

Postgraduate Research Conference Project Outline

Stephen Gilmour

October 7, 2003

1 Introduction

Ant colony optimization (ACO) is a multi-agent heuristic for solving combinatorial optimization problems. Recently, research has been done into providing a template to ACO algorithms which reduces the problem by giving the algorithm some starting information. This information, with high probability, still allows an optimal solution to be found, but significantly reduces the search space.

Parameterized complexity is about finding tractable algorithms that are guaranteed to find optimal solutions for computationally expensive problems such as combinatorial optimization problems. A fundamental idea within parameterized complexity is reducing problems to their problem kernel. A problem kernel still represents enough information from the original problem such that an optimal solution can be found, but reduces the problem so that processing it is easier.

ACO with templates is a new and under-developed area of research. Parameterized complexity is far more developed and kernelization is similar to templates in that they both try to reduce the problem. The aim of this research is to develop a set of ACO with templates algorithms for combinatorial problems that use kernelization as the template.

The outline of this document is as follows. Section 2 will define combinatorial optimization, the minimal vertex cover problem, and the travelling salesman problem. Section 3 will give a discussion of parameterized complexity. Section 4 will give a discussion of ant colony optimization. Section 5 is a summary of the research problem being investigated. Section 6 contains what work has been completed thus far. Section 7 contains the research plan for the rest of this project and section 8 contains the expected outcomes for this project.

2 Combinatorial Optimization

Combinatorial optimization problems are problems where the optimal combination of some aspect of the problem is to be found, where the definition of

optimal is part of the problem specification. Examples are the minimal vertex cover problem and the travelling salesman problem. In the minimal vertex cover problem, we want to select the minimum subset of nodes from a graph such that every edge in the graph has a selected node at one of its ends. In other words, we want to find a minimum combination of nodes such that each edge is connected to at least one of these nodes.

In the travelling salesman problem, given a graph containing a number of nodes (cities) with weighted edges (costs associated with moving from one city to another), we want to find the cheapest roundtrip route that visits each node precisely once and then returns to the starting node. In other words, we want to find a combination of edges such that the total weight of all selected edges is minimized and the edges form a circuit around the graph so that every node is visited precisely once.

Many combinatorial optimization problems such as the minimal vertex cover problem and the travelling salesman problem are NP-complete. A problem is NP-complete if it is in NP and every other problem in NP can be transformed into it in polynomial time. Therefore, these problems are thought to be some of the hardest problems in NP because if they could be solved in polynomial time, then all other NP problems could also be solved in polynomial time.

It is because of the difficult nature of combinatorial optimization problems that it is this class of problem that we hope to work on. So far, the only combinatorial optimization problems that have been mentioned are the minimal vertex cover problem and the travelling salesman problem. We will use these two problems to demonstrate ideas throughout the rest of this proposal, but there are many others such as the KNAPSACK problem and the quadratic assignment problem. Crescenzi[3] contains a compendium of NP optimization problems, many of which are combinatorial optimization problems that this research could be applied to.

3 Parameterized Complexity

Traditionally, the minimal vertex cover problem takes as input a graph $G = (V, E)$ and the output is a subset of nodes from the graph G which form a minimal vertex cover. However, you can transform the minimal vertex cover problem into the decision problem “is there a vertex cover for G of size k ?”, where k is some positive integer. Within parameterized complexity, k is called the parameter for the problem and is used to bound the problem. For this example, the graph G and the parameter k form a parameterized language.

For the vertex cover decision problem, the best known algorithm runs in time $O(1.271^k + kn)$ [6]. This solution is said to be fixed parameter tractable (FPT) because as long as the parameter is fixed and not too large, the algorithm is tractable. FPT algorithms are desirable because the size of the problem can be very large as long as the parameter is fixed and is not too large. So for the vertex cover decision problem, there can be a large number of nodes n as long

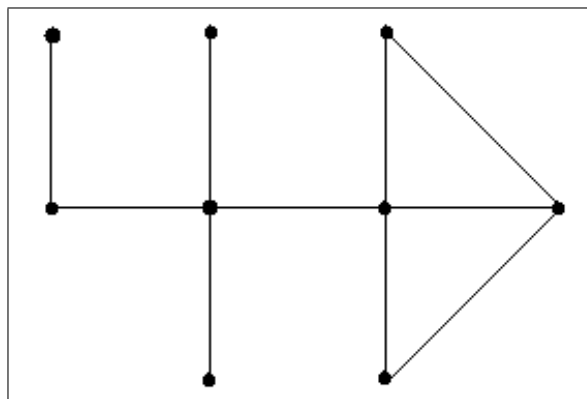


Figure 1: Original Graph

as the parameter k is small.

A practical way to apply fixed parameter tractability to computational problems is smart pre-processing. Kernelization is a process where a problem (G, k) is shrunk into its problem kernel (G', k') . The process of shrinking the problem can be performed in polynomial time, and the question of (G', k') can be answered in time bounded by a function of k simply by performing an exhaustive analysis of G' . Furthermore, after reducing a problem to its kernel, if the number of vertices in G' is more than k^2 , we can conclude that there is no vertex cover of size k . Here is a demonstration of how kernelization works.

Suppose we want to find a vertex cover of size $k = 4$ for the graph in figure 1. Downey[5] discusses five rules that can be used to kernelize a graph for the vertex cover decision problem. None of these five rules are intuitively kernelization rules, but all have been shown mathematically to reduce the problem whilst preserving enough information to allow an optimal solution to be found. However, for this particular graph, only two rules out of the five are required to find the problem kernel.

- Rule 2: If G has adjacent vertices u and v such that $N(v) \subseteq N[u]$, then replace (G, k) with $(G - u, k - 1)$.
- Rule 3: If G has a pendant edge uv with u having degree 1, then replace (G, k) with $(G - \{u, v\}, k - 1)$.

Figure 2 is the graph after applying rule 2. Four edges have been removed from the problem and one node (the circled node) has been included in the vertex cover. Now $k = 3$.

Figure 3 is the graph after applying rule 3. Another two edges have been removed from the problem and another node has been included in the vertex cover. Now $k = 2$.

Figure 4 is the graph after applying rule 3 again. Another two edges have been removed from the problem and another node has been included in the vertex

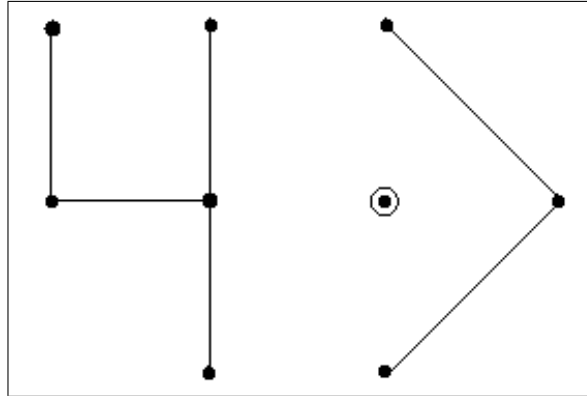


Figure 2: Graph after applying rule 2

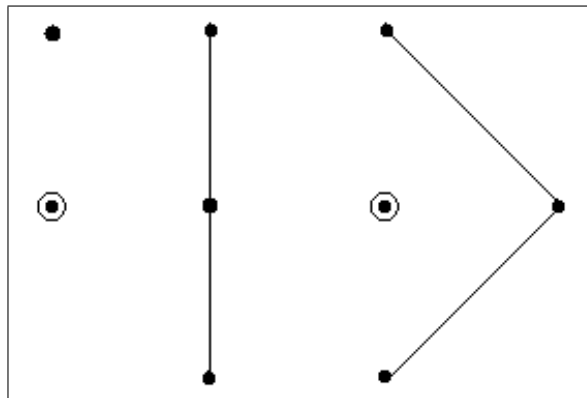


Figure 3: Graph after applying rule 3

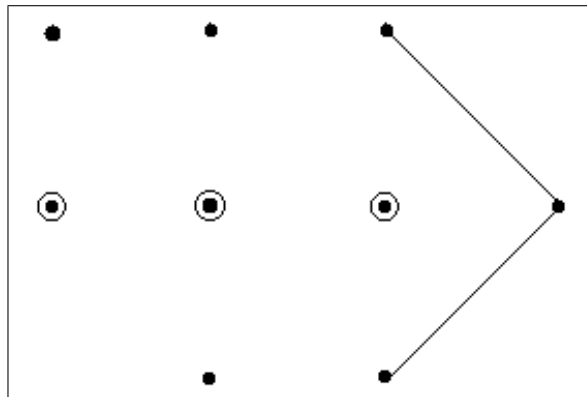


Figure 4: Graph after applying rule 3 a second time

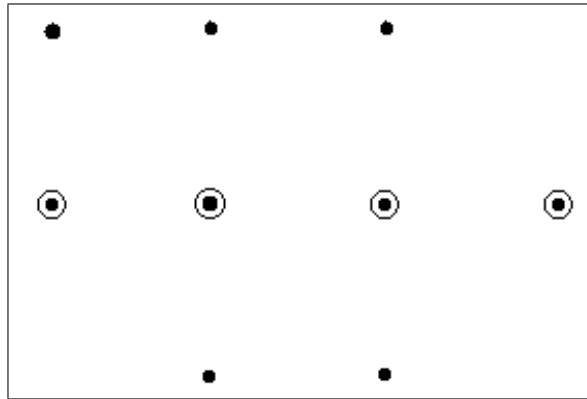


Figure 5: Kernel Graph

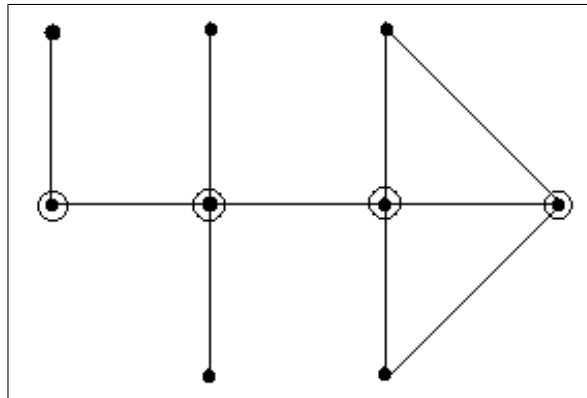


Figure 6: Vertex cover of size four

cover. Now $k = 1$.

Figure 5 is the graph after applying rule 3 again. Another two edges have been removed from the problem and another node has been included in the vertex cover. Now $k = 0$. In fact, all edges have now been removed. In this instance, kernelization has not only reduced the problem, but solved it. The problem kernel for this problem contains an empty set of edges. Figure 6 shows the vertex cover of size four which has been found for this graph.

As has been shown, kernelization is a process for reducing a problem to its problem kernel. The problem kernel contains enough information to find an optimal solution and can be found in a polynomial amount of time, but is significantly less computationally expensive. Once the problem kernel is produced, an optimal solution can be found in a tractable amount of time using any heuristic such as ant colony optimization, simulated annealing, neural nets, or genetic algorithms.

4 Ant Colony Optimization

Ant algorithms were first proposed as a multi-agent approach to difficult combinatorial optimization problems like the traveling salesman problem (TSP) and the quadratic assignment problem (QAP). Ant algorithms were inspired by the observation of real ant colonies where ants appear to find the most optimal path between food sources and their nest.

There are two core ideas that ant algorithms have taken from real ant colonies. The first idea is stigmergy. Stigmergetic communication is any indirect communication via the modification of the environment such that only local agents are able to access the information. For example with TSP, there are typically lots of ants that each find solutions to the problem using a simple heuristic and then drop pheromone to indicate the goodness of the solution they found along the path they took. Then, when other ants attempt to find better solutions, they are influenced in their decision making by the levels of pheromone within the environment.

The second idea is the coupling of autocatalytic (positive feedback) mechanism with the implicit evaluation of solutions. As already discussed, pheromone levels within the environment influence the decision making of the ants when finding solutions for TSP. In fact, ants are probabilistically more likely to follow a path with more pheromone rather than one with less. Since the goodness of solutions is represented within pheromone levels and ants are more likely to follow better solutions, a positive feedback mechanism is produced as trails with the most pheromone are likely to receive even more pheromone. The implicit evaluation of solutions is very much problem dependent, but an example from TSP is that shorter paths are found quicker because the path is shorter, and therefore pheromone is laid out quicker which is exploited because of the positive feedback system.

The major problem with the use of stigmergy and autocatalysis is premature convergence among the ants (stagnation). This is when some not very good paths are being followed by the entire population simply because of a local optimum or the initial random fluctuations caused a particular solution to be far better than all current solutions but still sub-optimal. Therefore, pheromone trail evaporation and stochastic state transitions are needed to complement this problem with stigmergy and autocatalysis.

Ant algorithms have been found to be very promising. For some combinatorial optimization problems they have reached world class performance [4]. A new area of research is giving ant algorithms a template. A template is a pattern used to organise activities. It is said that the ants “self-organise along a template” [1]. In other words, ants still develop a solution using stigmergy and autocatalysis, but the template guides them so that the final solution is more predictable. Templates can also be thought of as a mechanism for reducing the problem, such that the solution is easier to find. For example, graph partitioning.

There are several problems within graph partitioning, and one of them involves

dividing a graph into c clusters of approximately equal size while minimizing the number of connections between them. For this problem, the template is a grid that covers the environment (graph). The grid contains for each possible location the probability for items to be placed in that particular location in the graph. One possible template for this problem could have high probability in the corners of the graph and low probability in the centre. This means that the ants are probabilistically more likely to drop items in the four corners of the graph. This template works very well when the graph does naturally divide into four clusters, and works reasonably well if the number of clusters is close to four (if it is three, it should just form three clusters in three corners of the graph).

Templates provide a mechanism for reducing problems to make them easier to compute (it is easier to divide a graph into approximately four clusters than it is to divide it into an unknown number of clusters) and provides predictability in solutions. However, research into templates is still greatly under-developed. Thus far, templates have only been applied to graph partitioning and data analysis problems, and no attempt has been made to apply them to more prominent NP-complete problems such as the minimal vertex cover problem, travelling salesman problem, or the quadratic assignment problem.

5 Research Problem Summary

Kernelization is a pre-processing step that can be used on combinatorial optimization problems to reduce the problem to its problem kernel. The problem kernel is computationally less complex but is still guaranteed to allow an optimal solution to be found. Templates are a mechanism for reducing combinatorial problems so that ant colony optimization algorithms are able to find solutions more efficiently. Templates with high probability allow an optimal solution to still be found.

Templates and kernelization are connected in that they both aim to reduce a problem to make it computationally less complex. In fact, templates could be thought of as a kind of quasi kernelization. This research will investigate the relationship between templates and kernelization in order to improve the usefulness of templates for ant colony optimization algorithms. Therefore, the aim of this research is to develop a set of ACO with templates algorithms for combinatorial problems that are a hybrid of traditional templates and kernelization.

6 Progress Achieved

- Partial literature review covering parameterized complexity research.
- Partial literature review covering ant colony optimization for the travelling salesman problem.
- Project proposal complete.

- Development of a framework in which TSP ant algorithms can be implemented. This framework was written in Microsoft's CSharp and requires being rewritten so that it is more flexible. Currently it would be very difficult to implement non-TSP problems or kernelization within it.

7 Research Plan

7.1 Oct 2003 - Mar 2004

Develop software framework in which experimentation will take place.

- 8 weeks: Implement a software framework in which different combinatorial problems and multi-agent based algorithms (modules) can be plugged in and experimented with.
- 6 weeks: Implement a minimal vertex cover problem module which plugs into the software framework.
- 6 weeks: Implement a module for the software framework which solves the minimal vertex cover problem using ant colony optimization.
- 4 weeks: Continue to write literature review on ant colony optimization and parameterized complexity.

7.2 Apr 2003 - Aug 2004

Investigate methods of autonomously performing kernelization on graph problems using a distributed, local search approach.

- 8 weeks: There are already published rules for kernelizing the vertex cover problem. However, these rules are not designed to be implemented by a multi-agent system. Investigate how to implement these rules using a distributed, local search approach.
- 5 weeks: Implement a module for the software framework which performs kernelization of a vertex cover using a distributed, local search approach.
- 2 weeks: Compare efficiency of distributed, local search kernelization with polynomial results found for current non-distributed, top-down approach.
- 2 weeks: Compare efficiency of kernelization as a pre-processor followed by ant colony optimization to just ant colony optimization.
- 3 weeks: Continue to write literature review on ant colony optimization and parameterized complexity

7.3 Sep 2004 - Jan 2005

Investigate whether kernelization can be improved using probabilistic kernelization rules, which are similar to normal kernelization rules except that each reduction rule with high probability will still allow an optimal solution to be found.

- 8 weeks: Attempt to develop a set of probabilistic kernelization rules for the minimal vertex cover problem, which, with high probability, will still allow a minimal vertex cover to be found.
- 6 weeks: Implement a module for the software framework which performs probabilistic kernelization.
- 3 weeks: Compare size of problem kernel after probabilistic kernelization to size of problem kernel after regular kernelization.
- 3 weeks: Write up results from Apr 2003 - Aug 2004.

7.4 Feb 2005 - Jul 2005

Investigate whether agents can perform kernelization and ant colony optimization on a problem in parallel.

- 8 weeks: Investigate how agents can perform kernelization and ant colony optimization on a problem in parallel.
- 5 weeks: Implement a module for performing ant colony optimization and a module for performing kernelization such that the agents in both modules work together in parallel.
- 3 weeks: Compare kernelization and ant colony optimization in parallel to kernelization as a pre-processor followed by ant colony optimization.
- 3 weeks: Compare kernelization and ant colony optimization in parallel to just ant colony optimization.
- 3 weeks: Write up results from Sep 2004 - Jan 2005.
- 3 weeks: Write up results from Feb 2005 - Jul 2005.

7.5 Aug 2005 - Jan 2006

Complete thesis.

8 Expected Outcomes

- Contribution to the literature on how templates can be used to improve ant colony optimization algorithms.
- A multi-agent algorithm for performing kernelization on combinatorial problems.
- A multi-agent algorithm for performing probabilistic kernelization on combinatorial problems.
- A multi-agent algorithm for performing kernelization and ant colony optimization in parallel, suitable for all combinatorial problems.

- Paper describing how to perform distributed, local search kernelization and comparing kernelization as a pre-processor followed by ant colony optimization to just ant colony optimization.
- Paper proposing probabilistic kernelization and comparing it to regular kernelization.
- Paper describing how kernelization can be used as a template for ant colony optimization and comparing the algorithms with regular ant colony optimization algorithms and other heuristic algorithms for select problems.

References

- [1] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence From Natural to Artificial Systems*. A volume in the Santa Fe Institute studies in the science of complexity. Oxford University Press, 1999.
- [2] Alberto Colomi, Marco Dorigo, and Vittorio Maniezzo. Distributed optimization by ant colonies. *Proceedings of the First European Conference on Artificial Life, Paris, France*, pages 134–142, 1992.
- [3] Pierluigi Crescenzi and Viggo Kann. A compendium of np optimization problems. Technical report si/rr-95/02, Universita di Roma 'La Sapienza', 1995. <http://www.nada.kth.se/viggo/wwwcompendium/> [Accessed September 2003].
- [4] Marco Dorigo, Gianni Di Caro, and Luca Maria Gambardella. Ant algorithms for discrete optimization. *Proceedings of Artificial Life 5*, pages 137–172, 1999.
- [5] Rodney Downey, Michael Fellows, and Ulrike Stege. Parameterized complexity: A framework for systematically confronting computational intractability. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 1997.
- [6] Michael Fellows. Parameterized complexity: The main ideas and connections to practical computing. *Electronic Notes in Theoretical Computer Science*, 61, 2002.
- [7] James Montgomery. Towards a systematic problem classification scheme for ant colony optimisation. Technical report tr02-15, School of Information Technology, Bond University, Australia, 2002.