

Niching for Population-based Ant Colony Optimization

Daniel Angus

Centre for Information Technology Research
Faculty of Information & Communication Technologies
Swinburne University of Technology
Melbourne, Australia, 3122
dangus@ict.swin.edu.au

Abstract

Most Ant Colony Optimization (ACO) algorithms are able to find a single (or few) optimal, or near-optimal, solutions to difficult (NP-hard) problems. An issue though is that a small change to the problem can have a large impact on a specific solution by decreasing its quality, or worse still, by rendering it infeasible. Niching methods, such as fitness sharing and crowding, have been implemented with success in the field of Evolutionary Computation (EC) and are aimed at simultaneously locating and maintaining multiple optima to increase search robustness - typically in multi-modal function optimization. In this paper it is shown that a niching technique applied to an ACO algorithm permits the simultaneous location and maintenance of multiple areas of interest in the search space.

1. Introduction

In natural ecologies, a population of organisms is rarely spread uniformly (within an environment), but rather is typically distributed across a wide spatial area and divided into local or sub-groups. Resources available to individuals across a geographical distribution can differ, and so sub-groups of a species may specialise to exploit these differences. The effect of this speciation (or population level

natural selection) is referred to as ‘niching’ in the field of population ecology and population genetics [6, 17].

In a computation sense, niching may permit a more effective use of available resources by a search algorithm by either implicitly or explicitly dividing and searching different areas of the search space in parallel [13]. Such automatic resource re-allocation is useful in preserving useful population diversity for an extended search time. Niching techniques have also proved particularly useful when problem domain complexity is scaled up to include multiple global and local solutions; problems which are commonly referred to as ‘multi-modal’.

This investigation outlines two variants of the ACO algorithm [2, 4] which include forms of niching used in the field of Evolutionary Computation: *fitness sharing* and *crowding*. Fitness sharing involves the modification of the cost function to create and maintain niches, whereas crowding uses a population-based replacement strategy the effect of which is to modify the explicit historical information maintained by the algorithm. The algorithms presented here are extensions of the population-based ACO algorithm proposed by Guntsch and Middendorf [9, 10]. Their ant-inspired algorithm maintains a history in the form of a population of solutions rather than a pheromone map like other traditional ACO algorithms (although it still utilises a pheromone map generated from the population

for solution construction).

The algorithms presented here are evaluated by their ability to locate and maintain multiple, spatially distributed, near-optimal solutions rather than just the single best solution found overall. The Travelling Salesman Problem (TSP) [15] has received much attention by ACO researchers and is widely understood and accepted as a benchmark class of problem, as such a simple contrived instance of the TSP has been designed for a brief qualitative analysis. Unfortunately the TSP does not lend itself to a quantitative analysis of the performance of the algorithms presented. This is due to the difficulty in effectively measuring the modality of standard benchmark TSP instances which is discussed in more detail in Sec. 3.1. Instead, several Continuous Function Optimization (CFO) problems have been chosen as test problems. CFO problems are often used to analyse the performance of niching algorithms [8, 12] due to the ability to easily measure the modality of individual problem instances.

2. Algorithm Definitions

2.1. Population-based ACO

Traditional ACO algorithms encode the majority of historic ‘learned’ information in a pheromone map which contains the probability of any solution component being reused in future solution construction. The result being that once a solution updates the history (pheromone map) this discrete solution is discarded. Other EC methods such as Genetic Algorithms (GA) differ from this approach by maintaining learned information in the form of a population of stored solutions.

A population-based ACO algorithm called FIFO-Queue ACO was first introduced in [10]. This algorithm differs from other ACO algorithms in that it maintains a population of discrete solutions rather than only storing all solution information in a pheromone map. A pheromone map is still maintained, however its contents are always a direct reflection of the current state of the population. As a solution is added (to the population of

solutions) its associated component’s pheromone values are positively updated and as a solution is removed from the population the pheromone values associated with this solution are negatively updated.

The FIFO-Queue ACO algorithm was later renamed to Population-based ACO (P-ACO) [9] and has been used in applying ACO algorithms to continuous domains [20], and embedding ACO algorithms in hardware [18]. Maintaining a population of solutions allows for population level measurements and operations to be employed, such as measuring the similarity of stored solutions which is useful in controlling the diversity of stored solutions.

2.2. Determining the Similarity of Solutions

To implement niching, a metric is required to calculate the similarity of candidate solutions; knowing the spatial relationship of solutions in the search space is fundamental to implementation of a successful niching system. For continuous function optimization a common method used to obtain the difference between solutions, when using a real-value encoding scheme, is to calculate the Euclidean distance (1) where P and S are the solution vectors. For TSP, the difference is obtained by calculating the number of common edges used by two solutions and applying formula (2) to obtain a difference measurement. Using (2), 0 represents the solutions being exactly the same, whereas 1 represents the solutions being completely different, i.e. sharing no common edges.

$$difference_{euc} = \sqrt{\sum_{i=1}^n (p_i - s_i)^2} \quad (1)$$

Where: $P = (p_1, \dots, p_n), S = (s_1, \dots, s_n)$

$$difference_{tsp} = 1 - \frac{shared\ edges}{number\ of\ cities} \quad (2)$$

2.3 Fitness Sharing P-ACO

Once calculated, the similarity measurements can be used to derate¹ the quality of a solution according to its proximity to similar solutions. This niching method is known as ‘fitness sharing’ and was formalised by Goldberg and Richardson [8]. The adjusted quality (Q'_i), niche count (c_i) and sharing functions ($sh(d)$) used in this investigation are the same as those outlined in [7] and are reproduced for convenience (3, 4 & 5).

With (3, 4 & 5) if a solution is alone in a neighbourhood (i.e. its nearest neighbour is of a distance $> \sigma_{share}$), then its quality is unaltered since: $sh(d) = 1$ for itself, and this will be the only contribution to the niche count (c_i), ($Q'_i = Q_i$ in this case). If two identical solutions are acted upon with the fitness sharing equations then their respective quality will be halved since the niche count ($c_i = 2$), and this effectively halves the quality of each solution ($Q'_i = Q_i/2$).

$$Q'_i = \frac{Q_i}{c_i} \quad (3)$$

$$c_i = \sum_{j=1}^m sh(d_{ij}) \quad (4)$$

$$sh(d) = \begin{cases} 1 - \left(\frac{d}{\sigma_{share}}\right)^\alpha & \text{if } d < \sigma_{share}, \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

Table 1. Symbols used in 3, 4 & 5

Q_i	: Quality of individual solution i
c_i	: Niche count for solution i
d_{ij}	: Distance (difference) between solution i and solution j
$sh(x)$: Sharing function
σ_i	: Niche radius
α	: Power used to modify the shape of the sharing function

¹*Derate* is a common term in niching literature and refers to a solution’s quality being negatively adjusted.

To implement Fitness Sharing P-ACO (FS-PACO) a temporary pheromone map scheme is used. Rather than simply adding the best new solution and removing the oldest solution as in P-ACO, *all* solutions from a generation are added to the population, and the oldest members are removed. After this addition/removal process is complete, the fitness sharing function is applied to all solutions in the current population to de-rate their respective fitness’. These solutions are then used to construct a pheromone map which is used to create the next generation of solutions using the standard ACO random proportional rule [2, 4].

2.4. Simple Crowding P-ACO

The EC *crowding factor* model was first formalised as a diversity maintenance scheme by De-Jong [3], and later reworked as a niching strategy by Mahfoud [12]. Labelled *deterministic crowding*, it was designed to slow convergence and maintain diversity by limiting dominant building blocks in a population. The replacement policy in deterministic crowding involves a candidate solution competing (based on solution quality) for a place in the history with the most similar solution taken from a subset of the total population of current solutions. If a candidate solution is better than its most similar rival, the rival is replaced, otherwise the candidate solution is discarded. Deterministic crowding is useful when we are selecting a subset of the population to produce the next generation of solutions, for example, the parent candidate solutions in a Genetic Algorithm (GA). Since we are not selecting parents, as in a GA, another form of crowding has been implemented: *simple crowding* [1]. Simple crowding is a specialisation of Restricted Tournament Selection [11].

Simple crowding P-ACO (SC-PACO) involves each new candidate solution being compared against the entire current population of solutions. The closest current member of the population is replaced with this candidate solution if the candidate solution is of better quality. This replacement strategy effectively eliminates duplicate solutions in the history, which has the effect of widening the focus

of the search. The pheromone update and decay policy is the same as that of the P-ACO algorithm.

2.5. P-ACO for TSP and CFO

For the TSP the pheromone map is uniformly initialised and as is standard with most ACO algorithms each pheromone value corresponds to an edge connection of the search graph. In FS-PACO and SC-PACO a greedy pheromone contribution strategy is used (7) where the parameter λ is used to control the greedy bias of solution quality contribution to the pheromone map.

For function optimization problems the pheromone representation used by all algorithms tested (P-ACO, SC-PACO and FS-PACO) is that which was defined by [19]. The method described in this paper is to maintain a population of solutions where each stored solution correlates to a Probability Density Function (PDF). For FS-PACO and SC-PACO implementations tested in this paper the height and width of PDFs are based on the solution quality alone (the standard implementation also uses solution age). Solution quality is obtained (for the minimisation case) by applying the adjustment (6) to the value obtained from the CFO problem addressed. The minimum (f_{min}) and maximum (f_{max}) function values used in (6) are continually updated from the current population during algorithm execution which eliminates the requirement to know the function bounds a-priori.

$$solution\ quality = 1 - \frac{function\ value - f_{min}}{f_{max} - f_{min}} \quad (6)$$

$$\Delta\tau = \left(\frac{path\ length_{current\ best}}{path\ length} \right)^\lambda \quad (7)$$

3 Results

3.1. Travelling Salesman Problem

Although it is often easy to visualize a TSP from a tour creation perspective, it is much more difficult to visualize the solution space of a TSP than

that of a CFO. Techniques exist to map all solutions to an n-dimensional TSP into a 2-dimensional space [22], however these techniques fail to preserve the neighbourhood relationship of the TSP. This lack of visualization technique contributes to the difficulty in accurately measuring the modality of standard benchmark TSP instances, and therefore the normal criterion used to evaluate the performance of the niching algorithms (peak location and maintenance) is unavailable.

Instead of using a quantitative analysis method a qualitative analysis is used to show that for a single run on a trivially simple TSP instance a state-of-the-art ACO algorithm (Max-Min Ant Systems [5, 21]) fails to locate and maintain both of the two distinct optima past convergence, whereas FS-PACO and SC-PACO locate and maintain both optima past convergence. The 'crown' problem is a symmetric, 2-Dimensional Euclidean TSP containing 6 vertices (Fig. 1), is bi-modal (two global optima) with the optima being a reasonable distance apart (difference = 0.5)².

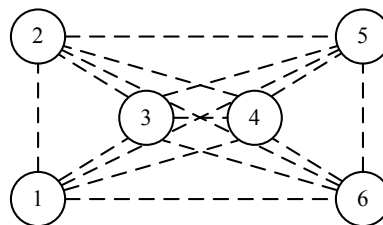


Figure 1. Crown TSP problem visual representation

The two optimal solutions present in the crown problem are shown in Fig. 2. For comparison purposes the MMAS algorithm was tested³, and the number of times either of the two optima were found was recorded (Fig. 3(a)). This experiment was repeated 100 times for consistency of reported results. The graphs presented in Fig. 3(a), Fig. 3(b) and Fig. 3(c), although illustrating single experi-

²The coordinates of this dataset are: $\{\{0,0\},\{0,100\},\{50,50\},\{100,50\},\{150,100\},\{150,0\}\}$

³Using the standard parameters as in [21]

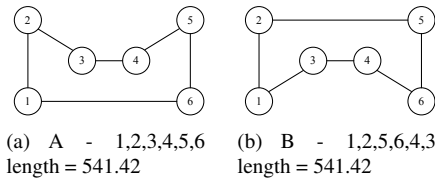


Figure 2. Optimal solutions to crown problem

mental runs, are indicative of each algorithms behaviour⁴.

Due to the small search space size of the problem it was not surprising that MMAS was able to find both optimal solutions in a short time, however this algorithm, which at the start was able to find both optima, quickly converged on a single optimal solution. Unlike MMAS, both FS-PACO and SC-PACO were able to locate and *maintain* both optimal solutions for the duration of the search.

The crown problem has demonstrated that on a simple problem, FS-PACO and SC-PACO are able to locate and maintain multiple optimal solutions in a single run, whereas under the prescribed configuration MMAS does not. This analysis was extended to a larger (albeit still small) TSP instance, the berlin52 problem [16]. The same observations of peak location and maintenance held for this problem as did on the small crown problem.

As it is not possible to easily measure the modality of standard TSPs a quantitative assessment of these algorithms niching ability on these problems cannot be produced. It is for this reason that attention has been concentrated on CFO, a class of problem whose modality is readily available.

3.2. Continuous Function Optimization

This study is concerned with each algorithm's ability to obtain not just a single best solution but multiple good solutions. The performance criterion used in this study is the number of distinct optima located and maintained. An optimum is considered

⁴Parameters and implementation details can be found at: <http://www.it.swin.edu.au/personal/dangus>

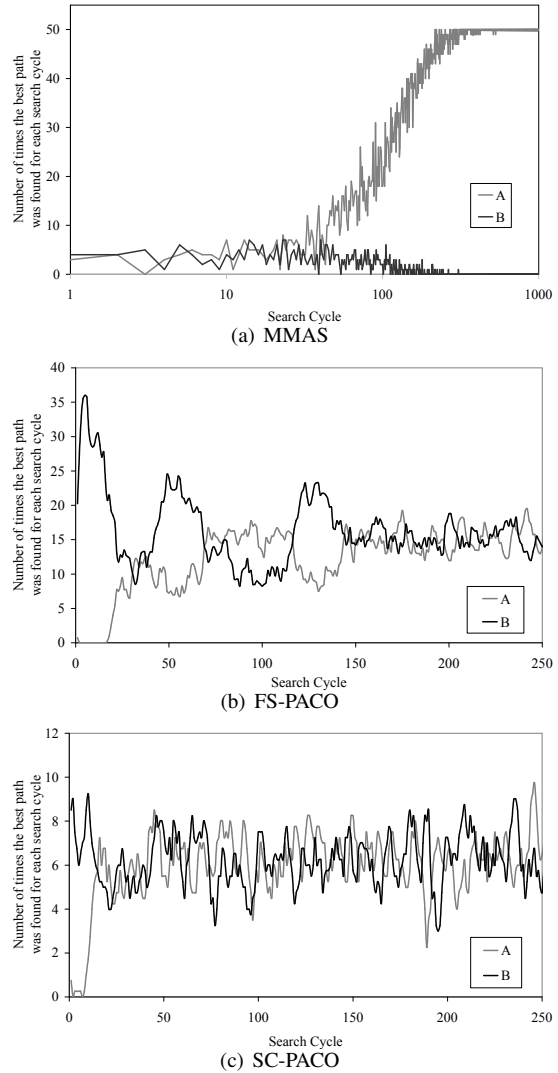


Figure 3. Crown problem: Occurrence of the two optimal paths over time for a single (although indicative) experiment run (A & B are the two distinct optimal paths)

found if a solution exists that is within a specified radius of this optimum. Within this radius the solution is considered to be close enough to the optimum that if desired a local-search technique would be able to find the exact location of the optimum. For all functions tested the radius was set to 0.1% of the bounds of the function. The stopping criteria for all algorithms was set to 50,000 function evaluations which is well past the convergence point for all algorithms tested for completeness and consistency.

Four common test functions (restricted in 2-Dimensions for easy visualization) were selected to be tested: 3-pot holes, Branin's R-Cos, Himmelblau's and Shekels's Foxholes Functions⁵. FS-PACO and SC-PACO were tested on these functions and their performance is compared to P-ACO.

The results (Tab. 2) are encouraging for SC-PACO which for two of the test instances was able to locate and maintain all distinct optima. Furthermore, SC-ACO located and preserved all three optima in 94% of trials on the 3 pot-holes function and located and preserved all three optima in 85% of trials on the Branin's R-Cos function. The results were not as encouraging for the FS-PACO algorithm which demonstrated less than ideal performance on all four test functions. On average, FS-PACO was able to locate and preserve close to half of the distinct optima for the test functions used. It is hypothesised that the reason for some of this poor performance was due to the sensitivity of the niche radius parameter (σ_i). Effective configuration of this parameter is dependent on apriori knowledge of the spacing and size of optima in the search space [23].

The performance of P-ACO was the poorest, as expected. As this algorithm was not designed for niching it was only able to identify (although to its credit consistently) a single optima for all four functions tested. Selection of the best solution per iteration for insertion into the population results in this algorithm quickly converging on an individual optimum. Figure 4 shows the typical sampling behaviour of each of the three test algorithms for a

single experimental run. It is speculated that the SC-PACO algorithm applied to higher dimension CFO problem will still exhibit useful niche formation and maintenance, however as the dimensionality of the problem instance increases so should the history size⁶.

4. Conclusions and Future Work

Both niching methods evaluated in this study were shown to be effective at identification and maintenance of multiple optimal solutions past convergence for an illustrative TSP instance and several CFO instances of varying modality. For all problem instances none of the non-niching ACO algorithms tested were able to maintain multiple optimal solutions. The niching methods presented assist in the maintenance of useful diversity, which is expected to improve algorithm robustness when applying ACO algorithms to more complex multimodal problem domains such as dynamic problems. In this type of problem dynamic changes can render some existing solutions infeasible at various times, and thus storing a wider range of solutions may lessen the chance of the algorithm history becoming completely infeasible.

The effects of niche radius (σ_i) and other parameters are not discussed in this paper as the aim of this paper is to demonstrate a proof of concept of niching ACO. A larger study into the effects that these parameters have on the algorithms behaviour on a wider range of test problems is under-way. It is also clear that the niching methods implemented in this investigation represent only a subset of available niching techniques, however this paper has demonstrated that niching with ACO algorithms is feasible and effective.

Other interesting areas for future work will be the implementation of niching-like behaviour in multiple colony ACO algorithms as it may allow for more effective information sharing between colonies. There is evidence to support the hypothesis that simple multi-colony schemes, which only

⁵Details of these functions can be found at: <http://www.it.swin.edu.au/personal/dangus>

⁶It was observed in preliminary experimentation that if the modality of the problem is larger than the history storage SC-PACO fails to maintain all useful niches.

Table 2. Results of testing FS-PACO & SC-PACO on a suite of 2-Dimensional multimodal CFO problem instances compared to the P-ACO algorithm (results averaged over 100 trials)

Algorithm	Problem (Number of optima)	No. Optima Located (Var)	% Optima Located (Var)
FS-PACO	3 Pot-holes Function (3)	1.21 (0.17)	40.33% (5.66%)
	Branin's R-Cos Function (3)	1.64 (0.23)	54.66% (7.66%)
	Himmelblau's Function (4)	2.19 (0.68)	54.75% (17.00%)
	Shekel's Foxholes (25)	10.45 (3.42)	41.80% (13.68%)
SC-PACO	3 Pot-holes Function (3)	2.94 (0.05)	98.00% (1.66%)
	Branin's R-Cos Function (3)	2.85 (0.13)	95.00% (4.33%)
	Himmelblau's Function (4)	4 (0)	100% (0%)
	Shekel's Foxholes (25)	25 (0)	100% (0%)
P-ACO	3 Pot-holes Function (3)	1 (0)	33.33% (0%)
	Branin's R-Cos Function (3)	1 (0)	33.33% (0%)
	Himmelblau's Function (4)	1 (0)	25.00% (0%)
	Shekel's Foxholes (25)	1 (0)	4.00% (0%)

share best-so-far information, are somewhat ineffective [14]. In this recent paper the authors suggest that a niching-like information sharing mechanism would help independent colonies avoid searching the same areas of search-space. We speculate that a niching extension applied to multi-colony ACO would allow for an increase in the diversity of solutions (thereby increasing robustness) without adversely affecting quality of the solutions since each independent colony can be allowed to concentrate on a single (distinct) area of the search space.

These niching techniques could also be useful in applying ACO algorithms to multiple objective optimization problems. In this class of problem the requirement is for the location and maintenance of multiple (pareto) optima rather than a singular optima.

References

- [1] J. Brownlee. Parallel niching genetic algorithms: A crowding perspective. Master's thesis, Swinburne University of Technology, 2004.
- [2] O. Cordon, F. Herrera, and T. Stützle. A review of the ant colony optimization metaheuristic: Basis, models and new trends. *Mathware & Soft Computing*, 9(2,3), 2002.
- [3] K. A. DeJong. *An analysis of the behaviour of a class of genetic adaptive systems*. PhD thesis, University of Michigan, 1975.
- [4] M. Dorigo and G. Di Caro. Ant colony optimization: a new meta-heuristic. In *Proceedings of the 1999 Congress on Evolutionary Computation (CEC99)*, pages 1470–1477. IEEE, 1999.
- [5] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, London, 2004.
- [6] N. Eldredge. *Macroevolutionary Dynamics: Species, Niches and Adaptive Peaks*. McGraw Hill, 1989.
- [7] D. E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley, 1989.
- [8] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49, 1987.
- [9] 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.
- [10] 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.

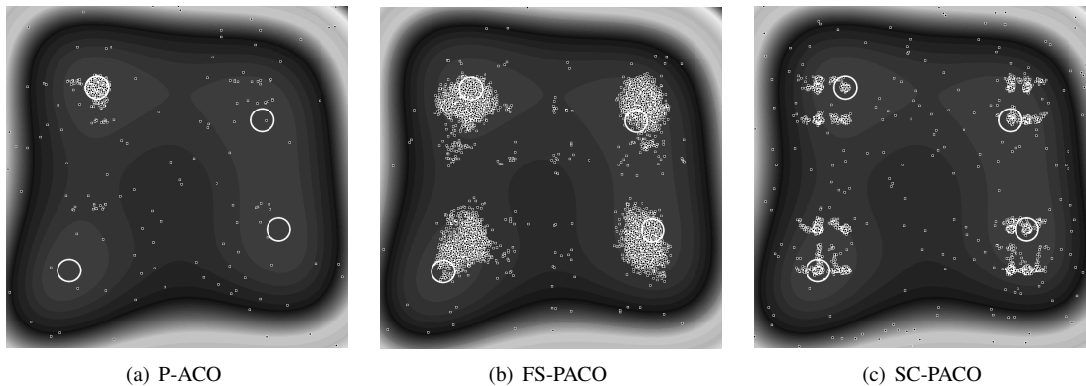


Figure 4. Diagrams indicating 50,000 samples generated in a single run by each algorithm applied to Himmelblau's Function (White circles indicate the four distinct optima, small white squares represent samples). It is clear that P-ACO has only found one optimum whereas FS-PACO and SC-PACO have located all four optima (although FS-PACO has not converged as well as SC-PACO to the centre of all four optima).

- [11] G. R. Harik. Finding multimodal solutions using restricted tournament selection. In L. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 24–31, San Francisco, CA, 1995. Morgan Kaufmann.
- [12] S. W. Mahfoud. *Niching methods for genetic algorithms*. PhD thesis, University of Illinois, 1995.
- [13] S. W. Mahfoud. *Evolutionary Computation 2: Advanced Algorithms and Operators*, chapter Niching Methods, pages 87–92. Institute of Physics Publishing, UK, 2000.
- [14] M. Manfrin, M. Birattari, T. Stützle, and M. Dorigo. Parallel ant colony optimization for the traveling salesman problem. Technical Report 2006-007, IRIDIA, March 2006.
- [15] G. Reinelt. TSPLIB - a traveling salesman problem library. *ORSA Journal on Computing*, 3:376–384, 1991.
- [16] G. Reinelt. Tsplib95, 1995. Available at: <http://www.iwr.uni-heidelberg.de/groups/comopt/software/tsplib95>.
- [17] R. E. Ricklefs. *Ecology*. Thomas Nelson & Sons Ltd, 1973.
- [18] B. Scheuermann. *Ant Colony Optimization on Runtime Reconfigurable Architectures*. PhD thesis, Universität Fredericiana zu Karlsruhe, 2005.
- [19] K. Socha. ACO for Continuous and Mixed-Variable Optimization. In M. Dorigo, M. Birattari, and C. Blum, editors, *Proceedings of ANTS 2004 - Fourth International Workshop on Ant Colony Optimization and Swarm Intelligence*, volume 3172 of LNCS, pages 25–36. Springer-Verlag, Berlin, Germany, 5-8 September 2004.
- [20] K. Socha and M. Dorigo. Ant colony optimization for continuous domains. Technical Report 2005-037, IRIDIA, December 2005.
- [21] T. Stützle and H. Hoos. Improvements on the ant system, introducing the MAX-MIN ant system. In *ICANNGA97 - Third International Conference on Artificial Neural Networks and Genetic Algorithms*, University of East Anglia, Norwich, UK, 1997.
- [22] S. Tsubakitani. A two-dimensional mapping for the traveling salesman problem. *Computers & Mathematics*, 26:65–73, 1993.
- [23] R. K. Ursem. When sharing fails. In *Proceedings of the Third Congress on Evolutionary Computation (CEC-2001)*, pages 873–879, 2001.

5. Acknowledgement

The author acknowledges the invaluable assistance received by Jason Brownlee and also acknowledges the guidance Prof. Tim Hendtlass has provided as well as the rest of the team at CIS.