

Literature Review of Optimisation Techniques to the Graph Colouring Problem

Scott Whitemore & Brendon Veronese

October 12, 2015

Abstract

This is an abstract.

Chapter 1

Literature Review

1.1 Introduction

1.2 Johnson - Simulated Annealing

1.3 Gravitational Swarm Intelligence

1.4 Genetic Algorithm

1.5 Flower Pollination

Chapter 2

Implementations & Original Contributions

2.1 Introduction

2.2 Random Brute Buckets

The Random Brute Buckets (RBB) algorithm presented here is a class of Successive Augmentation Technique. It is at its core an implementation of an insertion sort algorithm which samples vertices from the graph in a random order. This algorithm is guaranteed to find a valid colouring for graphs that are directed and undirected. It is however not bounded, and there is no guarantee that the algorithm will converge upon the best colouring.

2.2.1 Implementation and Problems

The RBB algorithm is presented in Algorithm 1 below.

Algorithm 1 Random Brute Buckets

```
1: procedure SOLVE(Graph  $g$ , long  $iterationLimit$ ) ▷  $g$  is predefined
2:    $currentIteration \leftarrow 0$ 
3:    $currentBestColouring \leftarrow \infty$ 
4:   Create empty list of buckets  $b$ 
5:   while  $currentIteration < iterationLimit$  do
6:     Populate vertex set  $V$ 
7:     while  $V \neq \emptyset$  &&  $|b| < currentBestColouring$  do
8:        $v \leftarrow$  random vertex from  $V$  ▷ Remove  $v$  from  $V$ 
9:        $vertexPlaced \leftarrow FALSE$ 
10:      for all  $bucket \in b$  do
11:        if  $bucket$  contains no conflicts with  $v$  then
12:          add  $v$  to  $bucket$ 
13:           $vertexPlaced \leftarrow TRUE$ 
14:          break for
15:        else continue
16:      end if
17:    end for
18:    if  $!vertexPlaced$  then ▷ create a place for the vertex
19:      create new bucket  $b_0$ 
20:      add  $v$  to  $b_0$ 
21:      add  $b_0$  to  $b$ 
22:    end if
23:  end while
24:  if  $|b| < currentBestColouring$  then
25:     $currentBestColouring = |b|$ 
26:  end if
27:   $currentIteration ++$ 
28: end while
29: return  $currentBestColouring$ 
30: end procedure
```

This algorithm always improves upon the best colouring (which is initially set to ∞ on line 3) in the first iteration, and upon subsequent iterations, if a better solution is stumbled upon randomly, that solution is set as the current best solution. The algorithm returns the best solution found after a set number of iterations set at runtime.

2.2.2 Workarounds

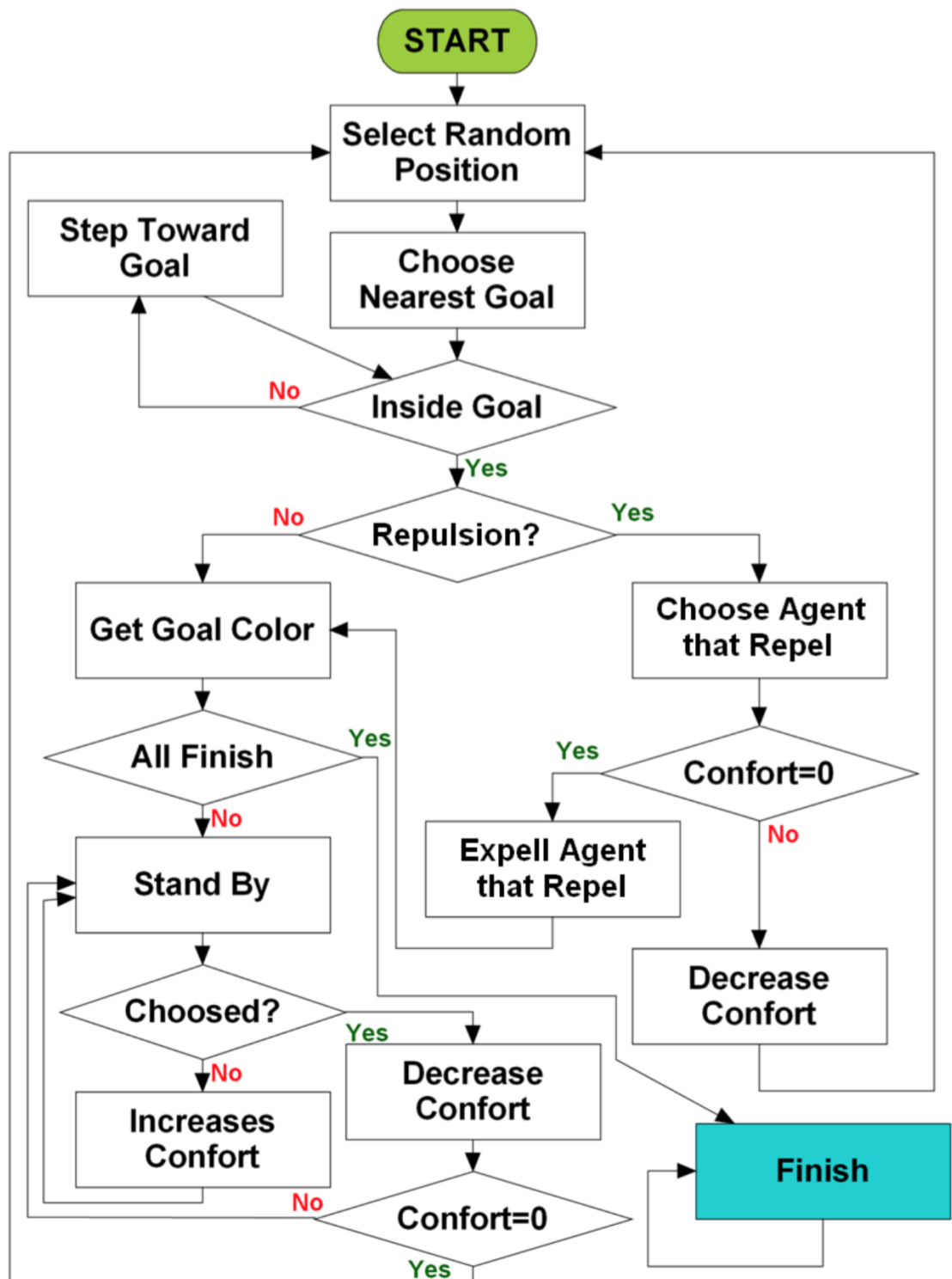
This algorithm is very fast at finding initial solutions. This algorithm does not try to improve upon solutions that it finds, so local minima offer no resistance to finding the solution. This algorithm is not guaranteed to find the best solution. Indeed even after infinite attempts, there is no guarantee that the best solution will be found. That being said, the algorithm is generally able to find solutions close to the optimal colouring for most graphs extremely quickly and with a minimum use of resources (compared to other algorithms presented in this paper).

2.3 Gravitational Swarm Intelligence

The Gravitational Swarm Intelligence Algorithm presented in this paper is a modified version of the algorithm presented in the paper "Gravitational Swarm for Graph Coloring" by Israel Carlos Rebollo Ruiz [1]. Swarm Intelligence is a model where the emergent collective behavior is the outcome of a process of self-organization, where the agents evolve autonomously following a set of internal rules for their motion, interaction with the environment and the other agents. In this algorithm, the agents are subjected to an environment subject to a version of Newtonian Gravity. Centres of attraction (wells) are placed in the environment and the agents move based upon their attraction to these wells. Each well represents a distinct grouping or colouring of the underlying graph. Agents experience a repulsive force if they try and enter a well already containing agents that prohibit a valid colouring. If an agent enters a well that is not prohibited, then it stops moving and takes on that well's colour. The algorithm ends once a set iteration limit is reached, or all agents settle into the wells, and hence a valid colouring is found.

2.3.1 Implementation and Problems

The action of each is presented below:



Each agent moves in the toric world until all agents fall into the "Stand By" State. Then once the last agent triggers the "All Finished \Rightarrow Yes" State and the current colouring is accepted as valid. This algorithm presented some issues when implemented for testing. The comfort statistic that each agent tracks can grow unboundedly if the agents repeatedly fall into wells that they are repulsed from. This causes all agents to grow in comfort, making the local minima deeper with every iteration and the chance of jumping out fall dramatically.

2.3.2 Workarounds

2.3.3 Additions

2.4 Genetic Algorithm

2.4.1 Implementation and Problems

2.4.2 Workarounds

2.4.3 Additions

2.5 Flower Pollination

2.5.1 Implementation and Problems

2.5.2 Workarounds

2.5.3 Additions

Chapter 3

Results & Discussion

3.1 Algorithms

Chapter 4

Bibliography

Bibliography

- [1] Israel Carlos Rebollo Ruiz. *Gravitational Swarm for Graph Coloring*.
<http://www.ehu.eus/ccwintco/uploads/3/3a/Tesis-Israel-final.pdf>, 2012.