Exploiting Antipheromone in Ant Colony Optimisation for Interactive Search-Based Software Design and Refactoring

Chris Simons
University of the West of England
Frenchay Campus
Bristol BS16 1QY United Kingdom
chris.simons@uwe.ac.uk

Jim Smith
University of the West of England
Frenchay Campus
Bristol BS16 1QY United Kingdom
james.smith@uwe.ac.uk

ABSTRACT

Preventing user-fatigue in interactive meta-heuristic search places as great an emphasis on efficiency as it does on effectiveness. Engagement may also be boosted if the system provides a sense of "responsiveness" - for example, avoiding unpopular solutions as well as exploiting preferred ones. In this paper we explore one possible way of achieving these goals using the concept of "anti-pheromones" in different forms of Ant Colony Optimisation. Taking search-based software design and refactoring as a case study, we use extensive offline experiments to investigate differences of timescale and method for applying anti-pheromones. Results confirm our predictions that most combinations are in fact counter-productive. However, applying high levels of anti-pheromone, only in the initial stages of a run, can rapidly steer the search away from unproductive regions, reducing the number of evaluations required by up to 20% without compromising solution fitness.

Keywords

Search-Based Software Engineering; Ant Colony Optimisation; Antipheromone

1. INTRODUCTION

Assisting the software engineer with the cognitively demanding task of design and refactoring has attracted much research attention within the field of Search-Based Software Engineering (SBSE) e.g. [3], [5]. In interactive metaheuristic search effective and efficient search performance is crucial to prevent user fatigue [8]. In SBSE, as elsewhere, the notion of fitness is fundamental to selection in metaheuristic search. However, it has been proposed that misfit of a candidate solution within its context is significant. For example, Alexander [1] suggests that designers are more adept at recognising solution/context ensemble misfit than good fit, and use their perceptions of misfit to derive improved candidate solutions. In agile software development methodologies, the notion of design as misfit rectification can be seen in design refactoring. Reflecting the observed practice of design misfit rectification, might the use of antipheromone be exploited in ACO search-based software design and refactoring to produce a more efficient search? To answer this question, we pose the following hypotheses:

1. Early stage application of antipheromone may help steer search away from poor solutions;
2. If search is in a good region then penalising "worst" solutions with antipheromone may be counter-productive.

GECCO '16 July 20-24, 2016, Denver, CO, USA

ACM ISBN 978-1-4503-4323-7/16/07.

DOI: http://dx.doi.org/10.1145/2908961.2909018

2. BACKGROUND

In 2002, Montgomery and Randall [4] noted that in nature, various ant species exploit many hormones in foraging behaviours. They suggested the use of antipheromone as a "repellent substance" to artificial ants, acting in combination with, but in contrast to, pheromone as an attractant. Building on Ant Colony System (ACS) [2], the authors suggested three extensions incorporating antipheromone by (i) subtracting pheromone for the worst solution path, (ii) making pheromone 'repellent' for poorer solutions, and (iii) making explorer ants that are arracted to areas of little pheromone.

3. PROPOSED APPROACH

Building on previous research [6], we extend Ant Colony Optimization by means of two mechanisms:

- *Phased Antipheromone*, where an initial phase of search (during which both antipheromone and pheromone are deposited) precedes a pheromone phase (in which only pheromone is deposited);
- *Subtractive Antipheromone,* in which a single ant, being the worst solution in the colony, removes or reduces pheromone from the worst solution path.

3.1 Solution Encoding

ACO generally uses a representation where candidate solutions (i.e. paths) are encoded by a permutation of a fixed set of values. However, each candidate solution is represented as a permutation of a set comprising design elements (i.e. attributes and methods) and "end-of-class" markers. Further details of the solution encoding are described at [6].

3.2 Cost Measure

The cost measure employed in the proposed approach draws on two metrics of design quality, which are to be minimised and lie in the interval (0,1]. The first relates to dependency coupling between design classes (f_{CBO}), while the

second reflects design symmetry (f_{NAC}). The cost measure combines the two elements with equal weight:

$$Cost(x) = 0.5 * (f_{CBO}(x) + f_{NAC}(x)) \quad (1)$$

Further details of the dependency coupling measure, f_{CBO} and the design symmetry cost measure, f_{NAC} are available at [6].

3.3 Search Algorithm

The ACO algorithm variants used in this paper are inspired by Simple-ACO [2] and MAX-MIN Ant System [7]. In the Simple-ACO algorithm all ants of the colony update the pheromone table in turn according to the cost value of their path. An initial phase of antipheromone deposit is implemented as a percentage of the total number of evaluations. Where the algorithm is in its antipheromone phase, we introduce an additional mechanism for the single worst ant in the colony, wherein pheromone values are reduced to 50%, 10% and 0% of their original values. Similarly, if the MMAS variant algorithm is in its antipheromone phase, we introduce two mechanisms. Either pheromone is reduced to the MAX-MIN minimum threshold parameter M_{min}, or a 50% reduction of the original pheromone value is applied, subject to the MAX-MIN minimum threshold.

4. METHODOLOGY

We conducted experiments with the two antipheromone ACO algorithm variants (Simple-ACO and MMAS) for each of four problem instances i.e. Cinema Booking System, Graduate Development Program, Select Cruises and a randomised instance [6]. The resulting data for *Cost* and number of Evaluations were analysed using a two-way Analysis of Variance.

5. RESULTS

For Simple-ACO variant algorithm, initial antipheromone phases were trialled by reducing pheromone values by 50%, 10% and 0%. Findings indicated that for all design problem instances, compared with no antipheromone deposit, values of *Cost* and Evaluations are inferior. Analysis of Variance indicated that the differences are significant.

For the MMAS algorithm variant, initial antipheromone phases were trailed with pheromone reductions to a 50% or original, and M_{min}. For 50% reduction, values of *Cost* and Evaluations are inferior, and the differences are statistically significant. However, for reductions to M_{min}, *Cost* values obtained are not significantly different from those obtained without antipheromone. Furthermore, the number of Evaluations is lower for antipheromone phases of up to 10%, and the reductions of up to 20% achieved are statistically significant. Reductions of 20%, 13% and 11% are achieved for the CBS, GDP and Randomised problem instances respectively. Results for the CBS problem instance are shown in figure 1; the picture is very similar for GDP and Random.

6. CONCLUSIONS

For search-based software design and refactoring, we investigated a number an antipheromone approaches to Ant Colony Optimisation, wherein pheromomone values are reduced for the single worst ant in the colony. Using an MMAS-based antipheromone variant, the number of evaluations required to achieve best cost measures is reduced up to

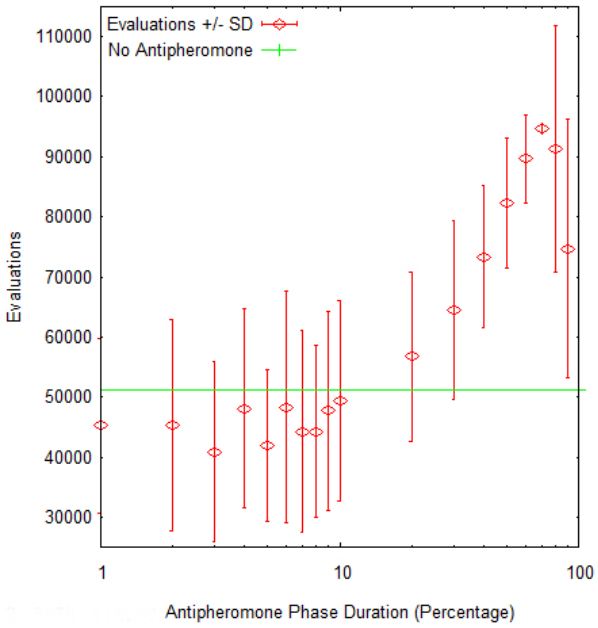

Figure 1: Mean Number of Evaluations required for best *Cost* in various antipheromone phase durations for CBS Problem Instance

20%, without compromising solution quality. We conclude early phase application (i.e. up to 10% of evaluations) helps steer search away from poor solutions, whereas extended application (i.e. above 20%) can be counter-productive, especially if search has arrived at promising regions.

7. REFERENCES

[1] C. Alexander. *Notes on the Synthesis of Form*, volume 5. Harvard University Press, 1964.

[2] M. Dorigo and T. Stutzle. *Ant Colony Optimization*. MIT Press, 2004.

[3] I. Moghadam and M. Ó Cinnéide. Code-Imp: A Tool for Automated Search-Based Refactoring. In *Proceedings of the 4th Workshop on Refactoring Tools (WRT '11)*. ACM Press, 2011.

[4] J. Montgomery and M. Randall. Anti-pheromone as a tool for better exploration of search space. In *Ant Algorithms*, Lecture Notes in Computer Science (LNCS) 2463, pages 100–110. Springer, 2002.

[5] C. Simons and I. Parmee. Elegant Object-Oriented Software Design via Interactive, Evolutionary Computation. *IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 42(6):1797–1805, 2012.

[6] C. L. Simons, J. Smith, and P. White. Interactive ant colony optimization (iaco) for early lifecycle software design. *Swarm Intelligence*, 8(2):139–157, 2014.

[7] T. Stützle and H. H. Hoos. Max–min ant system. *Future generation computer systems*, 16(8):889–914, 2000.

[8] H. Takagi. Interactive evolutionary computation: Fusion of the capabilities of ec optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296, 2001.