Comparison of a Memetic Evolutionary Algorithm to a Local Search when Finding Solutions to a Modified Maximum Clique Problem

Ian J. Baird

December 6, 2003
## Contents

1 Abstract 3

2 Introduction 3
   2.1 Motivation .......................... 3
   2.2 Background .......................... 4
   2.3 Research .............................. 5

3 Methodology 5
   3.1 Overview .............................. 5
   3.2 Design of the Local Search ............... 5
      3.2.1 Description ....................... 5
      3.2.2 Algorithm ......................... 6
   3.3 Design of the Memetic EA ............... 7
      3.3.1 Description ....................... 7
      3.3.2 Fitness Function .................. 7
   3.4 Testing ............................... 7
   3.5 Analysis .............................. 8

4 Experimental Results 9

5 Analysis 10
   5.1 Z-Test .................................. 10
      5.1.1 Null Hypothesis .................. 10
      5.1.2 Alternative Hypothesis .......... 10
      5.1.3 Level of Significance .......... 10
      5.1.4 Test Statistic .................. 10
      5.1.5 Sample Sizes ..................... 10
      5.1.6 Critical Region ................ 11
      5.1.7 Decision ......................... 11

6 Conclusion 12

7 Future Work 12
1  Abstract

This paper compares the application of a heuristic-based local search engine and a Memetic Evolutionary Algorithm to find solutions to constrained and modified Maximum Clique problems. The Maximum Clique problem has been shown to be an NP-Hard problem. The heuristics used in the local search are explained as well as it’s application to the Memetic EA. The resulting solutions are analyzed using statistical methods to conclusively prove the value of the Memetic Evolutionary Algorithm approach over the use of a local search when find the solution of a Modified Maximum Clique Problem.

Keywords

Memetic Evolutionary Algorithms, Maximum Clique, NP-Hard

2  Introduction

Subsection 2.1 will provide the motivation for this research, the following subsections will provide background on the Maximum Clique problem, evolutionary computation, and the practical applicability of the solution with finally the last subsection posing the questions addressed by this research.

2.1  Motivation

A graph in which each graph edge is replaced by a directed graph edge is called a digraph. A “clique” is a wholly connected subgraph within this graph (where each node is connected to the other via an edge). For the purpose of this experiment, a simple constraint and a simple relaxation are added to the definition of a clique. The constraint is that for each node in the clique, there must be at least one other node within the clique which an edge originating from the node must connect to. The definition of clique was relaxed by allowing a clique to be only partially connected, as long as each node in the collect is joined to another node by an edge originating from it. In fact, under this relaxation, a cycle (such as a ring structure) would be a completely allowable solution, whereas in the traditional Maximum Clique problem, it would be considered invalid. In addition to these modifications to
the definition of a clique, a certain number of cliques is desired. This desired number of cliques is dependant on the maximum size of the cliques.

Finding the largest clique within the digraph (with a maximum size of \( n \)) is known as the Maximum Clique problem. For this experiment a lower-bound is placed on the clique size. The unaltered Maximum Clique problem is known to be in the class of problems known as “NP-Hard.” NP-Hard is a “class of decision problems that contains all problems \( H \) such that for all decision problems \( L \) in NP there is a polynomial-time many-one reduction to \( H \).” [?] Even with the modifications made to this problem, it is conjectured that the new problem is still a member of the class NP-Hard. However, proving this is non-trivial and beyond the scope of this research.

In a practical setting, the initial digraph is created by having a class of primary-level (real or simulated) pick 5 other classmates whom they would most like to work with. Each student becomes a node in the graph, and a directed edge is drawn from one node to another node if and only if the student represented by the node at the beginning of the edge has chosen the student represented by the node at the terminal end of the edge. This
representation has also been called a sociograph \[ ? \] and can be used by primary-level teachers to organize a class into small groups of size 4, 5, and 6. A group of students in such a configuration corresponds to a clique in the digraph.

2.2 Background

The original Maximum Clique problem has been widely studied, as it is considered to be one of the fundamental problems in the field of Computer Science research. In Haynes’ paper “A Comparison of Random Search versus Genetic Programming as Engines for Collective Adaptation,” \[ ? \] he compares the performance of a Genetic Programming based approach to a pure random search and finds that the use of Genetic Programming, coupled with collective memory, is more effective at finding maximum cliques in an undirected graph than a pure random search when the search space is complex. For simpler search spaces, pure random search was shown to be more effective than the Genetic Programming based approach. In “Clique Detection as a Royal Road Function,” Haynes also conjectures that the other Genetic Algorithm based approach used in the experiment could be improved by the addition of local search or collective memory. \[ ? \] Other exotic EA-based methods have also been attempted. A Scatter-Search technique was employed by Luis Cavique \[ ? \] to solve the Maximum Clique problem, with some success. This EA based Scatter Search is slower, but was found to provide higher quality results than a pure local search algorithm.

2.3 Research

The following questions will be answered by this research:

- How to construct a Memetic EA for the modified Max Clique problem?
- For which parameters and datasets can we conclude that the Memetic EA produces higher quality solutions than the local search?
- Will the Memetic EA and/or the local search produce results more quickly than the hour it usually takes for a teacher to find the optimal groupings manually?
3 Methodology

3.1 Overview

The local search from *The Small Grouper For Windows* was used as a baseline to compare against the performance of the Memetic EA. The Z-Test, with \( \alpha = .05 \) was used to test and verify that the Memetic EA produced results which were significantly better than the local search based approach. A dataset of 18 students gleaned from an actual survey was used, along with randomly generated datasets of 30 and 40 students respectively.

3.2 Design of the Local Search

3.2.1 Description

The local search was implemented as part of *The Small Grouper for Windows*, an educational application targeted at the primary-level school teacher. The engine implements a “greedy algorithm” where the groups are seeded with group leaders and students are grouped in successive loops. If a student is ungrouped at the end of the grouping cycles, the student is grouped in any available underfull group. This approach provides acceptable results for small class sizes and executes very rapidly.

3.2.2 Algorithm

Refer to Algorithm 1 at the end of the report.

3.3 Design of the Memetic EA

3.3.1 Description

A Steady State EA served as the basic model for the Memetic EA, as it is usually better than a Generational EA for a static optimization problem where the data and the fitness function do not change with time. [?] A Memetic EA differs from a traditional EA in that it combines a heuristic based element of local search with a traditional EA in order to improve convergence speeds. The initial population for the Memetic EA was created using a set of heuristics borrowed from the local search. Group “leaders”, corresponding to nodes in which a significant amount of edges terminate, were seeded into the
solution, which was represented as an integer array. The leaders’ positions in the solution were marked as immutable for mutation and crossover. The rest of the solution was filled in by selecting students at random from the rest of the class list until all of the students were grouped. A rank-based selection model was used for the Memetic EA, where the number of individuals selected was made equal to the Culling Rate parameter. An equal number of individual solutions were culled from the population, to make room for the offspring. A uniform crossover was implemented, with every position in the solution (except for those marked as immutable) available for inclusion in the offspring. The Mersenne Twister Random Number Generator was used to generate uniformly distributed, low-reciprocity random numbers to drive the crossover and mutation operators. The mutation operator was a uniform mutation operator, where every position, except for those marked as immutable, has an equal probability of mutation. Mutation was implemented by swapping an element of the solution with another mutable element from the solution.

3.3.2 Fitness Function

Refer to Algorithm 2 at the end of the report.

3.4 Testing

Each dataset was processed by both the local search and the Memetic EA, creating groups (cliques) of size 4, 5, and 6. Each test was executed 30 times for each group size parameter setting (4, 5, or 6), making for a total of 90 tests per datafile. Three datafiles were tested for a grand total of 270 tests for the entire experiment. This number of tests was required in order to facilitate the proper use of the Z-Test for verification and analysis. Rough timings were taken but were not scaled into the final result as they were both found to be well within the limits.

3.5 Analysis

The Z-Test was performed for each group size on each dataset, in order to see if the Memetic EA produced higher quality results than the local search approach. Graphs, created using gnuplot, were created using the data from the output of these tests. For the Z-Test analysis, the Memetic
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Class Size 18</th>
<th>Class Size 30</th>
<th>Class Size 40</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population Size</td>
<td>100</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>Culling Rate</td>
<td>8</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Number of Tests</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
</tbody>
</table>

Figure 2: Memetic EA Parameter Settings

EA’s best solution’s fitness from each test was entered into a spreadsheet, transformed, and then compared against the result from the best solution found by the local search. If the Null Hypothesis was rejected, the Memetic EA was considered to have produced a higher quality result than the local search. Otherwise, the Memetic EA was considered to have not proven to produce a higher quality result than the local search, but was not considered to have produced an inferior result either.

4 Experimental Results

See Figures 3, 4, 5, and 6.

5 Analysis

5.1 Z-Test

5.1.1 Null Hypothesis

The Null Hypothesis assumes that the mean of the population of best fitnesses produced by the Memetic EA ($\mu_{EA}$) is equal to the mean of the population of best fitnesses produced by the local search.

$$H_0 : \mu_{EA} - \mu_{LS} = 0$$

5.1.2 Alternative Hypothesis

The Alternative Hypothesis, which will be assumed to be true if the Null Hypothesis is rejected, assumes the difference between the mean of the population of best fitnesses produced by the Memetic EA ($\mu_{EA}$) and the mean of the population of best fitnesses produced by the local search ($\mu_{LS}$) is greater than zero.
Figure 3: Table of “Best Fitness” Values
\[ H_1 : \mu_{EA} - \mu_{LS} > 0 \]

5.1.3 Level of Significance

An alpha value of 0.05 was chosen, as it is most often used with the Z-Test and represents a 5% Level of Significance.
\[ \alpha = 0.05 \]

5.1.4 Test Statistic

The Test Statistic is defined by the equation below, if it falls outside of the critical region, the Null Hypothesis is rejected for the test and the Alternative Hypothesis will be accepted.
\[ Z = \frac{\bar{X}_{EA} - \bar{X}_{LS}}{\sigma_{EA - LS}} \]

5.1.5 Sample Sizes

A sample size of 30 fitness values was chosen as this is the minimum sample size required to perform a valid Z-Test.
\[ n_{EA} = n_{LS} = 30 \]

5.1.6 Critical Region

The Critical Region represents the region under the Normal Curve which the Test Statistic must fall outside of for the Null Hypothesis to be rejected.
\[ z \geq z_{0.05} = 1.65 \]

5.1.7 Decision

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline
\multicolumn{3}{|c|}{Class Size 18} & & & & & & & & \\
\hline
GS 4 & GS 5 & GS 6 & & & & & & & & \\
\hline
AVG & 17.00 & 16.30 & 0.70 & 18.00 & 18.00 & 0.00 & 18.00 & 18.00 & 0.00 \\
STDDEV & 0.00 & 0.65 & 0.65 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
\mu_{EA} - \mu_{LS} & 0.70 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
z & 5.89 & 0.00 & 0.00 & & & & & & & \\
z_{0.05} & 1.65 & 1.65 & 1.65 & & & & & & & \\
reject h0? & yes & no & no & & & & & & & \\
\hline
\end{tabular}
\caption{Z-Test Results for Class Size 18}
\end{table}
Figure 5: Z-Test Results for Class Size 30

Note: Due to the limitations of the tools used to analyze the data, a very small number was added to all of the standard deviations which were equal to zero. While this does not affect the final output of the Z-Test, it does cause some noticeable irregularities, such as the very large number in the “Z-Test Results for Class Size 30” table.

Figure 6: Z-Test Results for Class Size 40

In the 7 tests where $z \geq 1.65$, the Null Hypothesis was rejected and it was concluded that for a specific parameter set and the input file, the Memetic EA provided higher quality solutions to than the local search. In the other 2 tests where the Null Hypothesis was unable to be rejected, it was concluded that the Memetic EA did not provide higher quality results than the local search was able to.

Speed was not an issue as both algorithms had sub-five minute running times for each test test on the P4 2Ghz testbed. The process (using sociograms) takes over an hour to complete by hand for anything above a trivial sized class size, so the two techniques explored in this experiment easily satisfied the running time requirement.
6 Conclusion

For the majority of the tests, the Memetic EA produced verifiably better results than the local search. While the local search performs well on the smaller sized datasets and for smaller groups, the Memetic EA produces higher quality results on the larger, more complex datasets. The Memetic EA also falls well within the allowable running time requirement (one hour) as laid out in the aforementioned research questions.

7 Future Work

The constraints and relaxations applied to the Maximum Clique problem in this experiment may have had the effect of changing the problem’s class from NP-Hard to P or NP. This should be verified by reducing this new problem into an instantiation of a previously classified problem which is known to be NP-Hard, P, or NP.

The Memetic EA can produce weakly connected subgraphs such as cycles instead of strongly connected cliques. It would be interesting to see if more strongly connected subgraphs (cliques) created a higher level of group cohesion in the classroom.

A scaling of the results in proportion to the running time of the algorithm would be appropriate for a more in-depth evaluation of the local search vs. the Memetic EA. This would allow one to decide which technique to use, not only by looking at the complexity of the data and the size of the groups asked for, but by any running time requirements there may be.
**Algorithm 1:** Local Search - “Greedy” Inner Loop

**Input:** A list of students who are not leaders $\text{class} = \{\text{student}_1, \text{student}_2, \ldots, \text{student}_n\}$, the number of groups $\text{numGroups}$ the user wishes to create, the maximum number of cycles allowed $\text{maxCycles}$, and the list of students who are “leaders” $\text{leaders} = \{\text{leader}_1, \text{leader}_2, \ldots, \text{leader}_{\text{numGroups}}\}$.

**Output:** A set of groups $\text{groups} = \{\text{group}_1, \text{group}_2, \ldots, \text{group}_m\}$.

$\text{GROUPSTUDENTS}(\text{class}, \text{numGroups}, \text{maxCycles})$

1. $\text{groups} \leftarrow \{\{\text{leader}_1\}, \{\text{leader}_2\}, \ldots, \{\text{leader}_{\text{numGroups}}\}_{\text{numGroups}}\}$
2. $\text{cycle} \leftarrow 0$
3. while $(\text{studentsGrouped} <> \text{totalStudents})$ and $(\text{cycle} < \text{maxCycles})$
   4. foreach student in students
   5.   $\text{studentGrouped} \leftarrow \text{false}$
   6.   foreach group in groups
   7.     foreach member in group
   8.       if student.chooses(member)
   9.         group $\leftarrow$ group + student
10.        students $\leftarrow$ students $-$ student
11.     break
12.   if studentGrouped = true
13.      studentsGrouped $\leftarrow$ studentsGrouped + 1
14.     cycle $\leftarrow$ cycle + 1
15.   foreach student in students
16.     groups $\leftarrow$ GROUPINUNDERFULL-GROUP(groups, student)
17. return groups
Algorithm 2: Memetic EA Fitness Function

**Input:** A list of integer elements $iaSoln$ representing a solution, and adjacency matrix of boolean values representing the diagraph $adjMat$, and a list of integer pairs $bounds$ representing the bounds of the groups contained in the solution.

**Output:** A float $fitness$ representing the fitness of the solution.

$FITNESS(iaSoln, adjmat, bounds)$

1. $fitness \leftarrow 0.0$
2. $\forall boundPair \in bounds$
3. \hspace{1em} $group \leftarrow iaSoln[boundPair[0] : (boundPair[1] - 1)]$
4. $\forall member \in group$
5. \hspace{1em} $\forall candidate \in group$
6. \hspace{2em} $\text{if } adjMat[member][candidate] = \text{true}$
7. \hspace{2em} $fitness \leftarrow fitness + 1.0$
8. \hspace{1em} $\text{break}$
9. $\text{return } fitness$