Discrete History Ant Systems

Daniel Angus, Jason Brownlee

Faculty of Information & Communication Technologies
Swinburne University of Technology
Melbourne, Australia
{dangus,jbrownlee}@ict.swin.edu.au

Abstract. Ant Colony Optimisation (ACO) algorithms are inspired by the foraging behaviour of real ants and are a relatively new class of algorithm which have shown promise when applied to combinatorial optimisation problems. In recent years ACO algorithms have begun to gain popularity and as such are beginning to be applied to more complex problem domains including (but not limited to) dynamic problems. Recent modifications to the fundamental ACO algorithms such as population based approaches are enabling ACO algorithms to be competitive to other known biologically inspired search techniques in addressing these more complex problems. This paper outlines a foundation population based ACO algorithm which is imbued with characteristics that allow for multiple extensions beneficial in addressing more complex problems.

1 Introduction

The Ant Colony Optimisation (ACO) class of algorithm is loosely based on the foraging behaviour of Argentine ants and was formalised by Dorigo et al. in 1999 [2]. One of the very first ACO algorithms, which was presented before the general ACO framework was developed, is the Ant Systems (AS) algorithm [4]. AS, like the majority of other ACO algorithms uses an iterative step-wise probabilistic solution construction process to construct candidate solutions to a search/optimisation problem domain, which are then evaluated and merged into a common history that is used for future solution construction (commonly known as a *pheromone map* or *pheromone matrix*). In merging the candidate solutions into a common history some information loss occurs, and it is hypothesised that some of this information should be retained for use by the algorithm to assist in future solution construction.

In this paper a new ACO algorithm, Discrete History Ant Systems (DHAS) is introduced. Unlike other ACO algorithms DHAS maintains a discrete list of past solutions (to a pre-specified limit) rather than merging all solution information into a pheromone map. DHAS is similar to both the AS algorithm and the FIFO-Queue ACO algorithm (a population based ACO algorithm) proposed by Guntsch and Middendorf [6, 7], however, DHAS differs from FIFO-Queue ACO in areas such as solution storage and maintenance (A detailed discussion of the similarities and differences between the two techniques is offered in Sect. 5).

DHAS is a foundation algorithm that has been designed to be extensible, as such, the focus of this paper is to demonstrate the general properties of the DHAS algorithm and also present preliminary performance comparisons to other ACO techniques, including AS and Ant Colony System (ACS) [3]. The test problem chosen in this study is the Travelling Salesman Problem (TSP) [11]. Instances

of this problem were chosen because they have known solutions, are easy to assess, easy to encode and provide a consistent basis for comparison within the field of ACO.

Section 2 outlines implementation specific details of DHAS and preliminary performance observations of an implementation of DHAS are offered in Sect. 3. Several simple modifications and their effects are described in Sect. 4. In Sect. 5, DHAS is compared with another population based ACO algorithm: FIFO-Queue ACO. Section 6 offers concluding remarks and perhaps most importantly, several areas for future extension of the DHAS algorithm.

2 Discrete History Ant Systems

The original ACO algorithm: Ant Systems (AS) [4], was selected as the basis for the new population based algorithm. This selection was made because it is the canonical ACO algorithm from which a majority of other ACO algorithms have been derived. AS, like the majority of ACO algorithms [3, 5, 13] requires a single pheromone mapping for the stepwise solution construction process. All individual solution components (roads/edges in a TSP for example) are assigned a single value representing their usefulness in the recent past, commonly referred to as a *pheromone value*.

While this singular history repository is computationally efficient for AS, it is not suitable for the implementation of a population based ACO approach. A population based approach requires the storage of many individual solutions which may refer to similar solution components, whereas the canonical ACO requires that a single quality metric be available for each solution component regardless of how many solutions it has been used in. In order to maintain the granularity of the history in DHAS while still making use of the step-wise construction procedure used in AS, a stack structure (Fig. 1) has been adopted which can be collapsed into a single *temporary pheromone mapping*, thereby fulfilling the requirements of discrete history components and singular solution component quality metric.

In AS, to direct the search via the most current search history, it is desirable that newer solutions contribute more to the direction of the search process than older solutions. This is referred to as update and decay, where update is insertion of new solutions into the history and decay is the gradual removal of old solutions from the history. These same properties are required in DHAS, however the implementation is subtly different. In DHAS the pheromone mapping is erased and reconstructed every search cycle; hence it is referred to as a temporary pheromone map. A search cycle is every ant in the population of ants creating a complete solution to the problem being optimised.

Algorithm 1 Discrete History Ant Systems

```
1: while not(stop condition) do
2:     Construct pheromone map
3:     Construct Solution(s)
4:     Update history
5: end while
```

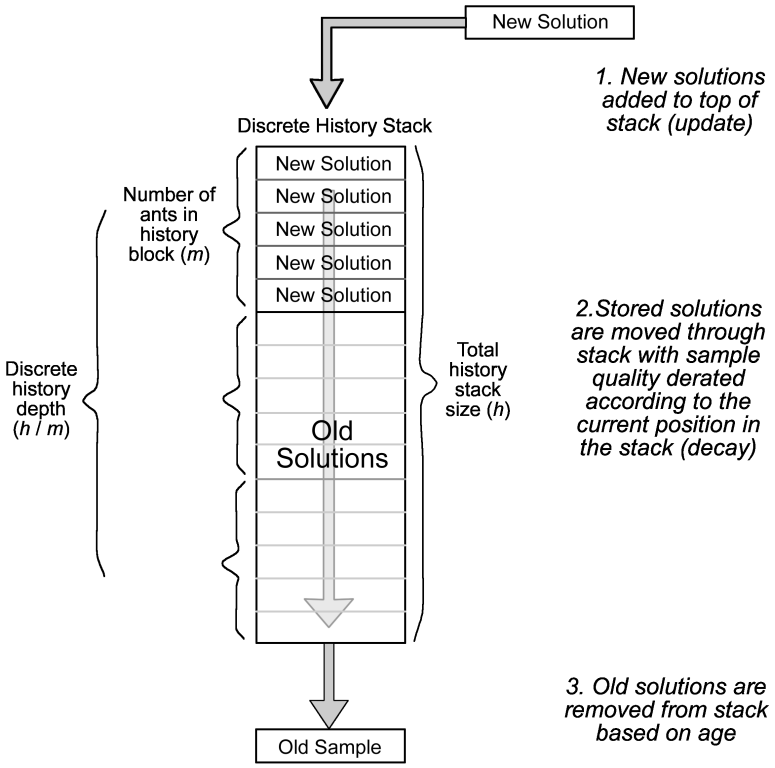

Fig. 1. Stack operation in discrete history ACO. New candidate solutions are added to the top of the stack, while existing solutions are moved through the stack until the limit h is reached at which time these solutions are removed from the stack.

The high level operation of DHAS is shown in Alg. 1 and is described as follows.[1]

1. The transformation of the discrete history of solutions into a single pheromone map for use in a traditional AS sense is achieved by the following steps:
   (a) Clear the temporary pheromone map by setting all values to zero.
   (b) Cycle through history adding the quality of each entry in the history to the corresponding location in the pheromone map.
   (c) Assign the value of the minimum pheromone value threshold to all values in the pheromone map that fall below a minimum pheromone value threshold. The threshold is set to the lowest quality solution contribution amount in the entire history stack by default.
2. Construction of solutions is the same as that of AS, once the pheromone map is constructed.

[1] Note: For the first cycle all values in the temporary pheromone map are set to a uniform value of 1 since there is no history available to build the temporary pheromone map.

3. History update involves the insertion of new solutions and the removal of old solutions from the discrete history stack. New solutions are evaluated before being inserted so that they can make a contribution to the pheromone mapping. History update also lowers the quality of every solution in the discrete history stack according to (1), the concepts of solution age and discrete history depth are clarified later.

$$Q' = Q - \left(Q \cdot \frac{\text{age of solution}}{\text{discrete history depth}} \right) \tag{1}$$

Where:

Q : Original quality of solution (For the TSP $Q \propto 1/\text{Path Length}$)
Q' : Adjusted quality

Figure 1 provides a more detailed view of step 3 (updating of history) in Alg. 1. Figure 1 also shows how all solutions are added to the discrete history (not just the best solutions) and that all solutions persist in the discrete history for an equal length of time.

To clarify the concepts of discrete history depth and age of solution, the age of a solution when it is first pushed onto the discrete history stack is set to zero. For every subsequent solution cycle the age of existing solutions is incremented by one until it reaches the discrete history depth limit (h/m) at which point it is popped from the stack and discarded. For the algorithm to function correctly the total history stack size (h) must be a multiple of the number of ants (m), otherwise there will be a non-uniform number of history entries for each depth-level of the stack. This restriction was imposed for initial implementation and analysis, however there is no reason why this restriction could not be relaxed in future implementations.

3 DHAS applied to the static TSP

This section presents the findings of the application of DHAS to a suite of TSPs. Results on all TSPs obtained with the AS and ACS algorithms (using default parameterisations) are included as benchmarks[2]. ACS is included as a benchmark as it is among the best performing ACO algorithms for these instances of the static TSP.

Before testing on larger TSPs a brief sensitivity analysis of DHAS was performed using the Berlin52 problem[3]. Three parameters: heuristic power, pheromone power and total history stack size were varied and the effect on tour length observed. The total number of unique parameterisation settings was 595, and each specific parameterisation setting was repeated 50 times with the average path length being recorded. These average path lengths are presented as histograms (bin size = 100) organised by a specific parameter (the results are presented in Figs. 2, 3 & 4).

[2] The results for ACS are taken directly from [3] and the AS results are from the author's own implementation of the AS algorithm implemented without elitism or local search functionality.
[3] All TSPs are available from [11].

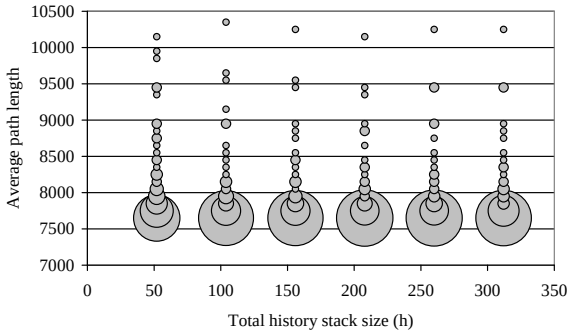

Fig. 2. Histogram highlighting the effect of varying the total history stack size on average path length found (bin size = 100, and the bubble size represents the number of bin entries) using 595 unique parameterisations (Each parameterisation setting was averaged over 50 trials).

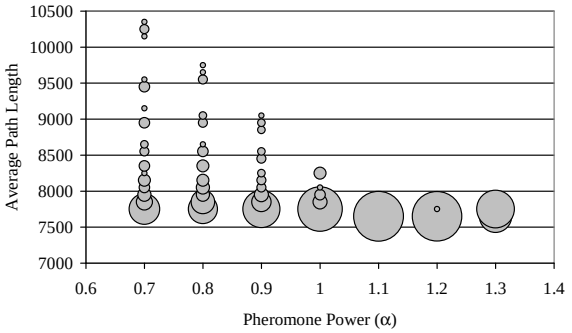

Fig. 3. Histogram highlighting the effect of varying the pheromone power(α) on average path length found (bin size = 100, and the bubble size represents the number of bin entries) using 595 unique parameterisations (Each parameterisation setting was averaged over 50 trials).

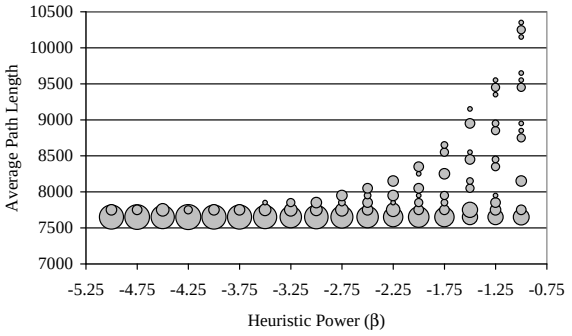

Fig. 4. Histogram highlighting the effect of varying the heuristic power(β) on average path length found (bin size = 100, and the bubble size represents the number of bin entries) using 595 unique parameterisations (Each parameterisation setting was averaged over 50 trials).

The results of the sensitivity analysis show that a total history stack size (h) greater than 104 (h = (number of ants = 52) × (history depth = 2)) makes no statistically significant difference to the average path length found, and that the use of similar heuristic and pheromone powers as other ACO algorithms ($\alpha \approx 1$,$\beta \approx -3$) make for the best performance of all parameterisation settings tried on the test problem. Although history size has little effect on the average path length found, it is worth keeping in mind that it does increase the computation time (observed to be an approximately linear increase with history size) required by the algorithm when creating the temporary pheromone matrix.

Using the configuration suggested by the sensitivity analysis some larger TSPs were attempted using DHAS. Each experiment was repeated 30 times with the best results obtained recorded along with the number of tours generated to achieve this solution (Tab. 1). In all cases DHAS outperforms AS in terms of the quality of the solution found, and DHAS is orders of magnitude faster to converge than AS and ACS, this result would suggest that the heuristic in DHAS might be able to be tuned to make the algorithm less greedy and increase the search efficacy, however this has not yet been tested.

Table 1. Comparison of DHAS to standard ACS & AS (All implemented without local search or elitism) when applied to the TSP. The best tour length found in all trials is included along with the number of tours generated to find this tour in brackets.

	Optimum	ACS	AS	DHAS
KroA100	21,282	**21,282** (4,820)	22,715 (15,000)	22,145 (**2,900**)
d198	15,780	**15,888** (585,000)	17,382 (30,000)	16,843 (**4,158**)
pcb442	50,779	**51,268** (595,000)	62,536 (60,000)	58,057 (**9,282**)
rat783	8,806	**9,015** (991,276)	11,303 (120,000)	10,541 (**13,311**)

Table 2. Parameters used for testing TSP problems presented in Tab. 1. Where n = number of cities & L_{nn} is an estimation of the optimal tour length

Algorithm	History Size	Number of Ants	Pheromone Power	Heuristic Power	Decay	q_0	Initial Pheromone
AS	-	n	1	-2	0.5	-	$n \times L_{nn}$
ACS	-	n	2	-2	0.1	0.9	$n \times L_{nn}$
DHAS	$3 \times n$	n	1	-3	-	-	1

4 Extensions to DHAS

4.1 Elite DHAS strategy

One possible area for extension of the DHAS algorithm is to include an elitist strategy. In the seminal works on AS [4] an elitist strategy was shown to improve algorithm performance. This

research suggests that an elitist strategy may improve the performance DHAS. To implement an elitist strategy DHAS continues to operate in the standard manner, however the best solution found to date is recorded and during the temporary pheromone mapping construction phase the elements of the elite solution are added to the temporary mapping as in (2). The quantity (e) determines the magnitude of the elite influence on the temporary pheromone mapping.

$$\tau_{ij} = \tau_{ij} + \frac{e}{\text{elite path length}}, \forall (i,j) \in T^{bs} \tag{2}$$

Where:

e : Magnitude of elite influence
τ_{ij} : Pheromone on edge (i,j)
T^{bs} : Elite solution (Best solution)

Experiments were run varying the magnitude of the elite influence (e). Figure 5 shows that a large elite magnitude can force the algorithm to converge on sub-optimal solutions, however a moderate elite influence can improve algorithm performance (in terms of the quality of the solution found).

4.2 Alternate solution contribution strategies

The contribution a specific solution makes to the temporary pheromone mapping is governed by its quality and age. An alternate approach is to remove the dependency on age to simplify (1) to that of (3). This approach goes against conventional ACO wisdom[4], however an analysis is still included to test if DHAS behaviour is still similar to canonical ACO algorithms. Another alternative contribution scheme is to decrease the influence of older solutions, more sharply than that of the standard linear age decay mechanism, by applying a power term (ψ) to the quality contribution as in (4).

$$Q' = Q \tag{3}$$

$$Q' = \left(Q - Q \cdot \frac{\text{age of solution}}{\text{discrete history depth}} \right)^{\psi} \tag{4}$$

To summarise, a comparison of the original contribution strategy and the two alternative contribution strategies is included in Fig. 6. These contribution strategies were tested on the Berlin52 problem and the results of running these three distinct contribution strategies while varying the history size is included in Fig. 7.

As in Sect. 3, increasing the total history size past 104 has little effect on the average path length found. However, the contribution scheme does have a measurable effect on the mean path length found. The linear age decay to a power increases the quality of solutions found over the standard linear age decay, whereas no decay decreases the average quality of solutions found.

[4] It was stated in Sect. 2 that it is desirable for newer solutions to contribute more to the direction of the search process than older solutions, hence the contribution of older solutions is often decayed

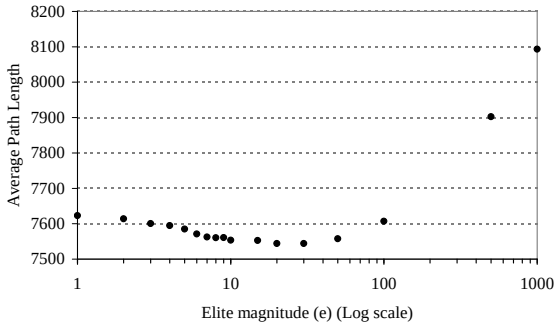

Fig. 5. Effect of varying the magnitude of the elite influence on average path length (average path length is reported for 100 repetitions of each parameterisation setting) presented on a log scale.

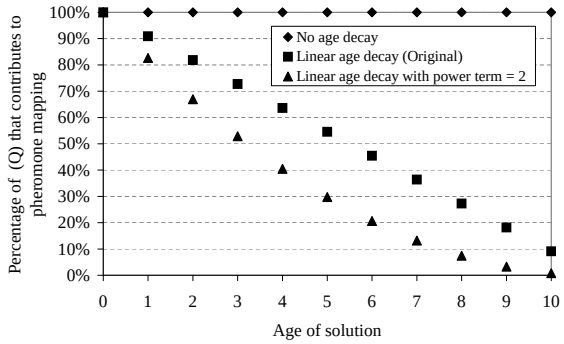

Fig. 6. Comparison of temporary pheromone mapping contribution strategies. Age = 0 represents a new solution being placed into the stack, and this age is incremented every solution cycle.

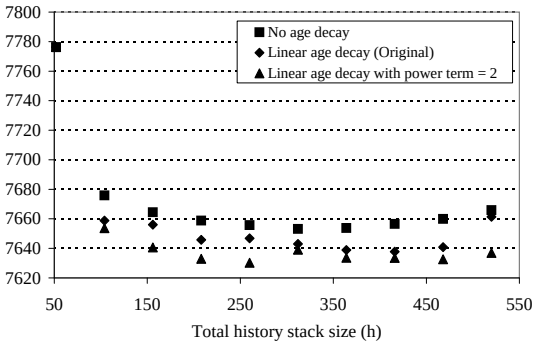

Fig. 7. Comparison of history contribution schemes with varied history size (average path length is reported for 100 repetitions of each parameterisation setting). For total history stack size > 104 the variance between the data points in each series in statistically insignificant, however the variance between each series while small is still meaningful.

5 Comparison and Contrast of DHAS and FIFO-Queue ACO

The FIFO-Queue ACO algorithm [7] is the most similar ACO algorithm to DHAS in the sense that both algorithms store history in a population stack and use this population stack to create/maintain a pheromone mapping. This section compares and contrasts the implementation and search behaviour of the DHAS and FIFO-Queue ACO algorithms.

FIFO-Queue ACO was designed specifically for application to dynamic optimisation problems. In dynamic optimisation the speed with which the search algorithm can adapt its history is a strong determinant of its performance. In reference to ACO applied to dynamic optimisation, Gunstch and Middendorf [7] comment: 'it will usually be faster to modify a few solutions directly than to modify the whole pheromone information of a usual ACO algorithm'. FIFO-Queue maintains a small history of solutions which can be quickly changed given a change in the problem being optimised. Each solution in the history directly influences the pheromone mapping so that very few changes occur to the pheromone mapping in any search cycle. To maximise search efficacy the FIFO-Queue algorithm only inserts the best solution from each search cycle into the history, thereby maintaining a small population of elite solutions. The FIFO-Queue ACO algorithm is adept at making fast changes in an unstable search environment and has been shown to be more efficient than canonical ACO algorithms for dynamic combinatorial optimisation [6]. The purpose for maintaining a population of solutions in FIFO-Queue ACO is to provide the algorithm with a quick way to adjust the pheromone mapping if a change occurs to the problem being optimised.

DHAS has been designed as a foundation algorithm with flexibility of application, and extensibility two of the major design goals. In Sect. 6 proposed extensions such as niching, parallel implementation and sequential search require access to a population of discrete solutions rather than a combined history which is how canonical ACO algorithms store historic search information. Hence the DHAS has been designed as a non-specialised algorithm with simplistic update procedures and a modular framework. Properties of the DHAS algorithm and FIFO-Queue ACO algorithm are summarised in Tab. 5.

Table 3. Comparison of DHAS to FIFO-Queue ACO

Algorithm	DHAS	FIFO-Queue ACO
Population Size	Large (100-300)	Small (1-10)
Suitable Problem Class	Static Optimisation	Dynamic Combinatorial Optimisation
Convergence Properties	Tunable (Slow-Fast)	Fast

Both FIFO-Queue ACO and DHAS achieve the same goal (although through differing means), a population based history representation for ACO. While FIFO-Queue ACO introduces a population based solution representation to speed up adjustments to the pheromone mapping, DHAS introduces a population representation for flexibility of application and extension. An important distinction between the two algorithms is the amount and type of historic information used by each algorithm with DHAS being in many ways a generalisation of the FIFO-Queue ACO algorithm. It may be useful to consider that FIFO-Queue ACO could easily be considered a specialisation of DHAS for dynamic combinatorial problems.

6 Conclusions and Future Work

This study has demonstrated that it is possible for an ACO algorithm to maintain a discrete list of past solutions rather than merging all solution information into a pheromone mapping. It is also evident that when provided with a similar amount of resource (number of evaluations) DHAS shows greater efficacy (measured as minimum and average path length) than AS in the case of the TSPs tested here. This result is impressive due to the fact that DHAS was tested without useful efficiency improvements such as elitism and alternative contribution schemes, which may explain why it was less efficient than ACS.

Although DHAS has access to large amounts of information previously discarded by other ACO algorithms such as AS, the DHAS experiments presented in this paper have made no real use of this extra information. Extensions taken from the field of Evolutionary Computation which could potentially make effective use of this extra information and which may prove fruitful and interesting are:

1. Niching - Niching permits a more effective use of allocated resources to a search algorithm on a single machine by either implicitly or explicitly partitioning a population and searching different areas of the search space in parallel [8]. This technique has proved useful when the complexity of the problem domain is scaled up to include multiple global and local deceptive optimal solutions, which are commonly referred to as multimodal problems. A niching extension for DHAS is presented in [1].
2. Island Population/Multiple Colony - Parallel implementations of ACO algorithms have achieved much attention and have all shown improvement over traditional singular implementations [9, 10, 12]. DHAS allows for a simplistic island population implementation since in DHAS historic information is treated as discrete solution entities which can be easily shared between other populations of solutions.
3. Parallel/Sequential Hybrid Search - Combinations of techniques such as DHAS & Particle Swarm or DHAS & Genetic Algorithm can be easily implemented. Particle Swarm and Genetic Algorithms represent history as discrete solutions, therefore these solutions can be easily inserted into or extracted from DHAS's discrete history allowing the DHAS algorithm to be run in sequence with or parallel to these other search techniques.

Some scope exists to increase the computational efficiency of routines such as the temporary pheromone mapping construction, although it was not of concern in this study due to the limited complexity of the problems tested.

References

[1] D. Angus. Niching ant systems. Submitted to ANTS2006, Fifth International Workshop on Ant Colony Optimization and Swarm Intelligence, September 2006.

[2] M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5:137–172, Spring 1999.

[3] M. Dorigo and L. M. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computing*, 1(1):53–66, 1997.

[4] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 26(1):29–41, Feb 1996.

[5] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, London, 2004.

[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] S. W. Mahfoud. *Evolutionary Computation 2: Advanced Algorithms and Operators*, chapter Niching Methods, pages 87–92. Institute of Physics Publishing, UK, 2000.

[9] R. Michel and M. Middendorf. An island model based ant system with lookahead for the shortest supersequence problem. In *PPSN V: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, pages 692–701, London, UK, 1998. Springer-Verlag.

[10] M. Randall and A. Lewis. A parallel implementation of ant colony optimization. *J. Parallel Distrib. Comput.*, 62(9):1421–1432, 2002.

[11] G. Reinelt. Tsplib95, 1995. Available at: `http://www.iwr.uni-heidelberg.de/groups/comopt/software/tsplib95`.

[12] T. Stützle. Parallelization strategies for ant colony optimization. In *PPSN V: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, pages 722–731, London, UK, 1998. Springer-Verlag.

[13] T. Stützle and H. Hoosa. 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.

Acknowledgement

The authors gratefully acknowledge the assistance Prof. Tim Hendtlass and Clinton Woodward have provided as well as the rest of the team at the Centre for Intelligent Systems & Complex Processes (CISCP).