

Niching for Ant Colony Optimisation

Daniel Angus

Abstract Evolutionary Computation niching methods, such as Fitness Sharing and Crowding, are aimed at simultaneously locating and maintaining multiple optima to increase search robustness, typically in multi-modal function optimization. Such methods have been shown to be useful for both single and multiple objective optimisation problems. Niching methods have been adapted in recent years for other optimisation paradigms such as Particle Swarm Optimisation and Ant Colony Optimisation. This paper discusses niching techniques for Ant Colony Optimisation. Two niching Ant Colony Optimisation algorithms are introduced and an empirical analysis and critical evaluation of these techniques presented for a suite of single and multiple objective optimisation problems.

1 Introduction

In natural ecologies, a population of organisms is rarely spread uniformly (within its 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 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 [18, 37].

In a computational sense, niching may permit a more effective use of available resources by a search algorithm by either implicitly or explicitly dividing

The University of Queensland
Brisbane
Queensland
Australia
d.angus@uq.edu.au

and searching different areas of the search space in parallel [31]. Such automatic resource re-allocation is useful in preserving population diversity for an extended search time. Niching techniques have proved particularly useful for problems that require multiple solutions to be located for a search algorithm, such as multi-modal and multi-objective optimisation problems.

Niching as an Evolutionary Computation (EC) concept was first formally applied to the Genetic Algorithm (GA), but has also been applied to other EC algorithms such as Particle Swarm Optimisation (PSO) [5, 6]. Some of the better known niching methods include Crowding [29, 30], Fitness Sharing [20, 21] and Clearing [33].

Previous work has applied Fitness Sharing and Crowding to Ant Colony Optimisation (ACO) [1, 2, 3, 4], all with promising results. This chapter discusses these works with a review of a variety of applications to single and multiple objective problem domains. The chapter is organised into three main sections which discuss, in order, the fundamentals of niching, an introduction to ACO and how to apply niching principles to ACO, and finally an empirical and theoretical analysis of several niching ACO algorithms.

2 Niching

Solution diversity is often mentioned as an important factor in population-based algorithm design. For some problems diversity may not be required to obtain optimal solutions while in others it is critical. In this section a particular diversity preservation technique called *niching* is introduced. Some historical aspects of niching in EC are introduced and examples of two specific niching techniques are presented.

2.1 *Niching in Evolutionary Computation*

The canonical GA tends to focus its search in an individual area of the search space. If multiple near optimal solutions are sought then such algorithms may prove ineffective in achieving this goal. Niching aims to diversify the search focus of an EC algorithm to multiple areas of the search space. In Goldberg [20] three key features of niching algorithms are presented:

1. Stable maintenance of subpopulations. Once an optimal area of the search space has been located a niching algorithm should maintain several population members in that location so as not to lose (forget) this area of interest.
2. The size of a subpopulation decreases according to the fitness of the area of interest. Considering the limited resources of an algorithm, exploration

of any area of the search space should be proportional to the potential quality of solutions returned from a niche.

3. Subpopulations should not compete. Since resources are most likely to be of a fixed size, a niching algorithm should not allow subpopulations to ‘fight’ for dominance of one area of the search space.

2.2 Crowding: Modifying the Replacement Mechanism

The *Crowding Factor* model was introduced as a diversity maintenance scheme by De Jong [11]. This scheme was not a niching method according to the criteria outlined in Sec. 2.1, as it was used to increase population diversity, not to locate and maintain multiple optimal areas of the search space [26]. However, the Crowding Factor model laid the foundation for much of the later work on niching and was most significantly reworked as a niching strategy by Mahfoud [29, 30].

Mahfoud’s technique, *Deterministic Crowding*, 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 population with the most similar of its parent solutions. If a candidate solution is better than its most similar parent, the parent is replaced, otherwise the candidate solution is discarded.

Another variation of Crowding is *Restricted Tournament Selection* (RTS) [26]. RTS is similar to Deterministic Crowding, but instead of a candidate solution only being compared against its parents, it is compared to a subset of the entire population. The size of this subset is called the window size and the most similar solution from this subset is sought and is replaced if the candidate solution is of higher fitness.

2.3 Fitness Sharing: Modifying the Selection Mechanism

Instead of modifying the replacement mechanism to introduce niching behaviour, as in Crowding, Fitness Sharing [21, 20] modifies the selection mechanism. Since most Genetic Algorithms use a fitness proportionate selection mechanism to select parent solutions, Goldberg and Richardson [21] found that modifying the selection of parents will avoid convergence to one area of the search space thereby introducing niche formation.

Fitness Sharing derates¹ solutions which occupy the same or a similar position in the search space. For example the adjusted quality (Q') of two solutions which occupy the same position in the search space will be half of their original quality (Q). The result of this adjustment is to remove the selection bias present through a solution being represented multiple times. The removal of this bias allows simultaneous convergence to multiple areas of the search space.

Niche formation, specifically with regard to Fitness Sharing, was explained by Goldberg and Richardson [21] using a variation of the k -armed bandit problem [27, 11, 38]. The problem involves a poker machine with k handles, each handle having set pay-out odds. There is also a population of gamblers, who wish to maximise their individual winnings from the poker machine. The variation introduced is that after every gambler has selected a handle they must share their winnings with everyone else who chose their particular handle. For example if there are two handles each having expected payouts of \$25 and \$75 and 100 gamblers all pull the better handle they would each receive $\$75/100 = \0.75 . If however the population divides across both handles proportionate to the expected payout of those handles the expected payout per gambler will be $\$75/75 = \$25/25 = \$1.00$. From this simple experiment it was shown that modifying the payout function in the k -armed bandit problem can introduce a reward for niche formation.

2.4 *Advantages and Disadvantages of Niching*

Most search algorithms are designed (or were initially designed) to find a single optimal solution to a difficult problem. This design trait is borne of the problems that exist in the literature where the goal tends to be optimisation of a single objective. Niching algorithms tend to be best applied in situations where multiple optimal solutions are required rather than a single optimum, such as in multi-modal and multiple objective problems.

Alternatively, niching algorithms can also be applied in situations where a single optimum solution is desired. This may be because some search advantage is gained over non-niching algorithms due to the specific search landscape of the problem. Niching algorithms in this sense 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 [31]. Such automatic resource re-allocation is useful in preserving useful population diversity for an extended search time.

Niching can also be advantageous if applied in a dynamic optimisation problem. In Schoeman and Engelbrecht [39] a PSO algorithm was adapted with a niching operator for application to a series of dynamic multi-modal

¹ Derate, a commonly used term in niching literature means 'to decrease the fitness of'.

function optimisation problems. The results showed that with moderate changes to the problem under study the algorithm was able to track peaks by maintaining stable niches around them. The algorithm was able to perform good resource reallocation as the peaks were removed, which is good for the long-term applicability of such a technique.

Since niching techniques spread the population across a wider search area, they can often waste computation by continually searching areas of the search space where no interesting optima exist. Also recombination of solution components from different niches can lead to the introduction of ‘lethals’ [8] which are solutions created between two optima but not located on an optima themselves. Some niching techniques introduce extra parameters in addition to the base algorithm. These parameters often come without good heuristics to set them and thus require extra sensitivity analysis to ensure best performance [43].

3 Niching for Ant Colony Optimisation

To be able to easily implement standard niching methods such as Crowding and Fitness Sharing with ACO, the ability to readily measure the distance (or difference) between solutions is required. Since most standard ACO algorithms encode solution quality information into a pheromone matrix without storing the actual solutions, access to required distance information between multiple generations of solutions is impractical. While it may be possible to achieve a niche-like behaviour in ACO, it is much more straightforward to use Population-based ACO to implement standard niching techniques such as Crowding and Fitness Sharing, since in PACO, access to a multi-generational population in the traditional EC sense is guaranteed. In this section an introduction to ACO and Population-based ACO is offered and two previously introduced Population-based ACO algorithms (Crowding PACO and Fitness Sharing PACO) imbued with niching behaviour [1] are discussed.

3.1 *Ant Colony Optimisation*

Ant Colony Optimisation (ACO) [12, 15], is an optimisation methodology based on the foraging behaviour of Argentine ants. All ACO algorithms are responsible for the scheduling of three processes:

- Ants generation & activity
- Pheromone trail evaporation
- Daemon actions

Each of the processes listed allow for flexibility in their implementation, as a result many different ACO algorithms have been proposed in recent years. Notable examples include Ant Systems [16], Ant Colony Systems [14] and $MAX - MIN$ Ant Systems [41]. A visual representation of the process organisation of ACO is provided as Fig. 1.

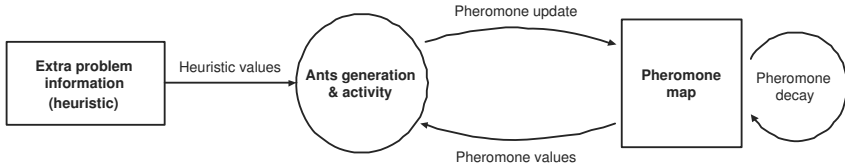


Fig. 1: Process organisation of the Ant Colony Optimisation Metaheuristic Framework

3.1.1 Pheromone Mapping

The pheromone mapping is the means by which solution components are able to be ranked and selected based on past usefulness. The pheromone mapping connects pheromone values from a pheromone map (usually a matrix structure) to specific solution components. The assumption usually being that if a prior solution is good then at least some of its parts (solution components) should also be good, and therefore a remixing of these components with other good components may lead to an optimal or near-optimal solution. A first step in defining an ACO algorithm is to define this pheromone mapping.

The problem domain will dictate how the pheromone mapping should be defined. In applying an ACO algorithm to a combinatorial optimisation problem such as the travelling salesman problem (TSP) it is not of interest which specific components are included, as any feasible solution will include every city once (and only once), it is the order of these components which is important in finding an optimal solution. For the TSP the transition points (edges/arcs) between the specific components can be assigned a specific pheromone value to reflect which order of cities is best. If a solution includes an edge connecting city a to city b and this solution is good then this goodness should be reflected by a higher pheromone level on this specific edge and other edges included in this solution.

3.1.2 Ants Generation and Activity

This process is responsible for the creation of new candidate solutions to the optimisation problem being addressed by the ACO algorithm, as well as the updating of pheromone values via the pheromone mapping. Through the execution of this process pheromone information is updated (increased or decreased depending on the implementation). A temporary population of (artificial) ants is used to construct feasible solutions to the problem being addressed. Each ant is evaluated upon the completion of a feasible solution and the solution information encoded through the pheromone mapping. After this encoding each individual ant is discarded and a new ‘empty’ ant is created in its place. This process is repeated until some stopping criterion is met.

An ant has the following properties:

- An ant searches for a minimum (or maximum) cost solution to the optimisation problem being addressed.
- Each ant has a memory used to store all solution components used to date, so that the candidate solution can be evaluated at the completion of solution construction; the memory can be used as a tabu list such as in the case of the TSP so that no component is reused.
- An ant can be assigned a starting position, for example an initial city in a TSP.
- An ant can include any feasible solution component (an example of a feasible solution component in a TSP would be a city which has not already been included in the candidate solution) until such time that no feasible components exist or a termination criterion is met (usually correlating to the completion of a candidate solution).
- Ants include solution components according to a combination of a pheromone value and a heuristic value which are associated with every solution component in the problem, the choice of which solution component is usually a probabilistic one.
- When including a new solution component in the growing candidate solution the pheromone value associated with the transition between these components (arc/edge in a TSP), or the solution component itself can be altered (*online step-by-step pheromone update*).
- An ant can retrace a candidate solution at the completion of a solution, updating the pheromone values of all transitions and/or solution components used in the solution (*online delayed pheromone update*).
- Once a candidate solution is created, and after completing online delayed pheromone update (if required) an ant dies, freeing all allocated resources.

3.1.3 Pheromone Trail Evaporation

Like the biological ant colony, the artificial ant colony employs a pheromone evaporation mechanism. This mechanism serves as a useful way of ‘forgetting’

older search bias [13]. As ACO algorithms use positive reinforcement, if pheromone was allowed to accumulate without decay the system would very quickly converge on a single solution since this solution would continue to be reinforced. This evaporation process can be thought of as a global pheromone update (it decreases all pheromone values by a set percentage), different to the previously mentioned local pheromone update process that increases or decreases specific pheromone values.

3.1.4 Daemon Actions

Daemon actions can be used to perform specialised functions which often require more knowledge than an individual ant is allowed [13, 17]. For example, a daemon action could inspect all solutions generated in one search cycle, identify the best solution and increment the pheromone values of its solution components more than the regular pheromone update (*offline pheromone update*). An alternative daemon action could be the application of a local search procedure.

3.2 Population-Based Ant Colony Optimisation

The Population-based Ant Colony Optimisation (PACO) [23, 24] algorithm was first introduced as a single objective optimisation algorithm designed for dynamic optimisation problems. PACO was later extended for a multiple objective optimisation problem, the single machine total tardiness problem with changeover costs [22, 25]. The defining difference between PACO and the canonical ACO algorithm is in the area of solution storage. Whereas most traditional ACO algorithms (e.g., Ant Systems [16], Ant Colony Systems [14], $MAX-MIN$ Ant Systems [41]) 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 [24]. PACO still uses the traditional ACS greedy transition rule [14] to construct new solutions. Figure 2, taken from Angus [4], provides a visual summary of the Population-based ACO algorithm using similar terminology to that defined in Dorigo et al. [17] and Cordon et al. [7], this can be contrasted with ACO as presented in Fig. 1.

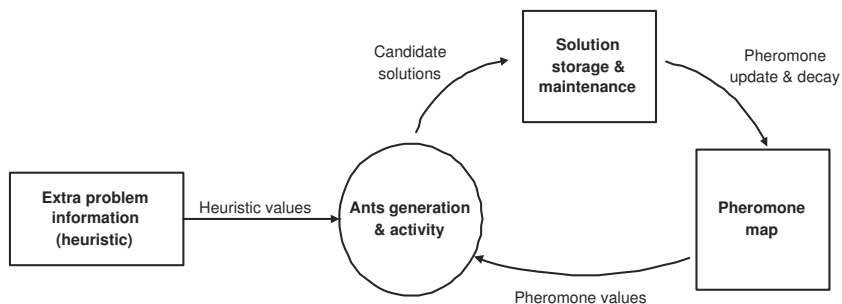


Fig. 2: Population-based ACO process organisation (reproduced from Angus [4])

3.3 Niching Ant Colony Optimisation Algorithms

As mentioned, at time of writing two examples of Niching ACO algorithms can be found in the literature. These algorithms both extend the PACO algorithm and are presented in this section.

3.3.1 Crowding PACO

In ACO algorithms the pheromone matrix tends to reflect the experience of the entire population, and unlike the GA, PACO algorithms do not select parent solutions for new solution construction. Rather, new individuals gain experience via the pheromone matrix as they progress towards a solution. Given this, Restricted Tournament Selection [26] is a suitable Crowding model since it compares new solutions against a subset of the entire population. The size of the subset selected is denoted as the Crowding window size, thus the application of this form of niching introduces one extra parameter to the basic PACO algorithm. The Crowding window size can be set anywhere between one and the size of the population. A general pseudocode representation of Crowding PACO (CPACO) is outlined in Alg. 1.

3.3.2 Fitness Sharing PACO

To implement Fitness Sharing with Population-based ACO a temporary pheromone matrix scheme is used. Rather than simply adding the best new solution and removing the oldest solution as in PACO, all solutions from a generation are added to the population, and the oldest population members are removed. After this addition/removal process is complete, the Fitness

Algorithm 1 Crowding Population-based Ant Colony Optimisation: CPACO

```

1: Uniformly initialise pheromone map values to  $\tau_{init}$ 
2: for  $j = 1$  to  $h$  do
3:   Create and evaluate random solution
4:   Insert random solution into history
5:   Add random solution information into pheromone map ( $+\Delta\tau$ )
6: end for
7: while stopping criterion not met do
8:   Construct  $m$  new solutions ( $s^{new}$ )
9:   Evaluate solutions
10:  Crowding history update
11: end while
12: procedure CROWDING HISTORY UPDATE
13:   for  $j = 1$  to  $m$  do
14:     Select random subset (size= $c$ ) of solutions ( $s$ ) from population ( $p$ )
15:     for  $k = 1$  to  $c$  do
16:        $d = \text{distance}(s_j^{new}, s_k)$ 
17:       if  $d < \text{leastDistance}$  then
18:          $\text{leastDistance} = d$ 
19:          $s_{closest} = s_k$ 
20:       end if
21:     end for
22:     if  $s_j^{new}.\text{quality} > s_{closest}.\text{quality}$  then
23:       Remove  $s_{closest}$  information from pheromone map ( $-\Delta\tau$ )
24:       Remove  $s_{closest}$  from population
25:       Add  $s_j^{new}$  to population
26:       Add  $s_j^{new}$  information into pheromone map ( $+\Delta\tau$ )
27:     end if
28:   end for
29: end procedure

```

Sharing function is applied to all solutions in the current population to de-rate their respective fitness values. These solutions are then used to construct a pheromone matrix which is used to create the next generation of solutions using the standard ACS pseudo-random proportional rule. A general pseudocode representation is provided as Alg. 2. In this pseudocode example p represents the population, and p_i the i^{th} member of that population.

3.4 Alternatives to Nicheing

As indicated by Horn [28] it is prudent to discuss alternative diversity preservation mechanisms alongside niching, as niching is itself a form of diversity preservation. Focusing on ACO algorithms, the issue of diversification versus intensification has been a driving force behind the development of ACO and

Algorithm 2 Fitness Sharing Population-based Ant Colony Optimisation: FSPACO

```

1: while stopping criterion not met do
2:   Construct temporary pheromone matrix
3:   Construct Solutions
4:   Update history (Replace oldest solutions)
5:   De-rate quality
6: end while
7: procedure DE-RATE QUALITY
8:   for  $j = 1$  to  $p_{\text{size}}$  do
9:     nicheCount = 0
10:    for  $k = 1$  to  $p_{\text{size}}$  do
11:       $d = \text{distance}(p_j, p_k)$ 
12:      if  $d < \sigma$  then
13:        shareValue =  $(1 - (d/\sigma)^\alpha)$ 
14:      else
15:        shareValue = 0
16:      end if
17:      nicheCount = nicheCount + shareValue
18:    end for
19:     $h_j.\text{quality} = h_j.\text{quality}/\text{nicheCount}$ 
20:  end for
21: end procedure

```

the balance between these factors is often cited as one of the distinguishing features of different ACO algorithms. This is particularly evident in algorithms such as Ant Colony Systems (ACS) [14] that use elite solutions to promote intensification and localised decay to ensure diversification.

Some research specifically targets the issue of diversification by adding features to basic ACO algorithms which introduce randomness [32, 19, 34, 35] dependent on different criteria such as the measured diversity of the population. For the problems addressed those techniques allow the modified algorithms to overcome some of the issues associated with premature convergence to suboptimal solutions.

While these cited approaches show improvement over the basic ACO algorithms, they are very different to niching. These modifications usually delay convergence to a single area of the search space. While niching aims at increasing diversity, it does not hold off convergence or prevent it, nor does it purposely introduce extra randomness. Niching aims at creating and maintaining stable subpopulations. This is quite different to slowing convergence or introducing randomness since niching algorithms can be tuned to be highly convergent, but convergent to multiple areas of the search space.

4 Applications of Niching Ant Colony Optimisation

To date only a handful of applications of niching ACO algorithms can be found in the literature. In this section three example applications are reproduced from the literature. The discussion focuses on highlighting the formation of niche behaviour and its effect on algorithm performance, including quality of solutions obtained and algorithm complexity.

4.1 Travelling Salesman Problem

At the outset applying a niching algorithm to the TSP does not seem to be a good idea since it has been shown that the TSP seems to benefit from greedy search behaviour, focused on one area of the search space at a time [42]. In their work on $\mathcal{MAX} - \mathcal{MIN}$ Ant System (MMAS), Stützle and Hoos [42] comment that Ant Systems (AS) performs poorly on the TSP in comparison to later ACO algorithms because AS does not exploit good solutions strongly enough. This is true since most of the best performing ACO algorithms for the TSP include variations such as greedy transition rules or pheromone update rules that strongly bias the reinforcement of elite solutions. Since elitism tends to correlate strongly with an increase in search efficacy this may indicate something about the problem: that the n -best solutions to a TSP all contain similar elements and thus are located in a similar area of the search space. In Angus [4] the best 100 solutions for the Burma14 problem were shown to occupy a similar space in the overall search space having on average 9 similar edges (out of 14) to all of the other best solutions. A baseline random sample indicated that from a random sample of 100 solutions only 2 out of 14 edges are the same on average when selecting from across the entire search space.

Since niching algorithms strive to maintain diversity, it is intuitive that niching algorithms are not suited to solving problems such as the single-objective TSP that benefit from strong convergent behaviour. However, cases may exist where a niching algorithm would be suited to solving the TSP. In Angus [1] a fabricated TSP, the Crown6 TSP, with two spatially separated optima was constructed. The Crown problem is a symmetric, 2-Dimensional Euclidean TSP containing 6 vertices which has the interesting property of containing two distinct yet equal global optima (Fig. 3). These optima are also a reasonable distance apart only sharing 3 out of 6 edges.

In Angus [2] the MMAS algorithm was tested² using 50 ants per iteration, and the number of times either of the two optima were found per iteration was recorded. This experiment was repeated 100 times for consistency of reported results. In every algorithm trial MMAS converged to one of the two

² Using the standard parameters as in Stützle and Hoos [41].

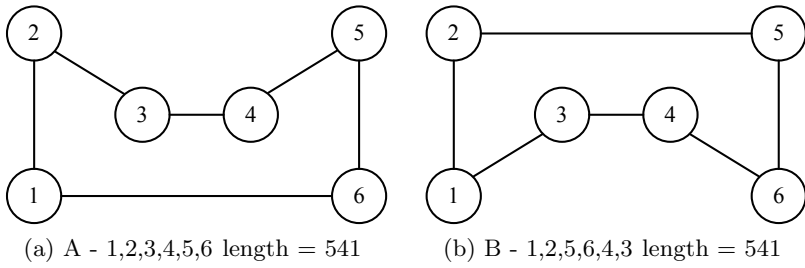


Fig. 3: Optimal solutions to Crown problem

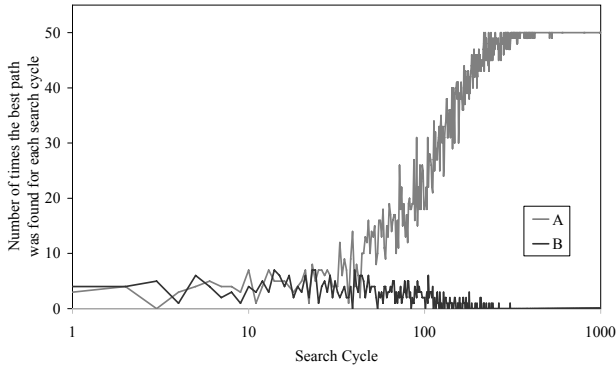
optima, while the niching PACO algorithms were shown to converge to both optima. Three graphs indicating single experimental runs (that are indicative of normal algorithm behaviour) of each of the algorithms are reproduced in Fig. 4a, Fig. 4b and Fig. 4c.

This Crown problem was a trivially small although quite unique TSP, with regard to the presence of two distinct optima. For completeness the niching PACO algorithms were also tested on several small-medium sized TSP from TSPLIB [36] to observe the effect of niching on the algorithm performance (with regard to locating the optimal solution). The results obtained, while not terrible were not as good as the MMAS algorithm with regard to the best solution found. This result was to be expected due to the reasons provided earlier.

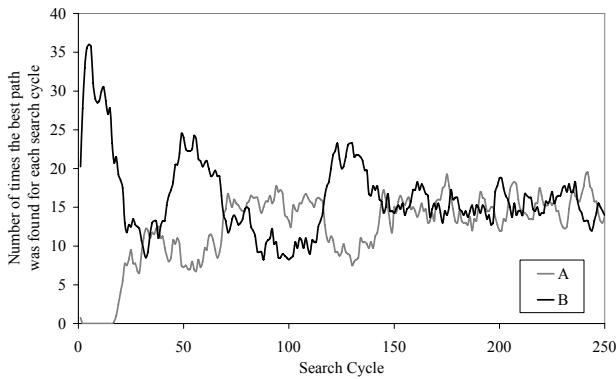
The overall findings from this study were that while the standard TSP instances of TSPLIB did not benefit from niching, the special case Crown problem did. While this problem was entirely contrived it did demonstrate the niching algorithms' strength in the simultaneous location and maintenance of multiple optima on a combinatorial problem domain. For the overwhelming majority of single objective TSP in TSPLIB [36] this property is probably not required, however there do exist other variations to the basic TSP that require the location and maintenance of diverse solutions, one such variation being the multiple objective TSP reported in Sec. 4.3.

4.2 Multimodal Function Optimisation

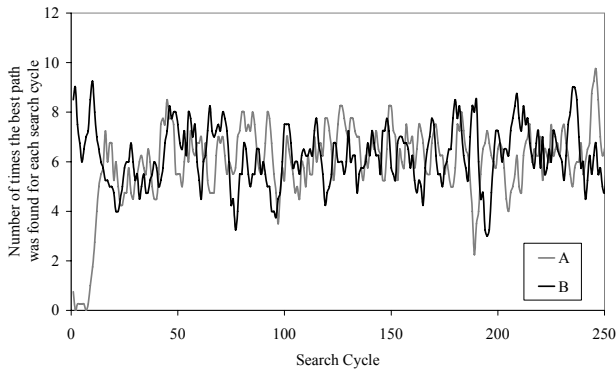
Multimodal function optimisation is a problem type that is useful in analysing niching behaviour. These problems have been used in many prior niching algorithm investigations mostly due to the ability to readily design challenging benchmark problems with features such as multiple diverse optima. The problems are also easily scaled to multiple dimensions which can rapidly increase the computational complexity, however such scaling does not necessarily come at a cost to the ability to analyse and comment on algorithms



(a) MMAS



(b) FSPACO



(c) CPACO

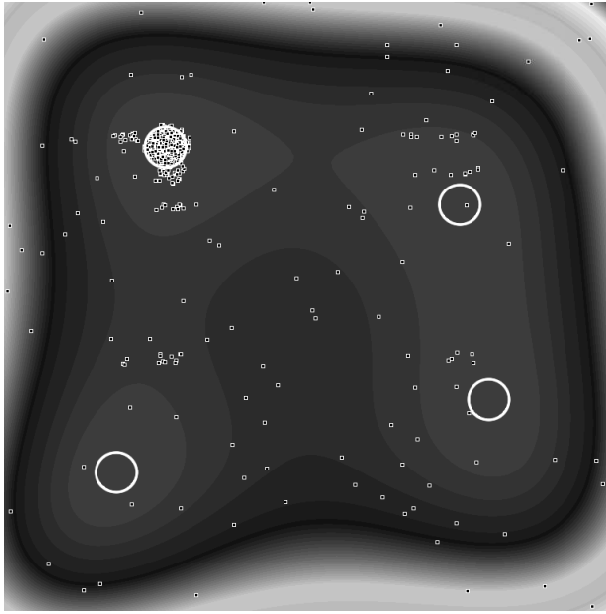
Fig. 4: Crown problem: Occurrence of the two optimal paths over time for a single (although indicative) experiment run (A & B are the two distinct optimal paths). Each algorithm tested constructs 50 solutions per iteration. Different running times are reported to better illustrate specific algorithm behaviour, importantly though the convergence characteristics do not change past the maximum number of iterations reported on the graphs.

applied to them. This is because these problems allow for easy visualisation of their search space, while still preserving their neighbourhood relationships.

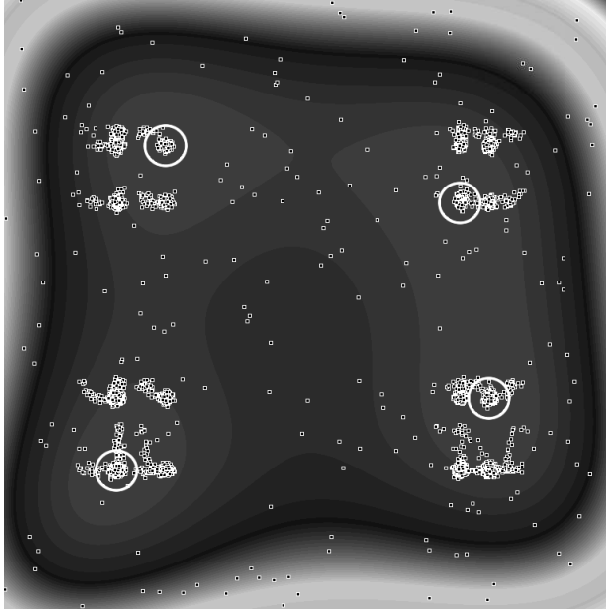
The domain of function optimisation is somewhat different to traditional combinatorial optimisation problems like the TSP. Variations of ACO algorithms have been applied to this domain with success [40]. In Angus [1] the Population-based ACO algorithm was modified for the function optimisation problem and was imbued with each of the fitness sharing and crowding niching techniques. Primarily this investigation was a qualitative one, aimed at observing sustained niche formation for a prolonged period of time. The Crowding PACO algorithm tested was successful in performing this task, while the Fitness Sharing PACO algorithm results were less impressive, attributed to high sensitivity in the parameter selection. Several figures indicating the distribution of all solutions for an entire algorithm run are reproduced from Angus [1] as Fig. 5. These figures indicate the fitness of the function as shaded, the dots represent all evaluated solutions (across multiple generations) for the entire algorithm run. In this example the algorithms tested were allowed 50,000 solution evaluations and each used a population size of 100, meaning that they were iterated for a total of 500 generations. As can be seen from these figures the sampling behaviour of the niching algorithms are spread across all peaks of interest, which is the expected behaviour of a niching algorithm. As a control the ACO algorithm without niching was found to converge and thus sample only one distinct peak.

A similar experiment was performed in Angus [4] to evaluate the effect of crowding window size on the stability of niche formation and maintenance. The problem used was the Shekel's foxholes problem which is comprised of 25 distinct optima of varying height. An optimal solution with a niching algorithm would be the distribution of a population across all 25 optima. In the study it was shown that decreasing the window size increased the frequency of replacement errors which lead to the loss of some of the optima. At an extreme this behaviour led to the loss of all but one optima. The measure used to analyse this behaviour was the max-peak ratio which is a sum of the relative distances from each optima to the nearest population member. Figure 6 indicates the effect of decreasing the crowding window size of CPACO for the Shekel's foxholes problem in two dimensions.

In general, results from Angus [1, 4] suggest that the Niching PACO algorithms were effective at solving a range of multi-modal function optimisation problems. These studies also compared the Niching PACO algorithms against state-of-the-art Niching EC algorithms and found that the results were comparable.

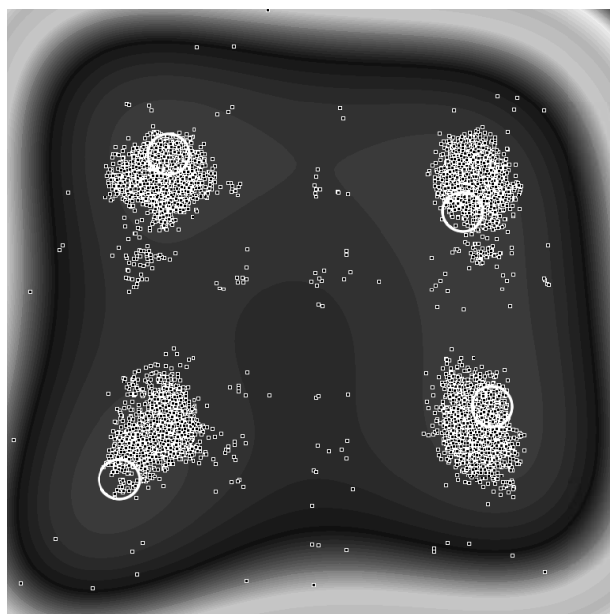


(a) Control (PACO)



(b) CPACO

Fig. 5: Diagrams indicating all of the 50,000 solutions generated in a single run (indicative of the usual search behaviour) by each algorithm applied to Himmelblau's Function (white circles indicate the four distinct optima, small white squares represent solutions).



(c) FSPACO

Fig. 5: (cont'd) Diagrams indicating all of the 50,000 solutions generated in a single run (indicative of the usual search behaviour) by each algorithm applied to Himmelblau's Function (white circles indicate the four distinct optima, small white squares represent solutions).

4.3 Multiple Objective Travelling Salesman Problem

Multiple Objective Optimisation (MOO) problems involve the simultaneous optimisation of two or more objective functions and most classic single objective optimisation problems have MOO variants. These problems often require a different approach to that of single objective optimisation due to decision makers often requiring multiple Pareto optimal, or near-Pareto optimal solutions; these Pareto optimal solutions being the best trade-off solutions. Given that more than one solution is required these problems are often solved using niching algorithms.

In Angus [2] the CPACO algorithm was extended for a suite of multiple objective travelling salesman problems (MOTSPs). The resulting algorithm, Multiple Objective CPACO (MO-CPACO) uses the key CPACO features of crowding replacement and probabilistic step-wise solution construction. To construct solutions MO-CPACO uses a weighted combination of heuristic information from all objective functions, and pheromone information which is based on the current state of the population. Due to there being more

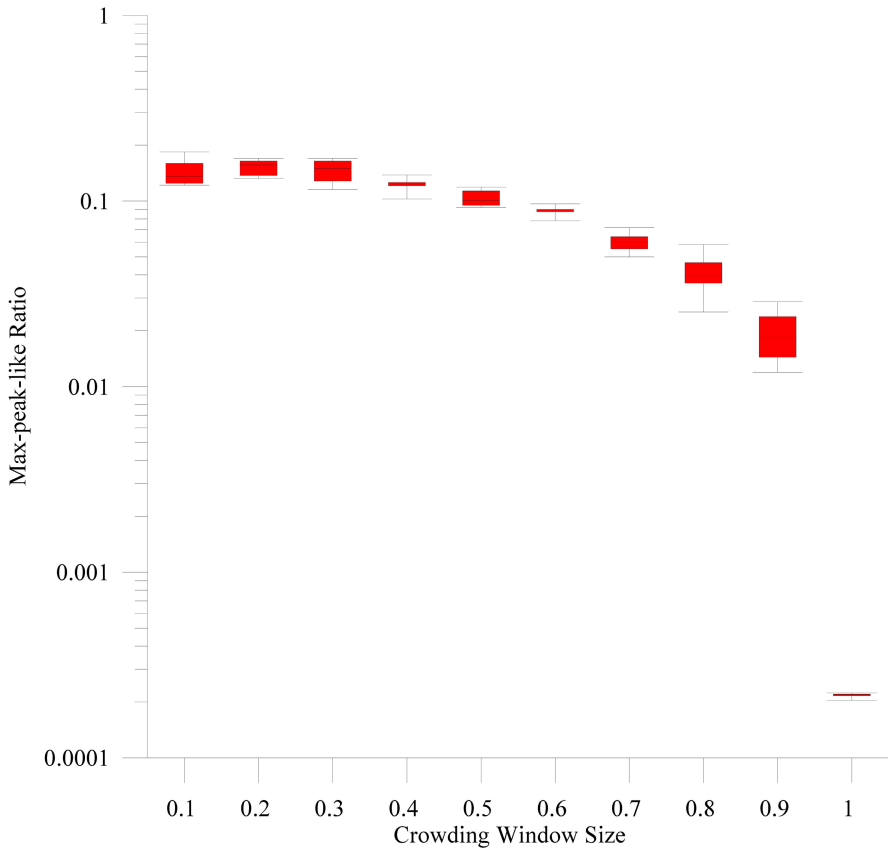


Fig. 6: Multiple box plots indicating the effect on the max-peak performance metric when the crowding window size of CPACO is varied from 0.1 to 1.0. For this max-peak performance metric zero indicates best performance while one indicates the poorest performance. Each parameter setting was repeated 100 times and allowed 50,000 solutions evaluations per individual run.

than one objective function the amount of pheromone deposited by each ant into the pheromone matrix is determined by a Non-Dominated Sorting technique [10] which ranks solutions according to their proximity to the Pareto front. This means that there is one heuristic matrix per objective and one pheromone matrix overall. All solutions are evaluated by each objective function after being created. These candidate solutions are then compared against a subset of the population. The closest solution (in terms of objective space distance) is replaced if the candidate solution is better in all objectives (crowding replacement). More specific algorithm details can be found in Angus [2].

In Angus [2] the MO-CPACO algorithm was tested on several multiple objective TSPs. Among the problems tested were the two objective KroAB100, KroAB150 and KroAB200 problems, each with 100, 150 and 200 cities respectively. These problems are taken from the TSPLIB [36]. As a comparison an original Multiple Objective ACO algorithm without niching was also tested with an equal number of solution evaluations and the same population size. Summary attainment surface comparison was used to evaluate any differences and these differences were evaluated using non-parametric statistical techniques as the data were non-normally distributed. These summary attainment surfaces show the best solutions obtained averaged over 100 independent trials. Given that every solution in the final population contains two objective values these solutions are plotted against both objectives simultaneously so as to easily visualise the trade-off between each objective. Like many multiple-objective problems these MOTSP don't have a single optimal solution, instead they contain several Pareto-optimal solutions.

Figures 7, 8 and 9, reproduced from the original investigation indicates that the MO-CPACO algorithm was able to obtain a better attainment surface than its non-niching equivalent. This finding suggests that the addition of niching to this ACO algorithm for these problems led to greater search efficacy. Such conclusions as to the usefulness of niching in multiple objective algorithm design are also commented on in Deb [9].

4.4 Key Findings

The problems listed in this section were so chosen for the original studies since they are applications which best demonstrate the advantages and disadvantages of adding niching to an ACO algorithm. The empirical results of these studies indicate that the use of niching benefited some problem domains while offering no substantial advantage to others. Some key findings are included here.

For the single objective TSP tested in Angus [1] there seemed to be a marked decrease in obtained solution quality. This was explained through a study of the neighbourhood relationship of an indicative TSP problem instance which highlighted that the TSP may offer a natural advantage to algorithms that tend to direct search effort in one area of the search space. A simple TSP problem (Crown6) was constructed to demonstrate that Niching ACO algorithms could exhibit niche formation on the TSP, but ultimately for this problem domain there seems to be no advantage in the application of these niching techniques.

Unlike the single objective TSP, multi-modal function optimisation problems do contain multiple spatially separated optima. For this particular problem domain Niching ACO algorithms were shown to be able to maintain stable niches [1, 4]. In these studies the Crowding PACO algorithm exhibited

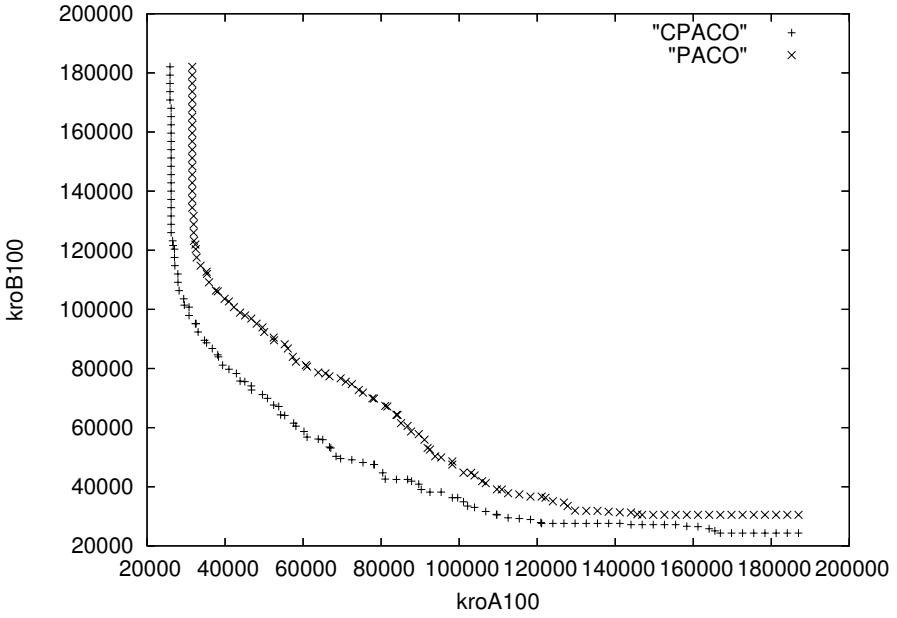


Fig. 7: 1% (best) attainment surface for kroA100 and kroB100 using MO-PACO (non-niching) & MO-CPACO (niching)

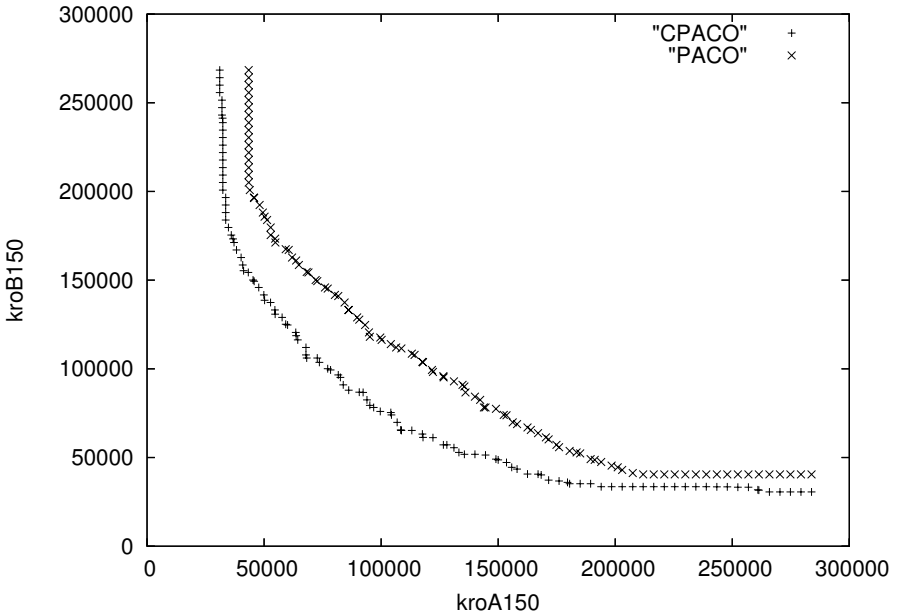


Fig. 8: 1% (best) attainment surface for kroA150 and kroB150 using MO-PACO (non-niching) & MO-CPACO (niching)

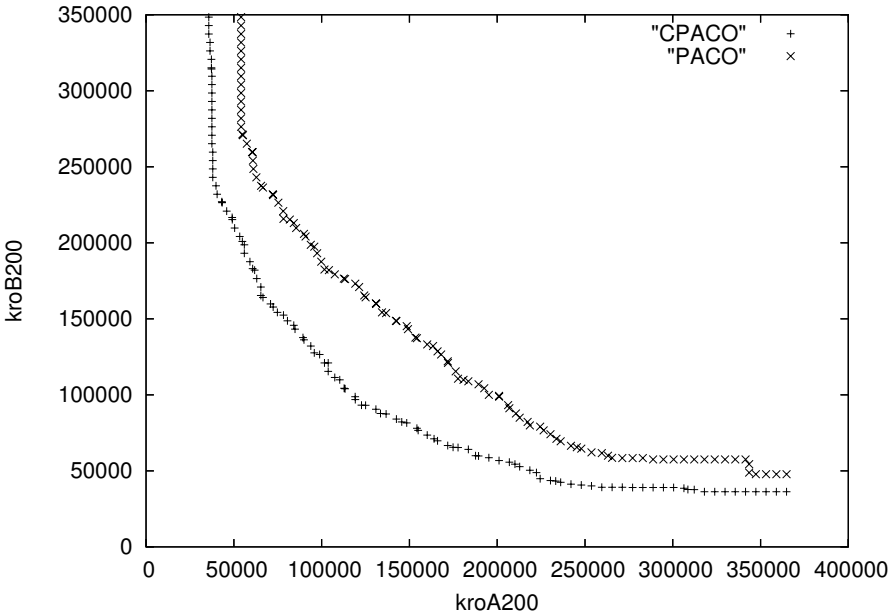


Fig. 9: 1% (best) attainment surface for kroA200 and kroB200 using MO-PACO (non-niching) & MO-CPACO (niching)

the best performance, while the Fitness Sharing PACO algorithm did not seem to perform as well, probably due to the parameter selection.

While the single objective TSP gave poor results for the Niching ACO algorithms, the application of the Crowding PACO algorithm to the multiple objective TSP [2, 4] saw a marked increase in algorithm efficacy and computational efficiency. For the problems tested, the Crowding PACO algorithm was able to maintain a very good distribution of solutions through the use of a niching population replacement operation.

5 Conclusion

This chapter has discussed current research into the application of niching to ACO algorithms. Important findings were the difficulty of applying basic niching techniques such as Crowding and Fitness Sharing to standard ACO algorithms. This difficulty was overcome through the use of the Population-based ACO algorithm paradigm. Once applied the niching algorithms were shown to be effective at sustaining multiple sub-populations distributed across points of interest in a search space. The key findings suggest that like many

niching EA these niching ACO algorithms are best applied to multi-modal or multiple objective problem domains.

Acknowledgements The author acknowledges time spent at Swinburne University of Technology preparing the research outlined in this publication. Specifically the assistance provided by Prof. Tim Hendtlass and other members of the Complex Intelligent Systems Lab, Swinburne University of Technology.

References

- [1] Angus, D.: 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)
- [2] Angus, D.: Crowding population-based ant colony optimisation for the multi-objective travelling salesman problem. In: 2007 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM 2007), pp. 333–340. IEEE, Piscataway (2007)
- [3] Angus, D.: Population-based ant colony optimisation for multi-objective function optimisation. In: Randall, M., Abbass, H.A., Wiles, J. (eds.) ACAL 2007. LNCS, vol. 4828, pp. 232–244. Springer, Heidelberg (2007)
- [4] Angus, D.: Niching ant colony optimisation. PhD thesis, Swinburne University of Technology (2008)
- [5] Brits, R.: Niching strategies for particle swarm optimization. Master's thesis, Department of Computer Science, University of Pretoria, South Africa (2002)
- [6] Brits, R., Engelbrecht, A.P., van den Bergh, F.: Scalability of niche PSO. In: Proceedings of the 2003 IEEE Swarm Intelligence Symposium (SIS 2003), pp. 228–234 (2003)
- [7] Cordon, O., Herrera, F., Stützle, T.: A review of the ant colony optimization metaheuristic: Basis, models and new trends. *Mathware & Soft Computing* 9(2,3) (2002)
- [8] Deb, K., Spears, W.M.: C6.2: Speciation methods. In: Bäck, T., Fogel, D.B., Michalewicz, Z. (eds.) *Handbook of Evolutionary Computation*. Institute of Physics Publishing (1997)
- [9] Deb, K., Agrawal, S., Pratab, A., Meyarivan, T.: A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. KanGAL report 200001, Indian Institute of Technology, Kanpur, India (2000)
- [10] Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 849–858. Springer, Heidelberg (2000)
- [11] DeJong, K.A.: An analysis of the behaviour of a class of genetic adaptive systems. PhD thesis, University of Michigan (1975)
- [12] Dorigo, M.: Optimization, learning and natural algorithms. PhD thesis, Politecnico di Milano, Italy (1992)

- [13] Dorigo, M., Di Caro, G.: The ant colony optimization meta-heuristic. In: Corne, D., Dorigo, M., Glover, F. (eds.) *New Ideas in Optimisation*, pp. 11–32. McGraw-Hill, London (1999)
- [14] Dorigo, M., Gambardella, L.M.: Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computing* 1(1), 53–66 (1997)
- [15] Dorigo, M., Stützle, T.: *Ant Colony Optimization*. MIT Press, Cambridge (2004)
- [16] Dorigo, M., Maniezzo, V., Colomi, A.: The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man and Cybernetics, Part B* 26(1), 29–41 (1996)
- [17] Dorigo, M., Bonabeau, E., Theraulaz, G.: Ant algorithms and stigmergy. *Future Generation Computer Systems* 16, 851–871 (2000)
- [18] Eldredge, N.: *Macroevolutionary Dynamics: Species, Niches and Adaptive Peaks*. McGraw-Hill, New York (1989)
- [19] Gambardella, L.M., Taillard, E., Agazzi, G.: MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. Tech. rep., IDSIA (1999)
- [20] Goldberg, D.E.: *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley, Reading (1989)
- [21] Goldberg, D.E., Richardson, J.: Genetic algorithms with sharing for multimodal function optimization. In: *Proceedings of the Second International Conference on Genetic Algorithms*, pp. 41–49 (1987)
- [22] Guntsch, M.: *Ant algorithms in stochastic and multi-criteria environments*. PhD thesis, Universität Fridericiana zu Karlsruhe (2004)
- [23] Guntsch, M., Middendorf, M.: Applying population based ACO to dynamic optimization problems. In: *ANTS 2002: Proceedings of the Third International Workshop on Ant Algorithms*, pp. 111–122. Springer, London (2002)
- [24] Guntsch, M., Middendorf, M.: A population based approach for ACO. In: Cagnoni, S., Gottlieb, J., Hart, E., Middendorf, M., Raidl, G.R. (eds.) *EvoIASP 2002, EvoWorkshops 2002, EvoSTIM 2002, EvoCOP 2002, and EvoPlan 2002*. LNCS, vol. 2279, pp. 72–81. Springer, Heidelberg (2002)
- [25] Guntsch, M., Middendorf, M.: Solving multi-criteria optimization problems with population-based ACO. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) *EMO 2003*. LNCS, vol. 2632, pp. 464–478. Springer, Heidelberg (2003)
- [26] Harik, G.R.: Finding multimodal solutions using restricted tournament selection. In: Eshelman, L. (ed.) *Proceedings of the Sixth International Conference on Genetic Algorithms*, pp. 24–31. Morgan Kaufmann, San Francisco (1995)
- [27] Holland, J.H.: *Adaptation in Natural and Artificial Systems: An Introduction With Applications to Biology, Control, and Artificial Intelligence*. MIT Press, Cambridge (1975)
- [28] Horn, J.: *The nature of niching: Genetic algorithms and the evolution of optimal, cooperative populations*. PhD thesis, University of Illinois (1997)
- [29] Mahfoud, S.W.: Crowding and preselection revisited. In: Männer, R., Mandrick, B. (eds.) *Parallel Problem Solving from Nature 2 (PPSN2)*, pp. 27–36. North-Holland, Amsterdam (1992)
- [30] Mahfoud, S.W.: *Niching methods for genetic algorithms*. PhD thesis, University of Illinois (1995)

- [31] Mahfoud, S.W.: Niching methods. In: Back, T., Fogel, D.B., Michalewicz, Z. (eds.) *Evolutionary Computation 2: Advanced Algorithms and Operators*, pp. 87–92. Institute of Physics Publishing, UK (2000)
- [32] Nakamichi, Y., Arita, T.: Diversity control in ant colony optimization. *Artificial Life and Robotics* 7(4), 198–204 (2004)
- [33] Petrowski, A.: A clearing procedure as a niching method for genetic algorithms. In: *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 798–803. IEEE, Los Alamitos (1996)
- [34] Randall, M.: Maintaining explicit diversity within individual ant colonies. In: *Recent Advances in Artificial Life*, ch. 17. World Scientific, Singapore (2005)
- [35] Randall, M., Tonkes, E.: Intensification and diversification strategies in ant colony system. *Complexity International* 9 (2002)
- [36] Reinelt, G.: Tsplib95 (1995), <http://www.iwr.uni-heidelberg.de/groups/comopt/software/tsplib95>
- [37] Ricklefs, R.E.: *Ecology*. Thomas Nelson & Sons Ltd. (1973)
- [38] Robbins, H.: Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society* 55, 527–535 (1952)
- [39] Schoeman, I., Engelbrecht, A.: Niching for dynamic environments using particle swarm optimization. In: Wang, T.-D., Li, X.-D., Chen, S.-H., Wang, X., Abbass, H.A., Iba, H., Chen, G.-L., Yao, X. (eds.) *SEAL 2006. LNCS*, vol. 4247, pp. 134–141. Springer, Heidelberg (2006)
- [40] Socha, K.: ACO for continuous and mixed-variable optimization. In: Dorigo, M., Birattari, M., Blum, C., Gambardella, L.M., Mondada, F., Stützle, T. (eds.) *ANTS 2004. LNCS*, vol. 3172, pp. 25–36. Springer, Heidelberg (2004)
- [41] Stützle, T., Hoos, H.: Improvements on the Ant System: Introducing the $MAX - MIN$ Ant System. In: *Third International Conference on Artificial Neural Networks and Genetic Algorithms*. Springer, Norwich (1997)
- [42] Stützle, T., Hoos, H.: $MAX - MIN$ Ant System. *Future Generation Computer Systems* 16(8), 889–914 (2000)
- [43] Watson, J.P.: A performance assessment of modern niching methods for parameter optimization problems. In: Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E. (eds.) *Genetic and Evolutionary Computation Conference*, vol. 1, pp. 702–709. Morgan Kaufmann, Orlando (1999)