Clique Detection via Genetic Programming

Thomas Haynes & Dale Schoenefeld
Department of Mathematical & Computer Sciences
The University of Tulsa
600 South College Avenue
Tulsa, OK 74104–3189
e–mail: [haynes,dschoen]@euler.mcs.utulsa.edu
(918) 631–[3234,3140]

Abstract

Genetic programming is applied to the task of finding all of the cliques in a graph. Nodes in the graph are represented as tree structures, which are then manipulated to form candidate cliques. The intrinsic properties of clique detection complicates the design of a good fitness evaluation. We analyze those properties, and show the clique detector is found to be better at finding the maximum clique in the graph, not the set of all cliques.

Category: Genetic Programming

1 Introduction

Determining whether an undirected graph contains a clique of size $\geq k$ is NP complete. In this paper, Genetic Programming (GP) [10] techniques are utilized to find cliques in a graph. A pure Genetic Algorithm (GA) approach was considered, but natural encodings resulted in variable length chromosomes. GPs are ideal for representing variable length chromosomes.

A collection of cliques in a graph can be represented as a list of a list of nodes which, in turn, can be represented by a tree structure, as in Figure 1. Since a collection of cliques can be cast into a tree structure, it is easy to represent in a GP system. The trick is that the GP S–expression is a data structure rather than a program.

1.1 Definition of Clique

Given a graph $G = (V, E)$[1], our goal is to obtain the set of all cliques of G. A subgraph $G\prime$ of G is a clique if

$$G\prime = \quad (V\prime, E\prime) \;\; where \quad V\prime \subseteq V,$$

[1]We assume that G is undirected.

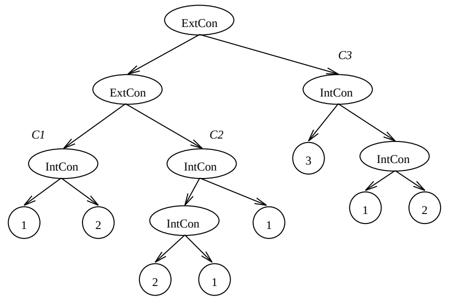

Figure 1: Example S–expression for 6 node graph.

$$E\prime \subseteq E, \quad and$$
$$V\prime \times V\prime - \{(v_i, v_i) \mid v_i \in V\prime\} = E\prime.$$

Less formally, a clique of G is a complete subgraph of G. We denote a clique by the set of vertices in the complete subgraph. Our goal is to find all cliques of G. Since the subgraph of G induced by any subset of the vertices of a complete subgraph of G is also complete, it is sufficient to find all maximal complete subgraphs of G. We refer to a maximal complete subgraph of G as a maximal clique. Figure 2 shows a graph with 6 nodes. The maximal cliques are $\{1,2,3,4,6\}$ and $\{3,4,5\}$.

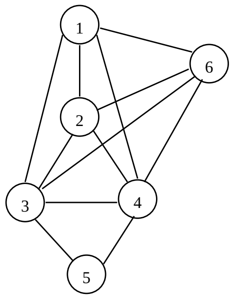

Figure 2: Example graph.

2 Genetic Programming

Holland's work on adaptive systems [8] produced a class of biologically inspired algorithms known as genetic algorithms (GAs) that can manipulate and develop solutions to optimization, learning, and other types of problems. In order for GAs to be effective, the candidate solutions should be represented as n–ary strings (though some recent work has shown that GAs can be adapted to manipulate real-valued features as well). Though GAs are not guaranteed to find optimal solutions, they still possess some nice provable properties (optimal allocation of trials to substrings, evaluating exponential number of schemas with linear number of string evaluations, etc.), and have been found to be useful in a number of practical applications [4].

Koza's work on Genetic Programming [10] was motivated by the representational constraint, i.e. fixed length encodings, in traditional GAs. He claims that a large number of apparently dissimilar problems in artificial intelligence, symbolic processing, optimal control, automatic programming, empirical discovery, machine learning, etc. can be reformulated as the search for a computer program that produces the correct input–output mapping in any of these domains. To facilitate this search, he uses the traditional GA operators for selection and recombination of individuals from a population of structures, and applies the operators on structures represented in a more expressive language than used in traditional GAs. The representation language used in GPs are computer programs represented as Lisp S–expressions. GPs have attracted a large number of researchers because of the wide range of applicability of this paradigm, and the easily interpretable form of the solutions that are produced by these algorithms [9, 11].

3 Encoding of the Problem

3.1 Representation Scheme

Each S–expression in a GP pool will represent sets of candidate maximal cliques. The function and terminal sets are $F = \{\textit{ExtCon, IntCon}\}$ and $T = \{\textit{1}\ldots\textit{\#nodes}\}$. *ExtCon* "separates" two candidate maximal cliques, while *IntCon* "joins" two candidate cliques to create a larger candidate clique. Graphs are encoded in the DIMACS Challenge file format, which can be found at [14].

3.2 Fitness Measure

The fitness evaluation will be composed of two parts: a reward for clique size and a reward for the number of cliques in the tree. Since we want to gather the maximal complete subgraphs, we want the reward for size to be greater than that for the number of cliques. We also want to make sure that we do not reward for a clique either being in the tree twice or being subsumed by another clique. The first will falsely inflate the fitness of the individual, while the second will invalidate the goals of the problem.

The algorithm for the fitness evaluation is:

1. Parse the S–expression into a sequence of candidate maximal cliques, each represented by an ordered list of vertex labels.

2. Throw away any duplicate candidate maximal cliques and any candidate maximal cliques that are subsumed by other candidate maximal cliques.

3. Throw away any candidate maximal cliques that are not cliques.

The formula for measuring the fitness is:

$$F = \alpha c + \sum_{i=1}^{c} \beta^{n_i},$$

where $c = \#\ \textit{of valid candidate maximal cliques}$ and $n_i = \#\ \textit{nodes in clique}_i$. Both α and β are configurable by the user. Note that β has to be large enough so that a large clique

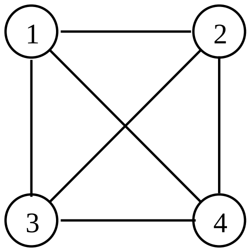

Figure 3: Example four node graph.

contributes more to the fitness of one S–expression than a collection of proper subcliques contributes to the fitness of a different S–expression. For example, consider the graph in Figure 3. It is clear that there is only one maximal clique: $C_4 = \{1,2,3,4\}$. However, there are four subcliques of cardinality three: $C_3 = \{\{1,2,3\},\{1,2,4\},\{1,3,4\},\{2,3,4\}\}$.

The various resultant fitnesses, as β is varied, and α is held constant, are visually shown in Figure 4. Note that it is not until β is larger than the cardinality of the clique that the desired result is found, see Table 1. Simply put, with the current fitness function, β must be chosen to respect the cardinality of maximum cliques in a graph. With other choices for α and β, our fitness function is more suited to determining the largest maximal clique in a graph, rather than the set of all maximal cliques.

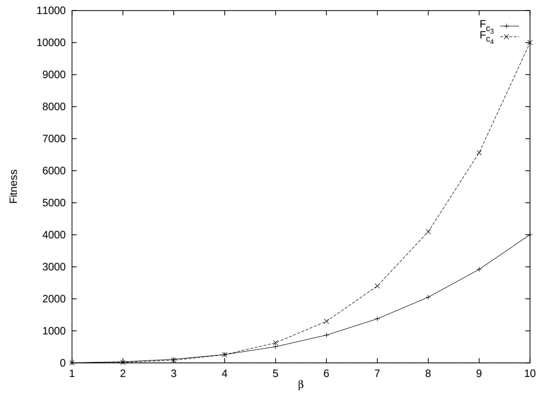

Figure 4: Fitness for four node graph.

4 Example Encodings

The encoding presented as Figure 1 is interpreted to produce $C = \{\{1,2,3\}\}$ as the set of candidate maximal cliques as follows. There are two problems in this S–expression. First, the candidate maximal clique denoted by $C1 = \{1,2\}$ is subsumed in the candidate maximal clique $C3 = \{1,2,3\}$ and, hence, $C1$ is thrown away. Second we eliminate duplication of node 1 and assume that $C2 = \{1,2\}$. As with $C1$, $C2$ is thrown away since $C2$ is subsumed by $C3$. With $\alpha = 50$ and $\beta = 10$ the fitness for this chromosome is 1050.

The S–expression corresponding to generation 0 for the 10 node graph presented in Figure 5 is presented in Figure 6. The set of candidate maximal cliques represented by this

α	β	F_{C_4}	F_{C_3}
1	1	2	5
1	2	17	36
1	3	81	112
1	4	257	260
1	5	626	504
1	6	1297	868
1	10	101001	4004

Table 1: Fitness for both the clique of cardinality four and four connected sets of cardinality three, for different β.

chromosome is: $C = \{\{6\}, \{2\}, \{4, 7, 8\}\}$. With $\alpha = 50$ and $\beta = 10$ the fitness for this chromosome is 1050.

5 Experimental Results

The clique detection problem was implemented in a modified version of GPengine [7], a Strongly Typed Genetic Programming system. Specifically, the problem of *ExtCon* functions having to have either a parent which is an *ExtCon* function or be the root, was addressed.

In the next subsections, we will present some contrived graphs to illustrate properties of our maximal clique detector. We have performed experiments with sample test cases from the DIMACS repository [14]. For the johnson16–2–4.clq graph, with 120 nodes and a largest clique of size 8, we were able to detect cliques of size 7. For the hamming8-2.clq graph, with 64 nodes and a largest clique of size 4, we were able to detect cliques of size 4. It should be realized that these graphs are intended for a different problem, i.e. finding the largest clique.

5.1 Example Graph I

Figure 5 is a ten node graph that was used to test the clique detection system. The best and average fitness per generation are plotted in Figure 7. The system steadily improves after generation 100, and the global optimum is found around generation 375. By generation 500, it is evident that a plateau has been reached, and that the average fitness is only increasing slightly. S–expressions for the candidate maximal cliques corresponding to generations 0, 100, 200, 300, and 400 can be found in [6]. The resultant maximal cliques for generations 0, 100, 200, 300, and 400 are shown in Table 2. Notice the steady addition of cliques of cardinality three. Clearly by generation 400, all of the maximal cliques have been found.

5.2 Example Graph II

Figure 8 is a ten node graph that was used to test the effect of varying β on the clique detection. From the discussion on α and β in Section 3.2 and since $C_{max} = 4$, the proposed

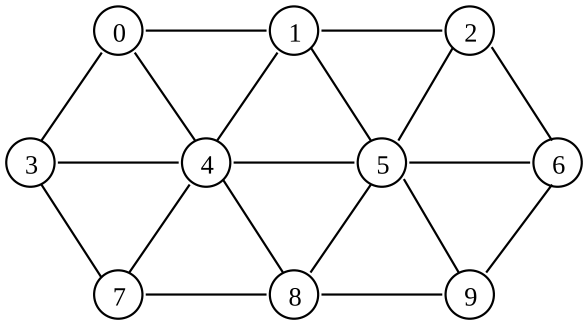

Figure 5: Example graph net10b.dat.

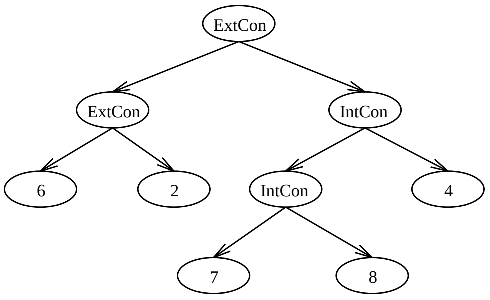

Figure 6: S–expression for generation 0's best chromosome.

optimal value for β is 5. In this section, we examine values of $\beta = 10$ and $\beta = 5$ on the graph in Figure 8.

The best and average fitness per generation are plotted in Figure 9 for $\beta = 10$. The system is stuck in a local optimum until about generation 500, at which point the average fitness makes a dramatic jump, and then steadily improves. The largest maximal clique is found in generation 0. The resultant candidate maximal cliques for generations 0, 100, 200, 300, 400, 500, and 600 are shown in Table 3.

The best and average fitness per generation is plotted in Figure 10 for $\beta = 5$. The system does not get stuck in a local optimum, but has a steady increase in fitness. The shape of

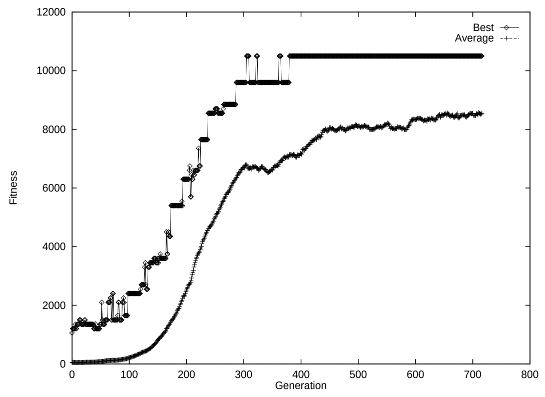

Figure 7: Best and Average fitnesses for graph net10b.dat.

Generation	Fitness	# of Snodes	List of Candidate Maximal Cliques
0	1054	9	{{6},{2},{4,7,8}}
100	2400	481	{{0,3,4},{3,7},{5},{0,1,4}, {6},{8},{1,2}}
200	6300	157	{4,5,8},{0,3,4},{1,2,5}, {9},{3,4,7},{4,7,8}, {0,1,4}}
300	9600	277	{4,5,8},{0,1,4},{1,2,5}, {2,5,6},{1,4,5},{0,3,4}, {4,7,8},{6,9},{3,4,7}, {5,8,9}}
400	10500	439	{5,6,9},{3,4,7},{1,2,5}, {4,7,8},{4,5,8},{0,3,4}, {1,4,5},{0,1,4},{2,5,6}, {5,8,9}}

Table 2: Cliques detected for 10 Node graph.

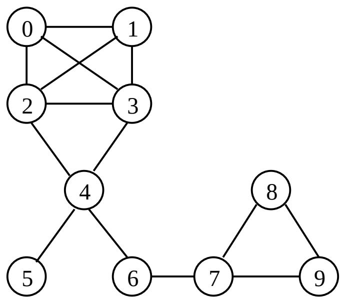

Figure 8: Example graph net10a.dat.

the curve is more like Figure 7 than Figure 9. The step–like increases between generations 100 and 300 indicate the incremental learning of different cliques. The resultant candidate maximal cliques for generations 0, 100, 200, 300, and 400 are shown in Table 4. With $\beta = 5$, it would have been nice if the candidate maximal cliques $C_{7,8,9} = \{\{7,8\},\{7,9\},\{8,9\}\}$, had coalesced into the clique $C = \{7,8,9\}$.

The difference between the two clique sets with $\beta = 10$ and $\beta = 5$ is that the larger value of β rewards for finding larger cliques. With too high a value of β, the clique detection system is actually hampered in the attempt to find all of the cliques in a graph.

6 Conclusions

In general, it is observed that the clique detector is better at discovering a large clique rather than all of the cliques in a graph. As was shown in Table 2, the largest clique is quickly found, while the list of all possible cliques takes much longer. As was shown in Table 3, the largest clique is found in the first generation. Since, in practice, the largest clique, C_{max}, in a graph is not known beforehand, a large value for β must be supplied. However, a value of

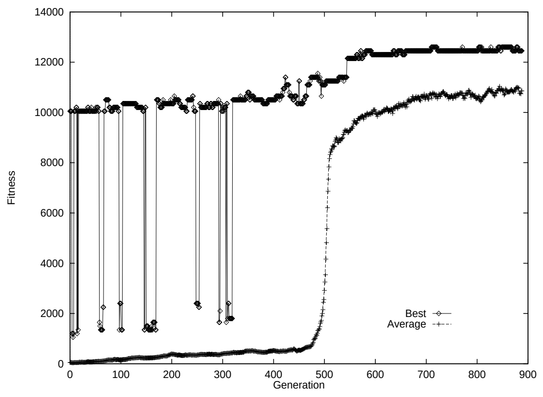

Figure 9: Best and Average fitnesses for graph net10a.dat, with $\beta = 10$.

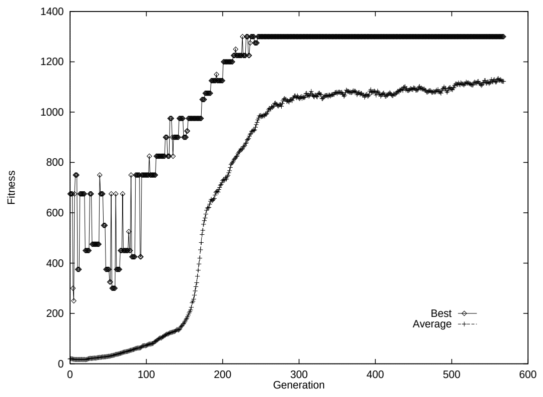

Figure 10: Best and Average fitnesses for graph net10a.dat, with $\beta = 5$.

$\beta > C_{max}$ will retard and even halt the learning of cliques smaller than C_{max}. This suggests the need for a two–pass clique detector:

1. Attempt to find the cardinality of the largest clique.
2. Attempt to find all of the cliques in the graph.

References

[1] Mihir Bellare and Madhu Sudan. Improved non–approximability results. In *Proceedings of the 26th Annual ACM Symposium on the Theory of Computing*, pages 184–193. ACM, 1994.

[2] Thang Nguyen Bui and Paul H. Eppley. A hybrid genetic algorithm for the maximum clique problem. In Larry Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 478–484, San Francisco, CA, 1995. Morgan Kaufmann Publishers, Inc.

Generation	Fitness	# of Snodes	List of Candidate Maximal Cliques
0	10050	15	{{ 0, 1, 2, 3 }, { 5 }, { 6 }}
100	2400	353	{{ 1, 2, 3 },{ 0, 2, 3 },{ 4, 6 } { 4, 5 }, { 9 }, { 7 } }
200	10350	125	{{ 3, 4 }, { 8, 9 }, { 0, 1, 2, 3 }, { 5 } }
300	10050	47	{ { 0, 1, 2, 3 } }
400	10500	137	{ { 7, 8 }, { 0, 1, 2, 3 }, { 6 }, { 9 }, { 4, 5 }, { 3, 4 } }
500	11100	709	{{ 6 }, { 0, 1, 2, 3 } { 5 }, { 4 }, { 7, 8, 9 } }
600	12300	359	{{ 7, 8, 9 }, { 2, 3, 4 }, { 0, 1, 2, 3 } { 5 }, { 4, 6 } }

Table 3: Cliques detected for 10 Node graph with $\beta = 10$.

Generation	Fitness	# of Snodes	List of Candidate Maximal Cliques
0	675	15	{{ 0, 1, 2, 3 }, { 5 }, { 6 } }
100	750	137	{{ 9 }, { 0, 1, 2, 3 } { 4, 5 }, { 7 } }
200	1125	233	{{ 4, 5 }, { 0, 1, 2, 3 } { 8, 9 }, { 6, 7 }, { 3, 4 } { 2, 4 }, { 7, 8 } }
300	1300	267	{{ 2, 3, 4 }, { 4, 5 } { 8, 9 }, { 7, 9 }, { 4, 6 } { 6, 7 }, { 0, 1, 2, 3 }, { 7, 8 } }
400	1300	267	{{ 4, 6 }, { 7, 9 }, { 6, 7 } { 2, 3, 4 }, { 7, 8 } { 0, 1, 2, 3 }, { 8, 9 }, { 4, 5 } }

Table 4: Cliques detected for 10 Node graph with $\beta = 5$.

[3] R. Chandraasekharam, S. Subhramanian, and S. Chaudhury. Genetic algorithm for node partioning problem and applications in VLSI design. *IEE Proceedings, Part E: Computers and Digital Techniques*, 140(5):255–260, Sep 1993.

[4] Lawrence Davis, editor. *Handbook of genetic algorithms.* Van Nostrand Reinhold, New York, NY, 1991.

[5] Anthony Hunter Dixon. *On the Efficiency of Clique Detection in Graphs.* PhD thesis, The University of British Columbia (Canada), 1973.

[6] Thomas Haynes. Clique detection via genetic programming. Technical Report UTULSA-MCS-95-02, The University of Tulsa, April 24, 1995.

[7] Thomas Haynes, Roger Wainwright, Sandip Sen, and Dale Schoenefeld. Strongly typed genetic programming in evolving cooperation strategies. In Larry Eshelman, editor,

Proceedings of the Sixth International Conference on Genetic Algorithms, pages 271–278, San Francisco, CA, 1995. Morgan Kaufmann Publishers, Inc.

[8] John H. Holland. *Adpatation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, MI, 1975.

[9] Kenneth E. Kinnear, Jr., editor. *Advances in Genetic Programming.* MIT Press, Cambridge, MA, 1994.

[10] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, 1992.

[11] John R. Koza. *Genetic Programming II, Automatic Discovery of Reusable Programs.* MIT Press, 1994.

[12] Thomas Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout.* Wiley, 1990.

[13] Ammanamanchi Srinivasa Murthy, Guturu Parthasarthy, and V. U. K. Sastry. Clique finding – a genetic approach. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 18–21, Piscataway, NJ, 1995. IEEE.

[14] DIMACS Repository. dimacs.rutgers.edu. ftp://dimacs.rutgers.edu/pub/challenge/graph/doc/ccformat.dvi.

[15] David Lindsay Springer. *Coloring and Clique Partitioning for Data Path Allocation.* PhD thesis, Carnegie–Mellon University, Sep 1991.

[16] N. Sriram. *Clique Optimization: An Approach to Parsimonious, Near–Optimal Ultrmetric Hierarchies.* PhD thesis, University of Oregon, Sep 1990.