

The Current State of Ant Colony Optimisation Applied to Dynamic Problems

Technical Report: TR009

Daniel Angus

dangus@ict.swin.edu.au

Centre for Intelligent Systems & Complex Processes
Faculty of Information & Communication Technologies
Swinburne University of Technology
Melbourne, Australia

April 21, 2006

1 Introduction

In recent years Ant Colony Optimisation (ACO) algorithms have been applied to more challenging and complex problem domains. One such domain, which is suggested in [1], is dynamic problems. The majority of dynamic problems addressed using biologically inspired computation techniques are from the general field of *Operations Research* and are usually modifications of popular static problem domains such as the travelling salesman problem and the vehicle routing problem. This document aims to summarise the current state of the field with regard to ACO algorithms and their application to dynamic problems. The document will attempt to answer questions such as:

- What general algorithmic characteristics do these novel approaches deem most important when addressing dynamic problems?
- What utility do these dynamic algorithms offer over standard approaches?

To address these questions several recent research papers have been reviewed and relevant findings presented (see Tab. 1. The author assumes that the reader is familiar with the concepts of Ant Colony Optimisation and combinatorial optimisation (specifically Travelling Salesman Problems and Quadratic Assignment Problems), if this is not the case, recommended references are [3, 2, 8].

Algorithm name	Optimisation problem	Dynamic variation
FIFO-Queue ACO	QAP, TSP	Regular removal and introduction of a varied number of locations/cities at periodic intervals
AS-DTSP	TSP	The weights of some edges in the TSP which are located on the current best path found are increased and after a time eventually decreased back to their original value

Table 1: ACO algorithms applied to dynamic problems

2 FIFO-Queue ACO

Guntsch and Middendorf introduced a population based approach for ACO in 2002 [7]. The specific algorithm tested at this time was named FIFO-Queue ACO due to the treatment of its population of ‘ants’. FIFO-Queue ACO sticks close to the original ACO inspiration by maintaining a pheromone mapping for use in the solution construction process.

In this paper the authors claim that the speed with which an algorithm can modify its population when addressing dynamic problems is an important performance characteristic.

“A standard elitist strategy for ACO algorithms is that an elitist ant updates the pheromone values in every generation according to the best solution found so far. But after a change of the problem instance the best solution found so far will usually no longer represent a valid solution to the new instance. Instead of simply forgetting the old best solution, it was adapted so that

it becomes a reasonably good solution for the new instance. Clearly, this is only reasonable when the changes of the instance are not too strong.”

The comment made by the authors here is referring to the algorithms solution robustness. That after a change in the search space a previously *optimal* solution will quickly become *sub-optimal* and that a traditional ACO technique is forced to adapt this now *sub-optimal* solution to continue to maintain the *optimal* solution.

“Instead of simply forgetting the old best solution, it was adapted so that it becomes a reasonably good solution for the new instance. Clearly, this is only reasonable when the changes of the instance are not too strong.”

By this next statement it is inferred that a traditional technique will only be able to perform on dynamic domains if the changes are not too fast. In that while a traditional technique may be able to ‘adapt’ to a new optimal location they can only do this if that optimal location exists for an extended time-period.

To address this point the FIFO-Queue ACO algorithm removes the traditional decay and update operations and replaces them with what the authors deem as being a ‘population’ based approach to pheromone maintenance. The best solution for each generation is pushed onto a FIFO stack and its corresponding solution components in the pheromone mapping increased proportional to the quality of the solution. This solution is moved down the stack in consequent generations until it is pulled from the stack; at which time the corresponding pheromone components it added to the pheromone mapping on insertion are removed. The outcome of this process is that the only two adjustments are made to the pheromone mapping at each generation, one for insertion, and one for removal. The best solutions for the last k generations are therefore maintained making this almost a pseudo-elitist strategy.

Another important feature in FIFO-Queue ACO is the use of a minimum pheromone value. The pheromone value is initialised uniformly and elite solutions added on top of the existing values. This way the base pheromone values are never decayed. The reasons provided for this is that: “Otherwise it could happen that pheromone values become zero because there is no solution in the population that puts pheromone on them”. To extrapolate this further although it is inferred by the authors that an important feature of a dynamic ACO algorithm is that some bounding on the minimum pheromone value occur to ensure that after a change event previously unused solution components are not impossible to include in a new candidate solution.

The algorithm has shown promise, performance-wise, when compared to MAX-MIN ACO and AS, even though the authors make it clear that they are not interested in performance on static problems, as long as “the population based approach performs at least not worse than the standard method on static problems”. Although the algorithm presented in this paper is not specifically tested on a dynamic problem, the authors claims about desirable algorithmic properties for good performance on dynamic problems are reasonable.

3 FIFO-Queue ACO applied to dynamic TSP & QAP

Leading on from their work in [7], Guntsch and Middendorf released a paper titled ‘Applying Population based ACO to dynamic optimization problems’ in the same year [6]. As the title

suggests this paper tested their FIFO-Queue ACO algorithm on instances of a dynamic TSP and a dynamic QAP.

Building upon the base FIFO-Queue ACO algorithm the authors try various population maintenance schemes including:

1. Age - No different to that used in [7], the oldest solution is removed from the population when a new candidate solution enters the population
2. Quality - If a new candidate solution is better than the worst solution in the population that population member is replaced, otherwise the candidate solution is discarded
3. Prob - A two stage process. Firstly the candidate solution is placed in the current population. Secondly a single population member is probabilistically chosen to be discarded proportional to the population members quality, that is, if a population member has a lower quality there is a higher chance of this member being discarded.
4. Age & Prob - The same as **Prob** however the new candidate solution is excluded from the removal procedure.
5. Elitism - Applied to any of the previous techniques, elitism maintains an extra population member - the best solution found so far, which is updated when a better solution is found, but never takes part in the replacement process.

To make the TSP problem dynamic half of the cities were removed from the standard TSP data set. These cities were added to a *replacement pool* from which, at regular intervals, a set number of cities was swapped between the replacement pool and the cities currently being optimised. Both the frequency (how often a swap was performed) as well as the magnitude (how many cities swapped) were varied to determine algorithmic performance on different dynamic scenarios. Likewise with the QAP, except instead of cities being swapped, locations were swapped.

As a baseline, the various FIFO-Queue ACO algorithm variants were tested against what was deemed the best performing (for static TSP & QAP) FIFO-Queue ACO algorithm, with the variants being run without restart after a dynamic change event (meaning the current population of solutions is maintained), and the baseline being restarted (meaning the population was cleared) after every dynamic change event. This baseline is a simplistic approach to dealing with dynamic change, however any algorithm should perform better than this baseline if it is to be considered for dynamic optimisation problems.

When a change occurs the current population of solutions will cease to be valid (they will contain locations/cities which no longer exist) to address this issue the authors propose a solution adjustment method. This solution adjustment method aims to provide a better starting point for the FIFO-Queue ACO algorithm than it would get from being re-initialised.

1. Remove all invalid cities/locations.
2. Append to the end of each solution any unused cities/locations in a greedy fashion.
3. Repair the pheromone matrix according to the changes made.

Initial results from testing of the algorithms (with varied population sizes, and without restarting the algorithm after a dynamic change event) are that all algorithm variants, except for **Quality**, exhibit good performance on the dynamic TSP when compared against the baseline algorithm. The

reasons for **Qualities** poor performance on the dynamic TSP is attributed to how this algorithm’s population quickly converge to a small area of the search space and after a dynamic change event struggle to explore beyond this local area. This is counter to observations on the dynamic QAP which showed good results for the **Quality** variant, as well as all other algorithm variants. There is no doubt the QAP problems tested favour algorithms with strong convergence properties.

The majority of algorithm variations performed better with a smaller population size, suggesting that in this case a large history does not provide any benefit when addressing dynamic problems, which agrees well with previous observations made in [7] regarding the speed with which an algorithm can adapt to a change may influence its quality.

4 Ant Systems for a dynamic TSP

Eyckelhof and Snoek modified the original ant algorithm - ant systems, to apply it to Dynamic TSP (AS-DTSP) [4]. The dynamic TSP was created by taking a standard TSP and increasing edge weightings on the current best path to introduce a ‘traffic jam’ in what is most likely the current most popular route. This analogy augers well with the two-armed bandit problem [5], in which a poker machine pays off at two different rates depending on which handle a punter pulls (usually \$60/\$40 per 100 punters). When eventually all of the punters realise which arm pays off more they focus their attention on this arm, however, by all of the punters using one arm it lowers the expected pay-off since all 100 punters have to share a finite winning pool of \$60. With the traffic analogy if all motorists are told that a particular road is flowing fast they will move onto this road, however, by all motorists focusing their attention onto this road they will increase the congestion thereby slowing the flow.

The authors decided to construct two test problems: a 25 city and 100 city TSP, both are non-standard TSP datasets with cities distributed in a 100x100 grid. On the 25-city TSP dynamic behaviour is introduced between iterations 100-150 where one traffic jam is steadily introduced, and iterations 200 and 250 where the first traffic jam disappears and another two traffic jams are introduced. Each traffic jam is introduced in increments of 10 units with one incremental step every 5 iterations. It is the desirable that the algorithms tested should be able to respond to the incremental change as it is occurring. For the 100-city TSP a new traffic jam is introduced and (except for when the first traffic jam is being introduced) the previous traffic jam is removed every 50 iterations in increments of 5 units every 5 iterations for 25 iterations. The 100-city TSP is run for 5000 iterations meaning a total of 100 traffic jams are introduced over the duration of the search.

One of the first observations made by the authors in testing the basic AS algorithm in a dynamic environment was that setting a lower pheromone bound was important. It was found that the pheromone values, when allowed to diminish away, would reach a state of entropy at which time they would never be included in the solution construction process. As such, previously poor paths could now represent good paths but would have no chance of being explored due to the restriction imposed by the level of pheromone.

Even with the introduction of a lower bound, it is still possible to obtain a large difference between pheromone values on currently good and poor edges due to the positive solution reinforcement procedure used by AS. Therefore the authors propose a method, *global shaking* (Equ. 1), to squash all pheromone values into a range, but preserve the relative rankings, i.e. if $\tau_{ij}(t) > \tau_{i'j'}(t)$ before

global shaking, then it will hold after *shaking*. This procedure encourages exploration on previously unused edges to assist the algorithm in the detection of new optima.

$$\tau_{ij} = \tau_0 \cdot (1 + \log(\tau_{ij}/\tau_0)) \quad (1)$$

The *global shaking* operation has a problem associated with its use on larger datasets which the authors address in a later section of the paper. It was believed that when a change occurs in an isolated section of the TSP dataset that it may not be necessary to ‘shake’ the entire pheromone mapping, in fact it may be undesirable. The authors introduce a *local shaking* operator which operates in the same way as *global shaking* however only in a defined radius around the area of change.

Observations from running experiments on the 25-city TSP with the original AS, original AS with reset (the pheromone mapping is reset when a change is detected) and AS with shaking (global, intermediate and local) were:

- All techniques seemed to be able to react to change and continue to find better results after traffic jams were introduced.
- AS with reset performed marginally better between the first traffic jam and second traffic jam.
- AS with global shaking out-performed the other techniques after the first traffic jam.

Observations from running experiments on the 100-city TSP with the same five algorithms were:

- AS with local shaking out-performed all other techniques in all measures, followed closely by AS with intermediate shaking.
- AS with global shaking and AS with reset performed poorly, as they were unable to converge in any area due to the frequency of dynamic change and constant disruption occurring to the pheromone mapping.

The general findings from this investigation were that the frequency of dynamic change is important when selecting an ACO technique to address a dynamic problem. Traditional ACO approaches may be able to accommodate dynamic problems if the frequency of change is not large, however when the frequency of change increases new ACO techniques are required to keep these techniques competitive. Also by tailoring the ACO algorithm to change only pheromone in the area of change ensures that the algorithm can respond efficiently and effectively to change. While this investigation has presented good research into the effects of frequency of dynamic change on ACO algorithms, there were no findings regarding the effects of the magnitude of dynamic change.

5 Conclusion

The authors of [7, 6, 4] consider the following points most important in the application of ACO to dynamic problems:

- Frequency of change - Traditional ACO algorithms can successfully adapt to low frequency change, however when the problem being optimised begins to change too frequently modification to the traditional approach is required.
- Modification to pheromone mapping - Modifications to the pheromone mapping (triggered after a dynamic change event) work best when the modification targets areas of disruption caused by dynamic change rather than attempting to modify the entire pheromone mapping.
- Tracking optimal solutions - Limiting the minimum pheromone value can assist the algorithm in being able to continue searching for an extended period of time (while changes are occurring) as current optimal solutions become sub-optimal and previous sub-optimal solutions become optimal. Limiting the amount of stored history to a very small number of previously found elite solutions ($\approx 1 - 6$) can assist in an algorithm's optimal solution tracking ability.
- Utilisation of prior information - It is more beneficial to retain previous algorithm history than to clear the algorithm history after a change, unless the change is of an extremely large magnitude, or if the optimisation process is to be run for a significantly long time after the change with no further problem changes taking place in this time.

General findings are that the modified ACO algorithms offer both efficacy and efficiency advantages over traditional ACO algorithms when applied to dynamic variations of TSP and QAP. Due to there currently being no standard for dynamic problems it is difficult to compare the ACO approaches discussed here to similar optimisation algorithms such as Particle Swarm, and Genetic Algorithms. For the moment the field seems content with comparing new ACO algorithms with traditional ACO algorithms and developing thoughts as to what properties an algorithm applied to dynamic problems might require. For future studies two useful metrics for comparison between techniques are the frequency and magnitude of change, as in the techniques reviewed these seem to be critical factors in determining an algorithm's success.

References

- [1] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press, New York, 1999.
- [2] R. E. Burkard, S. E. Karisch, and F. Rendl. QAPLIB a quadratic assignment problem library. *Journal of Global Optimization*, 10(4):391–403, 1997.
- [3] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, London, 2004.
- [4] C. J. Eyckelhof and M. Snoek. Ant systems for a dynamic TSP. In *ANTS '02: Proceedings of the Third International Workshop on Ant Algorithms*, pages 88–99, London, UK, 2002. Springer-Verlag.
- [5] D. E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley, 1989.

- [6] M. Guntsch and M. Middendorf. Applying population based ACO to dynamic optimization problems. In *ANTS '02: Proceedings of the Third International Workshop on Ant Algorithms*, pages 111–122, London, UK, 2002. Springer-Verlag.
- [7] M. Guntsch and M. Middendorf. A population based approach for ACO. In S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf, and G. R. Raidl, editors, *EvoWorkshops*, pages 72–81. Springer-Verlag, 2002.
- [8] G. Reinelt. Tsplib95, 1995. Available at: <http://www.iwr.uni-heidelberg.de/groups/comopt/software/tsplib95>.