

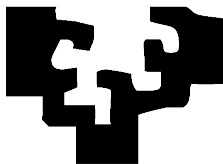
Gravitational Swarm for Graph Coloring

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. Professor Manuel Graña believed in me when I did not believe in myself. 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. Especially people who in one way or another have helped me.

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. My sister Afrika and my brother Txus for being my best friends. 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 July 19, 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 for Graph Coloring (GS-GC) 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 GS-GC stationary states correspond to GCP solutions. The Thesis provides also extensive empirical support of the GS-GC 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	14
2.2.2	Ant Colonies	15
2.2.3	Particle Swarms	17
2.2.4	Gravitational swarm	17
2.3	Graph Coloring Algorithms	17
2.3.1	Classical algorithms	18
2.3.2	Approximate algorithms	19
2.3.2.1	Graph Families	22
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
3.2.1	Special BT initialization	28
3.3	DSATUR	29
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 for Graph Coloring (GS-GC)	36
4	Gravitational Swarm Intelligence	39
4.1	Gravitational Swarm for GCP	40
4.2	Graph coloring problem	45
4.3	Gravitational Swarm	45
4.4	Gravitational Swarm for GCP	48
5	Parameter tuning	53
5.1	GS-GC model parameters	53
5.2	Experimental results on 30-50 KRG graphs	54
5.2.1	Chromatic number	55
5.3	Goal Radius	56
5.4	Comfort	59
5.5	Nom Parametric Tests	62
5.5.1	Friedman test	63
5.5.2	Friedman test to GS-GC	63
5.5.2.1	Goal Radius	64
5.5.2.2	Comfort	65
5.5.3	Post-Hoc test: Nemenyi's test	66
5.6	Concluding remarks	70
6	Graph Coloring Results	71
6.1	Experimental design	71
6.2	Trees and bipartite graphs	73
6.3	Kuratowski based planar graphs	74
6.4	Mizuno's 3-colorable	76
6.5	KRG graphs	79
6.6	DIMACS graphs	88
6.6.1	Test	92
6.7	Sequential chromatic number determination	93

6.7.1	Random Graphs	94
6.7.2	Leighton graphs	96
6.7.3	Real Graphs	97
6.8	Conclusions	98
7	Conclusions and further work	99
7.1	Future works	100
A	Graph experimental instances and graph generators	103
A.1	Introduction	103
A.2	Trees and bipartite graphs	104
A.2.1	Trees	104
A.2.2	Bipartite	104
A.3	DIMACS	105
A.3.1	Mycielski graphs	105
A.3.2	Queens Graphs	106
A.3.3	Miles graphs	107
A.3.4	Fullins graphs	107
A.3.5	Books graphs	108
A.3.6	Leighton Graphs	109
A.4	Random Graphs	109
A.5	Mizuno's Graphs	110
A.6	Planar Graphs	110
A.7	KRG graphs	112
A.8	Real Graphs	112
B	Graph Coloring Suite	115
	Bibliography	119

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	Simplified Flowchart	43
4.2	GS-GC agent behavior flowchart for GCP	44
5.1	Average success ratio vs Goal Radius	57
5.2	Average success ratio vs Goal Radius in 3D	58
5.3	Average Steps ratio vs Goal Radius	59
5.4	Average success ratio vs Comfort	60
5.5	Average success ratio vs Comfort in 3D	61
5.6	Average steps ratio vs Comfort	62
5.7	Nemenyi's diagram for 9 goal radius and 90% of acceptance . . .	67
5.8	Nemenyi's diagram for 9 goal radius and 95% of acceptance . . .	67
5.9	Nemenyi's diagram for 9 goal radius and 99% of acceptance . . .	67
5.10	Nemenyi's diagram for 5 goal radius and 95% of acceptance . . .	68
5.11	Nemenyi's diagram for 5 goal radius and 99% of acceptance . . .	68
5.12	Nemenyi's diagram for 6 comfort values and 95% of acceptance .	69
5.13	Nemenyi's diagram for 6 comfort values and 99% of acceptance .	69
5.14	Nemenyi's diagram for 5 comfort values and 90% of acceptance .	70
6.1	Trees and bipartite success ratio	73
6.2	Trees and bipartite average time in seconds	74
6.3	Trees and bipartite average number of steps	75
6.4	Kuratowski based graphs success ratio	75

6.5	Kuratowski based graphs average time in seconds	77
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	81
6.11	KRG average time in seconds	83
6.12	KRG average number of steps	84
6.13	Big KRG success ratio	85
6.14	Big KRG average time in seconds	86
6.15	Big KRG Average number of steps	87
A.1	Mycielski graph transition from M_3 to M_4	106
A.2	8 queens problem solution	107
A.3	United States Cites Graph	108
A.4	Books where the Books Graphs come from	109
A.5	Mizuno's MUGs for 3-colorable graphs generation [93]	111
A.6	Aleatory graphs, Kuratowski's planar graphs, Mizuno's 3-colorable graphs and KRG new developed graph type generator.	113
B.1	Graph Coloring Suite	116
B.2	Snapshot of a result file	117

List of Tables

5.1	Average accuracy results of GS-GC on the 20 KRG generated graphs of 30 nodes, 50 edges and 6 colors for varying comfort (rows) and radius (columns) parameter values	55
5.2	Friedman ranking for Goal Radius	64
5.3	Friedman ranking for Comfort	64
6.1	Layout of the experimental graphs	90
6.2	Results of SGB graphs	91
6.3	Results of CAR graphs	92
6.4	Algorithm Rankings for Friedman Test	93
6.5	Graphs of 100 nodes and 1000 edges. Between parentheses the minimum solution found.	94
6.6	Graphs of 100 nodes and 2000 edges. Between parentheses the minimum solution found.	95
6.7	Graphs of 100 nodes and 3000 edges. Between parentheses the minimum solution found.	95
6.8	Graphs of 100 nodes and 4000 edges. Between parentheses the minimum solution found.	95
6.9	Leighton Graphs results	97
6.10	Exam timetabling of Lewis [82] plus GS-GC	97

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

It is a well known fact of scientific research and other creative activities that the actual accomplished results of the work are an approximation to the fuzzy expectations set at the very beginning. Therefore, in this section we summarize what we identify as the main actual contributions of this Thesis:

1. The proposal of an innovative approach to the Graph Coloring Problem following the a Swarm Intelligence approach. Specifically, the proposed approach combines the mechanics of gravitation and other physical forces with the flocking behavior of birds. We call this approach Gravitational Swarm for Graph Coloring (GS-GC).

2. We have been able to prove that stationary states of the GS-GC correspond to solutions of the GCP. This proof does not deal with the dynamics of convergence to such stationary states. Dynamical convergence to stationary states is still an open research question for GS-GC.
3. We have performed an extensive computational evaluation of GS-GC, both regarding numerical sensitivities to parameter settings and comparison with state of the art algorithms for GCP.

From an operational point of view, the Thesis provides instrumental contributions allowing to perform the validation of GS-GC, and to have independent confirmation of our claims by independent researchers:

1. We have defined a new Graph class called KRG, and a constructive procedure to generate KRG graphs with a known chromatic number. This procedure allows us to generate arbitrarily difficult problem instances.
2. 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
3. Public implementations of both the graph suite generator and competing GCP solving algorithms.

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 $CG = \{g_1, \dots, g_K\}$
<i>nearencough</i>	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
<i>repulsion</i>	The opposition force exerted by a pair of agents with a link and the same color
$R(b_i, g_k)$	Repulsive forced exerted between the agent b_i and agents that repel 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 numer if cicles were an agents has a color without having repulsion forces in it's neighborhood
λ	Represent the probability of an agent to be expelled from a goal
<i>maxcomfort</i>	Is the maximum confort 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 and an intuitive description of the main SI approaches that have been used for GCP. The chapter contains a review of current approaches to solve the GCP.

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 adjacent nodes linked to it: $\mathcal{N}(v) = \{w \in V \mid (v, w) \in E\}$.

The Graph Coloring Problem (GCP), aka Graph Labeling, aka Node Coloring in graph theory, is an assignment of colors to nodes of a graph subject to the constraint that two adjacent nodes don't share the same color. The coloring of a graph with a fixed set of colors can have multiple solutions. The chromatic number $\chi(G)$ is the minimum number of colors that can be used to color the graph. Even minimal colorations may not be unique, in fact any permutation

of the color assignments is equally satisfactory.

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.

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,* denoted $P(G, k)$, is the number of possible coloring solutions of graph G using a given number k of colors.

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

Definition 6. *Total Coloring* is the 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.

Theorem 13. *Kuratowski. 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.

Theorem 14. *Four color theorem: any planar graph is 4-colorable.*

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 [42] 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".

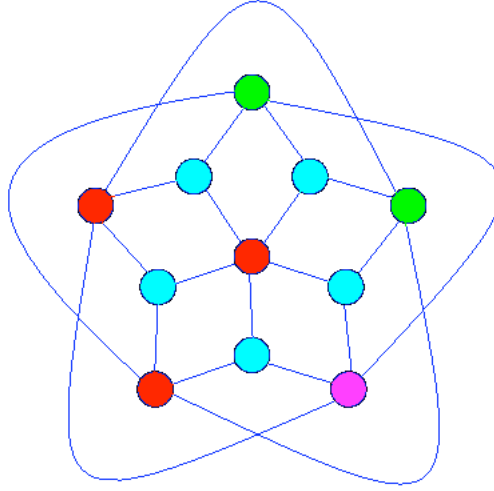


Figure 2.1: The Grötzsch Graph

2.2.1 Flocking behaviors

The inspiration in flocking birds to design computational systems appears in [119], where we first find the application of simple behavior rules followed autonomously by individuals of a group of virtual animals, in this case birds, to simulate the complex behavior of flocks, such as following the leader. Since then, simulation of flocking behavior has been used to control the navigation of self organized mobile multi-robot systems [2, 133]. El-abd [38] created a Particle Swarm Optimization (that we explain latter) based in flocking behavior.

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



Figure 2.2: Flocking Birds

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 (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 [21, 32] have found inspiration in the ant colonies. Briefly,

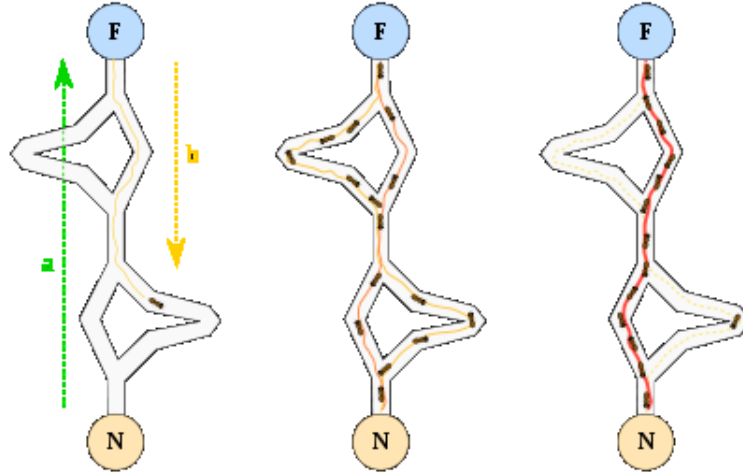


Figure 2.3: Ants using pheromones to find the shortest path

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 [66]. The first problem where the ant colony optimization (ACO) has been shown to provide feasible solutions was the classical Travel Salesman Problem (TSP) [32]. There are a lot of versions of ACO solving this problem, and ensuing applications, i.e. tracing the route of a vehicle [122]. Balaprakash [4] presents an ACO probabilistic TSP.

ACO can solve more problems, not only the TSP. Franks [43] show ants deciding how to get to moving targets. The ligand of proteins using the PLANTS (Protein Ligand ANT System) algorithm of Korb [73]. The formation of paths on multi-robot systems is approached with ACO in [100]. Ant clustering with locally weighted where the ants has some memory is discussed [106]. A reinforcement learning algorithm based algorithm can be found in [9]. More about trail formation and TSP in Shah [123]. The list is endless, and continuously 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 [65], additional convergence is discussed in [39]. The essential PSO is introduced in [109], but there are a lot of variants with specific convergence properties, such as [118], or the ensemble PSO [34]. Applications are not restricted to combinatorial problems, for instance [83] applies PSO in steganographic JPEG images .

2.2.4 Gravitational swarm

The Swarm computing natural inspiration is not necessarily coming from living beings. Rashedi [113] 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 [7] where we can find information about this problem, and more about

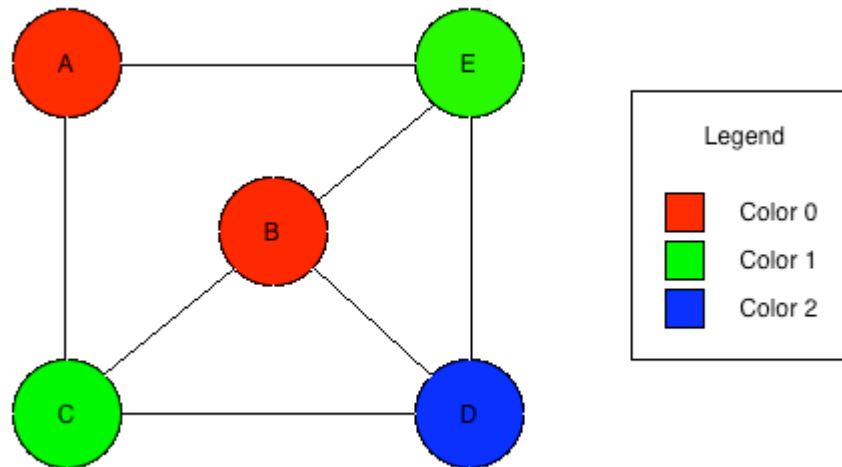


Figure 2.4: An instance of a Graph Coloring Problem solution.

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 also present some graph families with special features, that help to understand the different approaches for this problem. Then we show algorithms based in SI approach, although we are going to explain the GS-GC in more detail in a separate chapter.

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 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 sets the stage for a lot of GCP solving algorithms. Specifically, the contraction algorithm is based on Zykov's theorem of contraction [145], aims to reduce the complexity of a graph

reducing the computational time need to solve it.

Theorem 15. *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 a recent Odaira's paper [101]. In an earlier work, Corneil [24] developed an algorithm which searches through the Zykov tree in a depth-first manner. Dutton [36] searches non-adjacent nodes with a maximal number of common neighbors to contract until a complete graph is achieved. In Leighton's Recursive-Largest-First (RLF) algorithm [78] the contraction affects the largest path between nodes. Palubeckis present a more actual version of this algorithm in [102]. A modified LF-algorithm (Largest First) presented by Hansen in [56] is an optimization of RLF where the recursive technique is changed to a sequential way.

The most famous algorithm was developed by Brelaz [11] 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 [128], it is still a reference algorithm. Turner [134] 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 it's draw falls. Wood in 1997 [138] and more recently Mendez-Diaz [96] 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 theorem 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 [135] and Shanon [124] are used to achieve this.

Theorem 16. *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 17. *Shanon:* For any multigraph G , $\chi(G) \leq \lfloor \frac{3}{2}\Gamma(G) \rfloor$.

The COSINE algorithm by Hertz [59] and the Clique Covers (CC) algorithm of Klotz [71] are two examples of this kind of approaches. The authors of COSINE have published an updated work in [60]. The COSINE algorithm first tries to find an upper 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 [13], whose work was improved by Brelaz to make his DSATUR, Wilf [137] showed that if the number of color approaches infinity the order of Backtracking is $O(1)$, and Park [104] presented a new version and an application to network security.

The simulated annealing has been applied to GCP recently by Titiloye [131] using the Monte-Carlo path-integral. Simulated annealing method has been also used by Johnson [67] showing empirical result to the GCP. Nolte [99] focus the use of Simulated Annealing to 3-colouring problem. Bonomo [8] 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 [120] 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 [48] proposed in 1987 that today is still present in a lot of evolutionary and hybrid algorithm by its performance in local search. Blochliher [6] used a partial solutions method based in a Tabu Scheme for local search. The hybridization mention before can be seen in Mabrouk [87] presenting a parallel Genetic-Tabu algorithm for the GCP. Qu [112] 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 [63] proposed an heuristic for interval graphs, permutation graphs and trapezoid graphs, the interval graph is also presented in Yu [143] who de-

velops a parallel algorithm. Gaun [54] works over weighted graphs a particular problem that needs different approaches to solve it like the first-fit algorithm. Vredeveld [136] also works over weighted graphs focused in local search. Bouchard [10] 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 [31] 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 [121]. 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 [16]. Daniel [37] works on 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 [23]. 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 [88]. Yadav reduced the problem to acyclic node coloring of graphs of maximum degree 5[141]. Galinier and Hertz [47] present a more general algorithm for GCP using a memory based algorithm different from their famous Tabucol algorithm. Costa [26], 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 used in a large number of works. Marino [90] 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 [125] present a Genetic Algorithm for GCP. Bi-man develops a genetic algorithm with a new operator called Multipoint Guided Mutation [5]. Porumbel [111] presents a search space analysis for improving local search is GCP and solve it with a Genetic algorithm that appears in [110]. A parallel technique in genetic algorithm is also common like Sivanandam [126] who present a hybrid parallel genetic algorithm approach, Kokosiski [72] 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 [142] 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[129, 89].

There are different strategies like Abasian's[1] that uses a non-systematic

method based on a cultural algorithm to solve the GCP. Dukanovic[35] indicates a way to find the lower bound of a chromatic number. Mehrotra [91] 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[52] present a algorithm for fuzzy graphs.

2.3.2.1 Graph Families

Not all the graphs present the same complexity to be colored. Some times we can benefit from special graph features that help in the process of finding the chromatic number, as we have saw in the previous section. There was a challenge in 1993, the second DIMACS challenge [68, 69], where a lot of graphs were offered 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. For instance, that fact that planar graphs are 4 colorable [74, 75], eases the work on coloration of planar graphs [85][29]. In Appendix A we recall the Kuratowki's theorem.

A famous family of graphs is Mycielski graphs [95], 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 [92, 17]. But even though are a good starting point in the GCP, they are very easy graphs. Caramia [18] 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[76, 77].

Other classic graph family 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 [22].

Mizuno [93, 94] have generated 3 colorable graphs that are hard to solve. Using Mizuno's method we can build graphs which are very hard to color, although their chromatic number is only 3.

Other graph type can be found in Chang [20] and Petrosyan [108, 107] describing outer-planar graphs and trees. 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 [70, 71]. Less tested graph families found in the literature are also important, such as the one reported in [27] inspired in the DNA. The GCP is usually focused in undirected graphs, but directed graph can be colored as well [114].

Permutation graphs are treated in [98] and [3]. Graphs with long paths are solved in [144], but specific problems can appear solving these graphs as noted in [103]. Furmanczyk [46] used mixed graphs with directed and non directed edges. Herrmann in [58] deals with critical graphs.

2.3.3 Swarm Intelligence for Graph Coloring

As we have mentioned, the SI has been applied to the solution of a lot of mathematical problems, including 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 [25] where the ants are used to perform the coloring of a graph. We can see a survey of ACO applications in [55]. An algorithm using a chaotic ant colony is proposed in [51]. Borkar[9] introduces an incremental learning component. Lu[84] introduces a memetic algorithm for graph coloring also using ants. Dowsland [33] have improve the classical ACO algorithm. Another ant based algorithm appears in Bui [15].

The PSO has been used in Cui [28] for GCP where a modified PSO using disturbances is used, obtaining better results than standard PSO in planar graphs. Hsu [61] adding a modified turbulence to previous PSO, obtained better results in 4-colorable graphs solving them efficiently and accurately.

The use of flocking behavior SI has been forgotten, but SI is powerful enough to solve the GCP. We demonstrate [53] that SI can color graphs. Adding the gravitational law and a method for escaping from local optimum we build a competitive algorithm based in flockng SI [116, 115]. In the SI category we can include agent based algorithm like [140], but there are no more references.

2.4 GCP Applications

The GCP can be applied to a wide number of areas. In Demange [30] is applied in robotics. Clustering dynamics of nonlinear oscillator network using graph

coloring in Wu [139]. A monthly crew scheduling problem with preferential bidding in the airline industry is solved in Gamache [49]. Communication protocols in Buck [14].

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 [130] to solve it.

Experiments with graphs are shown in Lewandowski [80] applied to scheduling. Lewis [81] 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 Mycielski 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. Looking for smart initializations that may help to obtain solutions faster, we propose an initialization approach that looks for the biggest clique in the graph, using it to fix the initial number of colors needed to color the graph. Then starts the BT search to determine the coloring of the remaining nodes of the graph. The clique of a graph [12] 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.
3. DSATUR (Degree of Saturation): this algorithm developed by Br elaz [11] is a greedy color assignment algorithm which goes only once on each node. The algorithm tries to color first the highest degree nodes, as far as they are disconnected.
4. 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 [111]. 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.
5. Simulated Annealing [117]: 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[99].
6. Ant Colony Optimization (ACO): we have build an implementation following [51] 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.

7. Particle Swarm Optimization (PSO): we have built a PSO version of [61] 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.
8. Gravitational Swarm for Graph Coloring (GSGC): this algorithm is inspired in the Gravitation physic law of Newton, and the Boids swarm of Reynolds [119]. The gravitation law has been previously used in Swarm Intelligence for function minimization [113], unrelated to the GS-GC formulated as GSI in [116]. This algorithm does not try to mimic exactly a physical system obeying Newton's law. The GS-GC 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 agents that repel. 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 agents that repel settled in that goal, been an agent that repel a node that can't have the same color, then the agent tries to expel the agent that repel outside the goal to a random position before going itself inside. If it is no able to expel the agent that repel 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 decreasing 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 performs an orderly recursive search of color node assignments on the graph looking for a graph coloring with the required number of colors. The algorithm is exhaustive, therefore if any solution exists it will be found, but its computational cost is exponential in the number of nodes. If the order of the nodes doesn't change, the algorithm always returns the same coloring solution. If we try to solve the chromatic number problem, then all the search space must be explored. An intuitive description of the algorithm is as follows: each graph node can be viewed as a level of a search tree: Tree nodes at this level correspond to color assignments to this node. This tree is explored in a depth-first procedure. At each tree node, an evaluation of the correctness of the color assignment as a graph coloring is performed, if the test fails (i.e. two adjacent nodes have the same color) the search backtracks, pruning the subsequent subtree from the search. The order of picking the graph nodes may influence the search time employed to find a solution for the first time. 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.

Algorithm 3.1 provides an iterative specification of the BT, equivalent to a recursive search. Let us introduce some relevant notation, $G = (V, E)$ is the graph to be colored, whose nodes are enumerated in an arbitrary order $V = \{1, \dots, N\}$, $A_i = \{j \mid (i, j) \in E\}$ is the set of nodes adjacent to node $i \in V$, $W_i \subseteq C$ is the set of colors that have been tried on node i , $C = \{1, \dots, M\}$ is the set of colors, and c is the current color test.

3.2.1 Special BT initialization

Finding the biggest clique, we can start the process assigning colors to the nodes in the clique so the algorithm needs less backtracks than the original BT. It is important to order the nodes according to the clique nodes. This

Algorithm 3.1 Backtracking algorithm for graph coloring

Set initial values $\forall i \in V; W_i = \emptyset; \mathcal{C}(i) = 0$.

initialize $i = 1, c = 1$,

while $1 \leq i \leq N$

set $\mathcal{C}(i) = c$

if $\forall j \in A_i, \mathcal{C}(j) \neq \mathcal{C}(i)$ /test coloring correctness/

then $i \leftarrow i + 1$

elseif $c < M$

then $c \leftarrow c + 1$

else $W_i = \emptyset, \mathcal{C}(i) = 0$,

backtrack $i \leftarrow i - 1, c \leftarrow \mathcal{C}(i) + 1, W_i \leftarrow W_i \cup \mathcal{C}(i)$

endelseif

endwhile

if $i = 0$ the algorithm has failed.

Algorithm 3.2 Clique initialization of BT graph coloring

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 $|V_{BC}|$ colors.

3. Apply the BT algorithm to

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, $|V_{BC}|$ the cardinality of V_{BC} and $k \in \{1, 2, \dots, C\}$ set of colors, where C must be $C \geq |V_{BC}|$.

3.3 DSATUR

The DSATUR algorithm is a greedy color assignment algorithm whose underlying reasoning is that coloring first high degree nodes it is easier to find a correct

Algorithm 3.3 DSATUR graph coloring algorithm

```

set  $D_i = d_i; \mathcal{C}(i) = 0; \forall i \in V; V_1 = V; c^* = 1$ 

sort nodes in  $V_1$  such that  $D_i \geq D_{i+1}$ 

init  $i = 1$ 

while  $i \leq N$ 

    compute  $R_i = \{\mathcal{C}(j); j \in A_i \wedge \mathcal{C}(j) > 0\}$ 
    if  $D_i > |R_i|$  and  $|R_i| > 0$ 
        then  $D_i = |R_i|$ 
        sort nodes in  $V_i$  such that  $D_i \geq D_{i+1}$ 
    elseif  $|R_i| = 0$ 
        then  $\mathcal{C}(i) = 1$  /any used color/
        elseif  $|R_i| = c^*$ 
            then  $c^* = c^* + 1; \mathcal{C}(i) = c^*; V_{i+1} = V_i - \{v_i\}; i = i + 1;$ 
            else  $\mathcal{C}(i) = \min \{c^* - k; k \in R_i\}; V_{i+1} = V_i - \{v_i\}; i = i + 1;$ 
        endelseif
    endif

endifor

```

coloring minimizing the number of colors. There algorithm assigns colors to nodes in the order of the degree of saturation, which is either the node's degree for nodes whose adjacent nodes have not been colored, or the number of colored adjacent nodes. The algorithm does not search for a coloring with a restricted set of colors, but for the minimum number of colors, the chromatic number. Algorithm 3.3 provides an specification of the algorithm. Let be $d_i = |A_i|$ the node's degree, D_i the variable containing the degree of saturation, V_i denotes the remaining nodes to be colored, so that $V - V_i$ is the set of nodes that have already been colored.

Theorem 18. [11] *The DSATUR algorithm is exact (provides the chromatic number) for bipartite graphs.*

3.4 Tabu Search (TS)

We have implemented a straightforward application of the Tabu Search to graph coloring, where we consider the complete color assignment to the graph as the current tested solution. The tabu list stores color assignments, and the generation of a new solution involves changing some color assignment of a node. It is managed as a First-In-First-Out list, regardless of the solution quality. The solution quality $Q(\mathcal{C}_t(i))$ is the number of nodes whose color assignment is not correct, i.e.

$$Q(\mathcal{C}_t(i)) = |V - \{i | \forall j \in A_i; \mathcal{C}_t(j) \neq \mathcal{C}_t(i)\}|, \quad (3.1)$$

where A_i is the set of adjacent nodes to node i . The Tabu Search tries to minimize this energy-like function. This approach is different from the TabuCol algorithm [6, 48] where each node maintains an independent tabu list of color transitions. Our algorithm's computational flow is as follows:

- In the Initialization phase, we assign randomly colors to graph nodes for the set of M colors.
- Perform a number of iterations controlled by the time counter t , at each iteration perform
 - the generation of a new solution \mathcal{C}' from the previous iteration solution $\mathcal{C}_t(i)$, changing color of a node whose coloring is not correct. If this new solution is already in the tabu list then repeat the generation of a new solution until it is not in the tabu list. In Algorithm 3.4 we denote the tabu list at iteration L by L_t .
 - If the quality function of the new solution \mathcal{C}' is better (lower) or equal than the previous solution, then keep it and store the previous solution in the tabu list.
- If the assigned colors produce a successful graph coloring then iteration search is stopped.

The tabú list has a limit size, the tabu tenure, if we exceed the tabu tenure, then we delete the older instances of the Tabu tenure. In our current implementation, 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. As it is the algorithm avoids getting stuck in local minima because we can't repeat a color that has been previously used and also we change dynamically

Algorithm 3.4 Tabu Search for graph coloring

Generate an initial random solution $C_t(i) \in \{1, \dots, M\}; \forall i \in V, t = 0$.

set the initial tabu list $L_t = \emptyset$

for $t = 1, \dots, \text{maxiter}$

Generate C' a new solution from $C_{t-1}(i)$ s.t. $C' \notin L_{t-1}$

if $Q(C_{t-1}(i)) \geq Q(C'(i))$

then $C_t(i) = C'(i); L_t = L_{t-1} \cup C_{t-1}(i)$

if $Q(C_t(i)) = 0$ the search stops finding a correct coloring

endfor

the bad colored node. The description of the algorithm is given in Algorithm 3.4. The computational cost of this algorithm grows with the Tabu tenure and the number of graph edges, because of memory needed to keep the list, and time to travel through the list at each step of the algorithm to test if the solution has been already tested.

3.5 Simulated Annealing (SA)

We have implemented our own version of the classical Simulated Annealing (SA) algorithm [67, 117] tailored to the GCP. The cooling schedule is a very fast one, the temperature parameter β is the number of remaining computational steps. Thus the algorithm stops when the temperature reaches to zero if it hasn't find a graph coloring solution before. Algorithm 3.5 provides an specification of the SA algorithm. The energy function to be minimized is the coloring quality of equation (3.1). In this algorithm p denotes a random value between 0 and 1 generated to perform the decision of acceptance/rejection of the new solution. As in the Tabu search algorithm, candidate new solutions are produced by randomly changing the color assignment of a randomly picked node. Notice in the specification of the acceptance decision, that if the new candidate solution improves the previous one the exponent will be positive and the candidate solution will always be accepted.

Algorithm 3.5 Simulated Annealing for graph coloring

Generate an initial random solution $C_t(i) \in \{1, \dots, M\}; \forall i \in V, t = 0$.

for $t = 0, \dots, \text{maxiter}$

Generate C' a new candidate solution from $C_{t-1}(i); \beta = \text{maxiter} - t$

if $e^{-\frac{1}{\beta}(Q(C'(i)) - Q(C_{t-1}(i)))} > p$

then $Q(C_t(i)) = Q(C'(i))$

else $Q(C_t(i)) = Q(C_{t-1}(i))$

if $Q(C_t(i)) = 0$ successful stopping of the algorithm

endfor

3.6 Ant Colony Optimization (ACO)

We have defined and implemented an ACO algorithm for the GCP. Ants have two attributes: (1) a color that can not be changed, (2) the actual node where they are. Ant move from node to node of the graph across the existen graph edges, therefore they could reach any node from any other node that is connected to it by a path over the graph. IThe ACO algorithm starts creating N^2 ants, meaning that we create N ants for each one of the N nodes. Ants' intrinsic color is randomly drawn from the set of colors $C = \{1, \dots, M\}$. At each iteration dynamics of the ants over the graph are as follows:

1. For each node the color assignment is decided by majority voting of the ants placed in the node.
2. Ants that are in minority in the node, regarding the color assignment are sent to adjacent nodes. Target nodes whose color assignment is equal to the ant color are preferred, if there isn't any then the ant is moved randomly.
3. Nodes attract from the adjacent nodes the ants with the same color as their color assignment. Ties are solved in favor of the node with the maximum number of ants of the same color.
4. If a node is depleted of ants, then new N ants with random colors are generated for that vertice. Therefore the population of ants can grow unboundedly, though this is a very unlike situation.

Each iteration ends testing if the color assignment to the nodes is a correct graph coloring. Intuitively it is expected that this algorithm will converge to a

graph coloring, if the number of colors is above the chromatic number, because step #3 above ensures that no two adjacent nodes will have the same color assignment in a stationary state. Ants not contributing to the color assignment (minority ants) flow searching for nodes where they can stay. Step #4 ensures that no node will remain uncolored. If the graph's chromatic number is greater than M , then the algorithm will not converge to a stationary state, ants will keep moving indefinitely. A potential evolution of the system is an unbounded growth of the number of ants, but we have not found this behavior in our experiments. The algorithm stops after *maxiter* iterations. The amount of memory needed and the computational time of this algorithm can be very big, and code parallelization wouldn't help because we will need as many processes as ants, and the communication between process will be too much expensive. The detailed specification of the algorithm can be seen in Algorithm 3.6. In this algorithm, t is the iteration number, $a = (c_a, v_a) \in \mathcal{A}$ denotes an ant, where $c_a \in C$ is the ant's color, and $v_a \in V$ is the node where the ant is placed currently. Let us denote $\mathcal{A}_i = \{a \mid v_a = i\}$ the set of ants at node i , and \mathcal{A} the complete set of ants.

3.7 Particle Swarm Optimization (PSO)

The PSO is an innovative random search algorithm that is been applied to 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 defined and implemented a basic version of a PSO algorithm for GCP. Because the problem is discrete, the particle evolution equations will be applied to the probabilities of color assignment instead of the color assignment itself. Therefore, the PSO becomes a probabilistic PSO or Markovian PSO.

A particle corresponds to a node in the graph. Each particle's state is given by a probability distribution of the color assignment to the node:

$$s_i = \{p_1, \dots, p_M\},$$

where $p_k \geq 0$ and $\sum_k p_k = 1$. The global state of the system at time t is given by

$$S(t) = \{s_i(t); i \in V\}.$$

The evolution of the particles is guided by color assignment quality of equation

Algorithm 3.6 Ant Colony Optimization algorithm for graph coloring

Generate N random colored ants $a = (c_a, i) \in \mathcal{A}_i$ for each node $i \in V$

for $t = 0, \dots, \text{maxiter}$

compute $\mathcal{C}_t(i) = \arg \max_c |\{a | (c_a = c) \wedge (v_a = i)\}|$

 / migrate minority ants/

for $i = 1, \dots, N$

for each $a \in \mathcal{A}_i$ s.t. $c_a \neq \mathcal{C}_t(i)$

if $\exists j \in \mathcal{A}_i$ s.t. $c_a = \mathcal{C}_t(j)$

then $v_a = j$

else $v_a \in \mathcal{A}_i$ picked randomly

endfor

 / break conflicting colorings/

for each $(i, j) \in E$ s.t. $\mathcal{C}_t(i) = \mathcal{C}_t(j)$

if $|\mathcal{A}_i| > |\mathcal{A}_j|$

then $c_a = i; \forall a \in \mathcal{A}_j$

else $c_a = j; \forall a \in \mathcal{A}_i$

endfor

 / solve depleted nodes/

for each node i s.t. $\mathcal{A}_i = \emptyset$

generate N random colored ants $a = (c_a, i)$

endfor

if $Q(\mathcal{C}_t(i)) = 0$ stop report successful coloring

endfor

3.1. To determine the best local solution, we attend to the local version of this equation

$$Q(\mathcal{C}_t(i))|_j = |A_j - \{k | \forall k \in A_j; \mathcal{C}_t(j) \neq \mathcal{C}_t(k)\}|. \quad (3.2)$$

The dynamics of the system are as follows:

1. Particles' states are initialized as uniform probability distributions $\{p_k = \frac{1}{M}\}$.
2. Color assignments to nodes are performed sampling the local color distribution.
3. Global and local quality functions are computed, determining new local and global optimal solutions. The algorithm stops when a correct graph coloring has been reached.
4. Evolution of each particle state follows the conventional PSO equation.

The algorithm is presented in Algorithm 3.7, where s_i^l is the best local solution. $S^g = \{s_i^g; i \in V\}$ is the best global solution. Correspondingly, q_i^l is the best value of the local quality function of node i , and q^g is the best value of the global quality function. Parameters α_t and β_t control respectively the contribution of the local and global optima in the updating of the particle state. Finally, $r_i(t)$ denotes random additive term.

3.8 Gravitational Swarm for Graph Coloring (GS-GC)

For completeness, a pseudo-code of the Gravitational Swarm for Graph Coloring (GS-GC) algorithm for GCP is presented in Algorithm 3.8. Chapter 4 is devoted to a detailed description of the algorithm, as well as some convergence theoretical results.

Algorithm 3.7 Particle Swarm Optimization

Generate a initial state with uniform distributions per particle

$$S(0) = \left\{ s_i = \left\{ p_k = \frac{1}{M}; k = 1, \dots, M \right\}; i \in V \right\}.$$

initialize $S^g = S(0)$, $s_i^l = s_i(0)$, $q_i^l = q^g = \infty$ **for** $t = 1, \dots, \text{maxiter}$

/ evolve particles/

for each $i \in V$

$$s_i(t) = s_i(t-1) + \alpha_t s_i^l + \beta_t s_i^g + r_i(t)$$

endfor

/ compute graph color assignment/

for each $i \in V$ **obtain** $C_t(i)$ = sample of color probability distribution $s_i(t)$ **if** $q_i^l \geq Q(C_t(i))|_j$ **then** $q_i^l = Q(C_t(i))|_j$; $s_i^l = s_i(t)$ **endfor****if** $q^g \geq Q(C_t(i))$ **then** $q^g = Q(C_t(i))$; $S^g = S(t)$ **if** $Q(C_t(i)) = 0$ the algorithm stops successfully**endfor**

Algorithm 3.8 Gravitational Swarm for Graph Coloring

1. deploy goal colors GC and agents V randomly in the space.

2. assign a random position to agents that repel 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 for Graph Coloring (GS-GC) 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 GCP

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 GS-GC 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 $p_i(0) = (x_{0i}, y_{0i})$. Suppose 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 GS-GC 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.

GS-GC definition: We can model the system as a tuple

$$F = (B, CG, \{\vec{v}_i\}, K, \{\vec{a}_{i,k}\}, R) \quad (4.1)$$

where B is the set of GS-GC 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.

Agent velocity: The dynamics of each GS-GC agent in the world is specified by the iteration:

$$\vec{v}_i(t+1) = \begin{cases} 0 & c_i \in C \& (\lambda_i = 1) \\ d \cdot \vec{a}_{i,k^*} & c_i \notin C \\ v_r \cdot (p_r - p_i) & (c_i \in C) \& (\lambda_i = 0) \end{cases}, \quad (4.2)$$

where d is the distance of the agent's position p_i to the position of the nearest color goal g_{k^*} , \vec{a}_{i,k^*} represents the attraction force to approach the nearest goal, and v_r is the magnitude of a random vector moving the agent towards a random position p_r when it is expelled from a color goal. Parameter λ_i represents the effect of the degree of *Comfort* of the GS-GC agent. When a GS-GC agent b_i reaches a color goal in an instant t , its velocity becomes 0.

Dynamics inside the color goal: 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

$$\mathcal{N}(g_k) = \{b_i \text{ s.t. } \|p_i - g_k\| < \textit{nearenough}\}. \quad (4.3)$$

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

$$b_i \in \mathcal{N}(g_k) \Rightarrow c_i = k. \quad (4.4)$$

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 connected 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 *repulsion* which has value 1 if a pair of GS-GC 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:

$$R(b_i, g_k) = \sum_{N(g_k)} \textit{repulsion}(b_i, b_j). \quad (4.5)$$

Comfort dynamics: Each time step that the GS-GC agent stays in a color goal without been disturbed, its *Comfort* increases, until reaching a maximum value *maxcomfort*. When another GS-GC 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. If the *Comfort* values of the challenged agents are bigger than 0 then their *Comfort* decreases. If the *Comfort* reaches 0, then the agent is expelled from the color goal to a random position in space p_r with velocity v_r . In equation (4.2) when Comfort is positive the parameter has value $\lambda_i > 0$. If the Repulsion force is greater than zero and the *Comfort* of a GS-GC agent b_i inside that goal is equal to 0 then $\lambda_i = 0$ and b_i is expelled from the color goal. When all the GS-GC

agents stop, i.e. $\forall i, \vec{v}_i = 0$ we have $f(B, CG) = n$ of equation 4.6 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:

$$f(B, CG) = |\{b_i \text{ s.t. } c_i \in C \ \& \ R(b_i, g_{c_i}) = 0\}|. \quad (4.6)$$

This cost function is the 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 *nearenough* variable is small then 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 GS-GC 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: A simple version of the flowchart the internal logic working of each GS-GC agent is given in figure 4.1. The simplified flowchart start in the green transitory state. Then select a random position for 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 repulsion forces. If there are no repulsion forces the agent stop, if not the agent or an agent that repel 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 repulsion forces.

The flowchart of figure 4.2 shows the internal logic working of each GS-GC agent. This flowchart is defined for each agent, and can happend that two agents

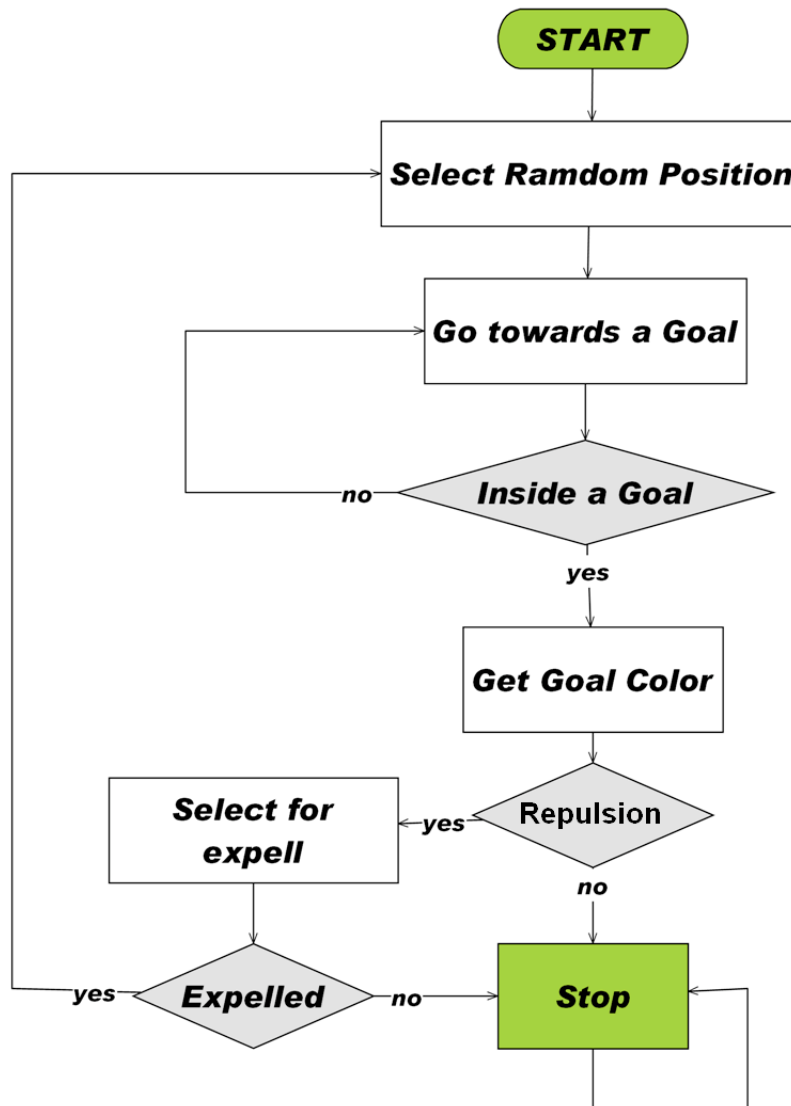


Figure 4.1: Simplified Flowchart

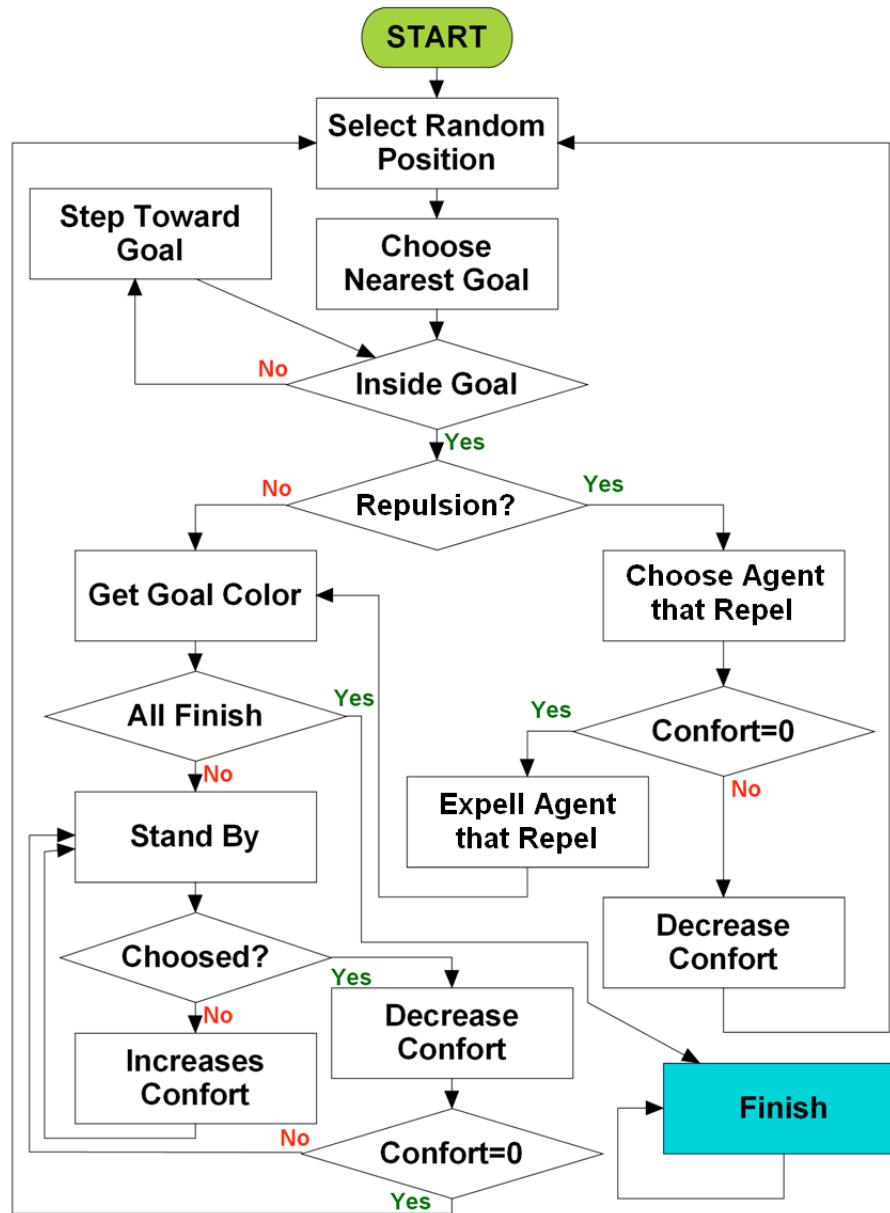


Figure 4.2: GS-GC agent behavior flowchart for GCP

arrives 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 repulsion forces 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.

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 19. 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 20. 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 21. Chromatic number: The chromatic number M^* is the number of colors of the minimal graph coloring C^* .

4.3 Gravitational Swarm

Definition 22. A Gravitational Swarm (GS) is a collection of particles $P = \{p_1, \dots, p_L\}$ moving in a 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:

$$\dot{s}_i(t) = -m_i(t) A_i(t) + \mu_i(t) R_i(t) + \eta(t), \quad (4.7)$$

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:

$$A_i(t) = \sum_{p_j \in P - r_i} m_j(t) (s_i - s_j) \delta_{ij}^A, \quad (4.8)$$

where

$$\delta_{ij}^A = \begin{cases} \|s_i - s_j\|^{-2} & \|s_i - s_j\|^2 > \theta_A \\ 0 & \|s_i - s_j\|^2 \leq \theta_A \end{cases}. \quad (4.9)$$

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

$$\delta_{ij}^R = \begin{cases} \|s_i - s_j\|^{-2} & \|s_i - s_j\|^2 \geq \theta_R \\ 0 & \|s_i - s_j\|^2 < \theta_R \end{cases}. \quad (4.10)$$

Remark 23. 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 24. We allow both mass and charge to be time varying. In exploratory computational experimental works [116, 115] 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 25. *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. \square

Lemma 26. *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 \square

- 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 all 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 27. *Global convergence of GS. If the conditions of Lemma 26 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 $\|s_i(t) - s_k(t)\| \geq \theta_R$. \square

Remark 28. The condition of Lemma 26 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 4.9 is changed to equation 4.11.

Remark 29. The Lemma 27 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 30. 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 4.8 is changed to

$$A_i(t) = \sum_{p_j \in P_C} \{m_j (s_i - s_j) \delta_{ij}^A\}. \quad (4.11)$$

Remark 31. Each vertex particle is attracted to its closest color particle according to the different color particle masses. The noise term in equation 4.7 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 32. *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 4.7 and 4.11 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 4.7 moves the GS-GC from such unstable stationary states.

□

Remark 33. The noise term in equation 4.7 has to be small enough not to push a vertex particle outside of a color particle region of influence.

Corollary 34. *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 35. When a vertex particle reaches a zero velocity it has attained a locally correct coloring of its corresponding vertex in the graph.

Definition 36. 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 37. 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 38. 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 39. *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:*

$$\bigcup_{p_j} \mathcal{N}(p_j) = P_V, \quad (4.12)$$

$$p_i \in \mathcal{N}(p_j) \Rightarrow \mathcal{N}(p_j) \cap r_i = \emptyset. \quad (4.13)$$

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 4.12 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 an stationary state.
 - If equation 4.13 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 40. Theorem 39 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 41. *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 42. Theorem 41 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 43. *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 39 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 39 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 44. 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 45. *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 43, therefore the chromatic number is M contrary to the initial assumption. \square

Remark 46. Theorem 45 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 47. 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 GS-GC 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 gives summary experimental results on the KRG graphs. Section 5.3 reports sensitivity results on the color goal radius. Section 5.4 reports sensitivity results on the comfort parameter. Section 5.5 reports the results of non-parametric statistical tests assessing the statistical significance of the results. Section 5.6 gathers the concluding remarks explaining the parameter settings for following computational experiments.

5.1 GS-GC model parameters

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

- The chromatic number,
- the color goal radius of its influence region and
- the Comfort.

The world size is irrelevant for GS-GC, because the agents' speed adapts to the world size. If the world is bigger, agents go faster. The color 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 limit on the number of steps to avoid enormous execution times, in the order 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. For this sensitivity experiment we have specifically 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.

5.2 Experimental results on 30-50 KRG graphs

The global experimental design consists in running GS-GC with changing the parameters as shown in the table 5.1, for the sensitivity exploration over the KRG graphs. Specifically table 5.1 gives the average accuracy results of GS-GC on the 20 KRG generated graphs of 30 nodes, 50 edges and 6 colors. For each graph we have repeated 1,000 times the application of GS-GC with specific parameter values limiting 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. Therefore, table 5.1 summarizes the results of running the GS-GC algorithm 20x48,000 times. Accuracy values correspond to the number of times that the GS-GC finds a correct coloring. The hypothetical chromatic number

Comfort	Radius								
	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 accuracy results of GS-GC on the 20 KRG generated graphs of 30 nodes, 50 edges and 6 colors for varying comfort (rows) and radius (columns) parameter values

(the number of color goals) is set to the exact value (6 in this table). The observation of the table shows that moderate values of comfort and color goal radius give the best results. Radius seems to play a more important role in GS-GC, but null comfort is catastrophic, results fall below pure random choice.

5.2.1 Chromatic number

This parameter can not be tuned because it is an input parameter related with the graph. If we know the chromatic number in advance, the algorithms can use this knowledge. If not, we must guess the chromatic number. Some theorems help to know *a priori* this number but we can not always apply them.

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

5.3 Goal Radius

The goal radius is a very important parameter because it determines the color of the agents. In our approach there is an attraction over the entire search space center towards the goals. When an agent is near enough to a goal then we assume that the agent's color becomes the color of the goal. But when we can say that an agent is near enough?. The goal radius determines 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.

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

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

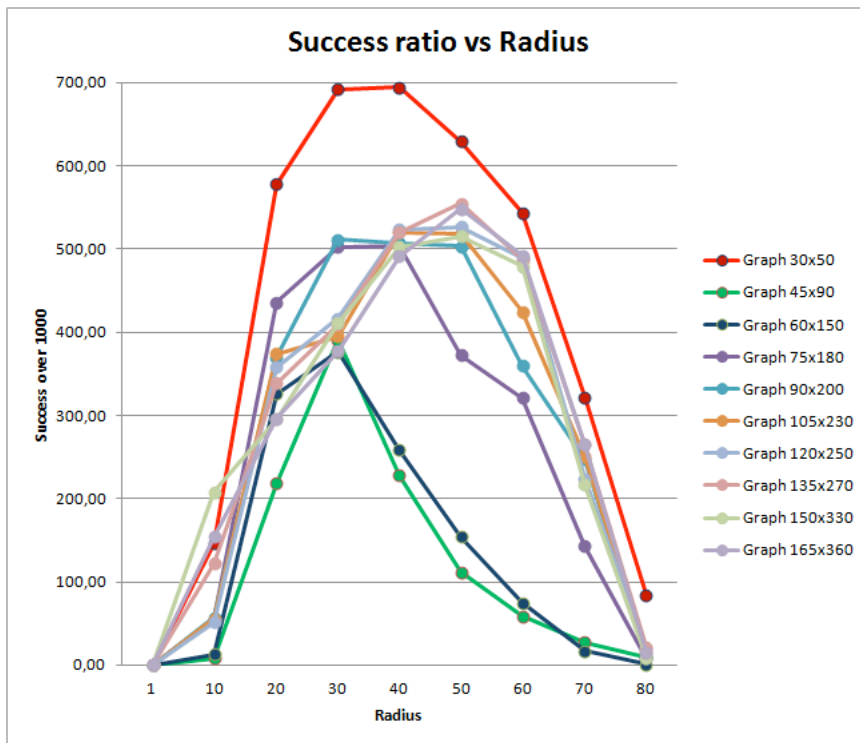


Figure 5.1: Average success ratio vs Goal Radius

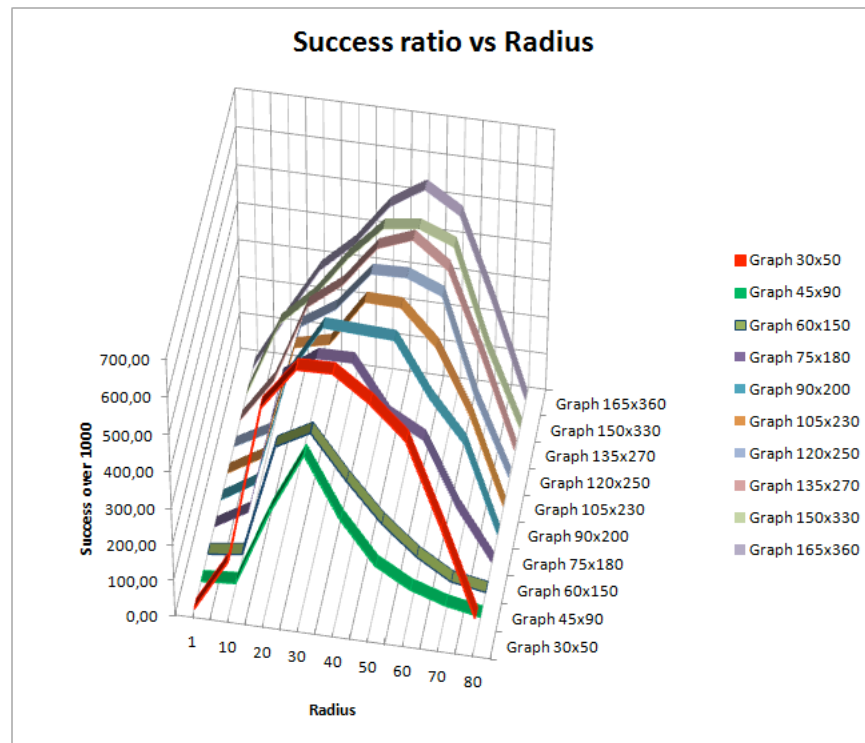


Figure 5.2: Average success ratio vs Goal Radius in 3D

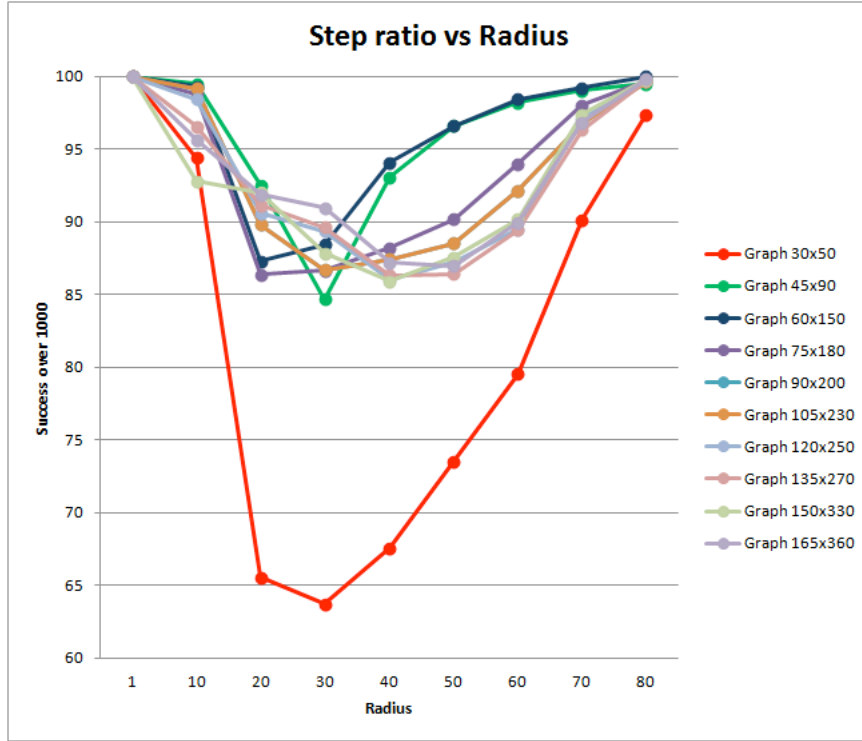


Figure 5.3: Average Steps ratio vs Goal Radius

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 lot 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.

5.4 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 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

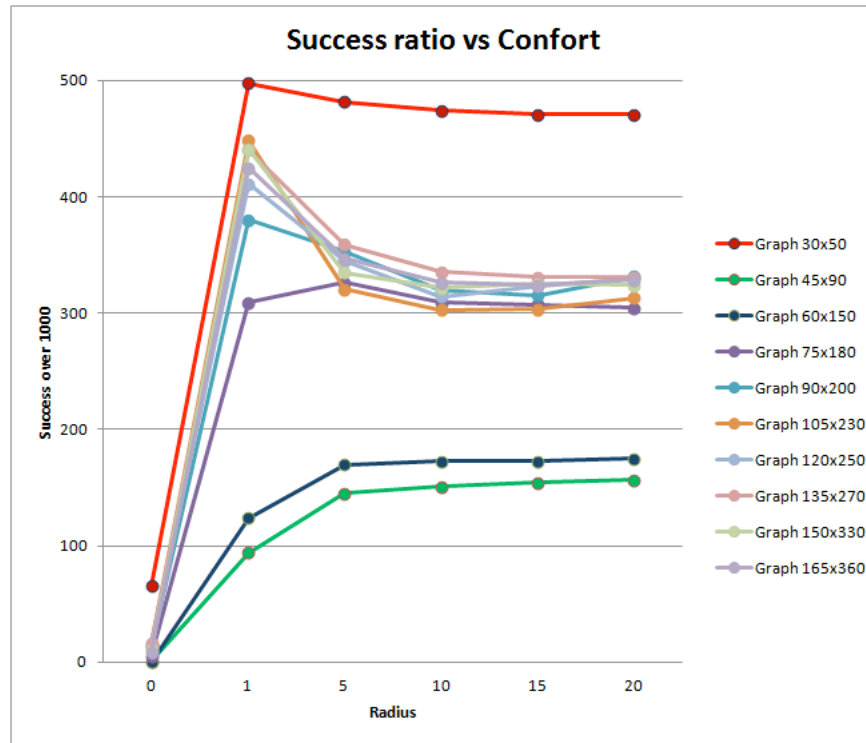


Figure 5.4: Average success ratio vs Comfort

reaching to a goal, the system behavior would be unstable because the agents will be jumping from one goal to another. The problem of the local minimum is that an agent reaching a color goal 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 it the algorithm fails. But which value 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 plotted again the results in 3D to have another point of view in figure 5.5.

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

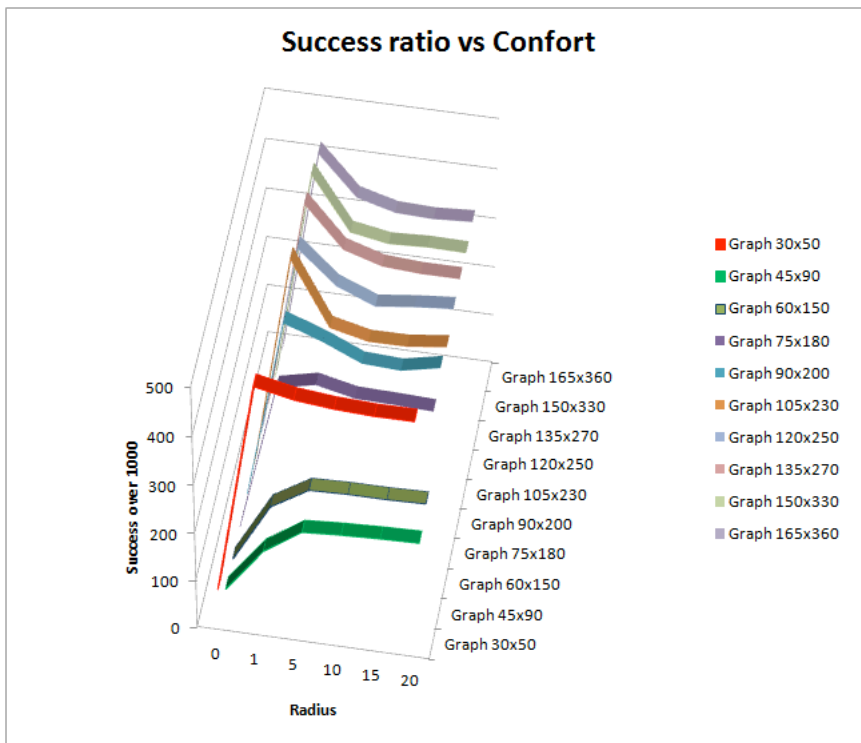


Figure 5.5: Average success ratio vs Comfort in 3D

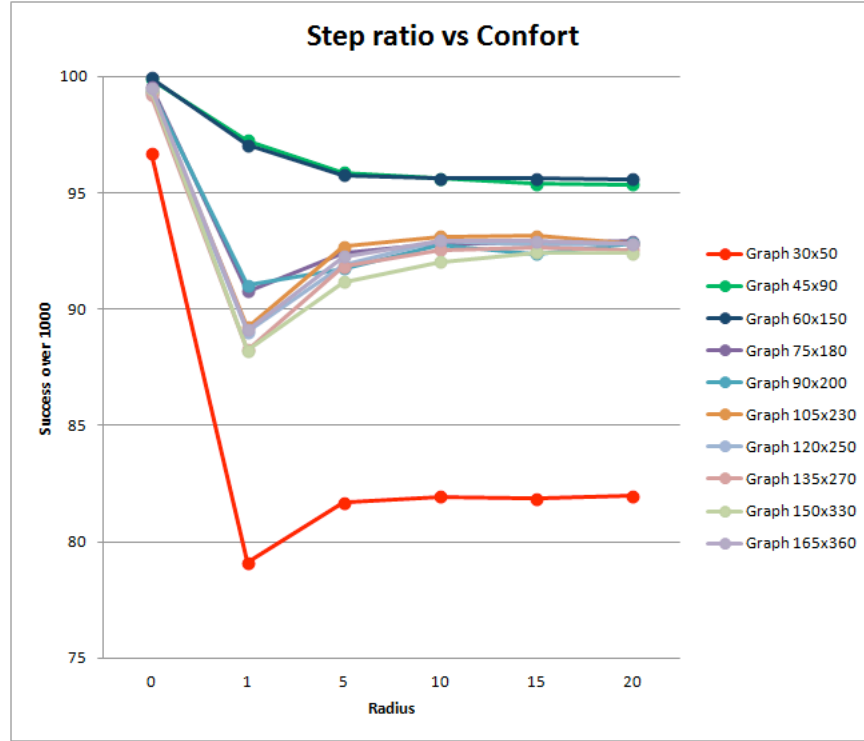


Figure 5.6: Average steps ratio vs Comfort

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.

5.5 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 [45, 44]. If the null hypothesis of this test is not comply then we can use a post-hoc test like Nemenyi's [97].

5.5.1 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 [105] 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 [64] 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 [127, 40] stochastic distribution with $k - 1$ and $(k - 1)(n - 1)$ degrees of freedom.

5.5.2 Friedman test to GS-GC

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

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\Comfort	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

each graph and value o goal radius and comfort and the average rankings R_k

5.5.2.1 Goal Radius

The Friedman value for goal radius is $\chi_F = 71.333$. The value of the chi-square statistic 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 $\chi^2 = 9.24$, $\chi^2 = 11.8$, 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.

5.5.2.2 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.

5.5.3 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 [97]. This test is similar to the Tukey's test [132] 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.

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

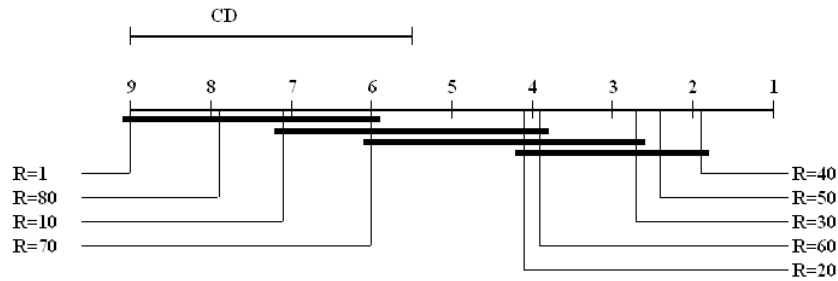


Figure 5.7: Nemenyi's diagram for 9 goal radius and 90% of acceptance

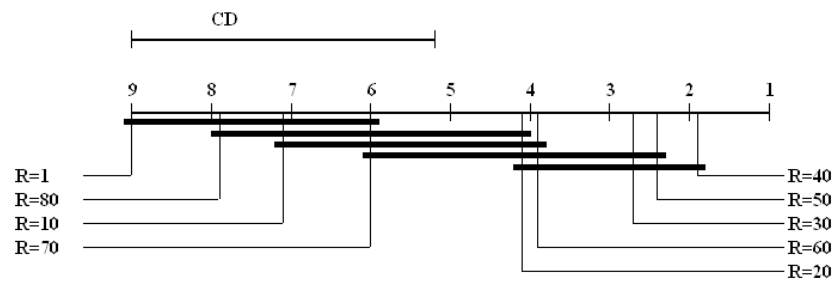


Figure 5.8: Nemenyi's diagram for 9 goal radius and 95% of acceptance

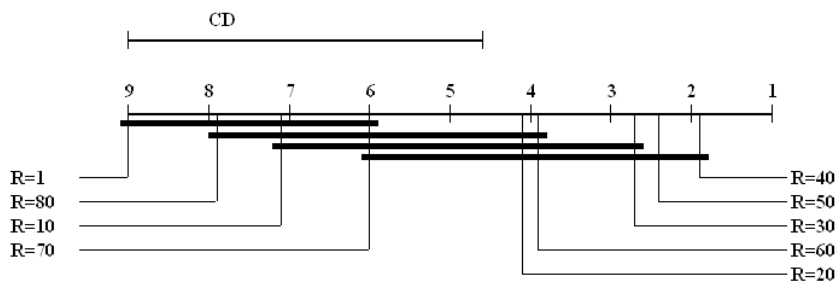


Figure 5.9: Nemenyi's diagram for 9 goal radius and 99% of acceptance

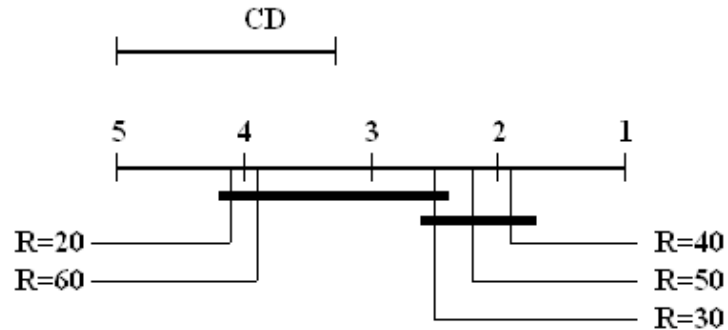


Figure 5.10: Nemenyi's diagram for 5 goal radius and 95% of acceptance

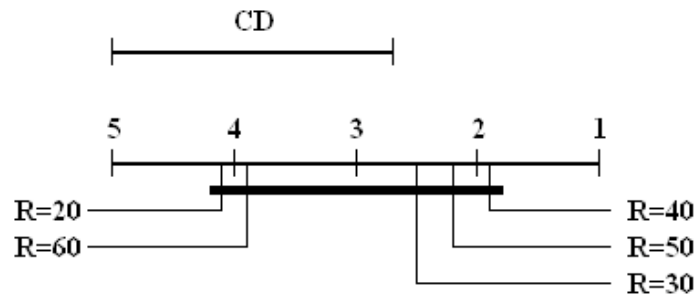


Figure 5.11: Nemenyi's diagram for 5 goal radius and 99% of acceptance

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, 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

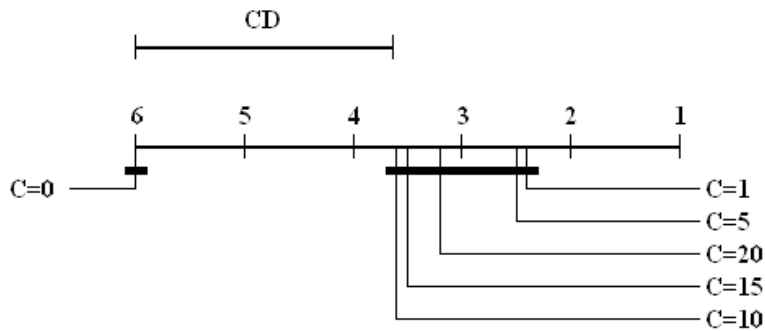


Figure 5.12: Nemenyi's diagram for 6 comfort values and 95% of acceptance

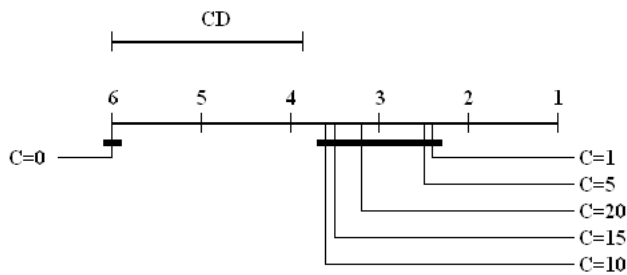


Figure 5.13: Nemenyi's diagram for 6 comfort values and 99% of acceptance

zero, but if 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.

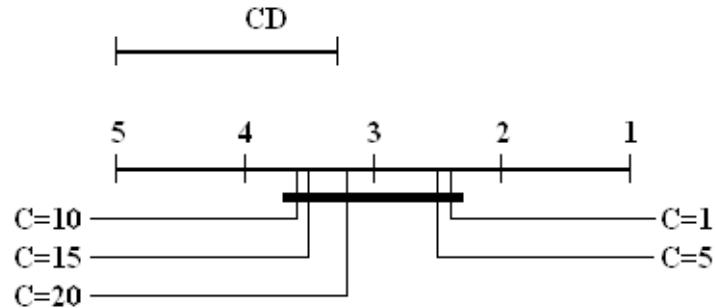


Figure 5.14: Nemenyi's diagram for 5 comfort values and 90% of acceptance

5.6 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 GS-GC algorithm get good results is a wide range of problems. We start testing on the Mycielski graphs [95]. 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 [11] famous DSATUR algorithm (DS)
3. Clique initialization of BT (CBT)
4. An stochastic simulated annealing (SA) [117].
5. A Tabu search (TS) [87].
6. And three Swarm Intelligence based algorithms:
 - (a) Ant Colony Optimization (ACO) [57]
 - (b) Particle Swarm Optimization (PSO) [39]
 - (c) Gravitational Swarm Intelligence (GS-GC) [116]

The first three 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 CBT single steps are computationally light, therefore we have allowed them 10.000.000 steps.
- The DS has as many steps as nodes.
- The SA with a medium complexity step, we have allowed 100.000 steps.
- The TS, PSO and our GS-GC have a maximum of 10.000 steps, because each iteration of PSO and GS-GC 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 graphs 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.

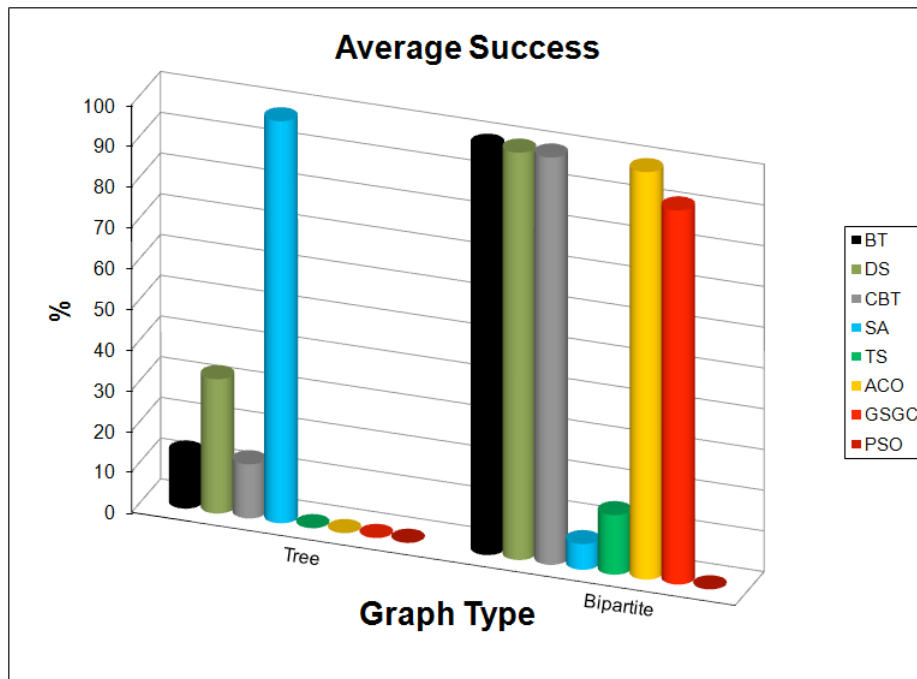


Figure 6.1: Trees and bipartite success ratio

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 GS-GC give 100% of success (GS-GC 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.

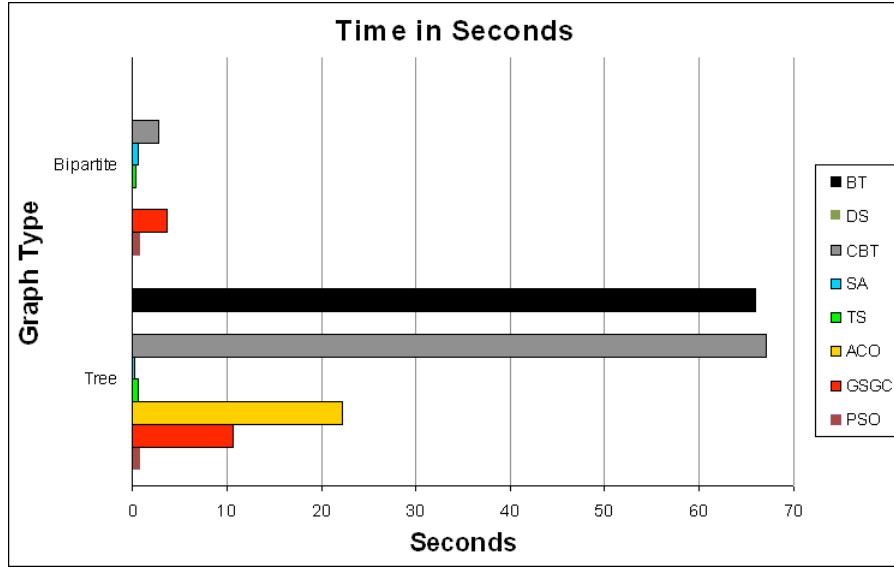


Figure 6.2: Trees and bipartite average time in seconds

In figures 6.2 and 6.3 we can see the average time and number of steps for each algorithm. The number of steps is normalized to 1.000. We can see the big difference between the time and the steps. The PSO time for trees is only a pair of seconds. This is one of the reasons 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 graphs each family. These families are grouped by the number of nodes, starting from 10 and increasing the number of nodes adding 10 more nodes until reaching a family with 250 nodes. The number of edges has been calculated $E = n * 2$.

In figure 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 GS-GC algorithm is robust against the size of the graph. Again 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

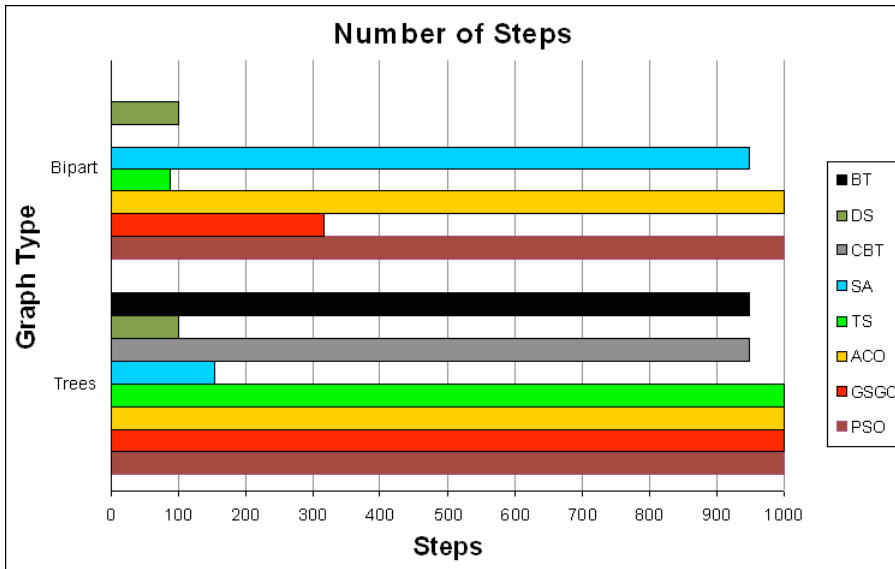


Figure 6.3: Trees and bipartite average number of steps

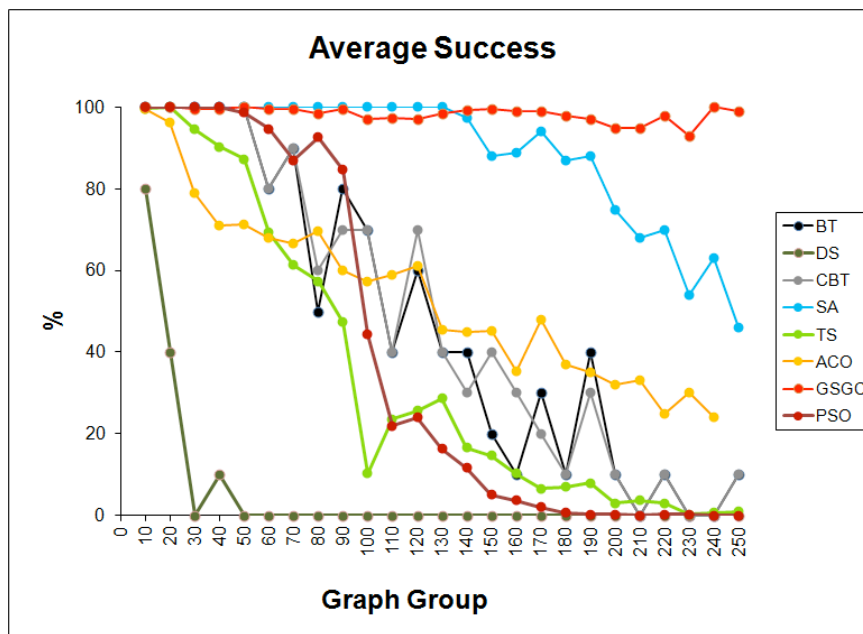


Figure 6.4: Kuratowski based graphs success ratio

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 GS-GC 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 GS-GC 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 GS-GC 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. GS-GC, PSO and SA time grow slowly compared with the other algorithms. DS is fast but obtain bad results. For the GS-GC 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 number of nodes and nodes is not the same for each MUGS, the resulting instances

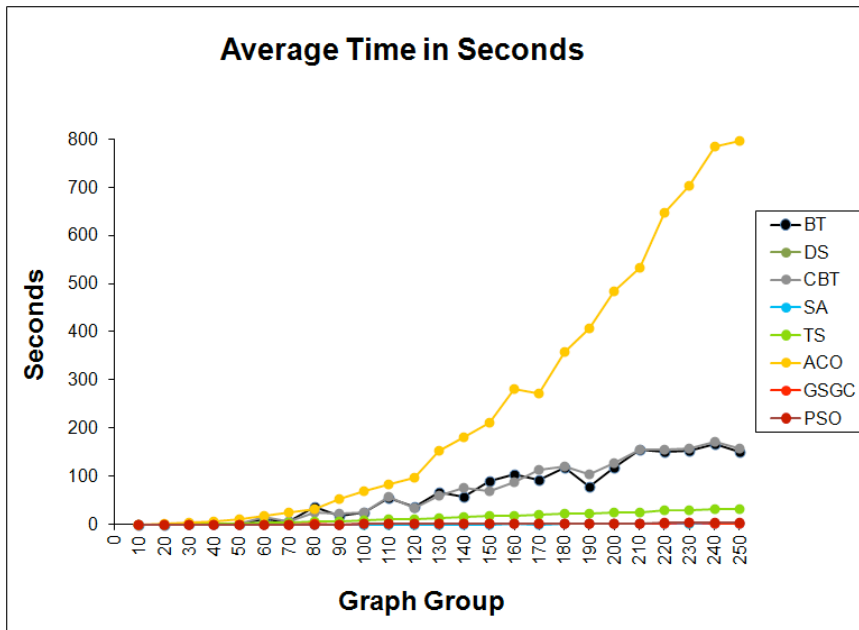


Figure 6.5: Kuratowski based graphs average time in seconds

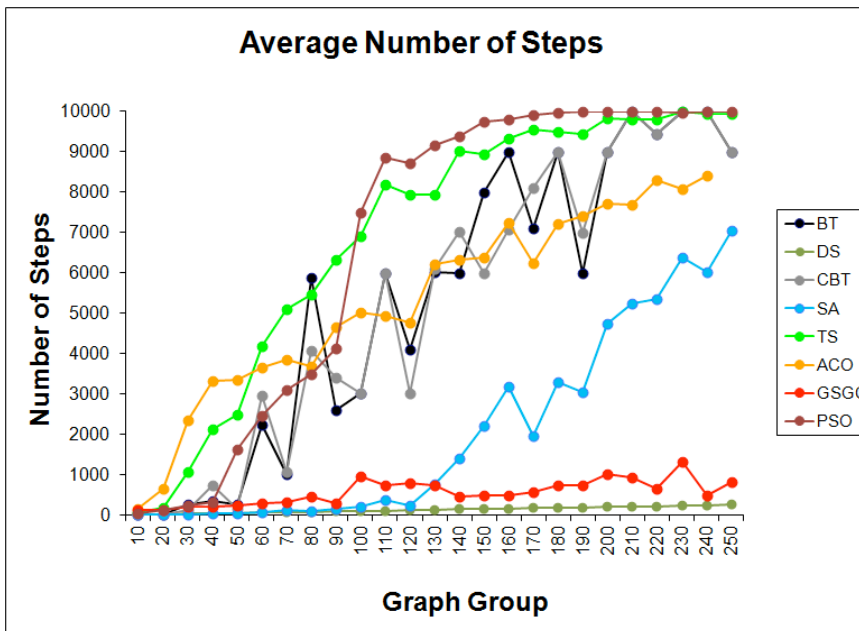


Figure 6.6: Kuratowski based graphs average number of steps

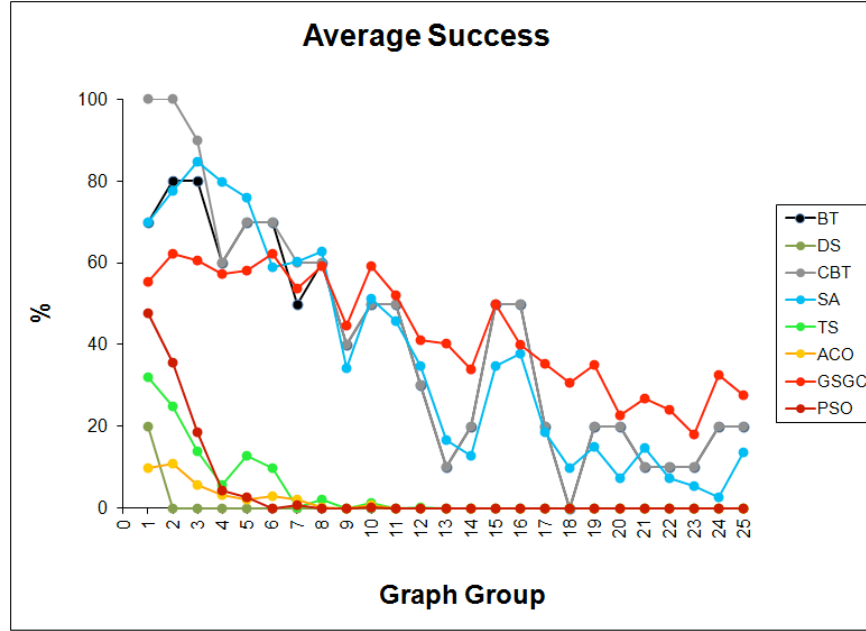


Figure 6.7: Mizuno's graphs success ratio

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 CBT non deterministic algorithms 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 GS-GC 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 CBT even though obtain good results, the computational time is very big. The DS works fast but with poor results. The ACO time is also big as we expected. PSO and TS are very quick and stable, because the perform bad results from the beginning and the success ratio don't affect them. SA is again the fastest and GS-GC this time penalized the dimension of the instances. As we have said the GS-GC algorithm

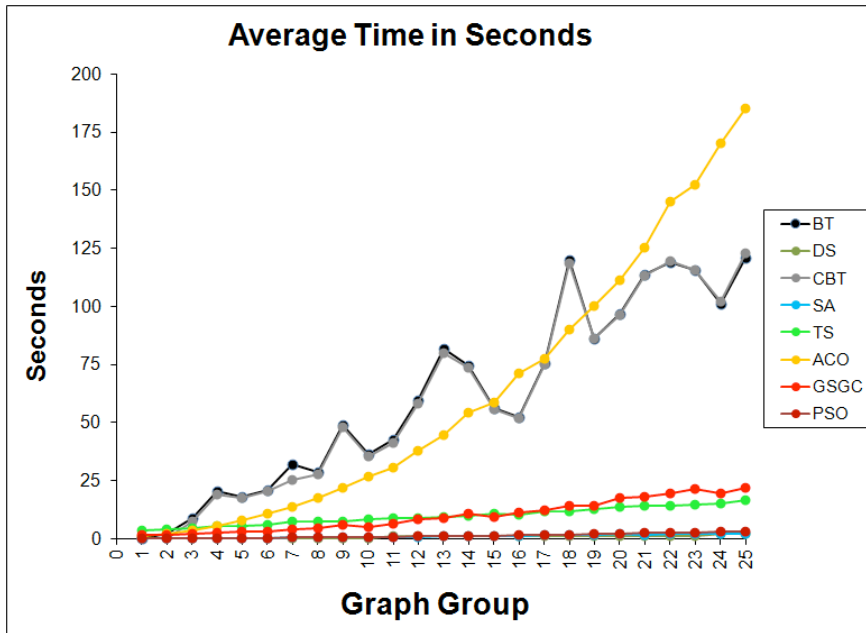


Figure 6.8: Mizuno's graphs average time in seconds

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 GS-GC's although the GS-GC 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 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 G lower bound is N , and the upper bound of the graph G -c is 4 so the chromatic number if the graph G is N .

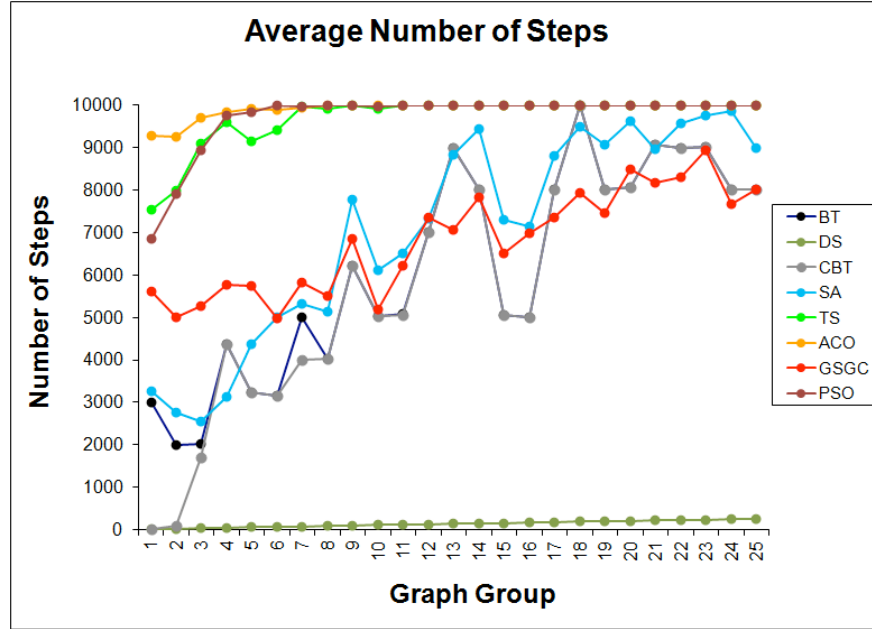


Figure 6.9: Mizuno’s graphs average number of steps

We have prepared two sets of experiments. The first one start 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 apply a $N=4$, $N=5$, $N=6$ and $N=7$ so finally we have 12 families of 10 graphs. We have use 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 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 GS-GC 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 GS-GC algorithm in KRG_100_300_7 family. The ACO results are pretty good in the 300 and 400 edge families, been the Swarm algorithms the best accurate in this

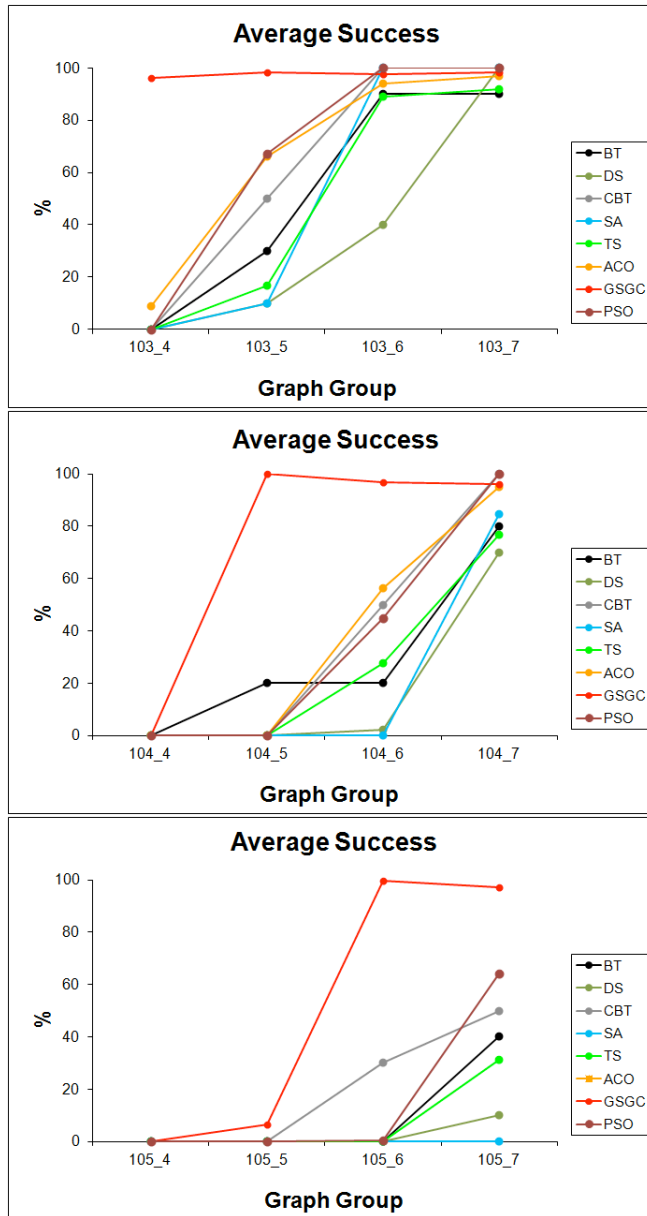


Figure 6.10: KRG success ratio

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 usually a good algorithm, here is the worst. The TS, BT and CBT behavior is similar as other test. The DS is very fast but poor.

In figure 6.11 we can see that the time in seconds need for TS and ACO is very big compared with the other methods. The BT, CBT and PSO are average in computational time. SA as we expected is the fastest in almost all the instances, but our GS-GC algorithm is very slow with complex graphs with small chromatic number but near as fast as SA in graphs with a big chromatic number, been 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 this experiments we have shown that the KRG graphs really complies 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, 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 GS-GC, 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 CBT, 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 CBT, 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 GS-GC 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.

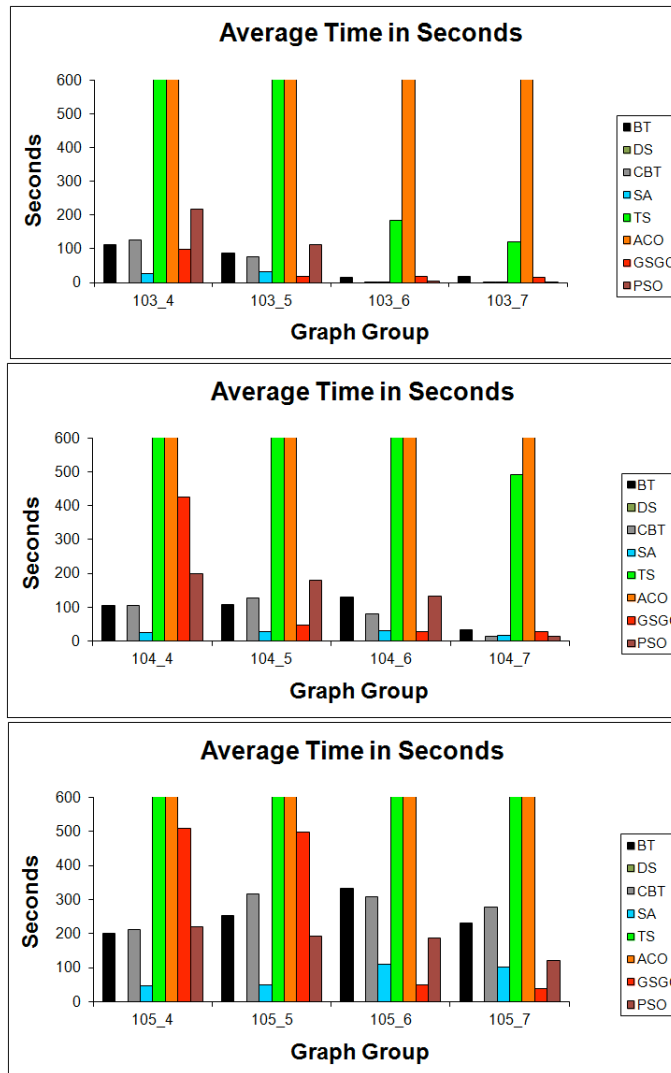


Figure 6.11: KRG average time in seconds

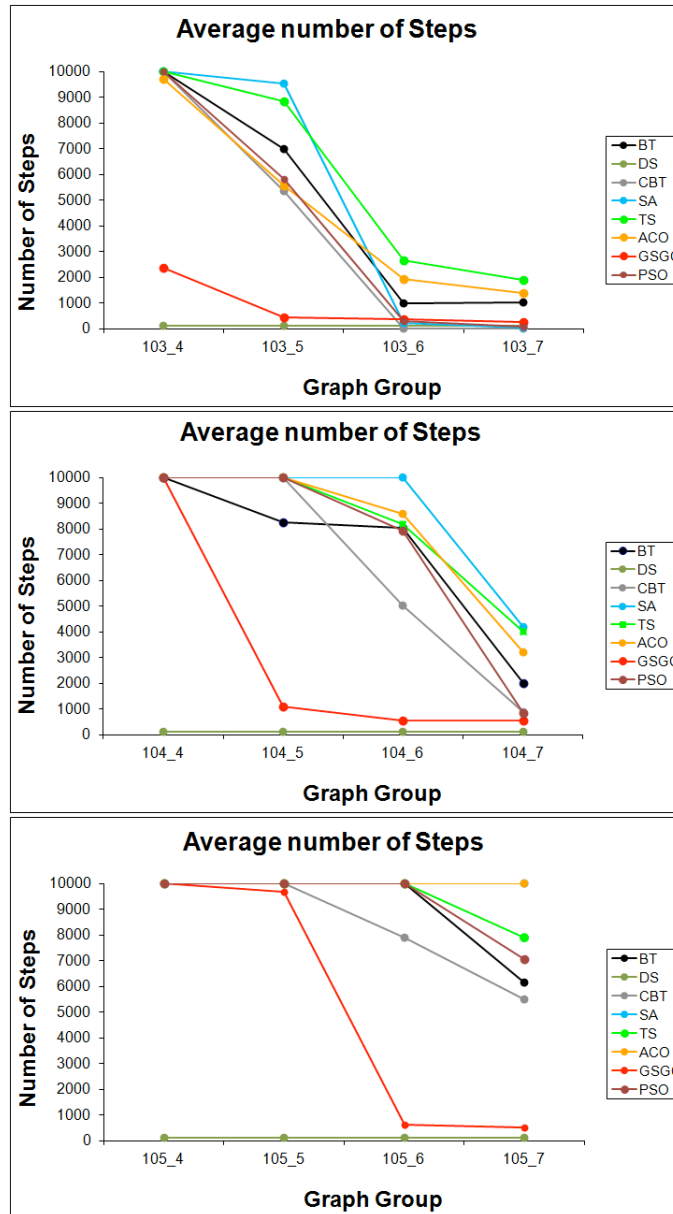


Figure 6.12: KRG average number of steps

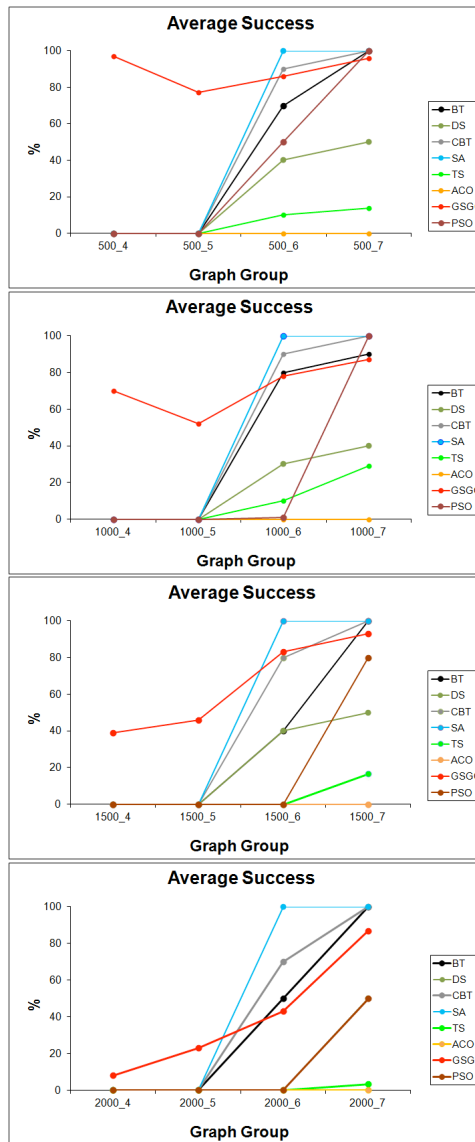


Figure 6.13: Big KRG success ratio

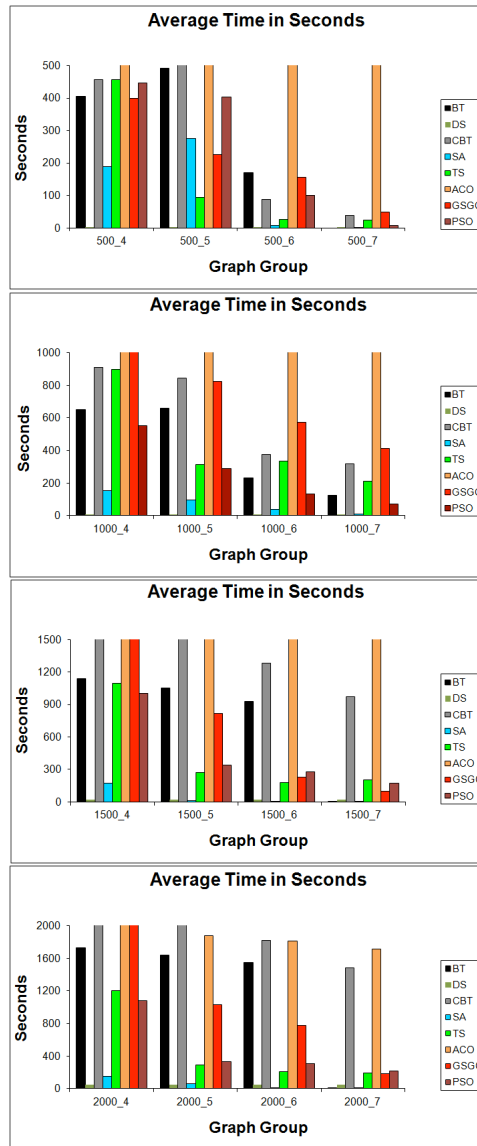


Figure 6.14: Big KRG average time in seconds

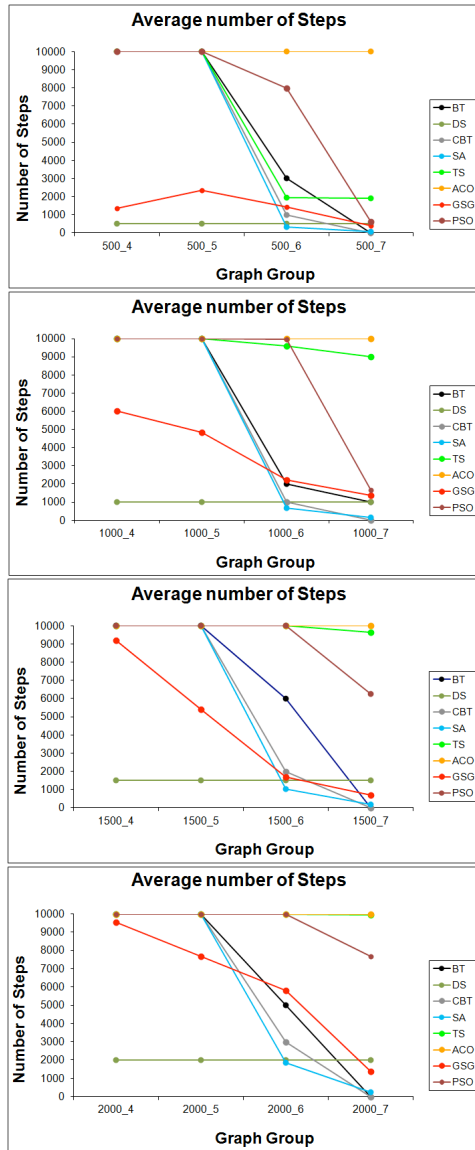


Figure 6.15: Big KRG Average number of steps

6.6 DIMACS graphs

The DIMACS challenge in 1993 [68] 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.

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 Caramia [17] graph family called CAR, 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 [68].

BackTrack, CBT 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 GS-GC 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 GS-GC manage the solve them. Finally the queens graphs show a very hard face. The greedy algorithm are the best. The results of the GS-GC, 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 [90] 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 GS-GC can solve all the books graphs. In [129] and [89] solve the SGB graphs using another genetic algorithm, but fail finding the optimum solution in the CAR family. The results are show in table 6.3. Here again made a good job in CAR family, and get good results in the SGB, perhaps they get better

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 6.1: Layout of the experimental graphs

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

Table 6.2: Results of SGB graphs

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 GS-GC haven't suffer any modification for any experiment. In [15] 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 [72] and [126] but the number of experiments is quite poor compare with ours. They perform well but in a reduced group of graphs.

In [1] a cultural algorithm is implemented to solve the GCP. The results are similar to ours but again with a small group of algorithms. In [62] 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 [5], 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 [125]. 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.

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

Table 6.3: Results of CAR graphs

6.6.1 Test

The Friedman test is a non-parametric statistical test developed by Milton Friedman [45] 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 value 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 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.

Method	BT	CBT	SA	TS	ACO	PSO	GS-GC
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
R_j	3.61	3.29	4.21	4.86	5.5	4.36	2.18

Table 6.4: Algorithm Rankings for Friedman Test

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. It 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.

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

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 CBT, 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, DIMACS Leighton's graphs and real graphs from "Exam timetabling problems" [86].

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 GS-GC algorithm gets the best result for all the families. We can't say that the result

Graphs100x2000	BT	DS	SA	TS	PSO	ACO	GS-GC
Random1	16	16	29.5(28)	18	18	34	15
Random2	16	17	29.5(28)	22	18	35	15
Random3	17	16	29.5(28)	19	18	34	15
Random4	18	17	29.4(28)	23	19	35	15
Random5	15	18	29.5(28)	19	19	34	15
Random6	18	16	29.4(28)	28	18	34	15
Random7	17	16	29.6(28)	17	18	34	15
Random8	16	17	29.5(28)	21	19	34	15
Random9	15	16	29.4(29)	23	19	34	15
Random10	18	17	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	DS	SA	TS	PSO	ACO	GS-GC
Random1	23	24	45(44)	27	28	50	23
Random2	24	24	45.5(44)	30	28	50	22
Random3	23	25	45.6(44)	34	27	50	22
Random4	26	24	45.8(44)	36	27	50	23
Random5	25	25	45.7(44)	31	27	50	23
Random6	24	24	45.4(44)	32	28	50	23
Random7	25	24	45.4(14)	32	29	50	22
Random8	25	25	45.5(44)	29	28	50	23
Random9	25	24	45.4(44)	31	27	50	23
Random10	25	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	DS	SA	TS	PSO	ACO	GS-GC
Random1	36	34	50	47	41	50	32
Random2	36	36	50	44	40	50	33
Random3	36	37	50	15	40	50	34
Random4	37	35	50	39	42	50	34
Random5	35	35	50	47	39	50	33
Random6	36	34	50	40	40	50	33
Random7	36	36	50	42	42	50	33
Random8	34	36	50	40	40	50	33
Random9	39	34	50	43	41	50	33
Random10	35	35	50	43	40	50	33

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

of the GS-GC 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 strange manner getting very bad results. The behavior of the SA algorithm is very strange 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 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 CBTDS has a limitation of a chromatic number of 8 so we haven't used it in this experiment.

The DS algorithm perform average results very fast, but it can't improve or even equal GSGC results.

6.7.2 Leighton graphs

Theorem 48. *Two finite graphs which have a common covering have a common finite covering [79].*

Based in this theorem Leighton show a way to built graphs [78] 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 our 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 [41] 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. The DSATUR algorithm is the fastest method, but get very poor results.

Graph	#node	#Edges	Density	#K	DS	GS-GC
le450_5a	450	5714	0.056	5	12	8
le450_5b	450	5734	0.057	5	12	8
le450_5c	450	9803	0.097	5	11	9
le450_5d	450	9757	0.096	5	12	9
le450_15a	450	8168	0.081	15	19	18
le450_15b	450	8169	0.081	15	19	18
le450_15c	450	16680	0.165	15	26	20
le450_15d	450	16750	0.166	15	27	20
le450_25a	450	8260	0.082	25	26	26
le450_25b	450	8263	0.082	25	26	27
le450_25c	450	17343	0.171	25	31	30
le450_25d	450	17425	0.172	25	32	30

Table 6.9: Leighton Graphs results

Graph	#K	TabuCol	PartCol	AntCol	HEA	HC	BT	DS	GSGC
sta-f-83	13	13.35	13	13.13	13	13	13	13	13
hec-s-92	17	17.22	17	17.04	17	17	19	21	18
kfu-s-93	19	20.76	19	19	19	19	19	21	20
yor-f-83	19	19.74	19	19.87	19.06	19	20	24	21
tre-s-92	20	20.58	20	20.04	20	20	23	25	23
car_f-92	27	39.92	32.48	30.04	28.5	27.96	27	34	36
car-s-91	28	39.1	30.2	29.23	29.04	29.1	28	37	37
pur-s-93	33	50.7	45.48	33.47	33.7	33.87	33	38	44

Table 6.10: Exam timetabling of Lewis [82] plus GS-GC

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 [82, 81], 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 from ours because this version of BT uses an especial ordering to improve the results.

The results of our GS-GC 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, GS-GC iterations and the appearing in [82] 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.

The DSATUR algorithm works better with these graphs, and again very fast.

6.8 Conclusions

We have test our GS-GC algorithm with a wide range of problems. We have compared the results obtained of our GS-GC 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 GS-GC 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 GS-GC 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 GS-GC: does the GS-GC 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 GS-GC 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 GS-GC may be very proficient in the adaptation to changes in the social structure by its own nature. The GS-GC does not stop its computation when reaching an stationary state, and if some external condition changes the relation between nodes

and thus GS-GC agents, the GS-GC 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 for Graph Coloring (GS-GC) 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 [68].
4. 250 Kuratowski's theorem based graphs [74].

5. 250 Mizuno's method graphs [93].
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 [82] 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 down 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 serve as 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 to 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.

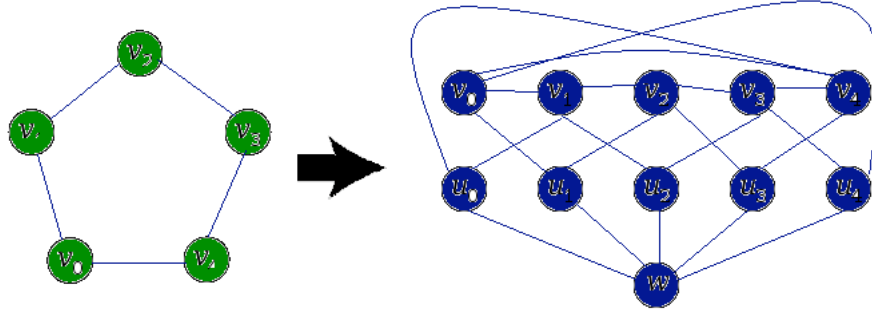
These graphs are very important because the scientific community used them to test different GCP algorithms allowing easy and fair comparison of different works. This is only partially true, because even though a lot of researchers use these graphs for testing new algorithms, the specific parametrization, stopping conditions, and computational resources used for the experiments are often unavailable, making 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 graphs.

A.3.1 Mycielski graphs

Among all the DIMACS graphs, the Mycielski graphs [95] 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:

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

Theorem 49. *There are triangle-free graphs with arbitrarily high chromatic number.*

The construction of the Mycielski graph follows a mathematical formula as 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 [50] without proof that the $n \times n$ queen graph is n -colorable whenever $n \bmod 6$ is 1 or 5.

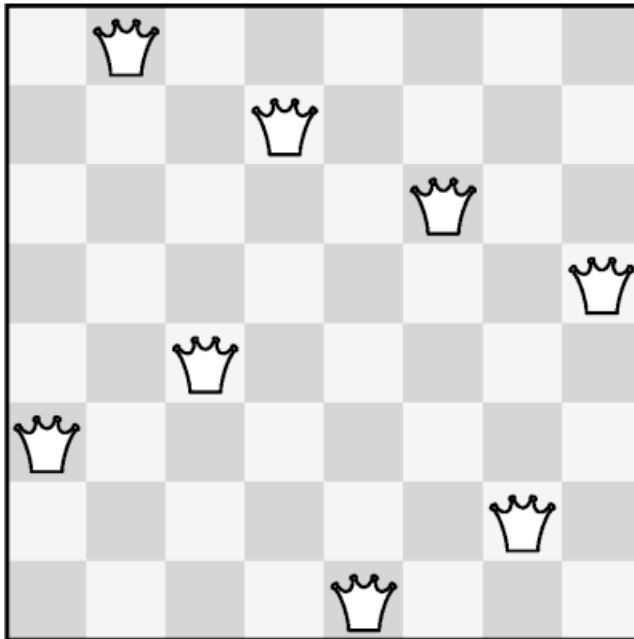


Figure A.2: 8 queens problem solution

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 between them is given 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.

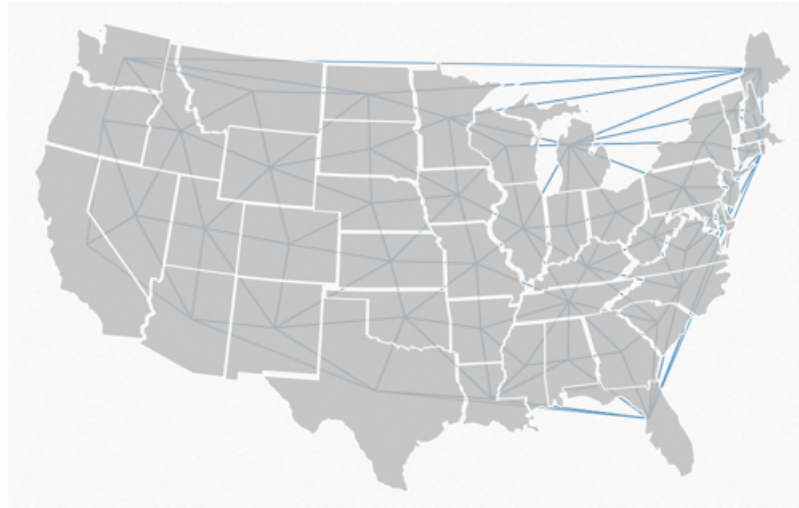


Figure A.3: United States Cites Graph

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 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.

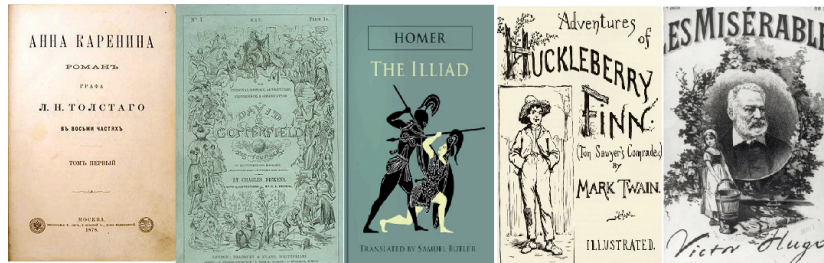


Figure A.4: Books where the Books Graphs come from

A.3.6 Leighton Graphs

This family of graphs is based on Leighton's graph covering theorem [79] that says:

Theorem 50. *Two finite graphs which have a common covering have a common finite covering.*

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 generated 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 [94]. 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 implemented a graph generation application the following Kuratowski's theorem:

Theorem 51. *A finite graph is planar if and only if it does not contain a sub-graph that is a subdivision of k_5 or $k_{3,3}$.*

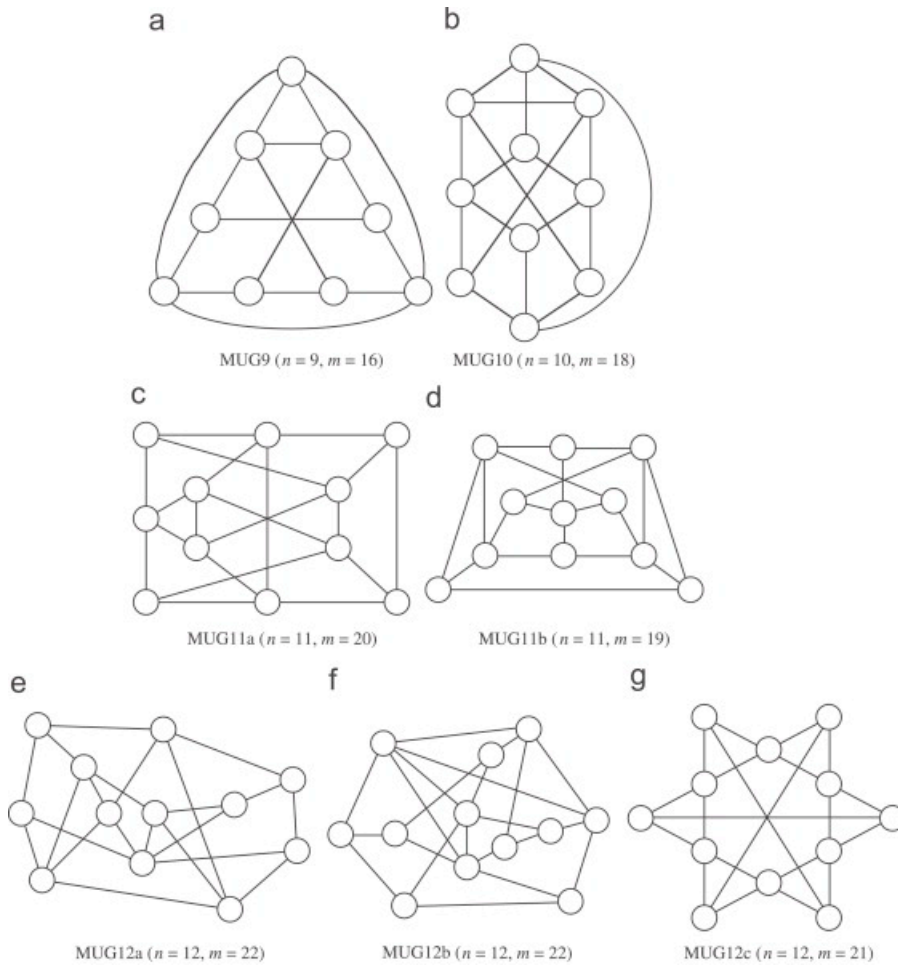


Figure A.5: Mizuno's MUGs for 3-colorable graphs generation [93]

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.

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 determined by the graph construction process. To build these graphs, we need to specify three parameters: the nodes V , the edges E and the desired chromatic number C . Taking into account the Kuratowski theorem, there are not any k_5 or $k_{3,3}$ sub-graphs in a planar graph. Choosing C nodes and connecting all the nodes forming a K_c sub-graph of n edges $n = \frac{C*(C-1)}{2}$. The resulting increased graph $G = \{V, E\}$ is C -colorable.

Proposition 52. *Given a planar graph, adding n edges between nodes such that we form a complete sub-graph K_c , the chromatic number of the increased graph is c the number of nodes 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 [11] method for graph coloring we can say that the chromatic lower-bound is C . Using the Bron-Kerbosch [12] 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, embeded 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 randomly generated graphs can have a pattern, induced by the generation method implementation. However, sometimes modeling a real life problem

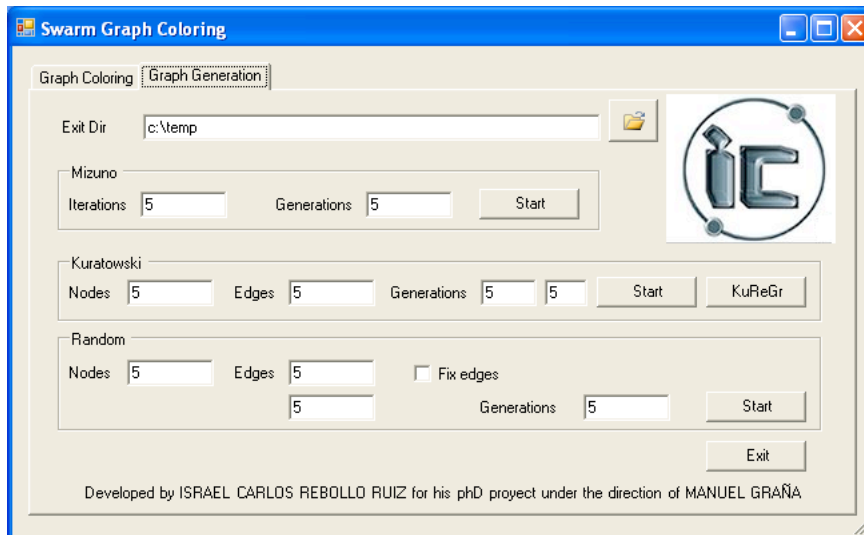


Figure A.6: Aleatory graphs, Kuratowski's planar graphs, Mizuno's 3-colorable graphs and KRG new developed graph type generator.

produces graphs without any special feature and arbitrary structures that can confuse even the most accurate algorithms.

For that reason we have included in this benchmarking graph collection some real life graphs arising from the modeling of a scheduling problem in some universities in the USA. Noteworthy, the exact chromatic number is unknown so that 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 [19].

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 GS-GC: 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.

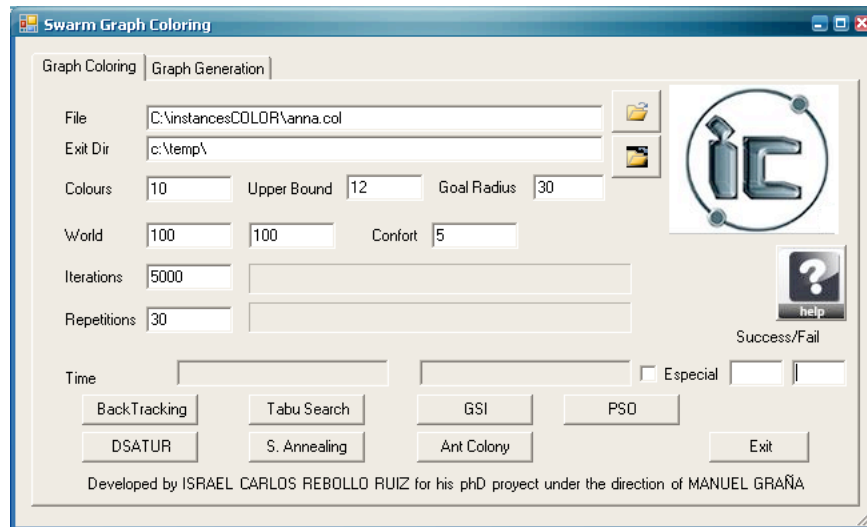
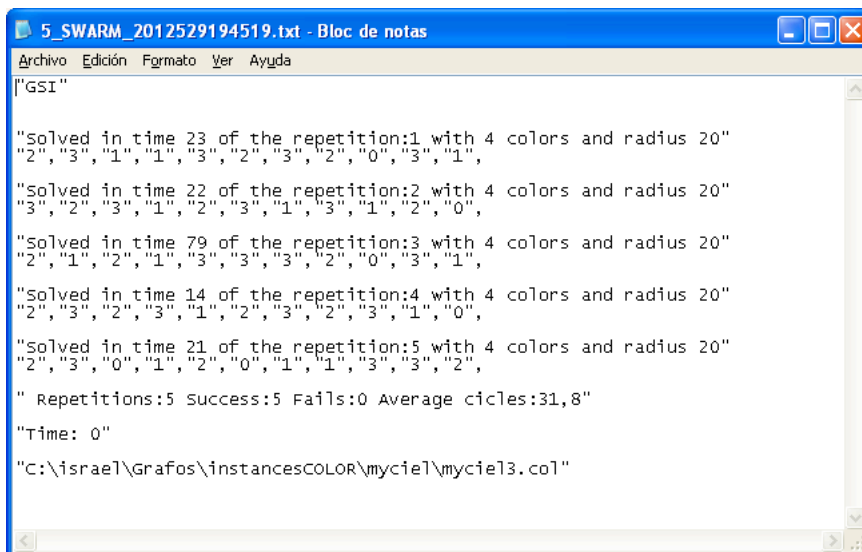


Figure B.1: Graph Coloring Suite

- The algorithm to be used for the solution (via separate buttons). Our GS-GC correspond to the GSI button, that comes from old versions of the 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] Reza Abbasian, Malek Mouhoub, and Amin Jula. Solving graph coloring problems using cultural algorithms. 2011.
- [2] G. Antonelli, F. Arrichiello, and S. Chiaverini. Flocking for multi-robot systems via the null-space-based behavioral control. *Swarm Intelligence*, 4:37–56, 2010.
- [3] Katerina Asdre, Kyriaki Ioannidou, and Stavros D. Nikolopoulos. The harmonious coloring problem is np-complete for interval and permutation graphs. *Discrete Applied Mathematics*, 155(17):2377 – 2382, 2007.
- [4] 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*, 3:223–242, 2009.
- [5] Debnath Bhattacharyya Biman Ray, Anindya J Pal and Tai hoon Kim. An efficient ga with multipoint guided mutation for graph coloring problems. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 3(2), june 2010.
- [6] I. Blochliger and N. Zufferey. A graph coloring heuristic using partial solutions and a reactive tabu scheme. *Computers & Operations Research*, 35(3):960 – 975, 2008.
- [7] J. A. Bondy. *Graph Theory With Applications*. Elsevier Science Ltd, 1976.
- [8] Flavia Bonomo, Sara Mattia, and Gianpaolo Oriolo. Bounded coloring of co-comparability graphs and the pickup and delivery tour combination problem. *Theoretical Computer Science*, 412(45):6261 – 6268, 2011.
- [9] V. Borkar and D. Das. A novel aco algorithm for optimization via reinforcement and initial bias. *Swarm Intelligence*, 3:3–34, 2009.

- [10] M. Bouchard, M. Cangalovic, and A. Hertz. About equivalent interval colorings of weighted graphs. *Discrete Applied Mathematics*, 157(17):3615 – 3624, 2009.
- [11] D. Brelaz. New methods to color the vertices of a graph. *Commun. ACM*, 22:251–256, April 1979.
- [12] C. Bron and J. Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM*, 16:575–577, September 1973.
- [13] J. Randall Brown. Chromatic scheduling and the chromatic number problem. *Management Science*, 19(4-Part-1):456–463, 1972.
- [14] Moritz Buck and Chrystopher L. Nehaniv. Communication and complexity in a grn-based multicellular system for graph colouring. *Biosystems*, 94(1-2):28 – 33, 2008.
- [15] T. N. Bui, T. H. Nguyen, C. M. Patel, and K. T. Phan. An ant-based algorithm for coloring graphs. *Discrete Applied Mathematics*, 156(2):190 – 200, 2008.
- [16] V. Campos, C. Linhares Sales, K. Maia, N. Martins, and R. Sampaio. Restricted coloring problems on graphs with few. *Electronic Notes in Discrete Mathematics*, 37(0):57 – 62, 2011.
- [17] M. Caramia and P. Dell’Olmo. A lower bound on the chromatic number of mycielski graphs. *Discrete Mathematics*, 235(1-3):79–86, 2001.
- [18] M. Caramia and P. Dell’Olmo. Coloring graphs by iterated local search traversing feasible and infeasible solutions. *Discrete Applied Mathematics*, 156(2):201 – 217, 2008.
- [19] M. Carter, G. Laporte, and S.Y. Lee. Examination timetabling: algorithmic strategies and applications. *Journal of the Operational Research Society*, 47:373–383, 1996.
- [20] H. Chang and X. Zhu. Colouring games on outerplanar graphs and trees. *Discrete Mathematics*, 309(10):3185 – 3196, 2009.
- [21] S. Chu, J. F. Roddick, and J. Pan. Ant colony system with communication strategies. *Information Sciences*, 167(1-4):63 – 76, 2004.

- [22] V. Chvatal. Coloring the queen graphs, 2004. Web repository (last visited July 2005).
- [23] Giuseppe Confessore, Paolo Dell’Olmo, and Stefano Giordani. An approximation result for the interval coloring problem on claw-free chordal graphs. *Discrete Applied Mathematics*, 120(1-3):73–90, 2002.
- [24] D. G. Corneil and B. Graham. An algorithm for determining the chromatic number of a graph. *SIAM J. Comput.*, 2(4):311–318, 1973.
- [25] D. Costa and A. Hertz. Ants can colour graphs. *The Journal of the Operational Research Society*, 48(3):295–305, March 1997.
- [26] M. C. Costa, D. de Werra, C. Picouleau, and B. Ries. Graph coloring with cardinality constraints on the neighborhoods. *Discrete Optimization*, 6(4):362 – 369, 2009.
- [27] M. Crochemore and R. Verin. On compact directed acyclic word graphs. In Arto Salomaa Jan Mycielski, Grzegorz Rozenberg, editor, *A Selection of Essays in honor of Andrzej Ehrenfeucht*, volume 1261 of *Lecture Notes in Computer Science*, pages 192–211. Springer Verlag, 1997.
- [28] 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*, 18(3):353 – 357, 2008.
- [29] D. de Werra, M. Demange, B. Escoffier, J. Monnot, and V.Th. Paschos. Weighted coloring on planar, bipartite and split graphs: Complexity and approximation. *Discrete Applied Mathematics*, 157(4):819 – 832, 2009.
- [30] Marc Demange, Tinaz Ekim, and Dominique de Werra. A tutorial on the use of graph coloring for some problems in robotics. (1).
- [31] Marc Demange, Tinaz Ekim, and Dominique de Werra. (p,k)-coloring problems in line graphs. *Theoretical Computer Science*, 349(3):462 – 474, 2005.
- [32] 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*, 26(1):29–41, 1996.

- [33] K. A. Dowsland and J. M. Thompson. An improved ant colony optimisation heuristic for graph colouring. *Discrete Appl. Math.*, 156:313–324, February 2008.
- [34] W. Du and B. Li. Multi-strategy ensemble particle swarm optimization for dynamic optimization. *Information Sciences*, 178(15):3096 – 3109, 2008.
- [35] I. Dukanovic and F. Rendl. A semidefinite programming-based heuristic for graph coloring. *Discrete Applied Mathematics*, 156(2):180 – 189, 2008.
- [36] R. D. Dutton and R. C. Brigham. A new graph colouring algorithm. *The Computer Journal*, 24(1):85–86, 1981.
- [37] Dániel and Marx. Parameterized coloring problems on chordal graphs. *Theoretical Computer Science*, 351(3):407 – 424, 2006.
- [38] M. El-Abd and M. Kamel. A cooperative particle swarm optimizer with migration of heterogeneous probabilistic models. *Swarm Intelligence*, 4:57–89, 2010.
- [39] J. Fernández and E. García. The pso family: deduction, stochastic analysis and comparison. *Swarm Intelligence*, 3:245–273, 2009.
- [40] R.A. Fisher. Frequency distribution of the values of the correlation coefficient in samples from an indefinitely large population. *Biometrika*, 10:507–521, 1915.
- [41] Charles Fleurent and Jacques Ferland. Genetic and hybrid algorithms for graph coloring. *Annals of Operations Research*, 63:437–461, 1996.
- [42] G. Folino, A. Forestiero, and G. Spezzano. An adaptive flocking algorithm for performing approximate clustering. *Information Sciences*, 179(18):3059 – 3078, 2009.
- [43] N. Franks, J. Hooper, M. Gumn, T. Bridger, J. Marshall, and A. Dornhaus. Moving targets: collective decisions and flexible choices in house-hunting ants. *Swarm Intelligence*, 1:81–94, 2007.
- [44] M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32:674–701, 1937.

- [45] Milton Friedman. A comparison of alternative tests of significance for the problem of m rankings. *Annals of Mathematical Statistics*, 11(1):86–92, March 1940.
- [46] H. Furmanczyk, A. Kosowski, B. Ries, and P. Zylinski. Mixed graph edge coloring. *Discrete Mathematics*, 309(12):4027 – 4036, 2009.
- [47] P. Galinier, A. Hertz, and N. Zufferey. An adaptive memory algorithm for the k -coloring problem. *Discrete Applied Mathematics*, 156(2):267 – 279, 2008.
- [48] Philippe Galinier and Alain Hertz. A survey of local search methods for graph coloring. *Comput. Oper. Res.*, 33(9):2547–2562, 2006.
- [49] Michel Gamache, Alain Hertz, and Jérôme Olivier Ouellet. A graph coloring model for a feasibility problem in monthly crew scheduling with preferential bidding. *Computers & Operations Research*, 34(8):2384 – 2395, 2007.
- [50] Martin Gardner. *The Unexpected Hanging and Other Mathematical Diversions*. The University of Chicago Press, Chicago Illinois, 1969. ISBN 0-226-28256-2.
- [51] 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*, volume 1, pages 512 –516, 2010.
- [52] D. Gomez and J. Montero. A coloring fuzzy graph approach for image classification. *Information Sciences*, 176(24):3645 – 3657, 2006.
- [53] M. Graña, B. Cases, C. Hernandez, and A. D’Anjou. Further results on swarms solving graph coloring. In D. Taniar et al., editor, *ICCSA 2010 Part III*, number 6018 in LNCS, pages 541–551. Springer, 2010.
- [54] D.J. Guan and Zhu Xuding. A coloring problem for weighted graphs. *Information Processing Letters*, 61(2):77 – 81, 1997.
- [55] W. Gutjahr. Mathematical runtime analysis of aco algorithms: survey on an emerging issue. *Swarm Intelligence*, 1:59–79, 2007.
- [56] J. Hansen, M. Kubale, U. Kuszner, and A. Nadolski. Distributed largest-first algorithm for graph coloring. In Marco Danelutto, Marco Vanneschi,

- and Domenico Laforenza, editors, *Euro-Par 2004 Parallel Processing*, volume 3149 of *Lecture Notes in Computer Science*, pages 804–811. Springer Berlin , Heidelberg, 2004.
- [57] H. Hernández and C. Blum. Ant colony optimization for multicasting in static wireless ad-hoc networks. *Swarm Intelligence*, 3:125–148, 2009.
- [58] F. Herrmann and A. Hertz. Finding the chromatic number by means of critical graphs. *Electronic Notes in Discrete Mathematics*, 5(0):174 – 176, 2000.
- [59] A. Hertz. A new graph coloring algorithm. *Operations Research Letters*, 10(7):411 – 415, 1991.
- [60] A. Hertz, M. Plumettaz, and N. Zufferey. Variable space search for graph coloring. *Discrete Applied Mathematics*, 156(13):2551 – 2560, 2008.
- [61] L. Hsu, S. Horng, and P. Fan. Mtpso algorithm for solving planar graph coloring problem. *Expert Syst. Appl.*, 38:5525–5531, May 2011.
- [62] Limin Hu. Distributed code assignments for cdma packet radio networks. *Networking, IEEE/ACM Transactions on*, 1(6):668 –677, dec 1993.
- [63] Shin ichi Nakayama and Shigeru Masuyama. A parallel algorithm for solving the coloring problem on trapezoid graphs. *Information Processing Letters*, 62(6):323 – 327, 1997.
- [64] Davenport J.M Iman, R.L. Approximations of the critical region of the friedman statistic. *Communications in Statistics*, pages 575–595, 1980.
- [65] M. Jiang, Y.P. Luo, and S.Y. Yang. Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm. *Information Processing Letters*, 102(1):8 – 16, 2007.
- [66] A. John, A. Schadschneider, D. Chowdhury, and K. Nishinari. Characteristics of ant-inspired traffic flow. *Swarm Intelligence*, 2:25–41, 2008.
- [67] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. Optimization by simulated annealing: An experimental evaluation; part II, graph coloring and number partitioning. *Operations Research*, 39(3):378–406, 1991.

- [68] D. S. Johnson and M. A. Trick. *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, volume 26. American Mathematical Society, 1993.
- [69] D.S. Johnson and M.A. Trick, editors. *Proceedings of the 2nd DIMACS Implementation Challenge*, volume 26. American Mathematical Society, 1996. DIMACS Series in Discrete Mathematics and Theoretical Computer Science.
- [70] R. J. Kang and T. Muller. Frugal, acyclic and star colourings of graphs. *Discrete Applied Mathematics*, In Press, Corrected Proof:–, 2010.
- [71] W. Klotz. Clique covers and coloring problems of graphs. *Journal of Combinatorial Theory, Series B*, 46(3):338 – 345, 1989.
- [72] Zbigniew Kokosinski and Krzysztof Kwarcianny. On sum coloring of graphs with parallel genetic algorithms. pages 211–219, 2007.
- [73] O. Korb, T. Stutzle, and T. Exner. An ant colony optimization approach to flexible protein-ligand docking. *Swarm Intelligence*, 1:115–134, 2007.
- [74] K. Kuratowski. Sur le problème des courbes gauches en topologie. *Fund. Math.*, 15:271–283, 1930.
- [75] K. Kuratowski. A half century of polish mathematics: Remembrances and reflections. Oxford, Pergamon Press, 1980.
- [76] 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*, 89(2):195–205, 2003.
- [77] M. Larsen, J. Propp, and D. Ullman. The fractional chromatic number of mycielski’s graphs. *J. Graph Theory*, 19:411–416, 1995.
- [78] F. T. Leighton. A graph coloring algorithm for large scheduling problems. *Journal of Research of the National Bureau of Standards*, 84(6):489–506, 1979.
- [79] Frank Thomson Leighton. Finite common coverings of graphs. *Journal of Combinatorial Theory, Series B*, 33(3):231–238, December 1982.
- [80] G. Lewandowski and A. Condon. Experiments with parallel graph coloring heuristics and applicationsof graph coloring. pages 309–334, 1994.

- [81] R. Lewis and J. Thompson. On the application of graph colouring techniques in round-robin sports scheduling. *Computers & Operations Research*, 38(1):190 – 204, 2011.
- [82] R. Lewis, J. Thompson, C. Mumford, and J. Gillard. A wide-ranging computational comparison of high-performance graph colouring algorithms. *Computers & Operations Research*, 39(9):1933 – 1950, 2012.
- [83] X. Li and J. Wang. A steganographic method based upon jpeg and particle swarm optimization algorithm. *Information Sciences*, 177(15):3099 – 3109, 2007.
- [84] Z. Lu and J. Hao. A memetic algorithm for graph coloring. *European Journal of Operational Research*, 203(1):241 – 250, 2010.
- [85] B. Luzar, R. Skrekovski, and M. Tancer. Injective colorings of planar graphs with few colors. *Discrete Mathematics*, 309(18):5636 – 5649, 2009.
- [86] S. Y. Lee M. W. Carter, G. Laporte. Examination timetabling : Algorithmic strategies and applications. *The Journal of the Operational Research Society*, 47(3):373–383, 1996.
- [87] 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*, 197(3):1192 – 1201, 2009.
- [88] E. Maistrelli and D.B. Penman. Some colouring problems for paley graphs. *Discrete Mathematics*, 306(1):99 – 106, 2006.
- [89] Timir Maitra, Anindya J. Pal, Minkyu Choi, and Taihoon Kim. Hybridization of ga and ann to solve graph coloring. In Tai-hoon Kim, Adrian Stoica, and Ruay-Shiung Chang, editors, *Security-Enriched Urban Computing and Smart Grid*, volume 78 of *Communications in Computer and Information Science*, pages 517–523. Springer Berlin Heidelberg, 2010.
- [90] 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*, pages 240–245, 2000.
- [91] A. Mehrotra and M. Trick. A column generation approach for graph coloring. *INFORMS Journal On Computing*, 8(4):344–354, 1996.

- [92] J. Miskuf, R. Skrekovski, and M. Tancer. Backbone colorings and generalized mycielski's graphs, 2008.
- [93] K. Mizuno and S. Nishihara. Constructive generation of very hard 3-colorability instances. *Discrete Appl. Math.*, 156(2):218–229, 2008.
- [94] K. Mizuno and S. Nishihara. Toward ordered generation of exceptionally hard instances for graph 3-colorability. In *Discrete Applied Mathematics archive*, volume 156, pages 1–8, January 2008.
- [95] J. Mycielski. Sur le colouage des graphes. *Colloquium Mathematicum*, 3:161–162, 1955.
- [96] Isabel Méndez-Díaz and Paula Zabala. A branch-and-cut algorithm for graph coloring. *Discrete Applied Mathematics*, 154(5):826 – 847, 2006.
- [97] P. B. Nemenyi. *Distribution-free multiple comparisons*. PhD thesis, Princeton University, 1963.
- [98] Stavros D. Nikolopoulos. Coloring permutation graphs in parallel. *Discrete Applied Mathematics*, 120(1-3):165 – 195, 2002.
- [99] A. Nolte and R. Schrader. Simulated annealing and graph colouring. *Comb. Probab. Comput.*, 10:29–40, January 2001.
- [100] S. Nouyan, A. Campo, and M. Dorigo. Path formation in a robot swarm. *Swarm Intelligence*, 2:1–23, 2008.
- [101] Rei Odaira, Takuya Nakaike, Tatsushi Inagaki, Hideaki Komatsu, and Toshio 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, pages 160–169, New York, NY, USA, 2010. ACM.
- [102] G. Palubeckis. On the recursive largest first algorithm for graph colouring. *Int. J. Comput. Math.*, 85:191–200, February 2008.
- [103] P.M. Pardalos and A. Migdalas. A note on the complexity of longest path problems related to graph coloring. *Applied Mathematics Letters*, 17(1):13 – 15, 2004.

- [104] Hyoung-Keun Park, Sun-Youb Kim, Yu-Chan Ra, and Seung-Woo 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*, pages 37–41, Washington, DC, USA, 2009. IEEE Computer Society.
- [105] Karl 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*, 50(5):157.175, 1900.
- [106] G. Peterson, C. Mayer, and T. Kubler. Ant clustering with locally weighted ant perception and diversified memory. *Swarm Intelligence*, 2:43–68, 2008.
- [107] P.A. Petrosyan. Interval edge-colorings of complete graphs and n-dimensional cubes. *Discrete Mathematics*, 310(10-11):1580 – 1587, 2010.
- [108] P.A. Petrosyan, H.Z. Arakelyan, and V.M. Baghdasaryan. A generalization of interval edge-colorings of graphs. *Discrete Applied Mathematics*, 158(16):1827 – 1837, 2010.
- [109] R. Poli, J. Kennedy, and T. Blackwell. Particle swarm optimization. *Swarm Intelligence*, 1:33–57, 2007.
- [110] 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*, 37(10):1822 – 1832, 2010.
- [111] D. C. Porumbel, J. Hao, and P. Kuntz. A search space cartography for guiding graph coloring heuristics. *Computers & Operations Research*, 37(4):769 – 778, 2010.
- [112] Rong Qu, Edmund K. Burke, and Barry McCollum. Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems. *European Journal of Operational Research*, 198(2):392 – 404, 2009.
- [113] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi. Gsa: A gravitational search algorithm. *Information Sciences*, 179(13):2232 – 2248, 2009.

- [114] André Raspaud and Eric Sopena. Good and semi-strong colorings of oriented planar graphs. *Information Processing Letters*, 51(4):171 – 174, 1994.
- [115] I. Rebollo and M. Graña. Further results of gravitational swarm intelligence for graph coloring. In *Nature and Biologically Inspired Computing*, 2011.
- [116] I. Rebollo and M. Graña. *Gravitational Swarm Approach for Graph Coloring*, volume 387 of *Studies in Computational Intelligence*. Springer-Verlag, 2011.
- [117] Israel Rebollo, Manuel Graña, and Carmen Hernandez. Aplicacion de algoritmos estocasticos de optimizacion al problema de la disposicion de objetos no-convexos. *Revista Investigacion Operacional*, 22(2):184–191, 2001.
- [118] Z. Ren, J. Wang, and H. Zhang. A new particle swarm optimization algorithm and its convergence analysis. In *Genetic and Evolutionary Computing, 2008. WGEC '08. Second International Conference on*, pages 319–323, 25-26 2008.
- [119] C. W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. In *Computer Graphics*, pages 25–34, 1987.
- [120] 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*, 36(7):2295 – 2310, 2009.
- [121] B. Ries. Complexity of two coloring problems in cubic planar bipartite mixed graphs. *Discrete Applied Mathematics*, 158(5):592 – 596, 2010.
- [122] A. Rizzoli, R. Montemanni, E. Lucibello, and L. Gambardella. Ant colony optimization for real-world vehicle routing problems. *Swarm Intelligence*, 1:135–151, 2007.
- [123] S. Shah, R. Kothari, and S. Chandra. Trail formation in ants. a generalized poly urn process. *Swarm Intelligence*, 4:145–171, 2010.
- [124] Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, July, October 1948.

- [125] Justine W. Shen. Solving the graph coloring problem using genetic programming. In John R. Koza, editor, *Genetic Algorithms and Genetic Programming at Stanford 2003*, pages 187–196, Stanford, California, 94305-3079 USA, 4 December 2003. Stanford Bookstore.
- [126] S. N. Sivanandam, S. Sumathi, and T. Hamsapriya. A hybrid parallel genetic algorithm approach for graph coloring. *Int. J. Know.-Based Intell. Eng. Syst.*, 9:249–259, August 2005.
- [127] George W. Snedecor. The method of expected numbers for tables of multiple classification with disproportionate subclass numbers. *Journal of the American Statistical Association*, 29(188):389–393, 1934.
- [128] Jeremy P. Spinrad and Gopalakrishnan Vijayan. Worst case analysis of a graph coloring algorithm. *Discrete Applied Mathematics*, 12(1):89 – 92, 1985.
- [129] Tai-hoon Kim Debnath Bhattacharyya 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*, 3(3):37–53, september 2010.
- [130] P. M. Talaván and J. Yáez. The graph coloring problem: A neuronal network approach. *European Journal of Operational Research*, 191(1):100 – 111, 2008.
- [131] O. Titiloye and A. Crispin. Quantum annealing of the graph coloring problem. *Discrete Optimization*, In Press, Corrected Proof:–, 2011.
- [132] J. W. Tukey. Comparing individual means in the analysis of variance. *Biometrics*, 5:99–114, 1949.
- [133] A. Turgut, H. elikkanat, and E. Fahin. Self-organized flocking in mobile robot swarms. *Swarm Intelligence*, 2:97–120, 2008.
- [134] J. S. Turner. Almost all k-colorable graphs are easy to color. *Journal of Algorithms*, 9(1):63 – 82, 1988.
- [135] V. G. Vizing. On an estimate of the chromatic class of a p-graph. *Diskret. Analiz*, 3:23–30, 1968.
- [136] Tjark Vredeveld and Jan Karel Lenstra. On local search for the generalized graph coloring problem. *Operations Research Letters*, 31(1):28 – 34, 2003.

- [137] Herbert S. Wilf. Backtrack: An $o(1)$ expected time algorithm for the graph coloring problem. *Information Processing Letters*, 18(3):119 – 121, 1984.
- [138] David R. Wood. An algorithm for finding a maximum clique in a graph. *Operations Research Letters*, 21(5):211 – 217, 1997.
- [139] Jianshe Wu, Licheng Jiao, Rui Li, and Weisheng Chen. Clustering dynamics of nonlinear oscillator network: Application to graph coloring problem. *Physica D: Nonlinear Phenomena*, 240(24):1972 – 1978, 2011.
- [140] X. Xie and J. Liu. Graph coloring by multiagent fusion search. *Journal of Combinatorial Optimization*, 18:99–123, 2009.
- [141] K. Yadav, S. Varagani, K. Kothapalli, and V.Ch. Venkaiah. Acyclic vertex coloring of graphs of maximum degree 5. *Discrete Mathematics*, 311(5):342 – 348, 2011.
- [142] Jiaqi Yu and Songnian Yu. A novel parallel genetic algorithm for the graph coloring problem in vlsi channel routing. *International Conference on Natural Computation*, 4:101–105, 2007.
- [143] Ming-Shing Yu and Cheng-Hsing Yang. A simple optimal parallel algorithm for the minimum coloring problem on interval graphs. *Information Processing Letters*, 48(1):47 – 51, 1993.
- [144] Zhao Zhang and Hao Li. Algorithms for long paths in graphs. *Theoretical Computer Science*, 377(1-3):25 – 34, 2007.
- [145] A. A. Zykov. On some properties of linear complexes. *Mat. Sb. (N.S.)*, 24(2):163–188, 1949.

