On Optimal Parameters for Ant Colony Optimization Algorithms.

Conference Paper · January 2005

Source: DBLP

CITATIONS
60

READS
818

2 authors, including:

Keith Clark
Imperial College London
100 PUBLICATIONS **2,746** CITATIONS

SEE PROFILE

On Optimal Parameters for Ant Colony Optimization algorithms

Dorian Gaertner and Keith Clark
Dept of Computing, Imperial College London,
180 Queens Gate, London, SW7 2AZ, UK
{dg00,klc}@doc.ic.ac.uk

Abstract

Ant Colony Optimization (ACO) is a meta-heuristic introduced by Dorigo et al. [9] which uses ideas from nature to find solutions to instances of the Travelling Salesman Problem (TSP) and other combinatorial optimisation problems. In this paper we analyse the parameter settings of the ACO algorithm. These determine the behaviour of each ant and are critical for fast convergence to near optimal solutions of a given problem instance. We classify TSP instances using three measures of complexity and uniformity. We describe experimental work that attempts to correlate 'types' of TSP problems with parameter settings for fast convergence. We found these optimal parameter settings to be highly problem-specific and dependent on the required accuracy of the solution. This inspired us to explore techniques for automatically learning the optimal parameters for a given TSP instance. We devised and implemented a hybrid ACO algorithm, similar to the one independently developed in [16], which uses a genetic algorithm in the early stages to 'breed' a population of ants possessing near optimal behavioural parameter settings for a given problem. This hybrid algorithm converges rapidly for a wide range of problems when given a population of ants with diverse behavioural parameter settings.

Keywords: Artificial Intelligence, Optimization, Parameter Learning and Genetic Algorithms

1 Introduction

Algorithms based on the foraging behavior of ants have first been introduced by Dorigo in [6] and were formalized as a new meta-heuristic termed *Ant Colony Optimization* in 1999 [7, 8]. Instances of ACO have been applied extensively to a variety of discrete combinatorial optimization problems like the Travelling Salesman Problem [5, 11, 20], the Quadratic Assignment Problem [12, 15] and the Network Routing Problem[3]. More recently, the approach has been extended to continuous search domains [18].

The meta-heuristic is instantiated with several parameters which have to be set manually. Almost all publications in the area of ACO applications refer to Dorigo's seminal paper [10] when it comes to the selection of parameter values. This original paper describes the *Ant System* and analyses the relative merits of certain parameter settings but does not look at the interdependencies between parameters. It does however stress the importance of parameter settings for quick convergence towards the best known solution and mentions the dependency of the parameters on the problem.

We conducted an exhaustive, empirical analysis of the sensitivity of the ACO algorithm to variations of parameters for different *instances* of the TSP. This problem was chosen since there exists an enormous amount of literature to benchmark our results against. We developed a taxonomy of problem instances and investigated different *classes* of Travelling Salesman Problems in order to correlate their characteristics and their optimal parameters. No such correlation was found, though we discovered a dependency between the settings and the required solution accuracy.

The algorithm uses distance information and

pheromones as heuristics as will be explained in Section 3. In order to achieve more accurate solutions to a given TSP instance, a greater importance must be placed on the distance information. When sub-optimal solutions suffice, focussing on the pheromone heuristic will find good solutions faster.

A novel algorithm was then established that eliminates the need to know the best parameters at compile-time. This algorithm, while developed independently, verifies the results of Pilat et al. [16].

The remainder of this paper describes our classification of the TSP instances in Section 2, the ACO meta-heuristic in Section 3 and optimal parameters for one particular TSP instance in Section 4. Section 5 analyses the dependency between problem instances and optimal parameter combinations and Section 6 introduces a hybrid algorithm that adopts the parameters at run-time. Finally, in Sections 7 and 8 we highlight related work and present our conclusions.

2 TSP classifications

The Travelling Salesman Problem (TSP) defines the task of finding a tour of minimal total cost given a set of fully connected nodes and costs associated with each pair of nodes. The tour must be closed and contain each node exactly once.

Instances of the TSP come in many different types, such as symmetric (euclidean or non-euclidean), asymmetric, dynamic and special (BWTSP [2], RATSP [14]) TSPs. But even within the class of symmetric euclidean instances, where distance between two cities is taken to be the geometric distance between them, differences can be found.

The first property, indicating the complexity of a problem instance, is the number of cities involved. The bigger that number, the less feasible exhaustive search of the space of all tours will be and the longer[1] it will generally take to find optimal solutions.

However, a *five by six* grid with 30 cities may be solved faster than a 20-city problem where the cities are placed randomly. Looking at the usual statistic measures like mean, median or mode of distances one realises that they are of little use when classifying problem instances. Scaling an instance will change these three metrics but should not affect any of the metrics in the proposed classification.

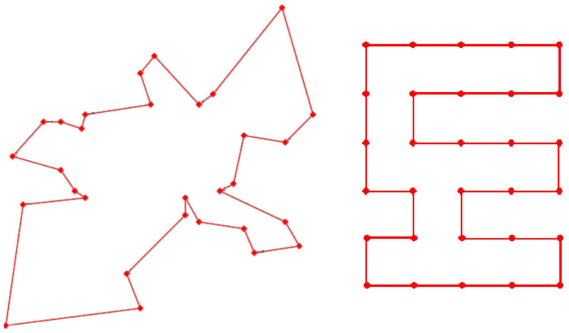

Figure 1: Best known solutions for two problem instance of the TSP—*Oliver30* and a *five by six* grid.

A property that takes the uniformity of the problem instance into account is termed normalised nearest-neighbour distances (nNNd) by the authors of this paper. For each city, the distance to the closest neighbouring city is determined and all these values are normalised to eliminate differences due to scaling. The standard deviation of the nNNds is then taken as an indication of how uniform the problem is. In the *five by six* grid shown on the right in Figure 1, the distances between each of the 30 cities and their nearest neighbours are equal and their standard deviation is zero. For a randomly generated problem this will not be the case and the nNNds will follow a normal distribution.

Figure 2: The *Dorian30* problem instance consisting of two clusters of fifteen cities each. Only two edges connect the clusters in the best solution.

A final property that looks beyond the nearest neighbours is the *coefficient of variation V* which takes all distances into account. This is important,

[1] *Speed* refers to the number of iterations the algorithm takes to find a solution of a specified quality.

since for a clustered problem instance like *Dorian30* in Figure 2, the previous metric neglects the inter-cluster edges. The coefficient of variation of a set of numbers X is defined as their standard deviation divided by their mean. It gives an indication of the relative magnitude of variation in relation to the mean of set X. The coefficient of variation is sometimes equated with the absolute relative deviation w.r.t. the mean expressed in percentage:

$$V = 100 * \left| \frac{Std.dev.}{mean} \right| percent \quad (1)$$

In Figure 2 we show a problem instance, termed *Dorian30*, where the standard deviation of the normalised nearest-neighbour distances is the same as that of a grid instance. However, the larger variation coefficient distinguishes it from a regular grid. The higher the value for this third metric, the more clustered the problem instance. The following table contrasts the metrics for several TSP problem instances.

Table 1: Metrics for selected problem instances

Instance	Cities	σ of nNNd's	V
Oliver30	30	0.16842	0.48008
Dorian30	30	0	0.71904
Grid-5x6	30	0	0.47734
Grid-7x7	49	0	0.47228
Random	50	0.08327	0.47093
Eilon50	50	0.04487	0.46251

While this is a naive approach to TSP classification, all three proposed metrics are not susceptible to scaling operations and distinguish different problem instances in the desired way.

3 ACO and its parameters

In 1996, the Ant System [10] was a novel heuristic designed to solve optimisation problems based on foraging behaviour found in nature. Many variations of it have been developed since, leading to the formulation of *Ant Colony Optimisation* (ACO) [8]. This new meta-heuristic abstracts away from particular implementations both of the algorithm and the problem instance under consideration.

We will briefly introduce one instance of ACO based on the *Ant System* (AS) [10] and the *Ant Colony System* (ACS) [5], which we chose in order to solve instances of the TSP[2]. This instance of the meta-heuristic uses a parallel search over a solution space using a memory structure that encodes *pheromones*. These were deposited on (and therefore reinforce) parts of the solution space that had previously been found to be of good quality.

At each time t, an ant in city i has to choose the next city j it goes to, out of those cities that it has not already visited. The probability of picking a certain city j is biased by the distance between i and j and the amount of pheromone on the edge between these two cities. Let τ_{ij} denote the amount of pheromone (also called trail) on the edge between cities i and j and let η_{ij} be the visibility of j from i defined as

$$\eta_{ij} = \frac{1}{distance(i,j)} \quad (2)$$

Then the bigger the product of τ_{ij} and η_{ij}, the more likely it is that j will be chosen as the next city. The trail and visibility are now weighted by parameters α and β and we arrive at the following formula (where $p_{ij}(t)$ is the probability of choosing city j from city i at time t and alw_k is the set of cites that are still allowed (unvisited) for ant k:

$$p_{ij}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha} * [\eta_{ij}]^{\beta}}{\sum_{k \in alw_k} [\tau_{ik}(t)]^{\alpha} * [\eta_{ik}]^{\beta}} & if j \in alw_k \\ 0 & otherwise \end{cases} \quad (3)$$

This probabilistic choice however does not guarantee that the optimal solution will be found. In some cases, a slightly worse than optimal solution may be found at the very beginning and some sub-optimal arcs may be reinforced by deposited pheromones. This reinforcement can lead to stagnation behaviour resulting in the algorithm never finding the best solution.

An improved performance is achieved by modifying the selection rule of the Ant System (Equation (3) above). An additional parameter q_0 with $0 \leq q_0 \leq 1$ is used in the ACS to control the level of exploration undertaken by the ants. If a random number q that

[2]The algorithm we propose is closer to AS but contains the pseudo-random-proportional rule from ACS—also, for simplicity, we chose to ignore the local updates to the pheromone matrix introduced by ACS.

is uniformly distributed over [0,1] is less than q_0 then we just use rule 3 to choose the next city probabilistically. However, if q is greater or equal to q_0, we deterministically pick the city j for which the weighted product of trail level and visibility is highest[3].

After all of the ants have completed their tours, the trail levels on all of the arcs need to be updated. The evaporation factor ρ ensures that pheromone is not accumulated infinitely and denotes the proportion of 'old' pheromone that is carried over to the next iteration of the algorithm. Then for each edge the pheromones deposited by each ant that used this edge are added up, resulting in the following pheromone-level-update equation:

$$\tau_{ij}(new) = \rho * \tau_{ij}(old) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \qquad (4)$$

where m is the number of ants and $\Delta\tau_{ij}^{k}$ is the amount of pheromone deposited by ant k onto the edge from i to j at the current iteration. This amount is based on a constant Q divided by the length of the tour found by ant k denoted L^k.

4 Optimal parameters for one TSP instance

The previous section described parameters $\alpha, \beta, \rho, q_0, m$ and Q, some of which can be fixed without affecting the validity of this research[4]. The comparative weight of τ_{ij} and η_{ij} in Equation (3) can be adjusted by fixing α and only varying β. Furthermore, Q is constant and the number of ants m can reasonably be set to the number of cities in the given TSP instance leaving three parameters to optimize—β, ρ and q_0.

Initial empirical tests verified Dorigo's claim that differing parameter combinations will vary the performance of the algorithm. We therefore tried to find the optimal combination of parameters for one particular instance of the TSP, Oliver30, defined in [19] and depicted in the left part of Figure 1.

In Table 2 we define a range for each of the three parameters and partition these ranges into fourteen, nine and eleven discrete values for parameters β, ρ and q_o, respectively. This yielded 1,386 different parameter combinations and we ran the algorithm ten times for each of these combinations. Each run was limited to 1,000 iterations but was stopped when the best known solution was found. It took several weeks (on a shared server with two dual 2.2GHz processors and 2GB RAM) to execute all 13,860 runs of this ACO algorithm for our multi-agent system implementation in the higher-order functional-imperative hybrid programming language April [4].

Table 2: The variable parameters

Parameter	Value Range
β	$0 < \beta < 15$
ρ	$0 < \rho < 1$
q_0	$0 \leq q_0 \leq 1$

This exhaustive search for the best parameter combinations was useful as a benchmark for evaluating the algorithm proposed in Section 6. Looking at the ten best parameter combinations we found the following settings to be optimal:

$$\beta = 6, \rho = 0.6, q_0 = 0.2 \qquad (5)$$

This combination took on average 34.2 iterations to find the *best known solution*—however, in many situations sub-optimal solutions of a high quality are sufficient. We therefore analysed the output files to determine which parameter combinations led to a solution within 1% of the best known solution using the fewest number of iterations. We found:

$$\beta = 10, \rho = 0.6, q_0 = 0.3 \qquad (6)$$

Finally, we investigated which combinations achieve results within 5% of the best known solution, finding:

$$\beta = 12, \rho = 0.6, q_0 = 0.2 \qquad (7)$$

Each of the combinations (5), (6) and (7) above are the averages of the best ten combinations for each respective task (0%, 1% and 5%), but also appeared as individual combinations within their respective top tens. For more detailed results, the reader is referred to our analysis in [11].

[3] This is opposite to the way q_0 is used in ACS.

[4] We leave out some of the less relevant parameters like the initial pheromone level τ_0 or the number of elitist ants γ which have been looked at elsewhere.

Our findings differ substantially from the standard parameter combination proposed by Dorigo. In [10], the Ant System is parameterised with $\beta = 5$ and $\rho = 0.5$, which is still quite similar[5]. However, in the paper that introduces ACS [5], the authors set $\beta = 2, \rho = 0.1$ and $q_0 = 0.9$ (equivalent to $q_0 = 0.1$ in our ACO version, due to the opposite way in which we treat q_0). Using this parameter combination, they report that the algorithm without local optimisation[6] fails to find the best known solution in all 25 test runs of 2,500 iterations each. Setting the parameters in question to our optimal values for the Oliver30 problem instance, the best known solution is found on average after 34.2 iterations without the use of a local optimiser.

The Eilon50 problem instance from [19] was another example, where using our method of finding optimal parameters results in improved solutions. The original ACS in [5] finds 427.965 as the shortest tour, whereas using optimal parameters, the algorithm finds an improved tour of length 427.855.

5 Parameter dependencies

The results in the previous section suggest that—for the Oliver30 problem instance—ρ is robust at a value of 0.6 and q_0 at a value of 0.2. Parameter β varies between 6 and 12 with lower values being more appropriate when more accurate solutions are needed.

This phenomenon can be understood using Equation 3. Parameter β regulates the importance of visibility η_{ij} where a high value for β gives high importance to the distance heuristic. This is due to the increased difference between different visibility values, as one referee pointed out. If, in an exemplary situation, there are two choices, one with $\eta_{ij} = 0.4$ and one with $\eta_{ij} = 0.5$, then setting $\beta = 2$ changes these values to 0.16 and 0.25, respectively. While 0.16 and 0.25 are lower in absolute value, the difference between the normalised values ($\frac{0.16}{0.41}$ and $\frac{0.25}{0.41}$) is larger than the difference between the normalised original values ($\frac{0.4}{0.9}$ and $\frac{0.5}{0.9}$). Hence, choosing the path with $\eta_{ij} = 0.5$ has a higher probability if a higher value of β is used. A high β value, changes the algorithm to something that more closely resembles a greedy search and is therefore not appropriate when the optimal solution is required.

The next question we tried to answer is concerned with the generality of the optimal parameter combination. We ran similar tests on a variety of other well-known TSP instances as well as random ones and found that different problem instances have different optimal parameter combinations.

Given this fact, we tried to establish a correlation between any of the metrics defined in Section 2 and any of the three parameters under investigation. We repeated our search for optimal parameters for problem instances that only varied in one metric. For example, keeping the number of cities constant, we compared the Oliver30 problem, a random 30-city instance and a *five by six* problem, where 30 cities are aligned on a grid.

Unfortunately, there was no statistically significant correlation to be found. Generally, the optimal value for ρ was around 0.6 but the other two parameters varied considerably. For example, problem instances where the cities are aligned on a grid, have a wide range of equally good parameter combinations. Random problem instances, on the other hand, require parameter combinations from a much narrower range of values.

6 Solution - GMACS

We showed in Section 4 that optimal parameters improve the performance of the algorithm. However, Section 5 revealed that no single combination of parameters is optimal for all TSP instances and that it is seemingly unfeasible to determine the optimal combination by looking at the characteristics of the problem instance. The cost of finding the optimal parameters experimentally, using an exhaustive search of the parameter space, is prohibitive.

We therefore modified the ACO instance using ideas from *Genetic Algorithms* to develop a *Genetically Modified Ant Colony System* (GMACS) [11]. In this algorithm, every ant is initialised with a random parameter combination where the parameter values are chosen from a sensible range. Over time, the entire population of ants evolves, breeding ants with better parameter combinations which find improved

[5] q_0 does not exist in Ant System which can be simulated by setting $q_0 = 0$ in ACS

[6] local optimisation here, is a process that attempts to improve a tour by swapping cities, for example

solutions to the given TSP instance. This section will present the issues that arose and the decisions that were made during the implementation process.

The general system is designed as follows. Many ant processes communicate with the environment process which uses a genetic engine to breed ants and replace the old generation of individuals with a new one. Conceptually, the genetic engine operates on the population of ants. However, for reasons of efficiency, the environment process mediates between the ant threads and the genetic engine.

We chose a real-valued encoding to represent an individual ant and fitness-proportionate selection as a method to determine individuals that are qualified to produce offspring. The worth $g(i)$ of an individual ant i is defined as the tour length found by the best greedy algorithm minus the tour found by ant i in the previous iteration of the algorithm. The fitness of ant i is then taken to be its worth divided by the average worth of the population.

For recombination we used a tournament method where two individuals are picked and the fitter one is chosen as a parent, then two more individuals are picked and the fitter one of those is chosen as the second parent. The two parents mate using uniform crossover. This is the only sensible way to recombine non-bitstring encoded individuals which have very short chromosomes. Each gene of the single offspring is selected randomly from the corresponding genes of the parents and then subjected to mutation with a small probability of 0.1. This is much higher than the standard choice of 0.001 but justified by the short chromosomes used for the encoding of the parameters. If a gene was chosen to be mutated, then the parameter value that is represented by the gene was subsequently increased or decreased by about 5%.

Further issues, like the generation gap and scaling of the fitness ranges, have also been investigated and the reader is referred to [11] for further details. We compared GMACS to the ACS with standard parameters and found that it performs equally well in most cases and slightly better in some cases. The computational overhead is negligible and Table 3 shows exemplary[7] evidence for the Eilon50 TSP instance:

[7]due to space restrictions

Table 3: Comparison of different ACO algorithms

Algorithm	Average tour length
ACS with Dorigo's params	434.44
GMACS	430.84
ACS with optimal params	429.21

7 Related Work

Surveying the literature, one has to point out, that the area of ant algorithms has matured a lot. Since its inception a decade ago, four international workshops have taken place, the last two of which have published their proceedings. Algorithms based on the ACO meta-heuristic have been applied to many different problems, but research on optimising ACO parameters has been sparse.

The issue of finding the best parameters was tackled by several researchers. Pilat et al. independently took a similar approach in [16] using genetic algorithms to optimise the ACS algorithm for the TSP. Our paper provides a motivation to optimise the parameter settings. We used exhaustive search and analysis to try and correlate certain types of problem instances with particular parameter combinations.

Another paper that tackles the problem of parameter selection is [17], which uses an ACO algorithm to find the best ants which then use an ACO algorithm to find the best tour. The author of this work modified ACS in such a way that it evolves parameters based on an extra pheromone matrix maintained solely for this purpose.

Guntsch and Middendorf in [13] describe a population based approach for ACO where all pheromone information corresponds to solutions that are members of the actual population. This differs from our solution where *ants* are the members of the population. Another attempt to combine genetic algorithms with ideas from ACO was termed GAACO by Acan in [1]. He uses both algorithms in parallel using the same problem representation for both and allowing solutions to migrate from one algorithm to another.

8 Conclusions and Future Work

In this paper, we showed that the performance of ACO algorithms depends on the appropriate setting of parameters which requires both human experience and luck to some extend. These parameters are dependent on the problem instance at hand and also on the required solution accuracy.

Combining ideas from genetic algorithms with the Ant Colony System, we showed, that manual setting of parameters is not required to achieve good performance of the algorithm. Starting with random parameter combinations from a sensible range, our GMACS algorithm evolves the parameters at runtime. The evolved parameters are usually similar to the standard ones, but yield improved solutions for certain problem instances.

In the future, we plan to apply the GMACS algorithm to other combinatorial optimisation problems to see whether we can reproduce the results we obtained for the Travelling Salesman Problem. Another area of interest is behavioural modelling of ants as agents where β could be interpreted as *trust* and q_0 as the ant's *attitude towards risk*.

We are also interested to see how GMACS performs in dynamic settings such as a TSP with changing distances between cities.

References

[1] Adnan Acan. GAACO: A GA + ACO hybrid for faster and better search capability. In *Ant Algorithms*, pages 300–301, 2002.

[2] Mélanie Bourgeois, Gilbert Laporte, and Frédéric Semet. Heuristics for the black and white traveling salesman problem. *Comput. Oper. Res.*, 30(1):75–85, 2003.

[3] Gianni Di Caro and Marco Dorigo. AntNet: Distributed stigmergic control for communications networks. *Journal of Artificial Intelligence Research*, 9:317–365, 1998.

[4] Keith L. Clark and Francis G. McCabe. April - Agent PRocess Interaction Language. In *Intelligent Agents - ECAI-94 Workshop on Agent Theories, Architectures, and Languages*, Lecture Notes in Computer Science, pages 324–340. Springer, 1994.

[5] M. Dorigo and L. M. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE transactions on Evolutionary Computation*, 1:53–66, 1997.

[6] Marco Dorigo. *Ottimizzazione, Apprendimento Automatico, ed Algoritmi Basati su Metafora Naturale*. PhD thesis, Politecnico di Milano, 1992.

[7] Marco Dorigo and Gianni Di Caro. Ant colony optimization: A new meta-heuristic. In Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 2, pages 1470–1477. IEEE Press, 1999.

[8] Marco Dorigo and Gianni Di Caro. *New Ideas in Optimization*, chapter The Ant Colony Optimization Meta-Heuristic, pages 11–32. McGraw-Hill, 1999.

[9] Marco Dorigo, Gianni Di Caro, and Luca M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5(2):137–172, 1999.

[10] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. The ant system: Optimization by a colony of coorperating agents. *IEEE Trans. on Systems, Man and Cybernetics-Part B*, 26(1):29–41, 1996.

[11] Dorian Gaertner. Natural algorithms for optimisation problems. Masters thesis, Imperial College London, 2004.

[12] L. M. Gambardella, E. Taillard, and M. Dorigo. Ant colonies for the QAP. *Journal of the Operational Research Society*, 1998.

[13] Michael Guntsch and Martin Middendorf. A population based approach for ACO. In Stefano Cagnoni, Jens Gottlieb, Emma Hart, Martin Middendorf, and Günther Raidl, editors, *Applications of Evolutionary Computing, Proceedings of EvoWorkshops2002: EvoCOP, EvoIASP, EvoSTim*, volume 2279, pages 71–80, Kinsale, Ireland, 3-4 2002. Springer-Verlag.

[14] Vicky Mak. *On the Asymmetric TSP with Replenishment Arcs*. PhD thesis, University of Melbourne, Australia, 2001.

[15] Vittorio Maniezzo and Alberto Colorni. The Ant System applied to the Quadratic Assignment Problem. *Knowledge and Data Engineering*, 11(5):769–778, 1999.

[16] Marcin L. Pilat and Tony White. Using genetic algorithms to optimize ACS-TSP. In *ANTS '02: Proceedings of the Third International Workshop on Ant Algorithms*, pages 282–287. Springer-Verlag, 2002.

[17] Marcus Randall. Near parameter free ant colony optimisation. In *ANTS Workshop*, pages 374–381, 2004.

[18] Krzysztof Socha. ACO for continuous and mixed-variable optimization. In *ANTS Workshop*, pages 25–36, 2004.

[19] T. Starkweather, D. Whitley, and D. Fuquay. Scheduling problems and travelling salesman: the genetic edge recombination operator. In J. David Schaffer, editor, *Proceedings of the 3^{rd} international conference on genetic algorithms*. Morgan Kaufmann, 1989.

[20] Thomas Stützle and Holger Hoos. Improvements on the ant-system: Introducing the MAX-MIN ant system. In *Proceedings International Conference on Artificial Neural Networks and Genetic Algorithms*. Springer, 1997.