Use of Differential Evolution for Graph Colouring

Scott Richmond-Wood
slhr201@exeter.ac.uk

ABSTRACT

Differential Evolution is a novel form of Evolutionary Algorithm that has been shown to be useful in global optimisation problems. In this paper, Differential Evolution is applied to the graph colouring problem with 3 colours in order to show its ability to solve combinatorial optimisation problems. Once the differential evolution algorithm has been implemented it is demonstrated on a number of DIMACS graphs and tested for optimal parameters.

KEYWORDS

Differential Evolution, Graph Colouring, Combinatorial Optimisation Problems

ACM Reference Format:
Scott Richmond-Wood. 2018. Use of Differential Evolution for Graph Colouring. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

The Graph Colouring Problem is a very common example of a combinatorial optimisation problem. The k-colouring of a graph G is an assignment of integers or colours $\{1, 2, ..., k\}$ to the nodes of G such that adjacent nodes have different colours. The problem has been shown to have applications in a number of scheduling tasks such as aircraft scheduling and bioprocessor tasks [17].

There have been a number of approaches to solving the graph colouring problem, including tabucol [12] and a number of classical integer programming models [16]. One common approach, and perhaps the most natural is a greedy heuristic. This involves sorting the vertices into a permutation and colouring sequentially [5].

This paper shall look at applying a technique known as Differential Evolution (DE) to the 3-Colouring problem (3-GCP). The technique has been shown to have some success by Fister and Brest [10]. The project shall look at applying a similar technique to a number of DIMACS graphs [1]. Then a number of experiments shall be run in an attempt to find optimal parameters.

The rest of this report shall be split into a number of sections. Section 2 shall be a literature review of relevant papers and further analysis of existing methods. Section 3 Shall elaborate on the design of the project, including pseudocode and deeper explanation of differential evolution. Section 4 shall show the results of the experiments and provide some analysis. Section 5 will discuss some improvements and avenues for further research that result from this project. Finally, Section 6 shall summarise the findings of this project.

Woodstock '18, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

2 LITERATURE REVIEW

A number of key references to the literature regarding the k-Colouring problem and DE are presented in this section, with a particular focus on how Differential Evolution has been used to solve the 3-Colouring problem in the past. Section 2.1 introduces the k-Colouring problem. A general description of DE is provided in Section 2.2. Finally, Section 2.3 provides references that show how DE can be adapted and applied to the k-Colouring problem.

2.1 k-Colouring Problem

Colouring theory in regards to graphs originates with a quite well-known problem of colouring all the countries on a map such that no two countries bordering countries shared a colour with the minimum number of colours. It appears to have been mentioned for the first time in 1852 from Morgan to Hamilton [24]. The infamous, Four Colour Problem became one of the most well-known problems in Discrete Mathematics. It caused a number of new directions in graph theory, such as chromatic polynomials by Birkhoff in 1912 [3] which was introduced to attempt to solve the problem by algebraic methods.

This problem was eventually solved with the Four Colour Theorem by Koch *et al.* in 1976 [24]. In which they used 1200 hours of computer time to find many combinations that would work. This was the first time that a famous mathematical problem was solved by extensive use of computers, making it rather important in the history of Computer Science. There were a few errors in the original proof, but they have since been corrected and the theorem was improved by Robertson *et al.* [20].

The problem has been solved, but the amount of time that it takes to run (whilst much shorter today) is still too long for most practical cases. There has been found an algorithm for determining whether a 2-colouring is possible for a graph in polynomial time, as it is just determining whether the graph is bipartite or not and thus can be found using bread-first or depth-first search [22].

The only found exact algorithms are a brute force search, which is $O(k^n)$ for a k-Colouring of a graph with n nodes, and a few dynamic programming techniques. One by Lawler, which made use of a bound on maximal independent sets [15] and gave a solution in $O(2.4423^n)$. Another by Björklund *et al.* [4], which made use of the inclusion-exclusion principle and Yates' algorithm [19]. It achieved a time of $O(2^n n)$.

Due to the time it takes to find exact solutions, many heuristics have been created to try and find good solutions or "near enough" solutions in much less time. The best known being the greedy algorithm mentioned in Section 1. The quality of the result given by this algorithm depend on the permutation chosen, there exists a permutation with the chromatic number such that the algorithm

gives a correct solution but the result can be arbitrarily bad based on the permutation.

2.2 Differential Evolution

DE is an evolutionary algorithm (EA) developed to solve real-valued optimisation problems, created by Storn and Price [21]. The algorithm is fast to converge, flexible, robust, and requires very few parameters to be set by the user. The disadvantage of the high convergence speed, however, is that the chance of getting stuck in a local optimum rises. Differential evolution is based on a mutation operator, which adds an amount obtained by the difference of two randomly chosen individuals of the current population. The problem has N decision variables, F (a real-value factor that controls amplification of differential variations) and CR (a real value, controlling the crossover operation) are parameters given by the user, g is the generation [14].

Algorithm 1: Basic Differential Evolution Algorithm

Generate initial population ;
do
  foreach *individual j in population* **do**
    Choose three numbers n_1, n_2, n_3 where $1 \leq n_1, n_2, n_3, \leq N$ with $n_1 \neq n_2 \neq n_3 \neq j$;
    Generate random integer $i_{rand} \in (1, N)$;
    foreach *parameter i* **do**
      $y^{i,g} = x^{n_1,g} + F(x^{n_2,g} - x^{n_3,g})$;
      $z_j^{i,g} = \begin{cases} y_j^{i,g} & \text{if rand()} \leq \text{CR or j} = j_{rand} \\ x_j^{i,g} & otherwise \end{cases}$
    end
    Replace $x^{i,g}$ with the child $z^{i,g}$ if $z^{i,g}$ is better ;
  end
until *not termination conditions*;

The termination condition is normally if a solution is optimal by the objective function or if the algorithm reaches a maximum number of iterations.

2.3 DE for k-Colouring

Adaptation of DE for discrete problems isn't too difficult, a simple adaptation of the objective function to limit the values to the required bounds is all that is required. In this case, we needed to clamp the values to integers which can be done either via a rounding, flooring, or ceiling methods. Fister and Brest [10] adapted DE for the 3-Colouring problem. In order to make it more effective, they made the algorithm self-adaptive and improved it with a local search heuristic. They compared their design to Tabucol [13], a Hybrid Evolutionary Algorithm (HEA) [11] and a differential algorithm using the SAW method (EA-SAW) [8]. The results indicated that their approach worked better then EA-SAW but not as well as HEA and Tabucol for finding solutions to 3-GCP.

3 DESIGN AND IMPLEMENTATION

This section covers the design of the code used, including the changes made to the standard DE Algorithm, solution representation, and the datasets used. We describe the representation of population members in Section 3.1. In Section 3.2, we describe the modifications made to the base DE Algorithm and some parameter choices. Section 3.3 describes the datasets and modifications made to them before running the tests.

3.1 Solution Representation

In order to run Differential Evolution we need a representation that can be multiplied by a scalar and that has defined arithmetic operations between two instances of said representation. The obvious choice is a vector, we use a vector where each element will represent the colour of a node in the graph. For each element of the vector, v_i, $v_i \in \{0, 1, 2\}$ this means that each element will represent one of our three colours.

3.2 Differential Evolution Algorithm

The base DE Algorithm is described above in Section 2.2. To summarise, it is an EA developed for solving real-valued problems. At each iteration, the algorithm mutates each candidate solution by a linear combination with other candidates picked at random from the current population [18]. Other strategies for combination have also since been developed. For this project, the 'best1bin' strategy [25] was selected.

3.2.1 best1bin. The best1bin strategy randomly selects two members of the population. The difference between them is used to mutate the best member of the population, this creates a new vector:

$$\vec{b'} = \vec{b_0} + m * (\vec{p_0} - \vec{p_1})$$

where $\vec{b'}$ is the new vector, $\vec{b_0}$ is the current best vector, m is the mutation rate, and $\vec{p_i}$ is the *ith* randomly selected member of the population. The trial vector is then created by choosing a random starting index and sequentially filling, in modulo, the vector with values either from $\vec{b'}$ or $\vec{b_0}$ based on a binomial distribution [7]. If the number from said binomial distribution is less than the recombination constant then the value is from $\vec{b'}$ else it is from $\vec{b_0}$. The final element chosen is loaded from $\vec{b'}$. The trial vector is then assessed by the fitness function, and replaces the original if it's better.

3.2.2 Continuous to Discrete. DE was designed for real-valued optimisation problems, but 3-GCP is a discrete problem. As described in Section 3.1, we use integers to represent each colour. The solution is rather simple, at some point throughout the program we round each element of the solution to ensure each element is an integer. We elect to do this during the fitness function so we can keep the original continuous vector and keep the variability of solutions. We also impose bounds on the DE, so that it will always return a value that will round to a valid integer for our colouring problem.

3.2.3 Fitness Function. The fitness function is perhaps the most important parameter when dealing with EAs of any kind. It gives an indication to the performance of an individual member of the population. The aim of the EA is to minimise this function. The obvious choice was to count the number of collisions by iterating through each node, checking its colour, checking its neighbours colours, and incrementing the function value for each equality. This

function results in an addition of 2 for each pair, which has no real effect on the function.

This function is very simplistic, but is quite effective. Other functions would likely require additional knowledge of the graph in question, or take longer to run.

Algorithm 2: Fitness Function

```
Get graph G and solution vector X ;
Let error = 0 ;
foreach node n in G do
    Get neighbours of n NL ;
    foreach nl in NL do
        if X[n] = X[nl] then
            error++ ;
        end
    end
end
```

3.3 Datasets

The graphs were downloaded in DIMACs format [6] and were converted into an edge list format, where each line is a pair of node ids indicating an undirected edge between the pair. A number of separate dataset were used, each of them used in past work on graph colouring. The sets vary in number of nodes, edge density, and optimal colouring.

Originally, the plan was to use a multitude of different datasets. But due to time constraints, the data was limited to the Mycielski set of graphs.

3.3.1 Mycielski Graphs. Mycielski graphs are a particular class of graphs, where the graph M_k of order k is triangle-free with chromatic number [23] k which has the smallest possible number of vertices [9]. The number of vertices in the k-Mycielski graph can be calculated as:

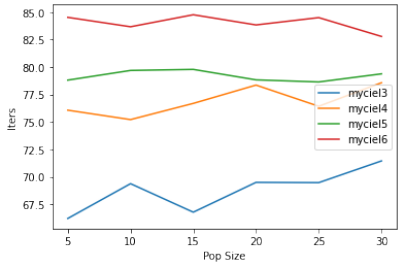
$$n(M_k) = \begin{cases} 1, & \text{if } n = 1 \\ 3.2^{n-2} - 1, & \text{if n} > 1 \end{cases}$$

We also know from the dataset site [2] that none of these graphs have no perfect 3 graph colouring, so we are looking for a minimum error as opposed to a perfect solution.

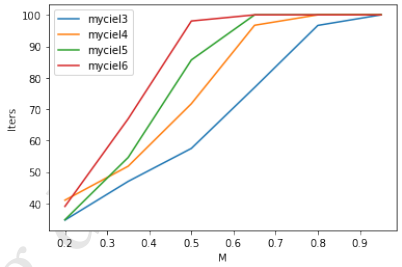
4 EXPERIMENTATION

After some base tests on some simple graphs, the algorithm was tested on some graphs Mycielski graphs [9]. Each graph was tested with a number of parameter sets which varied in population size, mutation rate, and recombination constant. Two measures were examined: the speed of convergence and the fitness of the solution. New results were constructed to illustrate the effect of each parameter on these values. This data was constructed by iterating through the set of values and averaging all the tests using each individual value for that parameter.

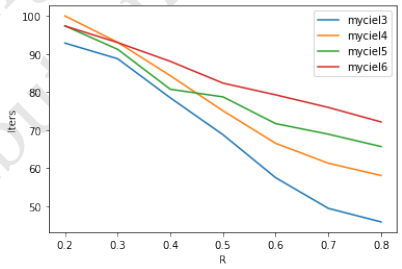
It was found that myciel3 yielded very few insights, likely due to the small size of the graph. It shows that the method operates well on small data though. While myciel4 provided a little insight, it still was difficult to gleam much as the variations were so small.



(a) Population Size



(b) Mutation Rate



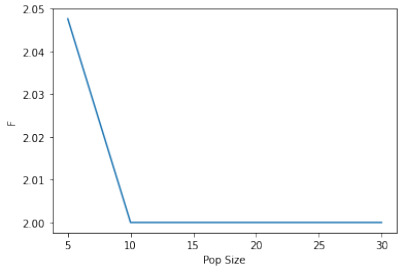
(c) Recombination Constant

Figure 1: Results for convergence testing on Mycielski graphs.

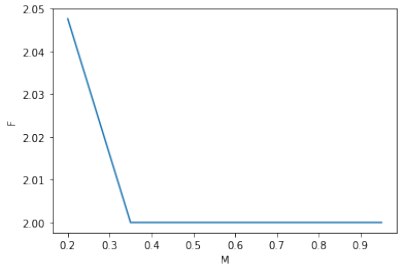
The graph in Figure 1a, show that population has very little effect on convergence speed for larger graphs. But it leads to better fitness as shown in Figures 4a and 5a. This is probably as due to having a larger population, there are more members and thus they are more spread out over the fitness landscape. Meaning it's less likely to get caught in a local optimum, but it's at the cost of speed.

Increasing mutation rate likewise increases convergence time but, whilst it begins to lower the fitness function, after a certain point increasing the mutation rate produces less fit solutions. This makes sense as increasing the mutation rate too far begins to essentially create random solutions, which won't exhibit many qualities of the old best solution.

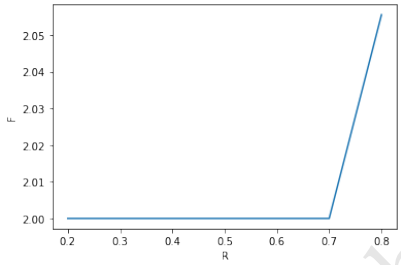
Lastly, the recombination constant seems to increase the speed of convergence as it increases. However,a high recombination rate appears to create less fit solutions if you examine Figures 4c and 5c. Again this makes sense, as the recombination constant defines how likely the element is from $\vec{b'}$ so as it increases the vector is under heavier influence from the random selection.



(a) Population Size

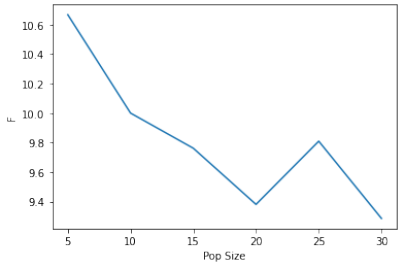


(b) Mutation Rate

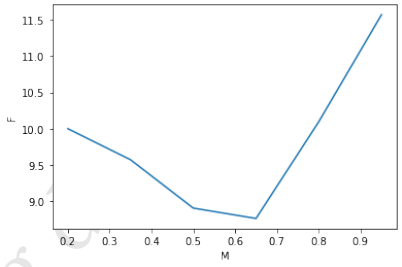


(c) Recombination Constant

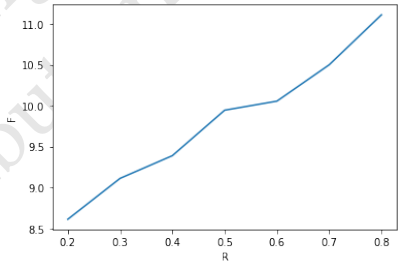
Figure 2: Results for fitness of myciel3.



(a) Population Size



(b) Mutation Rate



(c) Recombination Constant

Figure 3: Results for fitness of myciel4.

5 POSSIBLE IMPROVEMENTS AND FURTHER RESEARCH

It is unfortunate that time did not allow testing on the other datasets gathered. An interesting first avenue of investigation would be test the method on these other graphs. Further to this, there are a number of characteristics in graphs that may affect how well the method works. Comparing the number of nodes, vertices, average vertex order are all simple characteristics to monitor. Further to this research into how more complex structures (cycles, triangles, etc.) affect the convergence and fitness of solutions would also provide interesting insight.

There are also a number of improvements that could be made by using a hybrid approach similar to Fister *et al.* [10], or perhaps by changing the differential evolution strategy in some other way. As best1bin may not be the optimal strategy for this problem.

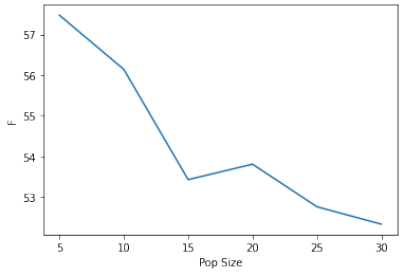
Lastly, a generalised expansion of the problem to a k-GCP as opposed to just the 3-GCP problem could be an interesting problem to see how well this methodology scales in both speed and accuracy.

6 CONCLUSION

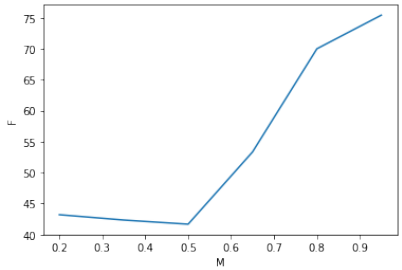
The Differential Evolution Algorithm has proved it can find results for the 3-GCP on smaller graphs (as shown in Figure 6) and makes good attempts even where a solution is impossible. The variations caused by population size, mutation rate, and the recombination constant become more and more prevalent as the graphs increase in size and demonstrate the trade-off between convergence and exploration. Further work could be done, following a number of different avenues as explained above. Overall, the project has been quite successful in achieving its aims.

REFERENCES

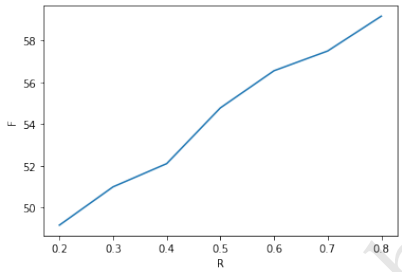
[1] URL: http://cedric.cnam.fr/~porumbed/graphs/.
[2] URL: https://mat.tepper.cmu.edu/COLOR/instances.html.
[3] George D Birkhoff and Daniel C Lewis. "Chromatic polynomials". In: *Transactions of the American Mathematical Society* 60.3 (1946), pp. 355–451.
[4] Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. "Set partitioning via inclusion-exclusion". In: *SIAM Journal on Computing* 39.2 (2009), pp. 546–563.
[5] John Adrian Bondy, Uppaluri Siva Ramachandra Murty, et al. *Graph theory with applications*. Vol. 290. Macmillan London, 1976.
[6] *Dimacs*. URL: https://www.cs.utexas.edu/users/moore/acl2/manuals/current/manual/index-seo.php/SATLINK____DIMACS.
[7] AWF Edwards. "The meaning of binomial distribution". In: *Nature* 186.4730 (1960), pp. 1074–1074.



(a) Population Size

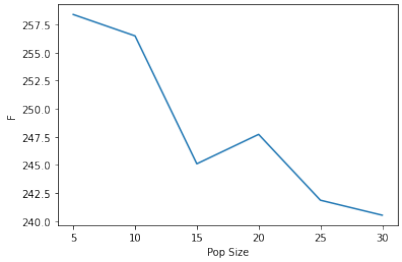


(b) Mutation Rate

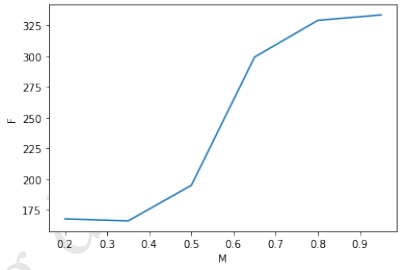


(c) Recombination Constant

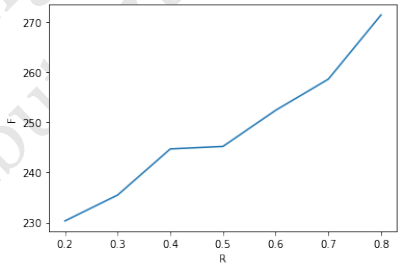
Figure 4: Results for fitness of myciel5.



(a) Population Size



(b) Mutation Rate



(c) Recombination Constant

Figure 5: Results for fitness of myciel6.

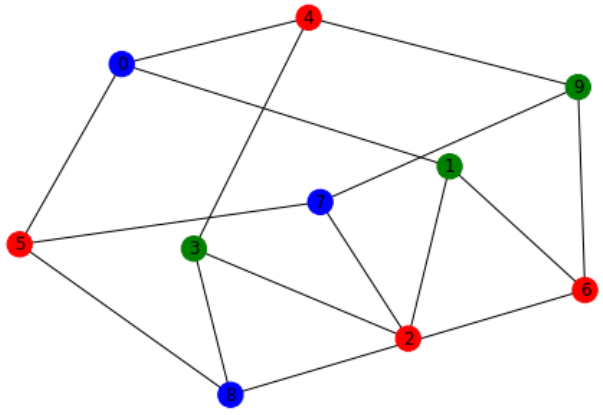


Figure 6: Result of algorithm on Petersen Graph.

[8] Ágoston E Eiben, Jan K Van Der Hauw, and Jano I van Hemert. "Graph coloring with adaptive evolutionary algorithms". In: *Journal of Heuristics* 4.1 (1998), pp. 25–46.

[9] Genghua Fan. "Circular chromatic number and Mycielski graphs". In: *Combinatorica* 24.1 (2004), pp. 127–135.

[10] Iztok Fister and Janez Brest. "Using Differential Evolution for the Graph Coloring". In: *IEEE Symposium on Differential Evolution* (Nov. 2012). DOI: 10.1109/SDE.2011.5952075.

[11] Philippe Galinier and Jin-Kao Hao. "Hybrid evolutionary algorithms for graph coloring". In: *Journal of combinatorial optimization* 3.4 (1999), pp. 379–397.

[12] Philippe Galinier and Alain Hertz. "A survey of local search methods for graph coloring". In: *Computers & Operations Research* 33.9 (2006), pp. 2547–2562.

[13] Alain Hertz and Dominique de Werra. "Using tabu search techniques for graph coloring". In: *Computing* 39.4 (1987), pp. 345–351.

[14] Shafqat Ullah Khan et al. "Correction of faulty sensors in phased array radars using symmetrical sensor failure technique and cultural algorithm with differential evolution". In: *The Scientific World Journal* 2014 (2014).

[15] Eugene L Lawler and LAWLER EL. "A Note on the Complexity of the Chromatic Number Problem." In: (1976).

[16] Enrico Malaguti and Paolo Toth. "A survey on vertex coloring problems". In: *International transactions in operational research* 17.1 (2010), pp. 1–34.

[17] Dániel Marx. "Graph colouring problems and their applications in scheduling". In: *Periodica Polytechnica Electrical Engineering (Archives)* 48.1-2 (2004), pp. 11–16.

[18] Alberto Moraglio, Julian Togelius, and Sara Silva. "Geometric differential evolution for combinatorial and programs spaces". In: *Evolutionary computation* 21.4 (2013), pp. 591–624.

[19] Hans Riedwyl. "Modifying and using Yates' algorithm". In: *Statistical papers* 39.1 (1998), pp. 41–60.
[20] Neil Robertson et al. "The four-colour theorem". In: *journal of combinatorial theory, Series B* 70.1 (1997), pp. 2–44.
[21] Rainer Storn and Kenneth Price. "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces". In: *Journal of global optimization* 11.4 (1997), pp. 341–359.
[22] Robert Tarjan. "Depth-first search and linear graph algorithms". In: *SIAM journal on computing* 1.2 (1972), pp. 146–160.
[23] N Vijayaditya. "On total chromatic number of a graph". In: *Journal of the London Mathematical Society* 2.3 (1971), pp. 405–408.
[24] V Voloshin. "Graph Coloring: History, results and open problems". In: *Alabama Journal of Mathematics, Spring/Fall* (2009).
[25] Evgeniya Zhabitskaya and Mikhail Zhabitsky. "Asynchronous differential evolution". In: *International Conference on Mathematical Modeling and Computational Physics*. Springer. 2011, pp. 328–333.