Multiple Ant Colony Systems for the Busstop Allocation Problem

Jasper de Jong [a] Marco Wiering [ab]

[a] University of Utrecht, Cognitive Artificial Intelligence
[b] University of Utrecht, P.O.Box 80.089, 3508TB Utrecht

Abstract

This paper introduces Multiple Ant Colony Systems, an optimization algorithm based on the Ant Colony System. The performance of this new algorithm is compared with the performance of a Greedy Algorithm and Simulated Annealing on a new optimization problem called the Busstop Allocation Problem (BAP). In the BAP, the goal is to construct a set of buslines (sequences of busstops) so that the average travel time of all passengers is minimized. Results show that the new algorithm outperforms the Greedy Algorithm and Simulated Annealing, which indeed makes it a promising acquisition to the range of existing Ant Algorithms.

1 Introduction

The research for Ant Algorithms is a growing field. These algorithms can be used to solve discrete combinatorial optimization problems using an artificial colony of ants. Previous results [1, 2, 6] have shown that Ant Algorithms are highly competitive with algorithms such as Tabu Search [8], Genetic Algorithms [9], and Simulated Annealing [11]. In this paper we introduce a new member of the class of Ant Algorithms, called Multiple Ant Colony Systems (MACS). The MACS algorithm can be used to solve particular combinatorial optimization problems which can be naturally modelled using a number of colonies consisting of a number of agents. Agents from different colonies collectively create solutions. An example of such a problem is multiple vehicle routing, in which multiple vehicles are working together to find a global solution to the problem at hand. To solve such problems, MACS uses several ant colonies which cooperatively search for a solution. Tackling such problems with conventional Ant Algorithms appears to be much less natural. We will use MACS to solve the Busstop Allocation Problem (BAP). In the BAP, we have to construct a set of buslines in such a way that the passengers waiting at the busstops will have a minimal average travel time for going to their destination. This problem can be naturally modelled and solved by the MACS algorithm.

This paper is organised as follows. In Section 2 we present the Busstop Allocation Problem. Section 3 shortly describes Ant Algorithms. The Multiple Ant Colony Systems algorithm is presented in Section 4. Then, we compare the performance of the new algorithm with two other algorithms, a Greedy Algorithm and Simulated Annealing, and show the test results in Section 5. Conclusions are described in Section 6.

2 Busstop Allocation Problem

A Busstop Allocation Problem (BAP) is the formalisation of the problem that arises when trying to construct m buslines, each one consisting of a sequence of busstops. In total there are n busstops, and one of the busstops will be the main busstop. This main busstop represents the central station. In our model all buslines will have to call at the main busstop. A solution to the BAP will be a collection of buslines. The main busstop is allocated to all buslines and all other busstops are allocated to precisely one busline. Thus, a solution requires: (1) a partitioning of the $n-1$ busstops to m buslines, and (2) an ordering of each subset of busstops in which the main busstop must also be included. In our model two busses drive on each busline in different directions starting from both ends of each busline. All $2m$ busses in the city start at the same time step (after a specific time interval the busses go for their next round). Once a bus passes the central station, passengers going to a busstop which is not allocated to their current busline exchange busses (after waiting until the other bus arrives). Formally the problem can be defined by two matrices:

$D = \{d_{ij}\}$ = Euclidean distance between busstop i and busstop j.
$T = \{t_{ij}\}$ = passengers waiting at busstop i having busstop j as destination.

Once a solution is generated we want to evaluate this solution to be able to compare this solution with other solutions. For this purpose we will use the average travel time (ATT). To compute the ATT, we first compute the travel time for one passenger: which is u_{ij} if i and j are allocated to the same busline and $u_{iz}+O+u_{zj}$ if i and j are allocated to two different buslines, where u_{ij} is the time needed for the bus to travel from busstop i to busstop j, z is the main busstop and O the time needed to change busses. Then we compute the ATT by averaging over all passengers. The lower this ATT is, the faster people can get (on average) from their start busstop to their destination busstop. The goal is to minimize the ATT.

3 Ant Algorithms

Foraging ants deposit a chemical substance called pheromone as they move from the nest to a food source and vice versa, which other foragers follow. This collective foraging behaviour enables ants to find the shortest path from the nest to some food source. Optimization algorithms inspired by the collective foraging behaviour of ants are called Ant Algorithms [4]. There is a wide variety of ant algorithms, some of them combine local search in their way of finding solutions, but all these algorithms share some basic properties: (1) They consist of an artificial colony of cooperating ants, (2) Ants make discrete moves, (3) Ants lay down pheromone on their chosen paths, and other ants use these pheromone trails in their local decision policies. The first Ant Algorithm was the Ant System (AS). The first use of the AS was on the Traveling Salesman Problem (TSP) [7].

Ant System. An AS is a collection of artificial ants. During one iteration each ant builds one complete tour (solution to the TSP). It does this by moving from one city to another until all cities have been visited. While moving from one city to another, ants prefer cities that are connected by short edges containing a high amount of pheromone. All edges of the graph will be updated at the end of each iteration. This updating causes all ants to alter the amount of pheromone on the edges they have travelled. Edges that did not belong to any ant's tour will loose some pheromone because of evaporation. Ants that constructed a short tour will deposit a relatively large amount of pheromone on the visited edges while ants that constructed a long tour will deposit a smaller amount of pheromone on the visited edges. AS has more recently been used on the Quadratic Assignment Problem [12] and the Vehicle Routing Problem [2].

Ant Colony System. To improve upon the results of the Ant System on various optimization problems, the Ant Colony System (ACS) was developed. The first use of the ACS was also on the TSP [5, 6]. An ACS is a colony of ants. Just as in the AS, ants form solutions by moving from one city to another city until all cities have been visited. They still prefer cities that are connected by short edges containing a high amount of pheromone. In the ACS, however, edges are updated after an ant has travelled an edge. This local updating decreases the amount of pheromone (pheromone evaporation) on the just travelled edge. This will ensure that not all ants will search in the narrow neighbourhood of the best tour. When all ants have constructed their tours, a global update will be performed. The global update causes pheromone evaporation on all edges, except on the edges belonging to the global best tour which will receive some pheromone.

4 Multiple Ant Colony Systems

Because a solution to the BAP consists of a collection of buslines and each busline consists of a sequence of busstops it seems natural to try to solve a BAP with several cooperating ACSs. We have therefore developed an algorithm called Multiple Ant Colony Systems (MACS). This algorithm is based on the workings of the ACS but is developed in such a way that each busline is represented by a separate ACS. This means that the pheromone levels of each ACS are updated separately. Given a BAP consisting of n busstops and m buslines we initialise a MACS consisting of m ACSs. All ACSs will consist of an equal number of ants r, which are numbered from 1 to r. At the end of each iteration, r solutions (each consisting of m buslines) will have been constructed.

We define a fully connected graph $G = (V, E)$ where V is the set of vertices (each vertice represents a busstop) and E is the set of edges. Each edge possesses information about its length and the level of pheromone of each separate colony. Just as in the ACS, solutions are built by ants moving from one busstop to another. Each ant will build a busline. This means that m ants with the same number collectively construct a complete solution (whereby each of them is solving one of the m sub-problems). Because there are now several colonies of ants, all ants with the same number of each colony will choose a busstop to move to and only the most promising ant (to be defined later) is allowed to extend its busline.

In the original ACS algorithm, start locations were chosen at random. We developed the MACS algorithm in such a way that it can account for the fact that some busstops will be better busstops to start a busline with than others. To model this, we extend the graph G by introducing a source node s which is connected to each busstop (with equal, negligible length). Instead of choosing the busstop to start a busline with at random, ants start at the source node and the pheromone level on the edges originating from the source node will influence the probability of choosing a busstop as the starting busstop.

We will call η^l_{ij} the visibility of edge (i,j) of colony l. The visibility is defined as the inverse of the time in seconds to get from busstop i to busstop j [1]. The amount of pheromone on the edge (i,j) of colony l is denoted by τ^l_{ij}. If ant k of colony l is at busstop i (respectively at source node s) it uses equation 1 (respectively equation 2) to decide what the next busstop (j) will be. This next busstop should be a member of the set of busstops J_k, the busstops which are still unvisited by all k-th ants of all colonies constructing a complete solution together. When ant k of colony l has not yet visited the main busstop, J_k will include the main busstop to ensure that all ants will at some point visit the main busstop.

$$j = \begin{cases} \arg\max\limits_{h \in J_k} \{[\tau^l_{ih}] \cdot [\eta^l_{ih}]^\beta\} & \text{if } q \leq q_0 \quad \text{(exploitation)} \\ S & \text{otherwise} \quad \text{(biased exploration)} \end{cases} \tag{1}$$

Here q is a random number ($0 \leq q \leq 1$), parameter q_0 ($0 \leq q_0 \leq 1$) determines the relative importance of exploitation versus exploration (if q_0 is set high, there will be more exploitation), and S is a random busstop, selected according to the probability distribution given in equation 2.

$$p^{kl}_{ij} = \begin{cases} \frac{[\tau^l_{ij}] \cdot [\eta^l_{ij}]^\beta}{\sum_{h \in J_k} [\tau^l_{ih}] \cdot [\eta^l_{ih}]^\beta} & \text{if } j \in J_k \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

In both equation 1 and equation 2, parameter β controls the importance of visibility versus pheromone. If this is set to zero, no a-priori knowledge (the preference for short edges) will be used. Empirical studies have shown that the use of visibility leads to much faster convergence to good solutions, however.

Each time an ant has moved from some busstop i to some busstop j, it changes the pheromone levels of edge (i,j) of its colony by applying the local update rule of equation 3.

$$\tau^l_{ij} = (1 - \rho) \cdot \tau^l_{ij} + \rho \cdot \tau_0 \tag{3}$$

Parameter ρ ($0 \leq \rho \leq 1$) represents the evaporation rate and τ_0 is the initial

[1]The time to get from one busstop to another is computed by dividing the distance by the average busspeed.

pheromone level. Each time an ant has travelled along an edge, the pheromone level of this edge will drop a little. When all solutions have been made, pheromone levels of edges belonging to the global best solution (the best solution the algorithm has found thus far) will increase a little while pheromone levels of all other edges will decrease a little. The global update is done by applying equation 4.

$$\tau_{ij}^l = (1-\alpha)\cdot\tau_{ij}^l + \alpha\cdot\Delta\tau_{ij}^l \tag{4}$$
$$\text{where } \Delta\tau_{ij}^l = \begin{cases} (L_{gb})^{-1} & \text{if edge (i,j)} \in \text{colony } l \text{ of } S_{gb} \\ 0 & \text{otherwise} \end{cases}$$

Parameter α ($0 \leq \alpha \leq 1$) represents the learning rate. The global best solution is represented by S_{gb} and the ATT of the global best solution is represented by L_{gb}.

The pseudo code of a MACS is shown in figure 1. S_k denotes the solution formed by all k-th ants of all colonies and L_k denotes the ATT of that solution. Initially all edge pheromone levels of all colonies are set to τ_0. Now, for a number of predefined iterations, the algorithm will produce r (the number of ants) solutions. When expanding a partial solution in exploitation mode, the most promising ant is the ant with the largest arg max value for going to its next busstop. In exploration mode, the most promising ant is chosen randomly.

5 Test results

We have developed 4 different instances of a BAP. Problem 1 consists of 12 busstops and 2 buslines, problem 2 consists of 20 busstops and 3 buslines, problem 3 consists of 25 busstops and 4 buslines, and problem 4 consists of 30 busstops and 4 buslines. The performance of MACS has been compared with two other algorithms.

The first algorithm is a straightforward Greedy Algorithm (GRA). This algorithm starts with a number of random solutions and selects each iteration the best possible move for each solution. The GRA thus carries each random solution to its local minimum. The implementation of this algorithm is based on the implementation of the Multi Greedy algorithm for the QAP as described in [3].

The second algorithm is the well known Simulated Annealing (SA) [11]. This algorithm has a parameter that represents the temperature. The temperature is set to an initial value at the start of the algorithm and it will decrease according to a cooling rate as the algorithm continues. The algorithm starts with a random solution and constructs new solutions by performing random swaps. As the temperature decreases, the probability that a worse solution replaces the current solution will also decrease. The implementation of this algorithm is based on the implementation of the SA algorithm for the QAP as described in [3].

Obviously the number of colonies in an MACS is the same as the number of buslines used to solve the problem. During all tests the number of ants was equal to the number of busstops and the initial pheromone levels were set to the inverse of the product of the number of busstops and the best time found by the Greedy Algorithm [2]. All results were obtained during 10 test runs. All algorithms

[2] Lots of testing revealed to us that the initial amount of pheromone is quite important.

```
1  Set $t = 0$
   For $l = 1$ to $m$ do
      Set all edge pheromone levels $\tau_{ij}^l$ to $\tau_0$

2  For $l = 1$ to $m$ do
      For $k = 1$ to $r$ do
         Choose busstop $j_{start}^l$ with probability $p_{sj_{start}^l}^{kl}$ given by equation 2
         $J_k = \{j1, ..., jn\} - j_{start}^l$
         Update pheromone level of edge $(s, j_{start}^l)$ using equation 3

3  For $w = 1$ to $n - 1$ do
      For $k = 1$ to $r$ do
         Choose random $q$ $(0 \leq q \leq 1)$
         If $q \leq q_0$ then
            For $l = 1$ to $m$ do
               Store busstop with its value using equation 1
            Choose busstop $j$ with largest arg max of all stored busstops
         Else
            For $l = 1$ to $m$ do
               Choose and store busstop using equation 2
            Choose busstop $j$ at random from stored busstops
         Set $J_k = J_k - j$ and $S_k = S_k + j$
         Perform local update on last edge according to equation 3

4  For $k = 1$ to $r$ do
      Compute $L_k$ for each solution $S_k$, and update $S_{gb}$ and $L_{gb}$ if $L_k < L_{gb}$
   For $l = 1$ to $m$ do
      For all edges (i,j) $\in S_{gb}$ do
         Update edge according to equation 4
   Set $t = t + 1$

5  If $t < t_{max}$ then
      Goto step 2
```

Figure 1: Multiple Ant Colony Systems pseudo code

were given approximately the same computational time during a run. For exact parameter settings, we refer to [10].

Results are shown in table 1. It is clear that the MACS algorithm performs a lot better than the other two algorithms. Results of the MACS algorithms were

Using the heuristic to initialize it to the best time of the Greedy Algorithm saves time exploring different parameter settings. Note that the algorithm cannot use this a-priori information in a very sensible way, such as: now the algorithm knows a particular reachable (upper) bound.

		GRA	SA	MACS
Problem 1	best	**351.181**	**351.181**	**351.181**
	av	358.036	363.758	**351.181**
	sd	6.36	11.34	0
Problem 2	best	474.118	465.773	**452.421**
	av	492.608	496.313	**465.424**
	sd	9.56	19.43	8.29
Problem 3	best	582.956	585.856	**558.245**
	av	606.695	605.922	**571.441**
	sd	13.15	15.62	6.29
Problem 4	best	719.250	707.963	**686.391**
	av	748.622	733.081	**700.767**
	sd	20.19	26.23	8.16

Table 1: Best ATT found, average ATT and the standard deviation values of the Greedy Algorithm, Simulated Annealing and Multiple Ant Colony Systems. Results obtained during 10 runs. Best results are in bold.

significantly better (t-test, $\alpha = 0.01$) on all problems.

6 Conclusions

We have proposed an optimization algorithm called Multiple Ant Colony Systems (MACS), which is based on the behaviour of ants. MACS is a natural extension of the Ant Colony System, which consists of multiple cooperating ant colonies. To test the performance of the MACS algorithm we have compared its performance with the performance of a Greedy Algorithm (GRA) and Simulated Annealing (SA) on 4 different instances of an optimization problem called the Busstop Allocation Problem (BAP).

The BAP is a new optimization problem consisting of n busstops and m buslines. A valid solution is a set of m buslines each one consisting of a sequence of busstops. Given a solution, we can compute the average travel time (ATT) which needs to be minimized.

The results showed that the MACS algorithm outperformed both GRA and SA on all test problems. Thus, the results demonstrate that the MACS is a promising new Ant Algorithm suitable for solving particular combinatorial optimization problems. In future work we intend to use MACS on different optimization problems such as multiple vehicle routing and job-shop scheduling problems.

Acknowledgments

Thanks to Marco Dorigo and Christian Blum for many helpful comments.

References

[1] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence, from Natural to Artificial Systems.* Oxford University Press, 1999.

[2] B. Bullnheimer, R. F. Hartl, and C. Strauss. An improved ant system algorithm for the vehicle routing problem. Technical Report POM Working Paper No. 10/97, University of Vienna, 1997.

[3] A. Colorni, M. Dorigo, and V. Maniezzo. Algodesk: an experimental comparison of eight evolutionary heuristics applied to the quadratic assignment problem. *European journal of Operational Research*, 81:188–204, 1995.

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

[5] M. Dorigo and L. M. Gambardella. Ant colonies for the traveling salesman problem. *Biosystems*, 43:73–81, 1997.

[6] M. Dorigo and L. M. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Trans. Evol. Comp.*, 1:53–66, 1997.

[7] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Trans. Systems, Man, and Cybernetics*, 26:29–41, 1996.

[8] F. Glover. Tabu search, part I. *ORSA Journal on Computing*, 1(3).

[9] J. H. Holland. *Adaptation in Natural and Artificial Systems.* Ann Arbor, MI:The University of Michigan Press, 1975.

[10] J. de Jong. Multiple ant colony systems for the busstop allocation problem, August 2001. Master's thesis, Cognitive Artificial Intelligence, University of Utrecht.

[11] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[12] V. Maniezzo, A. Colorni, and M. Dorigo. The ant system applied to the quadratic assignment problem. *IEEE Trans. Knowledge and Data Engineering*, 11, 1998.