

Crowding Population-based Ant Colony Optimisation for the Multi-objective Travelling Salesman Problem

Daniel Angus

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

Abstract—Ant inspired algorithms have recently gained popularity for use in multi-objective problem domains. One specific algorithm, Population-based ACO, which uses a population as well as the traditional pheromone matrix, has been shown to be effective at solving combinatorial multi-objective optimisation problems. This paper extends the Population-based ACO algorithm with a crowding population replacement scheme to increase the search efficacy and efficiency. Results are shown for a suite of multi-objective Travelling Salesman Problems of varying complexity.

I. INTRODUCTION

Ant Colony Optimisation (ACO) [7], [10], an optimisation methodology based on the foraging behaviour of Argentine ants, has been shown to be useful in finding optimal or near-optimal solutions to optimisation problems. The first applications of ACO algorithms were to combinatorial optimisation problems such as the Travelling Salesman Problem (TSP) and Quadratic Assignment Problem (QAP), although in recent years the paradigm has been applied to a much wider range of problem domains with (usually) successful results. One such problem domain is Multiple Objective Optimisation (MOO).

MOO is concerned with finding multiple ‘trade-off’ solutions in order to optimise many (in many cases conflicting or orthogonal) objectives. MOO problems are found frequently in the real-world. A commonly cited example is designing an automobile, where a designer may be attempting to simultaneously decrease cost and increase safety and comfort. In this example it is fairly clear that the designer must be willing to make trade-offs between all of these objectives because they are non-complimentary. For any MOO problem there is a set of optimal trade-off solutions which are referred to as the Pareto set, after the economist Vilfredo Pareto.

This study is concerned with the evaluation of a new Population-based ACO algorithm: Crowding PACO (CPACO) [2], against a set of combinatorial MOO problems. The CPACO algorithm maintains solution diversity through a selective population replacement scheme, crowding, taken from Evolutionary Computation [6], [19]. The purpose of

maintaining diversity in MOO is to encourage the algorithm to converge across the Pareto front rather than to one specific location. The performance of the CPACO algorithm is measured by its ability to not only locate the Pareto front but to spread the population uniformly across it. As such two measures have been used to compare the performance of the algorithms tested, attainment surface comparison and the C metric.

II. POPULATION-BASED ANT COLONY OPTIMISATION

To date there have been many ant-inspired algorithms proposed for multi-objective optimisation problems. An excellent review and analysis paper on the subject is [12]. In this paper eight major ant-inspired algorithms along with two state-of-the-art Genetic Algorithms (NSGA-II & SPEA2) were implemented and compared. For the particular test cases used (instances of a bi-criteria TSP) the ant-inspired algorithms performed well. One algorithm in particular, Population-based ACO (PACO), performed consistently within the top three ant-inspired algorithms. This section describes PACO for single and multi-objective optimisation.

A. PACO for Single Objective Optimisation

The Population-based Ant Colony Optimisation (PACO) algorithm was first introduced in [14], [15] and later extended for a multi-criteria optimisation problem, the single machine total tardiness problem with changeover costs, in [13], [16]. The defining difference between PACO and the canonical ACO algorithm is in the area of solution storage. Whereas most traditional ACO algorithms (Ant Systems, Ant Colony Systems, Max-Min Ant Systems) store solution information from an (artificial) ant in a pheromone matrix only, PACO stores solutions in a population and then uses this population to make adjustments to the pheromone matrix. At any time the pheromone matrix will be a direct reflection of some or all of the stored population.

In the single-objective PACO algorithm as solutions enter the population a positive update on the pheromone matrix is

performed, and as solutions leave the population a negative update is performed to adjust the pheromone matrix values. This process removes the requirement for a traditional ACO decay operation and results in a significant speed improvement over traditional pheromone maintenance operations [15]. PACO still uses the traditional ACS greedy transition rule [9] to construct new solutions. Figure 1, taken from [1], provides a visual summary of the differences between a traditional ACO algorithm and a Population-based ACO algorithm using terminology defined in [1], [4], [8].

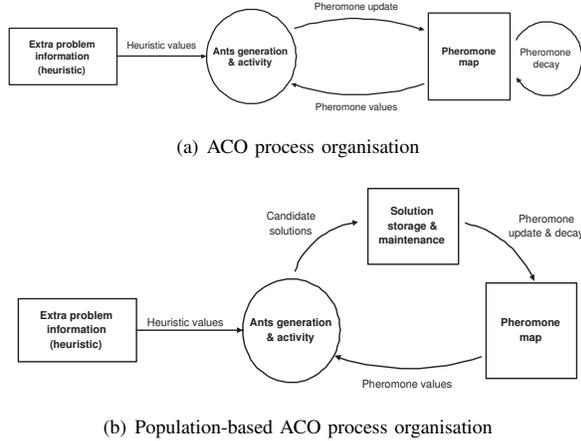


Fig. 1. Process organisation of a traditional ACO algorithm versus a population-based ACO algorithm (taken from [1])

B. PACO for Multiple Objective Optimisation

When applied to multi-objective problems PACO maintains a different pheromone matrix for each objective. For each iteration of the algorithm, where iteration refers to every artificial ant creating a complete solution, a random ant is selected from the population (Q) along with its k closest neighbours¹ to form a sub-population P . At any time Q will contain the complete set of non-dominated solutions found to date. The ants in P are then used to update the individual pheromone matrix for each objective. When available a separate heuristic matrix is used for each objective; in the case of the TSP these heuristic matrices are simply the corresponding edge weights for each individually defined TSP as is the case for most ACO algorithms applied to the TSP.

PACO uses an *average-rank-weight method* to weight the importance of each objective. These weightings (w) are used to bias the solution construction towards satisfying specific objectives over others. Briefly, the *average-rank-weight method* measures how well each solution in P satisfies each individual objective. Objectives which are better satisfied by the solutions in P relative to the entire population Q are given a higher rank and a subsequently larger weighting.

¹Closest neighbour refers to a match in the decision space, not the objective value space. This distance is usually obtained by taking a Euclidean or Hamming distance measure.

Once the pheromone matrices have been created and the objective weightings defined the transition probabilities are calculated using (1), where h is the total number of objectives. The Ant Colony Systems greedy transition rule [9] is then used to create one or more new solutions.

$$p_{ij} = \sum_{d=1}^h \left(w_d \cdot \frac{[\tau_{ij}^d]^\alpha \cdot [\eta_{ij}^d]^\beta}{\sum_{l \in N_i^k} [\tau_{il}^d]^\alpha \cdot [\eta_{il}^d]^\beta} \right) \quad (1)$$

Once created, each new solution (s) is evaluated for each objective. For s to be inserted into Q it must be checked for dominance against the entire population Q . If s is found to be non-dominated by all members of Q , meaning that no solution in Q is better in all objectives than s , then s is inserted into Q . If s is inserted into Q then Q must be checked for dominance by s . If any existing solutions in Q are dominated by s they are removed from Q .

III. CROWDING PACO

The Crowding Population-based ACO algorithm (CPACO) was first defined in [2] where it was applied to a small set of TSP and multi-modal function optimisation problems. In that paper the algorithm was shown to be able to locate and maintain a population of diverse solutions across multiple optimal and near-optimal locations of the search spaces tested.

Rather than use the super/sub-population scheme as in PACO or one of the population replacement schemes defined in [13], [14] CPACO uses a crowding replacement scheme [6], [19] similar to restricted tournament selection [17]. With this scheme a single population (S) of preset size is maintained, which is initialised with randomly generated solutions. Every generation a population of new solutions is created (Y) and each new solution is compared against a randomly selected subset S' of S to find its closest match (in the case of the TSP the closest match is determined by the solution with the largest number of common edges) and only replaces the existing solution if the new solution is better (in the case of multi-objective problems better is taken to mean strongly-dominating). This procedure is outlined in Alg. 1.

Algorithm 1 CPACO Crowding Replacement Scheme

- 1: **for** $i = 1$ to Y_{size} **do**
- 2: $S' = c$ randomly chosen solutions from S
- 3: $S'_j =$ closest match from S' to Y_i
- 4: **if** $Y_i \succ S'_j$ **then**
- 5: Replace S'_j with Y_i
- 6: **end if**
- 7: **end for**
- 8: Discard Y

Whereas PACO uses individual pheromone and heuristic matrices for each objective, CPACO uses a single pheromone matrix with individual heuristic matrices. Each iteration a new pheromone matrix is calculated as follows:

- 1) All solutions (s) in the population (S) are assigned an integer rank according to the dominance ranking

procedure used by the NSGA-II algorithm [5]. Figure 2 gives an example of the resultant solution ranks assigned using this ranking procedure.

- 2) All elements in the pheromone matrix are initialised to some initial value (τ_{init}).
- 3) All solutions in the population increment their corresponding elements in the pheromone matrix according to the inverse of their rank, i.e. $\Delta\tau_{ij}^s = 1/(s_{rank})$.

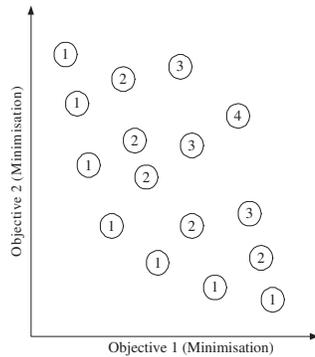


Fig. 2. Example of the resultant ranks assigned to a set of solutions using the NSGA-II dominance ranking procedure

Rather than use the PACO *average-rank-weight method* to determine weightings for each objective, CPACO assigns each ant a set of unique heuristic exponent correction factors (λ), similar to the procedure outlined in [3]. This allows each unique ant to exploit heuristic matrices by different amounts while still using a common pheromone matrix. The procedure used to assign these values is outlined in Alg. 2 and the transition probabilities are calculated using (2), where h defines the number of objectives.

Algorithm 2 CPACO Heuristic Scaling Value Assignment Procedure

- 1: **for** $i = 1$ to h **do**
 - 2: $R_i = \text{random}[0, 1]$
 - 3: **end for**
 - 4: Sort R in ascending order
 - 5: $\lambda_1 = R_1$
 - 6: **for** $i = 2$ to $h - 1$ **do**
 - 7: $\lambda_i = R_i - R_{i-1}$
 - 8: **end for**
 - 9: $\lambda_h = 1 - R_h$
-

$$p_{ij} = \frac{[\tau_{ij}]^\alpha \cdot \prod_{d=1}^h [\eta_{ij}^d]^{\lambda_d \beta}}{\sum_{l \in N_i^k} [\tau_{il}]^\alpha \cdot \prod_{d=1}^h [\eta_{il}^d]^{\lambda_d \beta}} \quad (2)$$

IV. TEST SETUP

The ‘Kro’ set of Travelling Salesman Problems have been selected to test the CPACO algorithm. This set consists of a

series of eight different 100, 150 and 200 city TSP which are combined to create four bi-objective TSPs of varying complexity and a single quad-objective TSP. The individual instances are labelled KroA100, KroB100, KroC100, KroD100, KroA150, KroB150, KroA200 and KroB200 and are available from [21]. Each of the multi-objective TSPs contains a discontinuous, convex Pareto front.

Two metrics have been selected to measure and compare the performance of the PACO and CPACO algorithms: summary attainment surface analysis and dominance ranking. These metrics have been chosen since together they take into consideration an algorithm’s ability to not only find solutions close to the Pareto front, but also its ability to obtain a diverse range of solutions along the Pareto front.

A. Summary Attainment Surface

The summary attainment surface plotting method [18], an extension of the work of [11], was developed to determine the median performance of a stochastic multi-objective optimisation algorithm over several experimental runs. The summary attainment surface demonstrates a given algorithm’s ability to find solutions close to and distributed across the Pareto front. All summary attainment surfaces produced in this study have been created with the aid of Knowles’ software package available from http://dbkgroup.org/knowles/plot_attainments/.

B. Dominance Ranking (C metric)

Dominance Ranking is based on two algorithms’ non-dominated sets being compared to determine if any solutions from one algorithm dominate solutions from the other. The premise is that if, at an extreme, one algorithm’s non-dominated set completely dominates another algorithms non-dominated set ($C=1$) then the first algorithm can be said to be better. This is reflected in the Coverage metric (C metric) [22].

V. RESULTS & ANALYSIS

A. Attainment Surface & Coverage Measures

Both PACO and CPACO were run 50 times and allowed 50,000 solution evaluations on each bi-objective problem using the parameter settings given in Tab. III. The 1% (best) and 50% (average) attainment surfaces for each of these problems are presented as Figs. 3, 4, 5 & 9 and Figs. 6, 7, 8 & 12 respectively. These figures indicate that for all bi-objective problems tested CPACO was able to obtain better mean and absolute attainment surfaces than PACO. Attainment surfaces are also included as Figs. 10, 11, 13, 14 for CPACO applied to the quad-objective problem, kroABCD100, the results of which are discussed in Sec. V-D.

Table I contains the calculated C metrics for the CPACO and PACO algorithms. These figures indicate that for all bi-objective problems tested CPACO obtained a better non-dominated set than PACO.

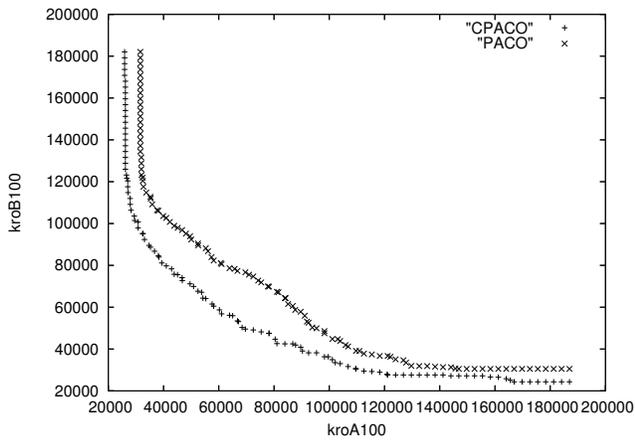


Fig. 3. 1% (best) attainment surface for kroA100 and kroB100 using PACO & CPACO

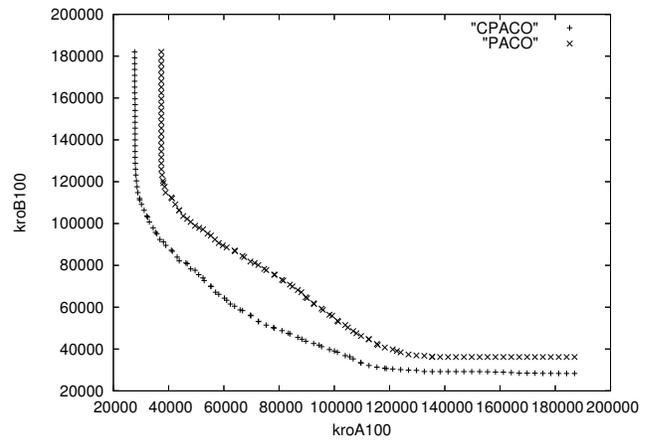


Fig. 6. 50% (average) attainment surface for kroA100 and kroB100 using PACO & CPACO

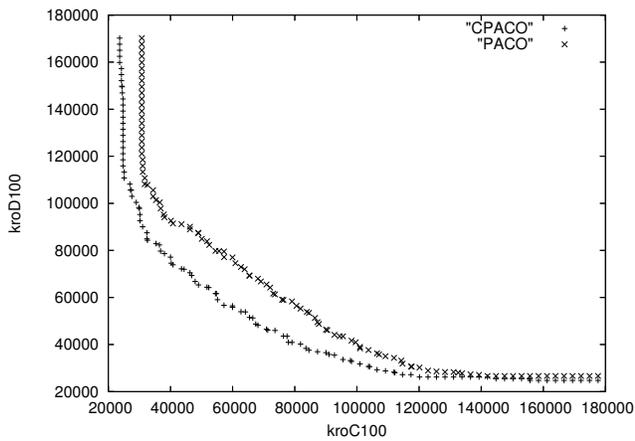


Fig. 4. 1% (best) attainment surface for kroC100 and kroD100 using PACO & CPACO

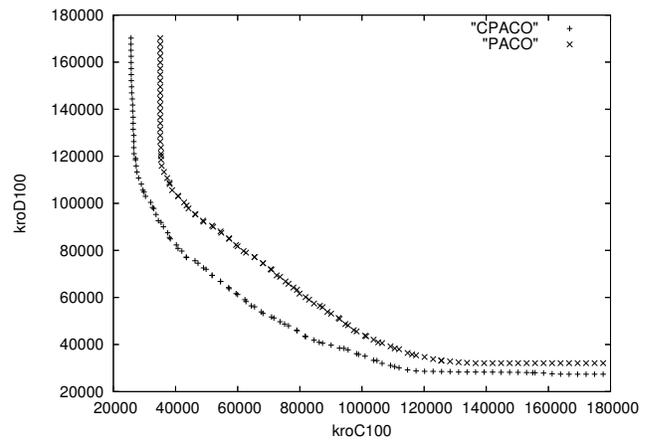


Fig. 7. 50% (average) attainment surface for kroC100 and kroD100 using PACO & CPACO

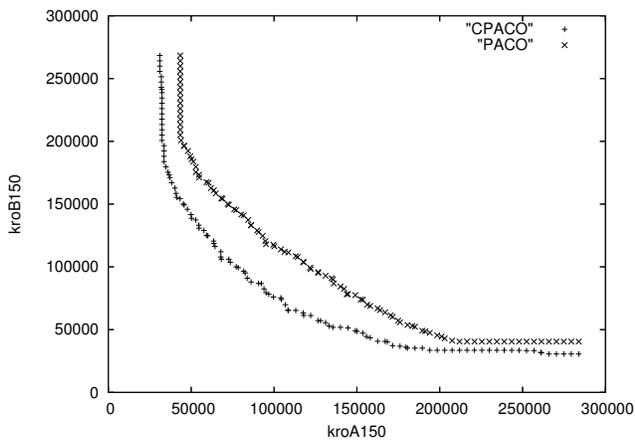


Fig. 5. 1% (best) attainment surface for kroA150 and kroB150 using PACO & CPACO

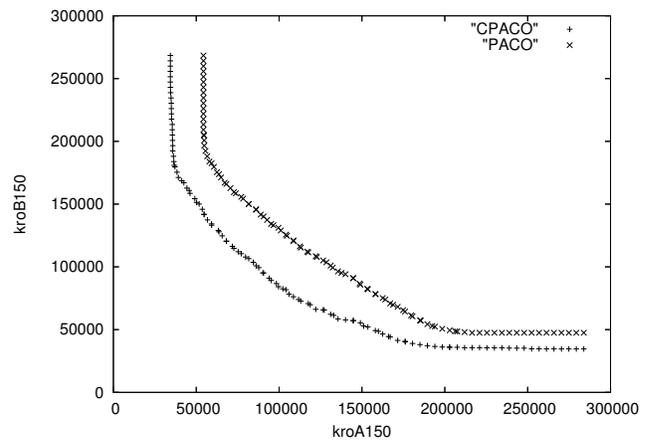


Fig. 8. 50% (average) attainment surface for kroA150 and kroB150 using PACO & CPACO

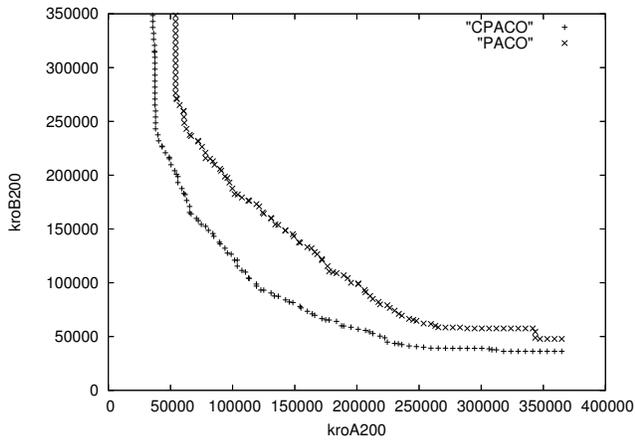


Fig. 9. 1% (best) attainment surface for kroA200 and kroB200 using PACO & CPACO

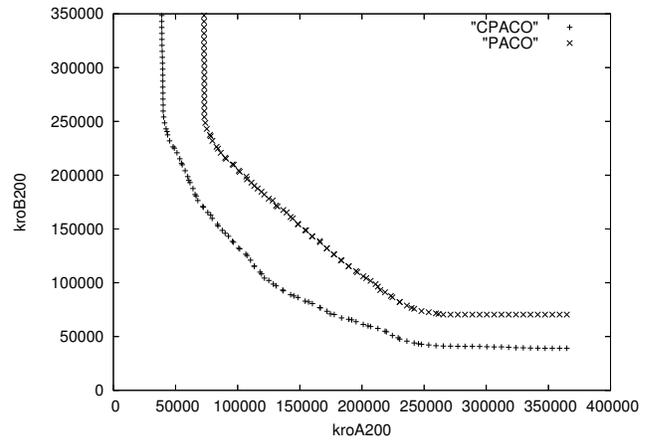


Fig. 12. 50% (average) attainment surface for kroA200 and kroB200 using PACO & CPACO

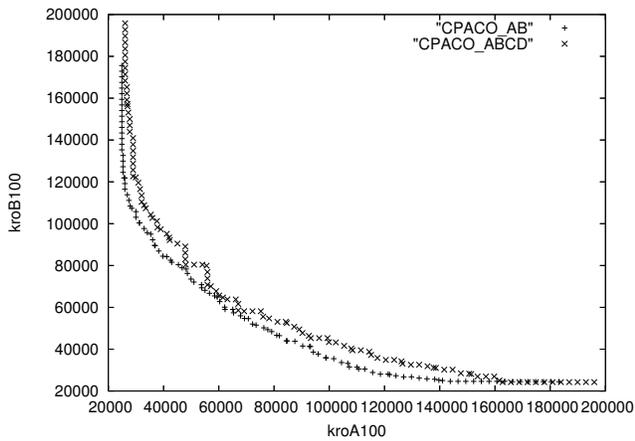


Fig. 10. 1% (best) attainment surface for CPACO applied to kroA100, kroB100, kroC100 & kroD100 (*CPACO_ABCD*), and CPACO applied to only kroA100 & kroB100 (*CPACO_AB*)

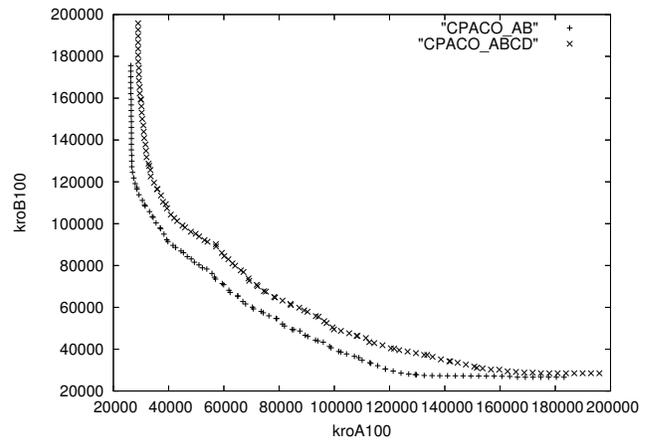


Fig. 13. 50% (average) attainment surface for CPACO applied to kroA100, kroB100, kroC100 & kroD100 (*CPACO_ABCD*), and CPACO applied to only kroA100 & kroB100 (*CPACO_AB*)

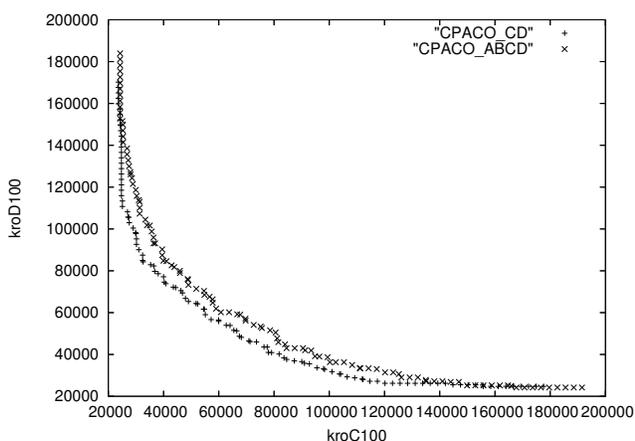


Fig. 11. 1% (best) attainment surface for CPACO applied to kroA100, kroB100, kroC100 & kroD100 (*CPACO_ABCD*), and CPACO applied to only kroC100 & kroD100 (*CPACO_CD*)

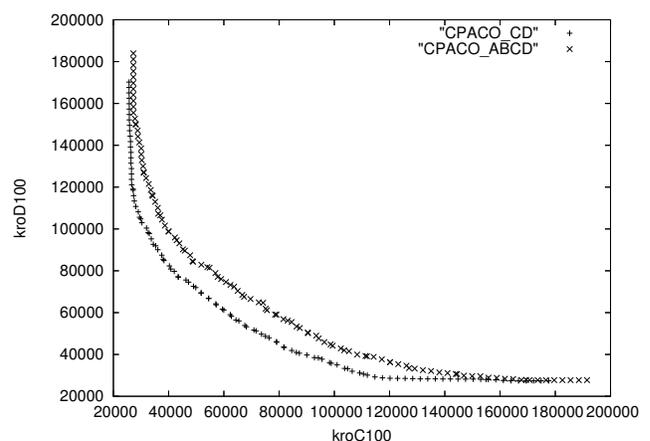


Fig. 14. 50% (average) attainment surface for CPACO applied to kroA100, kroB100, kroC100 & kroD100 (*CPACO_ABCD*), and CPACO applied to only kroC100 & kroD100 (*CPACO_CD*)

TABLE I
TABLE OF RESULTS FOR C METRIC (COVERAGE)

Problem	PACO covers CPACO	CPACO covers PACO
KroA100/KroB100	0	1
KroC100/KroD100	0	1
KroA150/KroB150	0	1
KroA200/KroB200	0	1

B. General Observations

It is speculated that the PACO algorithm’s lack of diversity preservation, combined with a greedy selection strategy and a strongly biased pheromone matrix (PACO only uses a minimal subset to construct the pheromone matrix) most likely leads to premature convergence in suboptimal areas of the search space. While the strongly biased pheromone matrix allows the algorithm to improve existing solutions towards the Pareto front through small perturbations, it may be making it too difficult for the algorithm to construct solutions which are vastly different from those already in the population (Fig. 15).

The CPACO algorithm is able to locate and maintain a diverse set of solutions across the Pareto front which may also contribute to the algorithm’s ability to find better solutions along all areas of this front (Fig. 16). There may be some wasted effort in CPACO since it constructs solutions based on a pheromone matrix which reflects the performance of the entire population, regardless of a solution’s position on the approximate Pareto front; however this, combined with the greedy transition rule, makes for a good balance between exploration and exploitation. It is worth noting that the ranking of solutions using the NSGA-II ranking procedure is a vast improvement over early attempts which did not use any ranking and allowed solutions to update the pheromone matrix uniformly.

C. Computational Efficiency

The CPACO algorithm is comparably, if not less, computationally complex than the PACO algorithm. CPACO reconstructs the pheromone matrix every iteration like PACO reconstructs its pheromone matrices (one per objective) and, although CPACO includes more information in the matrix since it uses the entire population rather than a subset, this is offset by the fact that CPACO constructs many more solutions from the pheromone matrix per iteration than the PACO algorithm. PACO also requires the identification of k closest neighbours, an operation which has a worst case complexity of $O(N^2)$ where N is the population size. Using the parameters from this study for the KroA100KroB100 test problem:

- CPACO constructs 50 solutions per matrix which is composed of 50 solutions
- PACO constructs 1 solution per set of matrices which are composed of 6 solutions

Using these parameters PACO performs pheromone matrix modifications, on twice the number of pheromone matrices, six times more frequently than CPACO.

PACO does not require dominance ranking every iteration like CPACO since all solutions are assigned a uniform

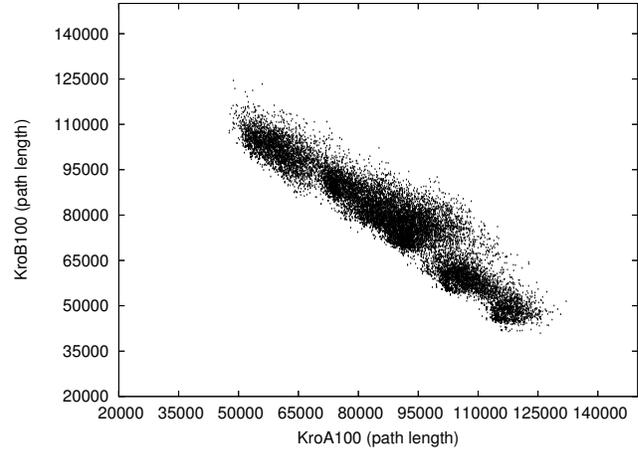


Fig. 15. PACO sampling behaviour for one complete experimental run (20,000 evaluations) of the algorithm on the KroA100KroB100 problem. Each point indicates an evaluated solution.

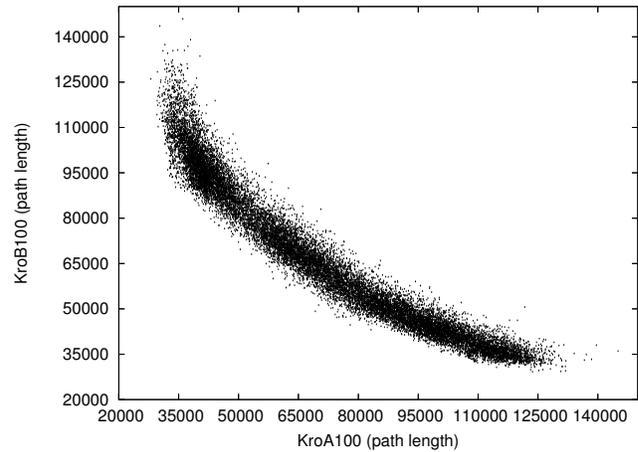


Fig. 16. CPACO sampling behaviour for one complete experimental run (20,000 evaluations) of the algorithm on the KroA100KroB100 problem. Each point indicates an evaluated solution.

pheromone update value. However, if the NSGA-II ranking procedure is used the worst case complexity of this ranking procedure is reduced from $O(hN^3)$ to $O(hN^2)$ where h is the number of objectives. PACO does use the average-rank-weight method to assign ranks to solutions which is of approximate worst-case complexity $O(hN^2)$, which is comparable.

When checking whether to insert a new solution into the population PACO performs a non-dominance check of worst-case complexity $O(hN)$. If the new solution is non-dominated by the population then it is inserted and the population is checked for non-dominance by the new solution with a worst case complexity of $O(hN^2)$. CPACO selects a subset (in this case 1/5 of N) of the population and uses a crowding comparison to find the closest subset member to the new solution and then performs a single non-dominance check, of total complexity $O((N/5)^2 + h)$.

The observed average final population size during experimentation was larger for PACO than the static population size

used for CPACO (Tab. II). Even though these figures indicate the average final population size, it was observed that the population size grew rapidly at the start of algorithm execution before fluctuating about the final recorded level.

TABLE II
AVERAGE FINAL POPULATION SIZE OF PACO VERSUS STATIC POPULATION SIZE OF CPACO FOR ALL TEST PROBLEMS

Problem	PACO final pop size (avg)	CPACO pop size
KroA100/KroB100	105	50
KroA150/KroB150	135	75
KroA200/KroB200	148	100

To summarize, both PACO and CPACO require the use of a distance sorting routine (albeit for different purposes), although CPACO does so on a subset of the population reducing the computation considerably. PACO maintains a larger population than CPACO, and performs approximately six times as many pheromone updates as CPACO (across multiple pheromone matrices). CPACO uses the non-dominance sorting routine to assign ranks to solutions, while PACO uses the average-rank-weight method to assign objective rankings (both of similar complexity). PACO requires non-dominance checking of the entire population per insertion of a new solution while CPACO only performs one non-dominance check per insertion. PACO maintains a larger population which is on average twice the size of CPACO. PACO thus appears to be a more computationally expensive algorithm than CPACO.

D. Quad-objective Optimisation

CPACO was tested on the KroABCD100 quad-objective problem to examine its performance on a larger than bi-objective problem. The results of the trials were analysed using attainment surfaces by isolating two objectives at a time (kroA100/kroB100 and kroC100/kroD100), even though the problem was attempted simultaneously on all four unique objectives. The attainment surfaces generated from the original bi-objective cases (kroA100/kroB100 and kroC100/kroD100) were used as comparisons against the quad-objective case isolated to two objectives. The ideal outcome of the analysis would be if little to no difference is observed between the previously obtained attainment surfaces and the quad-objective case analysed in two objectives.

The 1% (best) and 50% (average) attainment surfaces are included as Figs. 10, 11 and Figs. 13, 14 respectively. For this experiment the population size and number of ants were both increased to 200, the crowding replacement size was 40, and the number of iterations was increased from 50,000 to 100,000. All other parameters remain the same as stated in the Appendix, results are recorded from 50 repeats from random starting positions.

When isolating the quad-objective results in both the kroAB100 and kroCD100 objectives the original bi-objective attainment surfaces are better. This is not unexpected given that the quad-objective case is a much more difficult problem due to the added complexity of two extra objectives. It is interesting that the major difference occurs in the mid-point of

the approximate Pareto front which suggests that CPACO may not be making best use of its historic (pheromone) information. It may be that a scaling factor needs to be introduced to the pheromone update to strongly bias the top ranking solutions or that an approach similar to PACO where only a subset of the population is used to create the pheromone map needs to be introduced.

VI. CONCLUSION AND FUTURE WORK

This study has presented the CPACO algorithm for the multi-objective TSP. For the bi-objective test problems used CPACO was able to outperform the PACO algorithm based on the average attainment surface and C metrics. An added advantage is that while CPACO produced better results, it achieved them with approximately lower computational complexity than PACO.

CPACO was then applied to a quad-objective TSP to determine how well it scales as extra objectives are included. The results of this experiment, while impressive, were not quite as good as the bi-objective cases tried. The pheromone update and use of historic information being likely areas for future research to address this short-coming.

Given the promising results obtained a host of future work is intended for the CPACO algorithm including, but not limited to:

- Application to different classes of multi-objective problems, including ‘real-world’ instances.
- Combination of CPACO with local-search. In [20] good results were obtained on multi-objective TSP instances using a two phase local search. CPACO combined with a local search may allow CPACO to seed the local search with many good, diverse starting locations.
- Making CPACO more scalable to larger numbers of objectives.

REFERENCES

- [1] D. Angus. Niching for ant colony optimization. Technical report, Faculty of Information and Communication Technology, Swinburne University of Technology, 2006. Available from: <http://www.it.swin.edu.au/personal/dangus>.
- [2] D. Angus. Niching for Population-based Ant Colony Optimization. In *2nd International IEEE Conference on e-Science and Grid Computing, Workshop on Biologically-inspired Optimisation Methods for Parallel and Distributed Architectures: Algorithms, Systems and Applications*, 2006.
- [3] B. Barán and M. Schaerer. A multiobjective ant colony system for vehicle routing problem with time windows. In *Proceedings of the 21st IASTED International Conference*, Innsbruck, Austria, February 2003.
- [4] 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.
- [5] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature – PPSN VI*, pages 849–858, Berlin, 2000. Springer.
- [6] K. A. DeJong. *An analysis of the behaviour of a class of genetic adaptive systems*. PhD thesis, University of Michigan, 1975.
- [7] M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Italy, 1992.
- [8] M. Dorigo, E. Bonabeau, and G. Theraulaz. Ant algorithms and stigmergy. *Future Generation Computer Systems*, 16:851–871, 2000.

[9] 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.

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

[11] C. M. Fonseca and P. J. Fleming. On the performance assessment and comparison of stochastic multiobjective optimizers. In *PPSN IV: Proceedings of the 4th International Conference on Parallel Problem Solving from Nature*, pages 584–593, London, UK, 1996. Springer-Verlag.

[12] C. García-Martínez, O. Cordón, and F. Herrera. A Taxonomy and an Empirical Anlisis of Multiple Objective Ant Colony Optimization Algorithms for Bi-criteria TSP. *European Journal of Operational Research*, 2006. in press.

[13] M. Guntsch. *Ant Algorithms in Stochastic and Multi-Criteria Environments*. PhD thesis, Universität Fridericiana zu Karlsruhe, 2004.

[14] 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.

[15] 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.

[16] M. Guntsch and M. Middendorf. Solving Multi-criteria Optimization Problems with Population-Based ACO. In G. Goos, J. Hartmanis, and J. van Leeuwen, editors, *Proceedings of the Second International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003)*, volume 2632 of *LNCS*, pages 464–478. Springer, April 2003.

[17] G. Harik. Finding multiple solutions in problems of bounded difficulty. Technical Report 94002, Illinois Genetic Algorithms Laboratory (IlligAL), University of Illinois, 1994.

[18] J. Knowles. A summary-attainment-surface plotting method for visualizing the performance of stochastic multiobjective optimizers. In *ISDA '05: Proceedings of the 5th International Conference on Intelligent Systems Design and Applications*, pages 552–557, Washington, DC, USA, 2005. IEEE Computer Society.

[19] S. W. Mahfoud. *Niching methods for genetic algorithms*. PhD thesis, University of Illinois, 1995.

[20] L. Paquete and T. Stützle. A Two-Phase Local Search for the Biobjective Traveling Salesman Problem. In G. Goos, J. Hartmanis, and J. van Leeuwen, editors, *Evolutionary Multi-Criterion Optimization: Second International Conference (EMO 2003)*, volume 2932 of *LNCS*, pages 479–493. Springer, 2003.

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

[22] E. Zitzler and L. Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

VII. ACKNOWLEDGEMENT

The author acknowledges the guidance Prof. Tim Hendtlass has provided as well as the rest of the team at CIS.

VIII. APPENDIX

TABLE III
ALGORITHM PARAMETER SETTINGS

Algorithm	α	β	No. of ants	k	τ_{max}	q_0
PACO	1	3	1	5	1	0.9
CPACO	1	3	n/2	NA	NA	0.9

Algorithm	τ_{init}	History size (N)	Crowding amount (c)
PACO	$1/(n-1)$	NA	NA
CPACO	$1/(n-1)$	n/2	n/10