \{V, E\}$ that is C -colorable.

Theorem 50. *If we have a planar graph, if we add n edges between nodes such that we form a complete sub-graph, then the chromatic number of the graph is the order of the complete sub-graph.*

The Graph $G_k = \{V, (E - n)\}$ is planar using the Kuratowski theorem. The graph $G = \{V, E\}$ has a maximum clique of C , been a clique more or less a complete sub-graph where all the nodes have and edge between them. Using the Brèlaz [34] method for graph coloring we can say that the chromatic lower-bound is C . Using the Bron-Kerbosch [129] method to extract a clique, we can sure that there is any clique bigger than C . So the chromatic number is C .

In the figure A.7 we can see a screenshot of the implemented application, embedded in the graph coloring suite, that generates the aleatory graphs with a given number of nodes and a range of edges. Planar graphs based on kuratowski theorem and our new graphs KRG. And also the 3-colorable graphs of Mizuno.

A.8 Real Graphs

We have used synthetic graphs to prove that our algorithm can solve the GCP. These graphs follow a pattern which the algorithms can use to find the solution. Even the randomly generated graphs can have a pattern, induced by the generation method implementation. However, sometimes modeling a real life problem produces graphs without any special feature and arbitrary structures that can confuse even the most accurate algorithms.

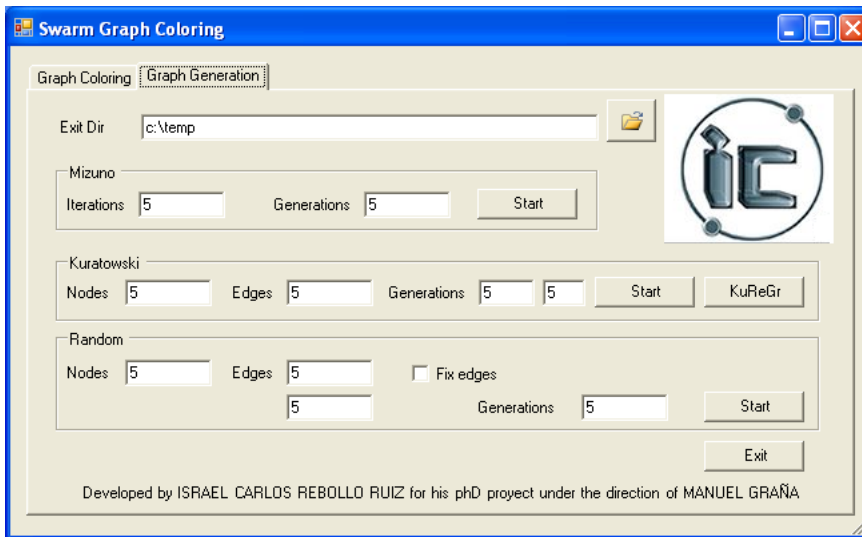


Figure A.6: Aleatory graphs, Kuratowski's planar graphs, Mizuno's 3-colorable graphs and KRG new developed graph type generator.

For that reason we have included in this benchmarking graph collection some real life graphs that represent a scheduling problem in some universities in the USA. Noteworthy, the exact chromatic number is unknown and for comparison we only have the result published by other researchers.

The problem is to schedule the examinations using the existing resources, in such a way that no two exams can be held in the same place in the same day. The full explanation of these graphs can be found in [145].

Appendix B

Graph Coloring Suite

We have implemented all the GCP solving methods in a single program. We call it a Graph coloring Suite, because we have in a unique environment all the algorithms. We can see a snapshot of the the program's user interface in figure B.1.

The user interface allows to select:

- The input graph file (which must be coded in DIMACS format),
- The output directory for the results.
- The hypothesis on the chromatic number.
- The upper bound on the chromatic number, if you want to perform a sequential search decreasing the hypothetical chromatic number of colors until reaching the lowest unsolved hypothesis.
- Besides there are four additional parameter needed only for the GSI: the Goal Radius, the world size and the Confort.
- The number of iterations that we are going to let to program before stopping it without finding a solution. The program stops when a solution is found or the maximum number of iterations is achieved.
- The number Repetitions of the algorithm that we want to execute. This value has no meaning in the BackTracking and DSATUR algorithms, as they are deterministic.
- The algorithm to be used for the solution (via separate buttons).

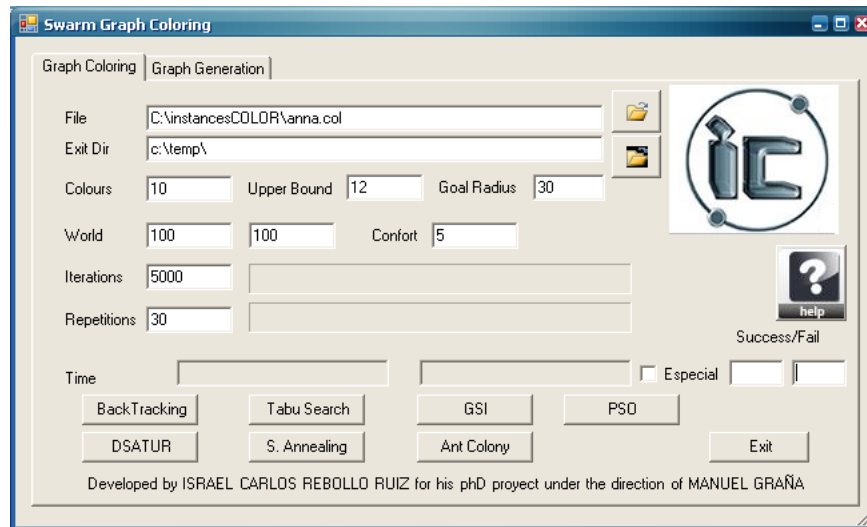
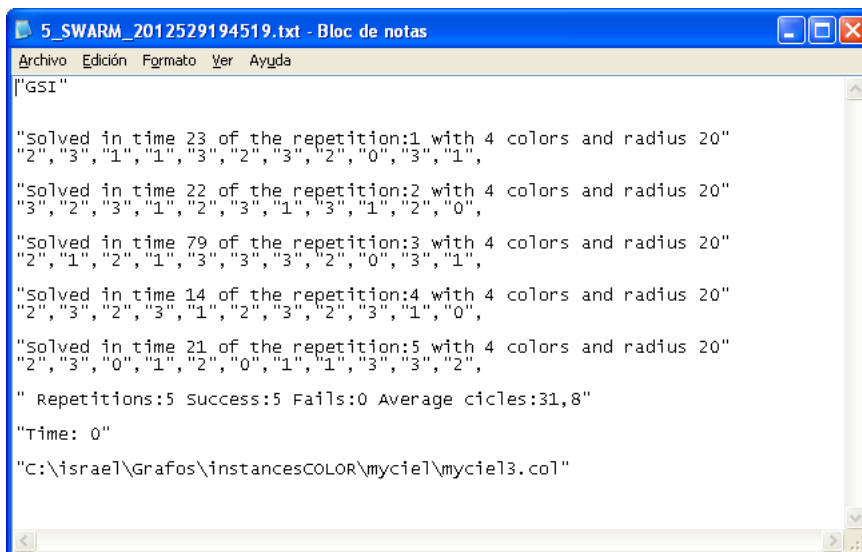


Figure B.1: Graph Coloring Suite

The program returns a file that contains information about the result of the applied algorithm. The first line is the algorithm used. Then for each repetition of the algorithm, the number of iterations need to find a solution and the number of repetition. If the algorithm didn't managed to find a solution then the number of iteration is equal to the maximum number of iteratiosn. After that, the solution appears if it is found, showing for each vertex the assigned color by the algorithm. The colors are represented by numbers starting from zero.

Finally, there is a summary report of the experiment. The number of repetitions. The numer of success experiment and the number of failures. Then the average number of iterations for all the experiments, success and failures. And the last two lines are for the total time of the experiments in seconds and the name of the file with the graph tested. The figure B.2 shows a snapshot of a result file.



```
5_SWARM_2012529194519.txt - Bloc de notas
Archivo Edición Formato Ver Ayuda
"GSI"

"Solved in time 23 of the repetition:1 with 4 colors and radius 20"
"2","3","1","1","3","2","3","2","0","3","1",

"Solved in time 22 of the repetition:2 with 4 colors and radius 20"
"3","2","3","1","2","3","1","3","1","2","0",

"Solved in time 79 of the repetition:3 with 4 colors and radius 20"
"2","1","2","1","3","3","3","2","0","3","1",

"Solved in time 14 of the repetition:4 with 4 colors and radius 20"
"2","3","2","3","1","2","3","2","3","1","0",

"Solved in time 21 of the repetition:5 with 4 colors and radius 20"
"2","3","0","1","2","0","1","1","3","3","2",

" Repetitions:5 Success:5 Fails:0 Average cycles:31,8"

"Time: 0"

"C:\israel\Grafos\instancesCOLOR\myciel\myciel3.col"
```

Figure B.2: Snapshot of a result file

Bibliography

- [1] K. Mizuno and S. Nishihara, “Constructive generation of very hard 3-colorability instances,” *Discrete Appl. Math.*, vol. 156, no. 2, pp. 218–229, 2008.
- [2] R. Lewis, J. Thompson, C. Mumford, and J. Gillard, “A wide-ranging computational comparison of high-performance graph colouring algorithms,” *Computers & Operations Research*, vol. 39, no. 9, pp. 1933 – 1950, 2012.
- [3] G. Folino, A. Forestiero, and G. Spezzano, “An adaptive flocking algorithm for performing approximate clustering,” *Information Sciences*, vol. 179, no. 18, pp. 3059 – 3078, 2009.
- [4] C. W. Reynolds, “Flocks, herds, and schools: A distributed behavioral model,” in *Computer Graphics*, pp. 25–34, 1987.
- [5] G. Antonelli, F. Arrichiello, and S. Chiaverini, “Flocking for multi-robot systems via the null-space-based behavioral control,” *Swarm Intelligence*, vol. 4, pp. 37–56, 2010.
- [6] A. Turgut, H. elikkanat, and E. Fahin, “Self-organized flocking in mobile robot swarms,” *Swarm Intelligence*, vol. 2, pp. 97–120, 2008.
- [7] M. El-Abd and M. Kamel, “A cooperative particle swarm optimizer with migration of heterogeneous probabilistic models,” *Swarm Intelligence*, vol. 4, pp. 57–89, 2010.
- [8] S. Chu, J. F. Roddick, and J. Pan, “Ant colony system with communication strategies,” *Information Sciences*, vol. 167, no. 1-4, pp. 63 – 76, 2004.

- [9] M. Dorigo, V. Maniezzo, and A. Coloni, "The ant system: Optimization by a colony of cooperating agents," *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS-PART B*, vol. 26, no. 1, pp. 29–41, 1996.
- [10] A. John, A. Schadschneider, D. Chowdhury, and K. Nishinari, "Characteristics of ant-inspired traffic flow," *Swarm Intelligence*, vol. 2, pp. 25–41, 2008.
- [11] A. Rizzoli, R. Montemanni, E. Lucibello, and L. Gambardella, "Ant colony optimization for real-world vehicle routing problems," *Swarm Intelligence*, vol. 1, pp. 135–151, 2007.
- [12] P. Balaprakash, M. Birattari, T. Statzle, Z. Yuan, and M. Dorigo, "Estimation-based ant colony optimization and local search for the probabilistic traveling salesman problem," *Swarm Intelligence*, vol. 3, pp. 223–242, 2009.
- [13] N. Franks, J. Hooper, M. Gumm, T. Bridger, J. Marshall, and A. Dornhaus, "Moving targets: collective decisions and flexible choices in house-hunting ants," *Swarm Intelligence*, vol. 1, pp. 81–94, 2007.
- [14] O. Korb, T. Stutzle, and T. Exner, "An ant colony optimization approach to flexible protein-ligand docking," *Swarm Intelligence*, vol. 1, pp. 115–134, 2007.
- [15] S. Nouyan, A. Campo, and M. Dorigo, "Path formation in a robot swarm," *Swarm Intelligence*, vol. 2, pp. 1–23, 2008.
- [16] G. Peterson, C. Mayer, and T. Kubler, "Ant clustering with locally weighted ant perception and diversified memory," *Swarm Intelligence*, vol. 2, pp. 43–68, 2008.
- [17] V. Borkar and D. Das, "A novel aco algorithm for optimization via reinforcement and initial bias," *Swarm Intelligence*, vol. 3, pp. 3–34, 2009.
- [18] S. Shah, R. Kothari, and S. Chandra, "Trail formation in ants. a generalized polya urn process," *Swarm Intelligence*, vol. 4, pp. 145–171, 2010.
- [19] M. Jiang, Y. Luo, and S. Yang, "Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm," *Information Processing Letters*, vol. 102, no. 1, pp. 8 – 16, 2007.

- [20] J. Fernández and E. García, “The pso family: deduction, stochastic analysis and comparison,” *Swarm Intelligence*, vol. 3, pp. 245–273, 2009.
- [21] R. Poli, J. Kennedy, and T. Blackwell, “Particle swarm optimization,” *Swarm Intelligence*, vol. 1, pp. 33–57, 2007.
- [22] Z. Ren, J. Wang, and H. Zhang, “A new particle swarm optimization algorithm and its convergence analysis,” in *Genetic and Evolutionary Computing, 2008. WGECC '08. Second International Conference on*, pp. 319–323, 25-26 2008.
- [23] W. Du and B. Li, “Multi-strategy ensemble particle swarm optimization for dynamic optimization,” *Information Sciences*, vol. 178, no. 15, pp. 3096 – 3109, 2008.
- [24] X. Li and J. Wang, “A steganographic method based upon jpeg and particle swarm optimization algorithm,” *Information Sciences*, vol. 177, no. 15, pp. 3099 – 3109, 2007.
- [25] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, “Gsa: A gravitational search algorithm,” *Information Sciences*, vol. 179, no. 13, pp. 2232 – 2248, 2009.
- [26] J. A. Bondy, *Graph Theory With Applications*. Elsevier Science Ltd, 1976.
- [27] A. A. Zykov, “On some properties of linear complexes,” *Mat. Sb. (N.S.)*, vol. 24, no. 2, pp. 163–188, 1949.
- [28] R. Odaira, T. Nakaike, T. Inagaki, H. Komatsu, and T. Nakatani, “Coloring-based coalescing for graph coloring register allocation,” in *Proceedings of the 8th annual IEEE/ACM international symposium on Code generation and optimization*, CGO '10, (New York, NY, USA), pp. 160–169, ACM, 2010.
- [29] D. G. Corneil and B. Graham, “An algorithm for determining the chromatic number of a graph,” *SIAM J. Comput.*, vol. 2, no. 4, pp. 311–318, 1973.
- [30] R. D. Dutton and R. C. Brigham, “A new graph colouring algorithm,” *The Computer Journal*, vol. 24, no. 1, pp. 85–86, 1981.

- [31] F. T. Leighton, "A graph coloring algorithm for large scheduling problems," *Journal of Research of the National Bureau of Standards*, vol. 84, no. 6, pp. 489–506, 1979.
- [32] G. Palubeckis, "On the recursive largest first algorithm for graph colouring," *Int. J. Comput. Math.*, vol. 85, pp. 191–200, February 2008.
- [33] J. Hansen, M. Kubale, U. Kuszner, and A. Nadolski, "Distributed largest-first algorithm for graph coloring," in *Euro-Par 2004 Parallel Processing* (M. Danelutto, M. Vanneschi, and D. Laforenza, eds.), vol. 3149 of *Lecture Notes in Computer Science*, pp. 804–811, Springer Berlin, Heidelberg, 2004.
- [34] D. Brelaz, "New methods to color the vertices of a graph," *Commun. ACM*, vol. 22, pp. 251–256, April 1979.
- [35] J. P. Spinrad and G. Vijayan, "Worst case analysis of a graph coloring algorithm," *Discrete Applied Mathematics*, vol. 12, no. 1, pp. 89 – 92, 1985.
- [36] J. S. Turner, "Almost all k-colorable graphs are easy to color," *Journal of Algorithms*, vol. 9, no. 1, pp. 63 – 82, 1988.
- [37] D. R. Wood, "An algorithm for finding a maximum clique in a graph," *Operations Research Letters*, vol. 21, no. 5, pp. 211 – 217, 1997.
- [38] I. Méndez-Díaz and P. Zabala, "A branch-and-cut algorithm for graph coloring," *Discrete Applied Mathematics*, vol. 154, no. 5, pp. 826 – 847, 2006.
- [39] V. G. Vizing, "On an estimate of the chromatic class of a p-graph," *Diskret. Analiz*, vol. 3, pp. 23–30, 1968.
- [40] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423, July, October 1948.
- [41] A. Hertz, "A new graph coloring algorithm," *Operations Research Letters*, vol. 10, no. 7, pp. 411 – 415, 1991.
- [42] W. Klotz, "Clique covers and coloring problems of graphs," *Journal of Combinatorial Theory, Series B*, vol. 46, no. 3, pp. 338 – 345, 1989.

- [43] A. Hertz, M. Plumettaz, and N. Zufferey, “Variable space search for graph coloring,” *Discrete Applied Mathematics*, vol. 156, no. 13, pp. 2551 – 2560, 2008.
- [44] J. R. Brown, “Chromatic scheduling and the chromatic number problem,” *Management Science*, vol. 19, no. 4-Part-1, pp. 456–463, 1972.
- [45] H. S. Wilf, “Backtrack: An $o(1)$ expected time algorithm for the graph coloring problem,” *Information Processing Letters*, vol. 18, no. 3, pp. 119 – 121, 1984.
- [46] H.-K. Park, S.-Y. Kim, Y.-C. Ra, and S.-W. Lee, “A study on the new bsc algorithm and design for network security,” in *Proceedings of the 2009 International Conference on New Trends in Information and Service Science*, (Washington, DC, USA), pp. 37–41, IEEE Computer Society, 2009.
- [47] O. Titiloye and A. Crispin, “Quantum annealing of the graph coloring problem,” *Discrete Optimization*, vol. In Press, Corrected Proof, pp. –, 2011.
- [48] 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*, vol. 39, no. 3, pp. 378–406, 1991.
- [49] A. Nolte and R. Schrader, “Simulated annealing and graph colouring,” *Comb. Probab. Comput.*, vol. 10, pp. 29–40, January 2001.
- [50] F. Bonomo, S. Mattia, and G. Oriolo, “Bounded coloring of co-comparability graphs and the pickup and delivery tour combination problem,” *Theoretical Computer Science*, vol. 412, no. 45, pp. 6261 – 6268, 2011.
- [51] Rhyd and Lewis, “A general-purpose hill-climbing method for order independent minimum grouping problems: A case study in graph colouring and bin packing,” *Computers & Operations Research*, vol. 36, no. 7, pp. 2295 – 2310, 2009.
- [52] P. Galinier and A. Hertz, “A survey of local search methods for graph coloring,” *Comput. Oper. Res.*, vol. 33, no. 9, pp. 2547–2562, 2006.

- [53] I. Blochliger and N. Zufferey, "A graph coloring heuristic using partial solutions and a reactive tabu scheme," *Computers & Operations Research*, vol. 35, no. 3, pp. 960 – 975, 2008.
- [54] B. B. Mabrouk, H. Hasni, and Z. Mahjoub, "On a parallel genetic-tabu search based algorithm for solving the graph colouring problem," *European Journal of Operational Research*, vol. 197, no. 3, pp. 1192 – 1201, 2009.
- [55] R. Qu, E. K. Burke, and B. McCollum, "Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems," *European Journal of Operational Research*, vol. 198, no. 2, pp. 392 – 404, 2009.
- [56] S. ichi Nakayama and S. Masuyama, "A parallel algorithm for solving the coloring problem on trapezoid graphs," *Information Processing Letters*, vol. 62, no. 6, pp. 323 – 327, 1997.
- [57] M.-S. Yu and C.-H. Yang, "A simple optimal parallel algorithm for the minimum coloring problem on interval graphs," *Information Processing Letters*, vol. 48, no. 1, pp. 47 – 51, 1993.
- [58] D. Guan and Z. Xuding, "A coloring problem for weighted graphs," *Information Processing Letters*, vol. 61, no. 2, pp. 77 – 81, 1997.
- [59] T. Vredeveld and J. K. Lenstra, "On local search for the generalized graph coloring problem," *Operations Research Letters*, vol. 31, no. 1, pp. 28 – 34, 2003.
- [60] M. Bouchard, M. Cangalovic, and A. Hertz, "About equivalent interval colorings of weighted graphs," *Discrete Applied Mathematics*, vol. 157, no. 17, pp. 3615 – 3624, 2009.
- [61] M. Demange, T. Ekim, and D. de Werra, "(p,k)-coloring problems in line graphs," *Theoretical Computer Science*, vol. 349, no. 3, pp. 462 – 474, 2005.
- [62] B. Ries, "Complexity of two coloring problems in cubic planar bipartite mixed graphs," *Discrete Applied Mathematics*, vol. 158, no. 5, pp. 592 – 596, 2010.

- [63] V. Campos, C. L. Sales, K. Maia, N. Martins, and R. Sampaio, "Restricted coloring problems on graphs with few," *Electronic Notes in Discrete Mathematics*, vol. 37, no. 0, pp. 57 – 62, 2011.
- [64] Dániel and Marx, "Parameterized coloring problems on chordal graphs," *Theoretical Computer Science*, vol. 351, no. 3, pp. 407 – 424, 2006.
- [65] G. Confessore, P. Dell'Olmo, and S. Giordani, "An approximation result for the interval coloring problem on claw-free chordal graphs," *Discrete Applied Mathematics*, vol. 120, no. 1-3, pp. 73–90, 2002.
- [66] E. Maistrelli and D. Penman, "Some colouring problems for paley graphs," *Discrete Mathematics*, vol. 306, no. 1, pp. 99 – 106, 2006.
- [67] K. Yadav, S. Varagani, K. Kothapalli, and V. Venkaiah, "Acyclic vertex coloring of graphs of maximum degree 5," *Discrete Mathematics*, vol. 311, no. 5, pp. 342 – 348, 2011.
- [68] P. Galinier, A. Hertz, and N. Zufferey, "An adaptive memory algorithm for the k-coloring problem," *Discrete Applied Mathematics*, vol. 156, no. 2, pp. 267 – 279, 2008.
- [69] M. C. Costa, D. de Werra, C. Picouleau, and B. Ries, "Graph coloring with cardinality constraints on the neighborhoods," *Discrete Optimization*, vol. 6, no. 4, pp. 362 – 369, 2009.
- [70] A. Marino and R. I. Damper, "Breaking the symmetry of the graph colouring problem with genetic algorithms," in *Genetic and Evolutionary Computation Conference (GECCO-2000), Late Breaking Papers*, pp. 240–245, 2000.
- [71] J. W. Shen, "Solving the graph coloring problem using genetic programming," in *Genetic Algorithms and Genetic Programming at Stanford 2003* (J. R. Koza, ed.), (Stanford, California, 94305-3079 USA), pp. 187–196, Stanford Bookstore, 4 Dec. 2003.
- [72] D. B. Biman Ray, Anindya J Pal and T. hoon Kim, "An efficient ga with multipoint guided mutation for graph coloring problems," *International Journal of Signal Processing, Image Processing and Pattern Recognition*, vol. 3, june 2010.

- [73] D. C. Porumbel, J. Hao, and P. Kuntz, "A search space cartography for guiding graph coloring heuristics," *Computers & Operations Research*, vol. 37, no. 4, pp. 769 – 778, 2010.
- [74] D. C. Porumbel, J. Hao, and P. Kuntz, "An evolutionary approach with diversity guarantee and well-informed grouping recombination for graph coloring," *Computers & Operations Research*, vol. 37, no. 10, pp. 1822 – 1832, 2010.
- [75] S. N. Sivanandam, S. Sumathi, and T. Hamsapriya, "A hybrid parallel genetic algorithm approach for graph coloring," *Int. J. Know.-Based Intell. Eng. Syst.*, vol. 9, pp. 249–259, August 2005.
- [76] Z. Kokosiski and K. Kwarciany, "On sum coloring of graphs with parallel genetic algorithms," pp. 211–219, 2007.
- [77] J. Yu and S. Yu, "A novel parallel genetic algorithm for the graph coloring problem in vlsi channel routing," *International Conference on Natural Computation*, vol. 4, pp. 101–105, 2007.
- [78] T.-h. K. D. B. T. Maitra, A. J. Pal, "Hybridization of genetic algorithm with bitstream neurons for graph coloring," *International Journal of u- and e- Service, Science and Technology*, vol. 3, pp. 37–53, september 2010.
- [79] T. Maitra, A. J. Pal, M. Choi, and T. Kim, "Hybridization of ga and ann to solve graph coloring," in *Security-Enriched Urban Computing and Smart Grid* (T.-h. Kim, A. Stoica, and R.-S. Chang, eds.), vol. 78 of *Communications in Computer and Information Science*, pp. 517–523, Springer Berlin Heidelberg, 2010.
- [80] R. Abbasian, M. Mouhoub, and A. Jula, "Solving graph coloring problems using cultural algorithms," 2011.
- [81] I. Dukanovic and F. Rendl, "A semidefinite programming-based heuristic for graph coloring," *Discrete Applied Mathematics*, vol. 156, no. 2, pp. 180 – 189, 2008.
- [82] A. Mehrotra and M. Trick, "A column generation approach for graph coloring," *INFORMS Journal On Computing*, vol. 8, no. 4, pp. 344–354, 1996.

- [83] D. Gomez and J. Montero, “A coloring fuzzy graph approach for image classification,” *Information Sciences*, vol. 176, no. 24, pp. 3645 – 3657, 2006.
- [84] D. S. Johnson and M. A. Trick, *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, vol. 26. American Mathematical Society, 1993.
- [85] D. Johnson and M. Trick, eds., *Proceedings of the 2nd DIMACS Implementation Challenge*, vol. 26, American Mathematical Society, 1996. DIMACS Series in Discrete Mathematics and Theoretical Computer Science.
- [86] K. Kuratowski, “Sur le problème des courbes gauches en topologie.,” *Fund. Math.*, vol. 15, pp. 271–283, 1930.
- [87] K. Kuratowski, “A half century of polish mathematics: Remembrances and reflections.” Oxford, Pergamon Press, 1980.
- [88] B. Luzar, R. Skrekovski, and M. Tancer, “Injective colorings of planar graphs with few colors,” *Discrete Mathematics*, vol. 309, no. 18, pp. 5636 – 5649, 2009.
- [89] D. de Werra, M. Demange, B. Escoffier, J. Monnot, and V. Paschos, “Weighted coloring on planar, bipartite and split graphs: Complexity and approximation,” *Discrete Applied Mathematics*, vol. 157, no. 4, pp. 819 – 832, 2009.
- [90] J. Mycielski, “Sur le colouage des graphes,” *Colloquium Mathematicum*, vol. 3, pp. 161–162, 1955.
- [91] J. Miskuf, R. Skrekovski, and M. Tancer, “Backbone colorings and generalized mycielski’s graphs,” 2008.
- [92] M. Caramia and P. Dell’Olmo, “A lower bound on the chromatic number of mycielski graphs,” *Discrete Mathematics*, vol. 235, no. 1-3, pp. 79–86, 2001.
- [93] M. Caramia and P. Dell’Olmo, “Coloring graphs by iterated local search traversing feasible and infeasible solutions,” *Discrete Applied Mathematics*, vol. 156, no. 2, pp. 201 – 217, 2008.

- [94] P. C. B. Lam, W. Lin, G. Gu, and Z. Song, "Circular chromatic number and a generalization of the construction of mycielski," *J. Comb. Theory Ser. B*, vol. 89, no. 2, pp. 195–205, 2003.
- [95] M. Larsen, J. Propp, and D. Ullman, "The fractional chromatic number of mycielski's graphs," *J. Graph Theory*, vol. 19, pp. 411–416, 1995.
- [96] V. Chvatal, "Coloring the queen graphs," 2004. Web repository (last visited July 2005).
- [97] K. Mizuno and S. Nishihara, "Toward ordered generation of exceptionally hard instances for graph 3-colorability," in *Discrete Applied Mathematics archive*, vol. 156, pp. 1–8, January 2008.
- [98] H. Chang and X. Zhu, "Colouring games on outerplanar graphs and trees," *Discrete Mathematics*, vol. 309, no. 10, pp. 3185 – 3196, 2009.
- [99] P. Petrosyan, H. Arakelyan, and V. Baghdasaryan, "A generalization of interval edge-colorings of graphs," *Discrete Applied Mathematics*, vol. 158, no. 16, pp. 1827 – 1837, 2010.
- [100] P. Petrosyan, "Interval edge-colorings of complete graphs and n-dimensional cubes," *Discrete Mathematics*, vol. 310, no. 10-11, pp. 1580 – 1587, 2010.
- [101] R. J. Kang and T. Muller, "Frugal, acyclic and star colourings of graphs," *Discrete Applied Mathematics*, vol. In Press, Corrected Proof, pp. –, 2010.
- [102] M. Crochemore and R. Verin, "On compact directed acyclic word graphs," in *A Selection of Essays in honor of Andrzej Ehrenfeucht* (A. S. Jan Mycielski, Grzegorz Rozenberg, ed.), vol. 1261 of *Lecture Notes in Computer Science*, pp. 192–211, Springer Verlag, 1997.
- [103] A. Raspaud and E. Sopena, "Good and semi-strong colorings of oriented planar graphs," *Information Processing Letters*, vol. 51, no. 4, pp. 171 – 174, 1994.
- [104] S. D. Nikolopoulos, "Coloring permutation graphs in parallel," *Discrete Applied Mathematics*, vol. 120, no. 1-3, pp. 165 – 195, 2002.
- [105] K. Asdre, K. Ioannidou, and S. D. Nikolopoulos, "The harmonious coloring problem is np-complete for interval and permutation graphs," *Discrete Applied Mathematics*, vol. 155, no. 17, pp. 2377 – 2382, 2007.

- [106] Z. Zhang and H. Li, "Algorithms for long paths in graphs," *Theoretical Computer Science*, vol. 377, no. 1-3, pp. 25 – 34, 2007.
- [107] P. Pardalos and A. Migdalas, "A note on the complexity of longest path problems related to graph coloring," *Applied Mathematics Letters*, vol. 17, no. 1, pp. 13 – 15, 2004.
- [108] H. Furmanczyk, A. Kosowski, B. Ries, and P. Zylinski, "Mixed graph edge coloring," *Discrete Mathematics*, vol. 309, no. 12, pp. 4027 – 4036, 2009.
- [109] F. Herrmann and A. Hertz, "Finding the chromatic number by means of critical graphs," *Electronic Notes in Discrete Mathematics*, vol. 5, no. 0, pp. 174 – 176, 2000.
- [110] D. Costa and A. Hertz, "Ants can colour graphs," *The Journal of the Operational Research Society*, vol. 48, pp. 295–305, March 1997.
- [111] W. Gutjahr, "Mathematical runtime analysis of aco algorithms: survey on an emerging issue," *Swarm Intelligence*, vol. 1, pp. 59–79, 2007.
- [112] F. Ge, Z. Wei, Y. Tian, and Z. Huang, "Chaotic ant swarm for graph coloring," in *Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on*, vol. 1, pp. 512 –516, 2010.
- [113] Z. Lu and J. Hao, "A memetic algorithm for graph coloring," *European Journal of Operational Research*, vol. 203, no. 1, pp. 241 – 250, 2010.
- [114] K. A. Dowsland and J. M. Thompson, "An improved ant colony optimisation heuristic for graph colouring," *Discrete Appl. Math.*, vol. 156, pp. 313–324, February 2008.
- [115] T. N. Bui, T. H. Nguyen, C. M. Patel, and K. T. Phan, "An ant-based algorithm for coloring graphs," *Discrete Applied Mathematics*, vol. 156, no. 2, pp. 190 – 200, 2008.
- [116] G. Cui, L. Qin, S. Liu, Y. Wang, X. Zhang, and X. Cao, "Modified pso algorithm for solving planar graph coloring problem," *Progress in Natural Science*, vol. 18, no. 3, pp. 353 – 357, 2008.
- [117] L. Hsu, S. Horng, and P. Fan, "Mtpso algorithm for solving planar graph coloring problem," *Expert Syst. Appl.*, vol. 38, pp. 5525–5531, May 2011.

- [118] M. Graña, B. Cases, C. Hernandez, and A. D’Anjou, “Further results on swarms solving graph coloring,” in *ICCSA 2010 Part III* (D. T. et al., ed.), no. 6018 in LNCS, pp. 541–551, Springer, 2010.
- [119] I. Rebollo and M. Graña, *Gravitational Swarm Approach for Graph Coloring*, vol. 387 of *Studies in Computational Intelligence*. Springer-Verlag, 2011.
- [120] I. Rebollo and M. Graña, “Further results of gravitational swarm intelligence for graph coloring,” in *Nature and Biologically Inspired Computing*, 2011.
- [121] X. Xie and J. Liu, “Graph coloring by multiagent fusion search,” *Journal of Combinatorial Optimization*, vol. 18, pp. 99–123, 2009.
- [122] M. Demange, T. Ekim, and D. de Werra, “A tutorial on the use of graph coloring for some problems in robotics,” no. 1.
- [123] J. Wu, L. Jiao, R. Li, and W. Chen, “Clustering dynamics of nonlinear oscillator network: Application to graph coloring problem,” *Physica D: Nonlinear Phenomena*, vol. 240, no. 24, pp. 1972 – 1978, 2011.
- [124] M. Gamache, A. Hertz, and J. O. Ouellet, “A graph coloring model for a feasibility problem in monthly crew scheduling with preferential bidding,” *Computers & Operations Research*, vol. 34, no. 8, pp. 2384 – 2395, 2007.
- [125] M. Buck and C. L. Nehaniv, “Communication and complexity in a grn-based multicellular system for graph colouring,” *Biosystems*, vol. 94, no. 1-2, pp. 28 – 33, 2008.
- [126] P. M. Talaván and J. Yáez, “The graph coloring problem: A neuronal network approach,” *European Journal of Operational Research*, vol. 191, no. 1, pp. 100 – 111, 2008.
- [127] G. Lewandowski and A. Condon, “Experiments with parallel graph coloring heuristics and applications of graph coloring,” pp. 309–334, 1994.
- [128] R. Lewis and J. Thompson, “On the application of graph colouring techniques in round-robin sports scheduling,” *Computers & Operations Research*, vol. 38, no. 1, pp. 190 – 204, 2011.

- [129] C. Bron and J. Kerbosch, "Algorithm 457: finding all cliques of an undirected graph," *Commun. ACM*, vol. 16, pp. 575–577, September 1973.
- [130] I. Rebollo, M. Graña, and C. Hernandez, "Aplicacion de algoritmos estocasticos de optimizacion al problema de la disposicion de objetos no-convexos," *Revista Investigacion Operacional*, vol. 22, no. 2, pp. 184–191, 2001.
- [131] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *Annals of Mathematical Statistics*, vol. 11, pp. 86–92, March 1940.
- [132] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the American Statistical Association*, vol. 32, pp. 674–701, 1937.
- [133] P. B. Nemenyi, *Distribution-free multiple comparisons*. PhD thesis, Princeton University, 1963.
- [134] K. Pearson, "On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling," *Philosophical Magazine*, vol. 50, no. 5, p. 157.175, 1900.
- [135] D. J. Iman, R.L., "Approximations of the critical region of the friedman statistic," *Communications in Statistics*, pp. 575–595, 1980.
- [136] G. W. Snedecor, "The method of expected numbers for tables of multiple classification with disproportionate subclass numbers," *Journal of the American Statistical Association*, vol. 29, no. 188, pp. 389–393, 1934.
- [137] R. Fisher, "Frequency distribution of the values of the correlation coefficient in samples from an indefinitely large population," *Biometrika*, vol. 10, pp. 507–521, 1915.
- [138] J. W. Tukey, "Comparing individual means in the analysis of variance," *Biometrics*, vol. 5, pp. 99–114, 1949.
- [139] H. Hernández and C. Blum, "Ant colony optimization for multicasting in static wireless ad-hoc networks," *Swarm Intelligence*, vol. 3, pp. 125–148, 2009.

- [140] L. Hu, "Distributed code assignments for cdma packet radio networks," *Networking, IEEE/ACM Transactions on*, vol. 1, pp. 668–677, dec 1993.
- [141] S. Y. L. M. W. Carter, G. Laporte, "Examination timetabling : Algorithmic strategies and applications," *The Journal of the Operational Research Society*, vol. 47, no. 3, pp. 373–383, 1996.
- [142] F. T. Leighton, "Finite common coverings of graphs," *Journal of Combinatorial Theory, Series B*, vol. 33, pp. 231–238, December 1982.
- [143] C. Fleurent and J. Ferland, "Genetic and hybrid algorithms for graph coloring," *Annals of Operations Research*, vol. 63, pp. 437–461, 1996.
- [144] M. Gardner, *The Unexpected Hanging and Other Mathematical Diversions*. Chicago Illinois: The University of Chicago Press, 1969. ISBN 0-226-28256-2.
- [145] M. Carter, G. Laporte, and S. Lee, "Examination timetabling: algorithmic strategies and applications," *Journal of the Operational Research Society*, vol. 47, pp. 373–383, 1996.