university of groningen / faculty of science and engineering

A Comparison Between Ant System and Max-Min Ant System in Multi Colonial Systems

Bachelor's Project Thesis

Dekel Viner, s2612925
Supervisor: Dr M.A. Wiering

Abstract: This thesis describes the use of the ant system and Max-Min ant system in single and multi colonial structures. The differences between both ant colony systems are compared in the fitness of the solutions they produce, and the branching and exploration of the search space. The study shows that the shape of the fitness landscape is highly directive towards the final outcome of the search, and that there is a low variation in solutions of similar algorithms. We also observe that Max-Min ant system and the original ant system have distinctly different trail patterns. The improved performance of the Max-Min is not due to a higher branching of the search but mostly due to higher exploitation and random variability caused by having a minimum trail. Finally, no significant improvements in the multi colonial systems studied over a single colony system were found.

1 Introduction

Swarm intelligence (Bonabeau, Dorigo, and Theraulaz, 1999) is a problem solving approach that takes inspiration from social cooperative behavior of insects and other animals. In this category ants have inspired numerous techniques and algorithms, the most prominent being a class of meta-heuristics known as ant colony optimization (Dorigo, Birattari, and Stützle, 2006). Ant colony optimization (ACO) is inspired by the foraging behaviour of some ant species. These ants search for food and leave pheromone on the ground in order to direct other ants to the food source (Dorigo et al., 2006; Deneubourg, Aron, Goss, and Pasteels, 1990). Ants tend to follow paths where the pheromone concentration is higher and through this mechanism they generate a remarkably effective way of finding the shortest route to transport food. The information transfer employed by these ants is stigmergic, meaning it is an indirect form of information transfer mediated by the environment. In the early ninetees the first ACO algorithm was proposed and since then, numerous modifications and improved algorithms have been introduced. ACO as a class of meta-heuristics has been shown to successfully solve numerous combinatorial problems. Initially ACO algorithms have been used to solve conventional NP-hard problems such as the travelling salesman problem (Dorigo, Maniezzo, and Colorni, 1996; Stützle and Hoos, 2000), scheduling problems (Deng, 2002), and more recently ACO techniques have been applied to image processing (Khanna and Arora, 2016), clustering (Inkaya, Kayalgil, and Özdemirel, 2015), data mining and machine learning (Parpinelli, Lopes, and Freitas, 2002) and many more. The original ACO algorithm known as the ant system emerged in the ninetees (Dorigo et al., 1996). It has shown promising results in solving instances of the travelling salesman problem (TSP) to near optimal solutions, but it was later revealed to be inferior to the more specific state-of-the-art algorithms that existed and emerged during the time (Stützle and Hoos, 2000). This led to many modifications and enhancements that form the ant colony optimization class that we have today. As a stochastic agent based reinforcement learning approach, improvements on ant system focus on two aspects: enhancing the exploration of the search space and enhancing the exploitation of promising solutions found in the search space to better direct the search. Often a higher exploitation of found solutions leads to a faster stagnation of the search and hence a reduced exploration of the search space.

However, one of the emerging algorithms following the ant system that managed to strike a good balance between exploring and exploiting of the search space is the Max-Min ant system (Stützle and Hoos, 2000). Furthermore, ACO techniques have emerged to be great candidate algorithms for parallelisation (Middendorf, Reischle, and Schmeck, 2002), but instead of the parallelisation to be agent based it is colony based. This means that instead of a single colony system we now can have a multiple colony system similiar to a multiple island approach in genetic algorithms (Middendorf et al., 2002; Whitley, Rana, and Heckendorn, 1999). Information is shared between the different colonies expanding the variety of the search. This thesis will investigate the ant system and the Max-Min ant system under several different information sharing structures applied on the travelling salesman problem. The aim is to try to get a better understanding of the search exploration and exploitation characteristics of these different algorithms and structures. The algorithms will be compared based on the fitness of their final solutions, the progression of solutions produced per iteration and the average branching factor during the progression of the search. The average branching factor is a measurement to determine how much the search is spread out in the search space (Gambardella and Dorigo, 1995). There are not many ACO related scientific papers that study the branching factor of ACO algorithms, and even fewer that present the branching factor of algorithms through the progression of a search. I hope this paper will serve to disambiguate the workings of ACO algorithms and their traversal of search spaces. and aid in the development of better algorithms in the ACO paradigm.

2 ACO Algorithms

ACO algorithms make use of simple agents called ants. The ants iteratively construct candidate solutions to an optimization problem. A solution consists of solution components which each ant iteratively adds until a solution is generated. Theoretically ACO algorithms can be applied to any combinatorial optimization problem by defining solution components. A model of combinatorial optimization problems is needed in order to clarify the guidelines underwhich an ACO algorithm could be applied. The model shown was taken from (Dorigo et al., 2006; Blum and Roli, 2003; Talbi, 2009). A combinatorial optimization model $P = (S, \Omega, f)$ consists of:

- A search space S that is described over a finite set of discrete decision variables (solution components) $X = \{x_1 ... x_n\}$.
- A set Ω of constraints among the variables.
- An objective function $f : S \longrightarrow \mathbb{R}$ to be minimized.

A decision variable X_i takes in values from a possible domain set $D_i = \{v_i^1, ..., v_i^{|D_i|}\}$. The search space S is the set of all possible assignments. A feasible solution $s \in S$ is a complete assignment of values to decision variables that satisfies all the constraints in Ω. From the set of solutions S, a most desirable solution is a global optima.

Definition 2.1. A solution $s^* \in S$ is considered a global optima if and only if $f(s^*) \leq f(s), \forall s \in S$.

Ideally a global optima is reached in a search, but approximate algorithms, which ant colony optimization falls under often just reach a good enough solution also known as a local minimum (local optimum). In order to define a local minimum we first need to define the concept of neighborhood.

Definition 2.2. A neighborhood structure is a function $N : S \longrightarrow 2^S$ that assigns to every $s \in S$ a set of neighbours $N(s) \subseteq S$. $N(s)$ is called the neighbourhood of s.

We can now define the concept of local minima.

Definition 2.3. A local minimum with respect to a neighborhood structure N is a solution $\hat{s}$ such that $\forall s \in N(\hat{s}) : f(\hat{s}) \leq f(s)$

In ACOs the choice of solution components is guided by (artificial) pheromone often referred to as trail which is a value tied to every specific component. Depending on the specifics of the algorithm either a single ant or multiple ants will leave a trail behind depending on the quality of the solution they produced. The amount of trail also being proportionate to the solution quality determined by the objective function.

2.1 ACO for the travelling salesman problem

The travelling salesman problem consists of a set of cities and a set of distances between the cities. The goal is to find the shortest tour between all the cities and return to the start city. Every city has to be visited once and only once. More formally, the goal is to find a Hamiltonian tour of minimal length on a fully connected graph. Representing the TSP as a fully connected graph is simple, each vertex represents a city and each edge represents a connection between two cities. Every edge $(i, j) \in A$ is assigned a value d_{ij} which represents the distance between cities i and j. Every edge represents a potential solution component, and hence all edges have a corresponding trail value that stochastically directs the ants when constructing solutions. For every edge(i, j) there is also a corresponding trail value τ_{ij}.

3 Search Space Characteristics

3.1 Fitness-distance correlation

When studying the effectiveness of optimization algorithms to certain problems often the fitness landscape of the problems is considered. The fitness landscape can be envisioned as a terrain filled with solutions. Valleys and mountains represent local maxima and minima. Solutions that are similar lie close to each other and the more solutions differ from each other the further they are apart in the landscape. In fact every move in the landscape can be viewed as a single change to a solution component. Formally the fitness landscape can be described as follows, taken from (Stützle and Hoos, 2000):

- a set S of all possible solutions.
- a fitness function f that assigns a certain fitness value to every potential solution.
- a neighbourhood structure $N \subseteq S \times S$

The relative location of solutions is presented in the neighborhood structure. The distance $d(s, s')$ between two solution s and s' can be viewed as the minimum number of moves or adaptations necessary to traverse from one solution to the other. In order to understand the fitness landscape, it is helpful to consider the fitness-distance correlation (FDC). The FDC can be computed as follows:

$$\rho(F, D) = \frac{Cov(F, D)}{\sqrt{Var(F)}\sqrt{Var(D)}} \tag{3.1}$$

$Cov(F, D)$ describes the covariance between the fitness of local optima and the distance to the nearest global optimum. The covariance can be computed with the following formula:

$$Cov(F, D) = \frac{\sum_{i=1}^{n}(f_i - \bar{f})(d_i - \bar{d})}{n - 1} \tag{3.2}$$

here $F = \{f_1...f_n\}$, is the set of fitness values of local optima, and $D = \{d_1...d_n\}$, is the set of corresponding distances. The covariance describes a general relationship between the two variables. The statistical formula for measuring variance for some set $X = x_1...x_n$ is as follows:

$$Var(X) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n} \tag{3.3}$$

It indicates variability within a given set of values.

ACO algorithms focus the search around good solutions and therefore the shape of the fitness landscape for a given problem is an important indicator to the effectiveness of an ACO algorithm. For the TSP which is a minimization problem, a high positive correlation indicates that good solutions exist at low distances to other good solutions. In other terms good solutions are concentrated in the search space, which means that ACO algorithms could converge towards good solutions in the search space.

An estimation of the FDC for TSPs can be computed empirically as has been done (Stützle and Hoos, 2000). The distance between solutions can be defined as the number of different solution components. Which in the case for the TSP amounts to the number of different arcs in the tours. Formally, the distance between solutions s and s' is given by (Stützle and Hoos, 2000):

$$d(s, s') = n - |(i, j) : (i, j) \in s \wedge (i, j) \in s'| \tag{3.4}$$

In which n is the number of cities. In (Stützle and Hoos, 2000) a 3-opt local search was used to find local minimas for several TSP instances. Using 2500 local minimas to empirically compute the FDC and form plots of the local minimas distance to the global optimum against the percentage deviation from the global optimum. It was shown that symmetric TSPs tend to have a positive relatively high FDC coefficient (Stützle and Hoos, 2000). From this it is derived that TSPs make good candidate problems for ACO algorithms.

3.2 The branching factor of ACO algorithms

An important factor when considering how an ACO algorithm is functioning is its branching factor. Introduced in (Gambardella and Dorigo, 1995), the branching factor is an indication of how much the search is concentrated around a specific solution. In order to compute the branching factor, let τ_r^{max} and τ_r^{min} be the maximum and minimum trails leaving node r. Let $\delta_r = \tau_r^{max} - \tau_r^{min}$, and let λ be a parameter value such that $0 \leq \lambda \leq 1$. The branching factor of node r is then the number of edges exiting from node r which have a trail value greater then $\lambda * \delta_r + \tau_r^{min}$. The mean λ-branching factor is then the average branching factor across all the nodes for a give value of λ (Gambardella and Dorigo, 1995). Considering that in ACO the search tends to concetrate itself around good solutions and that symmetic TSPs tend to have a high positive FDC, the branching factor could be an indication of why a run of an algorithm concluded in a local optimum.

4 Ant-System

Algorithm 4.1 depicts the general format of the ant system and most ACO algorithms. Let m represent the number of ants in a system. Every cycle of the algorithm, m ants each construct a solution to the TSP by iteratively choosing an unvisited city based on a probability distribution given by formula 4.1.

$$p_{ij} = \frac{(\tau_{ij}^{\alpha})(\eta_{ij}^{\beta})}{\sum_{k \in s}(\tau_{ik}^{\alpha})(\eta_{ik}^{\beta})} \tag{4.1}$$

Here τ_{ij} is called trail and is the amount of (artificial) pheromone on the edge(i, j). η_{ij} is a heuristic

Algorithm 4.1 Iteration of a general Ant System algorithm

```
chooseStartingCities
WHILE ants have not visited all the cities
DO
  FOR each Ant DO
    chooseNextCity {formula(4.1)}
  END FOR
END WHILE
updateDeltas {formulas(4.2)(4.3)}
updateTrails {formula(4.4)}
```

used to ensure that closer cities have a higher priority, it equals $1/d_{ij}$ where d_{ij} is the distance of edge(i, j). α and β are parameter values that determine the relative influence of the trail and the distance heuristic respectively. In this thesis we restrict the use of formula 4.1 to only be applied to the 20 nearest cities of an ant's current city. The reason formula 4.1 is only applied to the 20 nearest cities is in order to reduce the computational time, especially for large TSP instances. Another reason is because traversing arbitrarily large edges outside of this subset will no longer be a phenomena of local improvements (Johnson, 1990, as cited in Bently, 1992). If all 20 of the nearest cities have already been visited then the next city is the one with the maximum value for $(\tau_{ij}^{\alpha})(\eta_{ij}^{\beta})$.

Once the set S for each ant no longer contains unvisited cities, all the ants have constructed a solution. Trail deltas ($\Delta\tau$) and the edges trail (τ) are then updated. In this version of the ant system all m ants will update the trails. In some versions only the m_b best ants update the trails, such that $m_b \leqslant m$. $\Delta\tau_{ij}$ is the amount of trail to be added to an edge(i, j) and is given by the sum of the individual ants contributions, presented in equation 4.2.

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{4.2}$$

An individual ant's contribution is then represented by $\Delta\tau_{ij}^{k}$, for some ant k, and is computed using equation 4.3. (Stützle and Hoos, 2000).

$$\Delta\tau_{ij}^{k} = \frac{Q}{L^{k}} \tag{4.3}$$

Q is the contribution amount and it is set as a parameter, L^k is the solution length of ant k. An ant

only contributes to an edge(i, j) if edge(i, j) is located on its path. The new trail value is given by equation 4.4.

$$\tau_{ij} = \rho * \tau_{ij} + \Delta\tau_{ij} \tag{4.4}$$

The new trail equals the old trail value multiplied by the perseverance constant ρ added by the summed contribution of all the ants. ρ is set as parameter for the perseverance rate of the trail pheromone. Another way to look at it is that $1-\rho$ is the evaporation rate, it has the following restriction $0 \leq \rho < 1$.

5 Max-Min Ant System

Max-Min ant system differs from the original ant system algorithm in two ways:

- Firstly, in an afford to increase the exploitation of the search space, only the best ant may leave pheromone on the trails. The best ant is decided by either having the current iteration's best solution or by having the global best solution.
- Secondly, in an effort to limit the stagnation of the search, the amount of trail on each edge is limited to the range $[\tau_{min},\tau_{max}]$.

The reason we are having a max-min bound is to avoid a complete stagnation around a solution that is sub-optimal and that does not allow for further exploration. Therefore choosing appropriate τ^{max} and τ^{min} parameter values is crucial for controlling how much exploration will be allowed. Here the method for determining the trail limits from (Stützle and Hoos, 2000) is adopted. This method is a thought out way of determining appropriate trail limits. Trail limits are set with relation to the maximum amount of trail that could possibly be placed with accordance to the problem. Stützle and Hoos (2000) have shown that the maximum amount of trail is asymptotically bounded to:

$$\frac{1}{1-\rho}\frac{1}{f(s^{opt})} \tag{5.1}$$

here $f(s^{opt})$ is the fitness value of the optimal solution. In the case of the TSP it is the shortest possible route. The global best solution found thus far is used in order to estimate the optimal asymptotic bound. By replacing $f(s^{opt})$ with an empirically found global best solution, the asymptotic bound can be estimated:

$$\tau_{max} = \frac{1}{1-\rho}\frac{1}{f(s^{gb})} \tag{5.2}$$

Equation 5.2 gives the formula for setting the maximum trail, $f(s^{gb})$ is the global best solution in a run of the algorithm. In order to assign a τ_{min} Stützle and Hoos (2000) used the following logic: there is a certain probability at which the optimal best solution can be found. Call that probability P_{best} and assume that at every decision point the probability of choosing a component of P_{best} is identical and call that probability P_{dec}. Therefore, in a TSP with n cities we can conclude that $P_{dec}^{n} = P_{best}$. Hence, P_{dec} is given by the following formula:

$$P_{dec} = \sqrt[n]{P_{best}} \tag{5.3}$$

We make an assumption that after an infinite number of iterations the probability of choosing an element of an optimal solution at each decision point will equal τ_{max} and choosing a solution component from different solutions will equal τ_{min}. Take $n/2$ to be the average number of possibilities at every decision point. Then equation 5.4 gives the probability of a single decision from the optimal solution.

$$P_{dec} = \frac{\tau_{max}}{\tau_{max} + ((n/2)-1)\tau_{min}} \tag{5.4}$$

Rearranging the formula 5.4 gives equation 5.5 for τ_{min}:

$$\tau_{min} = \frac{\tau_{max}(1-P_{dec})}{((n/2)-1)P_{dec}} \tag{5.5}$$

P_{best} is set as a parameter, if $P_{best} = 1$ then t_{min} would equal 0. If P_{best} is set to be very small it could be that $t_{min} > t_{max}$, in that case it is set that $t_{min} = t_{max}$. If $t_{min} = t_{max}$ the trails become meaningless, and only the heuristic information is used.

6 Multiple Colony Ant System Optimization

For the multiple colony approaches we use three different structures, all structures were implemented with either ant system colonies or Max-Min

colonies. It is possible to have structures comprised of a variety of different ACO variants. For example one could use a structure consisting of a combination of Max-Min colonies and ant system colonies. Here for simplicity, only structures containing exclusively Max-Min or the ant system were experimented with.

- The first structure is what we can refer to as a simple multi colony structure. Here there is a number of independent colonies with the ants divided equally between the colonies.
- The second structure is what we call a central colony structure. The central colony structure is similar to the simple multi colony structure, however in this case there is also a central colony which receives information from the other independent colonies, and ultimately takes over the search.
- The final structure is what we call a ring structure. Introduced by Middendorf et al. (2002) as a circular exchange of locally best solutions structure. This structure can be seen as a linked list, Every N_e number of iterations each colony shares information with the colony to the right in the list.

The structures are described in more detail below.

6.1 Multi Colony Ant System

Let m_b represent the overall number of ants in the ant population. Let N_c be the number of independent colonies. Divide the m_b ants among the number of active colonies N_c, with the remaining ants being distributed as evenly as possible. Algorithm 6.1 shows pseudo code for a single iteration of a simple multi colony structure. It is important to note that in algorithm 6.1 "nextIterationOfColony" refers to an iteration like in algorithm 4.1. Notice that there is also an optional central system. With the central colony system after 80% of the total number of iterations has passed, the central colony takes over. In the initial 80% of the total number of iterations the central colony is only being trained by the outside colonies but it generates no solutions. Training occurs as the outside colonies pass on their best solutions to update the trail deltas and ultimately the trails of the central colony. In the final 20% of the total number of iterations all the ants shift to the central colony which runs independently with the outside colonies being deactivated. The hope is that the external colonies will converge around different routes and when the central colony takes over it would be able to select the best solution components from the different routes and generate a better solution. This also adds to further diversification in the search.

Algorithm 6.1 Iteration of a Multi Colony Ant System

```
for each colony do
    nextIterationOfColony
    updateCentralColonyDelta (optional central ant colony system)
end for
updateCentralColonyTrail (optional central ant colony system)
```

Algorithm 6.2 Iteration of a Ring Colony structure

```
for each colony do
    nextIterationOfColony
    IF total Iterations mod N_c = 0 THEN
        updateNextColonyDelta
    END IF
end for
```

6.2 Ring Ant System

The Ring ant system works by having N_c colonies connected to each other in a manner similar to a linked list. Algorithm 6.2 shows the pseudo code for a single Ring ant system iteration. Each colony runs independently, and every N_e iterations that the individual colonies run, they transfer their own best found solution to the right adjacent colony in the figurative recursive linked list.

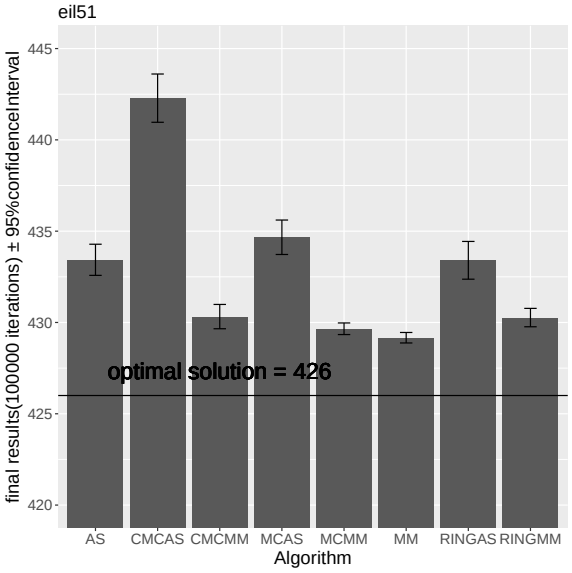
7 Experiments

The experiments were conducted on 5 problems of various sizes: eil51.tsp, berlin52.tsp, eil76.tsp, kroA100.tsp and ch130.tsp. The problems are taken from TSPLIB (Reinelt, 1997). For all problems and algorithms the parameters remained the same: $\alpha = 1$, $\beta = 2$, the number of iterations was set to

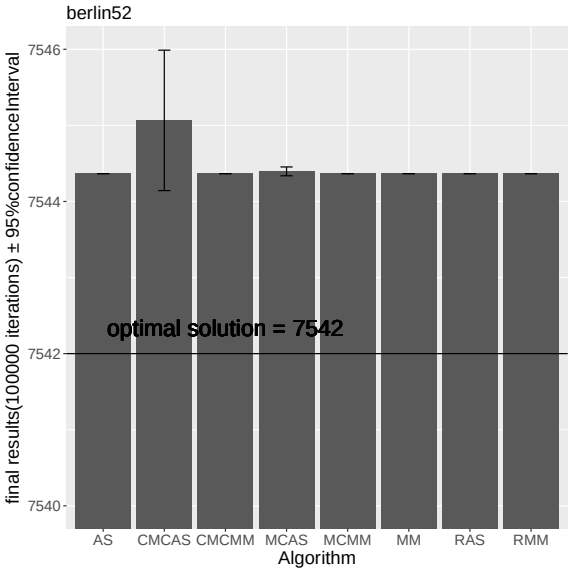
$100,000$, $\rho = 0.98$, Q $= 1$. The trail was initialized at $100,000$ for ant system and an ambiguously high number for Max-Min. For the multi-colonial algorithms, we had 4 colonies. The parameters were chosen based on standards in literature for the ant system and the Max-Min ant system (Dorigo et al., 1996; Stützle and Hoos, 2000). ρ was set relatively high to reduce stagnation. For the multi-colonial algorithms with a central colony, there were four external colonies and one central colony. The Ring structured systems were constructed by five colonies. Experiments were ran on ant system (AS), Max-Min ant system (MM), multi colony ant system (MCAS), multi colony Max-Min (MCMM), central multi colony ant system (CMCAS), central multi colony Max-Min ant system (CMCMM), Ring ant system (RAS), Ring Max-Min ant system (RMM). Every iteration for each individual colony the global best solution was measured and the mean branching factor of the trails in the colonies was measured with a branching coefficient $\lambda = 0.1$. Ten trials were conducted for every algorithm.

8 Results

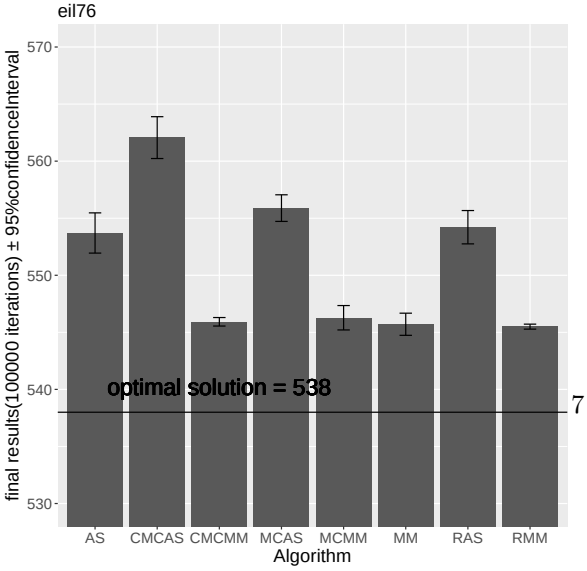
The purpose of the experiments is to obtain a better understanding of the workings of the different ACO algorithms. As a reminder the algorithms which are tested are the ant system(AS), the Max-Min ant system(MM), the multi colony ant system (MCAS), the multi colony Max-Min ant system(MCMM), the ring ant system(RAS), the ring Max-Min ant system(RMM), the central multi colony ant system(CMCAS) and the central multi colony Max-Min ant system(CMCMM). The results that are shown are selected in order to highlight the attributes of the algorithms which we are interested in. For every colony in an algorithm each iteration the global best solution of the colony is measured and the mean branching factor is computed. Note the difference between a global best solution of a colony and the global best solution of an algorithm. The algorithm global best is the best solution obtained across the whole algorithm while the colony global best is the best solution obtained only within that colony.



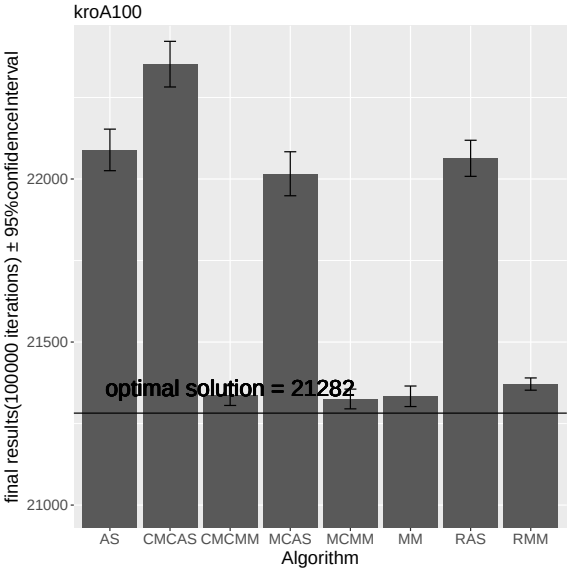
(a)



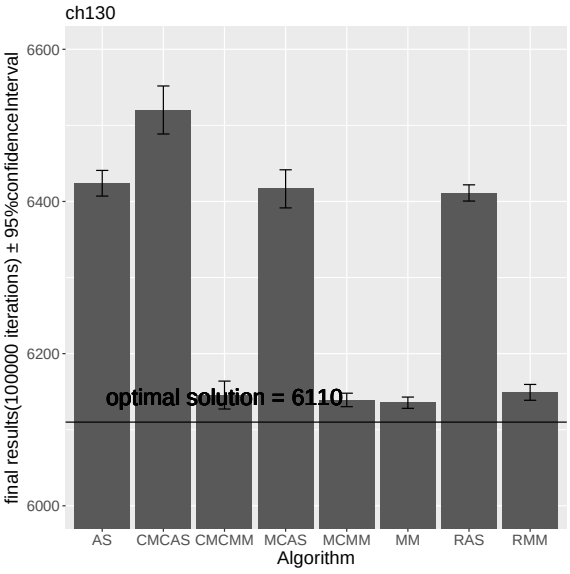
(b)



(c)



(d)



(e)

Figure 8.1: plots for the final solutions after $100,000$ iterations, averaged over 10 runs with a 95% confidence interval. The problems are eil51(a), berlin52(b), eil76(c), kroA100(d) and ch130(e). The algorithms depicted are the ant system(AS), Max-Min ant system(MM), multi colony ant system (MCAS), multi colony Max-Min ant system(MCMM), central multi colony ant system(CMCAS), central multi colony Max-Min ant system(CMCMM), Ring ant system(RAS) and Ring Max-Min ant system(RMM).

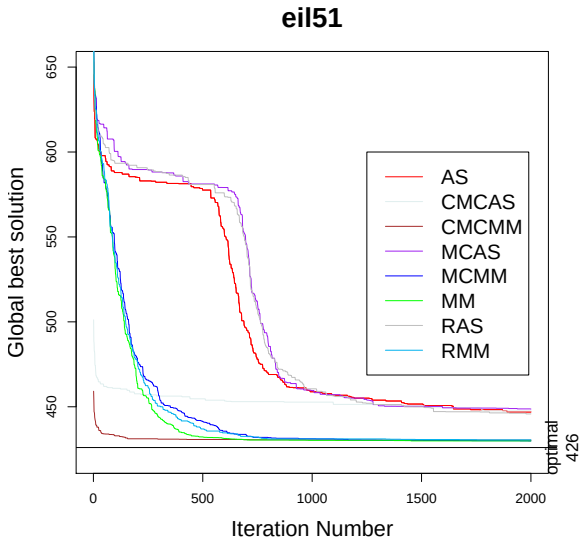
8.1 Final solutions

The first results, shown in figure 8.1, depict solution fitness values after $100,000$ iterations. For AS, MM, MCAS, MCMM, RAS and RMM the algorithm global best fitness value is used. For CMCAS and CMCMM the central colonies global best fitness values are used. The results are averaged over 10 runs and the error bars show the 95% confidence interval. Looking at figure 8.1 it is noticeable that for most problems the Max-Min based algorithms produce distinctly different results compared to the ant system based algorithms. The results show that Max-Min based structures overall perform better than ant-system based structures. Among the Max-Min structures there appears to be no significant differences in the final solutions. For the ant system based structures, generally the CMCAS's central ant system colony performs poorer than the other structures.

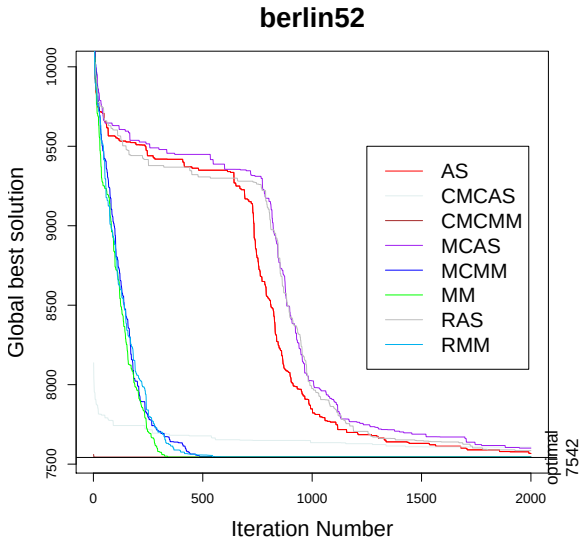
The 95% confidence interval is relatively small. For stochastic algorithms we might expect more deviation, but ACO algorithms seem to be very reliable in the results they produce. Therefore, it appears that for a specific search space, similar algorithms produce similar results and converge on similar local minima. The local minima in the search space can be seen as a sort of bread crumb trail leading the algorithms through the fitness landscape. which is why similar algorithms tend to produce very similar results. When looking at the berlin52 results we can see that most of the algorithms produced near identical results. That can only be attributed to the fitness landscape, which is also shown by the small confidence interval produced. It is interesting that none of the algorithms ever reaches the optimal solution which leaves room for further research into why this occurs.

8.2 Solution per iteration

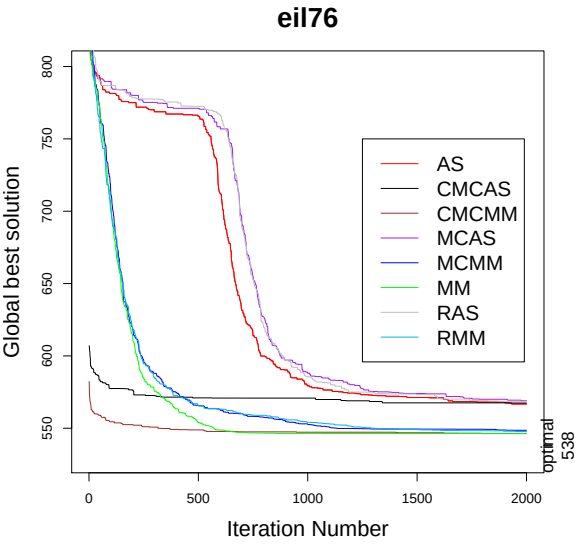
Figure 8.2 depicts the global best solution per iteration for the first 2000 iterations. For AS, MM, MCAS, MCMM, RAS and RAM the algorithm global best solution per iteration is used and averaged over 10 runs. For CMCAS and CMCMM the central colony's global best solution is used for the iterations 80001-82000. These iterations are the first 2000 iterations after the central colony is no longer being trained and has taken over the search.



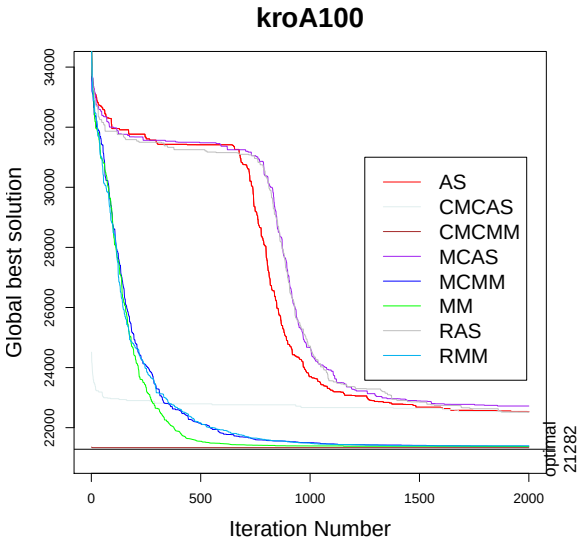
(a)



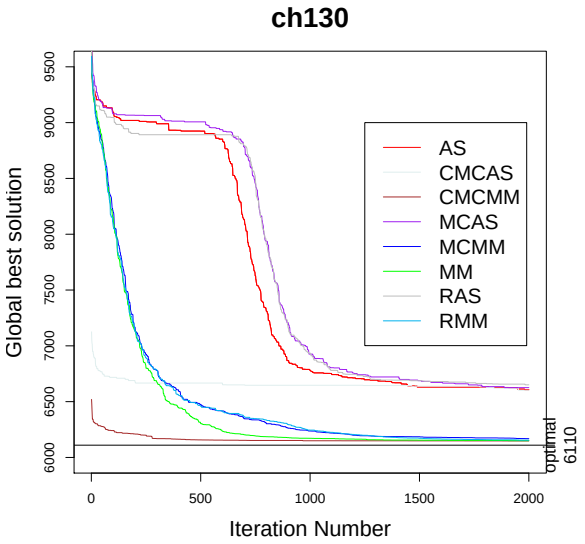
(b)



(c)



(d)



(e)

Figure 8.2: plots for the global best found solution per iteration. The global best solution obtained by an algorithm at an iteration is averaged over 10 runs, for the central colony structures it is the global best obtained by the central colony. The problems are eil51(a), berlin52(b), eil76(c), kroA100(d) and ch130(e). The algorithms depicted are the ant system(AS), Max-Min ant system(MM), multi colony ant system (MCAS), multi colony Max-Min ant system(MCMM), central multi colony ant system(CMCAS), central multi colony Max-Min ant system(CMCMM), Ring ant system(RAS) and Ring Max-Min ant system(RMM).

These results are also averaged over the 10 runs.

The results as expected show that Max-Min based algorithms converge on better solutions than ant system based algorithms. The central colonies tend to converge the fastest and reach good solutions quicker, which is expected as the trails are already directing the ants to the good solutions in the search space.

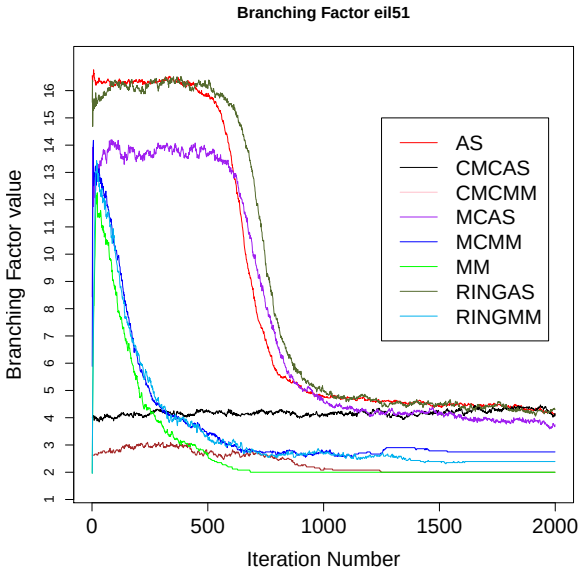
Figure 8.2 shows ant system based algorithms converge to one value as the curve flattens, and then there is a steep improvement in solutions and the true convergence is reached. This is due to high initial trail values and a low Q value. It takes some iterations until the high trail values get reduced enough for the added Q value to the trails to become a significant factor. The search only becomes effective as the trails are sufficiently reduced for the added trail to become significant. However, this does not impact the final results significantly as $100,000$ iterations are enough for the trail values to normalize and the search to stagnate.

The central colonies while being highly primed, also still converge. This is mostly due to the probabilistic nature of the search. While the search is heavily directed it still takes some iterations until the really good solutions are found. It is also due to the fact that the different colonies provide slightly different paths which leads to initial diversification of the search.

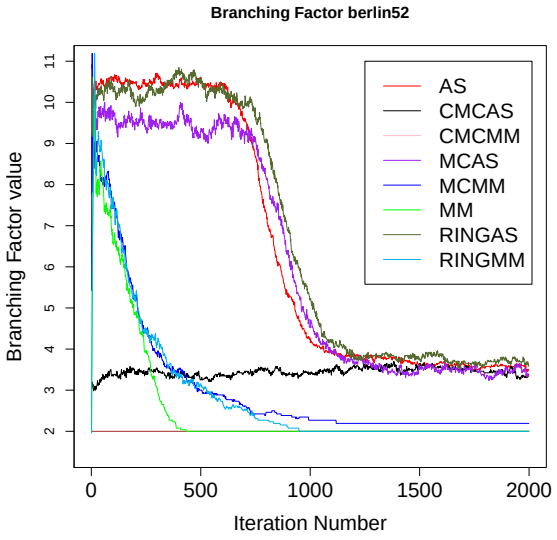
8.3 Branching factor per iteration

Finally, figure 8.3 depicts a colony's branching factor per iteration of the algorithm. In figure 8.3 the results are not averaged. For AS and MM the branching factors for the first 2000 iterations of a random trial are used. For MCAS, MCMM, RAS and RMM the branching factors of the first 2000 iterations of a single random representing colony in a random trial are used. For CMCAS and CMCMM the branching factors for iterations 80001-82000 of the central colonies in a random trial are used.

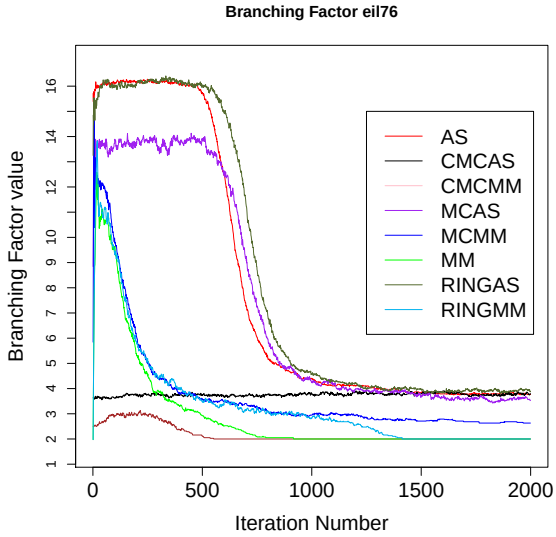
In Figure 8.3 we see that the branching factor of the Max-Min ant system converges to lower values than the ant system. This is due to the fact that the Max-Min ant system uses only a single ant to update its trails while the ant system uses $m_b = n$ number of ants. We also observe that the Max-Min algorithms tend to converge towards 2. A branching factor of 2 means that on average only



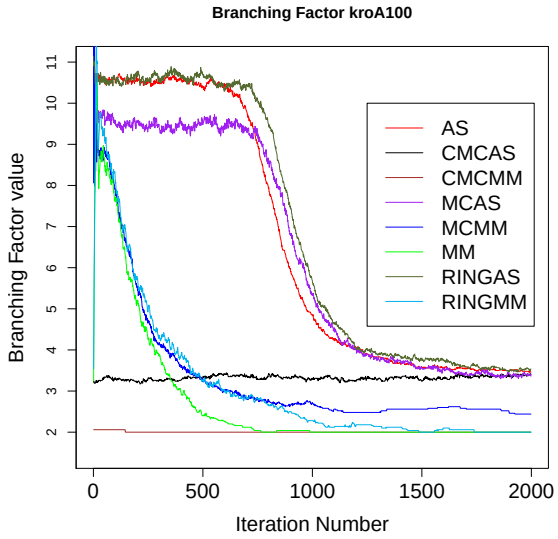
(a)



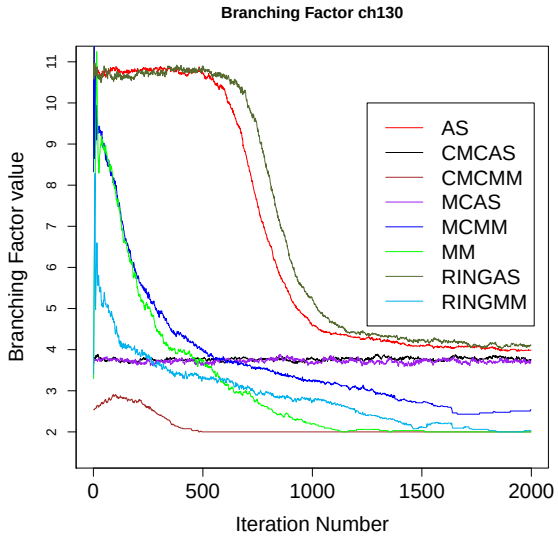
(b)



(c)



(d)



(e)

Figure 8.3: plots for the average branching factor per iteration. The average branching factor is shown for a random representing colony from a random trial. The problems are eil51(a), berlin52(b), eil76(c), kroA100(d) and ch130(e). The algorithms depicted are the ant system(AS), Max-Min ant system(MM), multi colony ant system (MCAS), multi colony Max-Min ant system(MCMM), central multi colony ant system(CMCAS), central multi colony Max-Min ant system(CMCMM), Ring ant system(RAS) and Ring Max-Min ant system(RMM).

two edges with significant trail values are exiting each node. This indicates that a search has either stagnated completely or is very close to. Of course the Max-Min ant system always has a probability to explore other options due to the minimum trail limit (Stützle and Hoos, 2000). Looking at Figure 8.3 we can see that the ant system based algorithms tend to converge to a branching factor between 3 and 4. In effect the branching factor of the ant system is also converging towards 2 but really slowly. An average branching factor between 3 and 4 means that the solutions generated by the algorithm will have some variability but are still in a small concentrated area of the search space. Therefore the added trail values of the different ants will be close to identical. Hence, when all the ants are directing the search, the search stagnates around an area of the search space. In the case with the ant system that is an area which is more spread out than the Max-Min ant system and further from the optimal. The Max-Min central colony initially has a branching factor greater than 2 and after a few runs converges onto 2, which is an indication that the different colonies do produce different results that lead to branching in the search space. Regarding the berlin52 problem the fitness landscape is very directive, so this effect does not take place. However, why the diversification of the external colonies does not lead to overall better performance is still to be explored.

9 Conclusion

The findings of this report are a reiteration of the findings that can be found in the literature about Max-Min and the ant system. However, the findings are presented in this paper in a new way as the results are tied back to the branching factor and the influence that the fitness landscape has on the search of an ACO algorithm.

The results show that ACO algorithms are very reliable and produce very similar results per run of an algorithm. This means that the fitness landscape of a problem is very directive to the progression of a search. The effectiveness of an ACO algorithm depends on the extent to which it concentrates the search around good solutions and the probability it has to explore outside these concentrated areas of the search.

The findings show unsurprisingly that the Max-Min based algorithms perform significantly better than ant system based algorithms. The reason for this better performance is that the Max-Min based algorithms converge but do not stagnate. The difference between convergence and stagnation of a search is that with convergence there is a still a low probability of solutions to be produced that are outside the concentrated area of the search space. With stagnation the search is bound to an area of the search space, and alternative areas are not explored (Stützle and Hoos, 2000).

The Max-Min ant system which uses only the best ant to update the trails, converges on a smaller area of the search space that is closer to optimal than the ant system. The ant system stagnates in a larger area of the search space, meaning that even though the algorithm is covering a larger section of the search space more frequently, that section is farther from the optimal.

The branching factor reiterated the difference between the convergence of Max-Min and the stagnation of ant-system. The trails of the Max-Min ant system converge to a smaller average branching factor than the ant system. This is due to the fact that in the Max-Min ant system only a single ant updates the trails. Because there is always a probability to explore other options the Max-Min ant system turns into a superior search algorithm. The reason that the Max-Min ant system does not find optimal solutions could be because the optimal solution lies too far from the concentrated area of the search space. Hence, the probability to find the optimum is simply too low. The ant system stagnates in a larger area of the search space, with a greater average branching factor. This means that overall there is a larger variety in the solutions produced. However, the solutions produced are all concentrated in the same area of the search space. Alternative areas of the search space having no chance of being explored.

The findings show that the most significant aspect of an ACO algorithm structure is the type of ACO colonies that makes up the structure. The type of colonies present in a structure set the boundaries of the individual searches that are conducted by these colonies. Once the boundaries are set it is a matter of probability to find the best solutions within those boundaries.

For the Max-Min based algorithms no significant difference was found between the different information sharing structures that were experimented with. For the ant system the center multi colony ant system performed significantly worse than the other ant system algorithms. In both the central multi colony ant system(CMCAS) and the central multi colony Max-Min ant system(CMCMM) the Central colonies obtain a more diversified section of good solutions in the search space than their single colony counterparts. For the CMCMM this is likely because the central Max-Min colony does not stagnate, and therefore acts very similarly to a Max-Min colony that was not trained. However, why the CMCAS performs poorer than a single colony of the ant system is not clear.

10 Discussion

The progression of the ACO paradigm lies in finding solutions of more complex problems and the development of better algorithms that produce better solutions more efficiently. The purpose of this paper is to provide clarity on the workings of ant colony optimization algorithms and their progression through the search space of optimization problems. A better understanding of the theoretical framework and functioning of the ACO paradigm may aid in the development of more complex algorithms and the application of the ACO paradigm to a wider range of problems.

This paper has failed to produce effective multi colony information sharing structures that performed better than their single colony counterparts. But this does not mean that information sharing structures of this sort could not exist. Instead of passing solutions one could imagine a system in which trails are passed between colonies. We observe that the Max-Min ant system and the ant system generate trails differently. A multi colony system that uses both the ant system and the Max-Min ant system could be imagined. Passing trails between the ant system colony to the Max-Min ant system colony would increase the branching of the search in the Max-Min colony. Passing the trail from the Max-Min colony to the ant system colony would remove the colony out of its local minima and allow it to explore areas of the search space that are closer to the optimal.

None of the algorithms managed to find optimal

solutions. But this paper may aid in the development of algorithms that could do so. In this paper we observe the difference in trails and average branching factors between colonies where a single ant updates the trails and colonies where multiple ants update the trails. In colonies of the Max-Min ant system where only one ant updates the trails the average branching factor is lower than that of the ant system, where multiple ants update the trails. One could imagine a Max-Min ant system in which multiple ants update the trails instead of just a single best ant. Doing this would increase the average branching factor of the search and increase the probability of finding the optimal solution.

Another area of research would be to investigate how far global optima are in the fitness landscape from the local optima found by the algorithm. This may help in aiding the investigation of why the ACO algorithms studied do not find optimal solutions.

There are infinitely more possibilities and combinations in which ACO algorithms could be adapted and structures could be formed. The ACO paradigm as a whole is a wide field with a large number of applications. Therefore, understanding how ACO algorithms work will aid in understanding how to design ACO algorithms which are appropriate for a specific problem.

References

J. L. Bently. Fast algorithms for geometric traveling salesman problems. *ORSA Journal on Computing*, 4(4):387–411, 1992.

C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM computing surveys*, 35(3):268–308, 2003.

E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence : From Natural to Artificial Systems*. New York: Oxford University Press, 1999.

J. L. Deneubourg, S. Aron, S. Goss, and J. M. Pasteels. The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*, 3 (2):159–168, 1990.

W. Deng, G. Lin. Ant colony optimization-based algorithm for airline crew scheduling problem. *IEEE Transactions on Evolutionary Computation*, 6(4), 2002.

M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26:29–41, 1996.

M. Dorigo, M. Birattari, and T. Stützle. Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1(4):28–39, 2006.

L. M. Gambardella and M. Dorigo. Ant-Q: A reinforcement learning approach to the traveling salesman problem. *Machine Learning - International Workshop Then Conference*, pages 252–260, 1995.

T. Inkaya, S. Kayalgil, and N.E Özdemirel. Ant colony optimization based clustering methodology. *Applied Soft Computing*, 28:301–311, 2015.

D. S. Johnson. Local optimization and the traveling salesman problem. In Michael S. Paterson, editor, *Automata, Languages and Programming*, pages 446–461, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.

K. Khanna and S.M. Arora. Ant colony optimization towards image processing. *Indian Journal of Science and Technology*, 9(48), 2016.

M. Middendorf, F. Reischle, and H. Schmeck. Multi colony ant algorithms. *Journal of Heuristics*, 8: 305–320, 2002.

R.S. Parpinelli, H.S. Lopes, and A.A. Freitas. Data mining with an ant colony optimization algorithm. *IEEE Transactions on Evolutionary Computation*, 6(4), 2002.

Gerhard Reinelt. Tsplib, 1997. URL `http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html`.

T. Stützle and H.H. Hoos. Max-Min ant system. *Future Generation Computer Systems*, 16:889–914, 2000.

E. G. Talbi. *Metaheuristics : from design to implementation*. John Wiley Sons, 2009.

D. Whitley, D. Rana, and R. B. Heckendorn. The island model genetic algorithm: On separability, population size and convergence. *CIT Journal of Computing and Information Technology*, 7(1): 33–47, 1999.