Ant colony optimization algorithms for the bi-objective shortest path problem in sparse networks

Based on a paper by Ghoseiri and Nadjari (2010)

Arie Trouwborst

Supervisor: Y.N. Hoogendoorn

Second assessor: S. Sharif Azadeh

A thesis presented in fulfillment of the requirements for the

Bachelor Econometrics and Operational Research

Erasmus School of Economics

July 3, 2020

Abstract

The Ant Colony Optimization algorithm was developed as a heuristic method for solving the traveling salesman problem. Since then, it became used for solving many combinatorial optimization problems, among which the bi-objective shortest path problem. We test the quality of the ACO algorithm as used by Ghoseiri and Nadjari (2010) for this problem by comparing its results to those obtained by an exact method, the label correcting algorithm. We show that this algorithm produces formidable results within shorter running time for large grid networks. Moreover, a comparison with another ACO method shows that the technique used is one of the best in the field of the ACO algorithms.

Contents

1 Introduction

The shortest path problem, which is about finding the shortest path from one node to another, is applied to network analysis in many fields, ranging from communication and transportation networks to fluid flow networks. Many from these are multi-objective in nature, which is why researchers started studying the multi-objective shortest path (MOSP) problem. A bi-objective shortest path problem is a form of a MOSP problem which is in turn an extension of the traditional single-objective shortest path problem (SP) (Ghoseiri and Nadjari, 2010). Therefore, MOSP results from SP by considering multiple criteria for the edges (Paixão and Santos, 2013). While solving the single-objective shortest path problem results in one optimal path, the solution of its multi-objective counterpart is generally not unambiguous, as it is assumed that the multiple objectives have a conflicting nature. That is, the optimal path for one objective may differ from the optimal path for another objective (Sastry et al., 2003). This usually results in a set of Pareto-optimal paths, i.e. one cannot improve one objective function without harming another. We will discuss this more extensively in Section 2. The decision-maker has to decide between the non-dominated paths by weighing the different objectives (Modesti and Sciomachen, 1998).

While the classical shortest path problem is solvable in polynomial time, the same is not guaranteed for MOSP problems (Hansen, 1980). Garey and Johnson (1979) indeed showed that the number of Pareto-optimal paths can grow exponentially with the number of nodes, which makes MOSP problems NP-hard (Serafini, 1987). However, significant research has recently been devoted to developing different methods for solving MOSP problems, many of which are label-based algorithms. These methods are categorized into two broad groups, label setting and label correcting algorithms. The original label correcting algorithm was developed by Vincke (1975), while the first label setting algorithm was described by Hansen (1980). For an extensive comparison between these two families, the reader is referred to Zhan and Noon (2000). Skriver and Andersen (2000) ranked their label correcting algorithm as the best among the then existing MOSP algorithms in terms of computational performance, solving instances up to 1,000 nodes. That is why Ghoseiri and Nadjari (2010) compare their ant colony optimization algorithm to the label correcting algorithm with regard to their running times and Pareto-optimal frontier.

Ghoseiri and Nadjari (2010) propose an ant colony optimization (ACO) algorithm for solving a bi-objective shortest path problem. Ant colony optimization is a heuristic technique for solving hard combinatorial optimization problems within reasonable running time inspired by the behaviour of

ant colonies. While searching for food, the ants start exploring the area surrounding their nest in a random manner. If an ant has found a food source, it deposits a chemical substance called pheromone during the way back to the nest (Dorigo and Blum, 2005). As the ants using a shorter path return to the nest faster, pheromone accumulates at a higher rate on the shorter paths. Beckers et al. (1992) showed that the ants tend to find shortest paths between their nest and food sources by choosing the path with more pheromone with higher probability. As the deposited pheromone vaporizes over time, the longer paths containing lower levels of pheromone will eventually get out of use (Goss et al., 1989).

Exploiting this characteristic of real ants, Dorigo et al. (1996) first developed Ant System (AS). One year later, Dorigo and Gambardella (1997) improved the algorithm and renamed it to Ant Colony System (ACS). Although it was initially applied to the Traveling Salesman Problem, it has now applications to many other combinatorial optimization problems such as vehicle routing, scheduling, knapsack and graph colouring (Dorigo and Stützle, 2003). As the number and range of applications grew, it was soon used to tackle multi-objective optimization problems as well. Mariano and Morales (1999) used a preliminary version of AS to tackle a multi-objective water distribution problem. In the same year, Gambardella and Taillard (1999) developed a multiple ant colony system to tackle a bi-objective vehicle routing problem considering both the number of vehicles and the total travel time as different objectives. Two years later, Iredi et al. (2001) considered a multi-objective problem where the objectives could not be ranked by importance. They also first used a different approach to handle the multiple objectives, which will be discussed in Section 3. The same approach is also used by Ghoseiri and Nadjari (2010).

Section 2 will describe the aforementioned problem in mathematical terms, while Section 3 explores the methodology of the ACO algorithm as used by Ghoseiri and Nadjari (2010). The results of their paper are replicated and a number of extensions will be explored in Section 4.

2 Problem description

Let us now make a more formal definition of a MOSP problem. Let $G = (N, E)$ be a directed graph where N is the set of n nodes and E the set of directed edges connecting any pair of the nodes. More formally, $E = \{(i, j) : i, j \in N, i \neq j\}$. Associated with every edge (i, j) is a cost vector $\{c_{ij}^1, ..., c_{ij}^k\}$, where k represents the number of objectives. In the problem under consideration, these two objectives represent cost and time. The goal of MOSP is finding the optimal path from

source node s to sink node t with regards to the k objective functions. The bi-objective shortest path problem, where $k = 2$, can be formulated as follows:

$$\text{min.} \quad f^1(x) = \sum_{(i,j)\in E} c_{ij}^1 x_{ij} \tag{1}$$

$$\text{min.} \quad f^2(x) = \sum_{(i,j)\in E} c_{ij}^2 x_{ij} \tag{2}$$

$$\text{s.t.} \quad \sum_{j:(j,i)\in E} x_{ij} - \sum_{j:(i,j)\in E} x_{ij} = \begin{cases} -1 & \text{if } i = s \\ 0 & \text{if } i \neq s,t \\ 1 & \text{if } i = t \end{cases} \tag{3}$$

$$x_{ij} \in \mathbb{B} \tag{4}$$

In this x_{ij} is a binary decision variable indicating whether edge (i, j) is part of the suggested path. Ghoseiri and Nadjari use the following graph to illustrate that two objective functions may lead to different optimal paths.

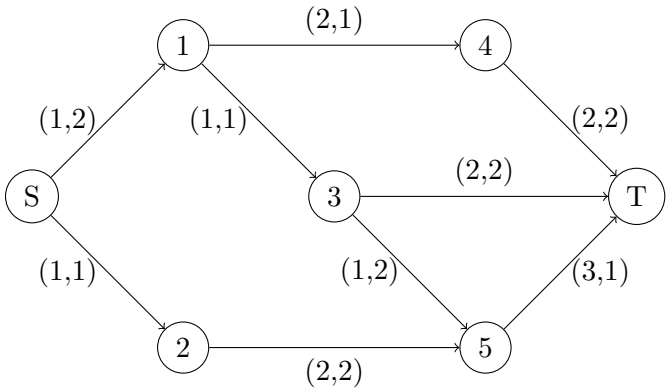


Figure 1: Multiple non-dominated solution in a network (Ghoseiri and Nadjari, 2010)

As follows from the graph, four paths exist from node S to T, two of which are dominated and two are non-dominated. Both path S-1-4-T and path S-1-3-5-T, associated with costs (5,5) and (6,6) respectively, are (strictly) dominated by path S-1-3-T (4,5). Path S-2-5-T (6,4) is however better than path S-1-3-T (4,5) in terms of c_2, but worse in terms of c_1. Therefore, both paths are Pareto-optimal, i.e. non-dominated, which means that they form the Pareto-optimal frontier. Determining the set of Pareto-optimal paths is the aim of solving an MOSP problem (Ghoseiri and

Nadjari, 2010). As mentioned before, the decision-maker has then to decide whether avoiding c_1 or c_2 is considered more important.

3 Methodology

In this section, we will show how the pheromone trail-following behaviour of ants while searching for food is applied to deal with the MOSP model. Also, a number of key concepts, such as the pheromone matrix, evaporation, heuristic parameters and local and global updates as originally used by the AS and ACS procedures are explained. Subsequently we will show how Ghoseiri and Nadjari's procedures are derived from the original.

3.1 Original AS and ACS procedures

The AS algorithm comes down to letting artificial ants explore the feasible area, generating solutions on the way. Although this is initially done mostly randomly, the search later focuses on (local) optima. When an ant completes its walk, a certain amount of artificial pheromone is left on the edges traveled by the ant, dependent on the quality of the solution. As the ants prefer higher levels of pheromone, more pheromone should be added for better solutions. To keep track of the current pheromone levels per edge, at the end of every iteration these values are updated in the pheromone matrix. The basic steps of the first iteration of the ant algorithms may be described as follows (Afshar, 2005):

1. The ant colony is set up by placing a number of m ants at the source node. Also, the pheromone matrix is initialized at some appropriate level at the start of the computation.

2. As long as the ants have not reached the sink node, a transition rule is used at the nodes to determine the next destination for every ant. This transition rule weighs the pheromone level, costs of the arc used and the distance to the sink node.

3. When all m ants have generated a trial solution, the associated cost is evaluated and the pheromone matrix is updated accordingly. By also taking evaporation into account, previous information is efficiently disregarded and stagnation around a local optimum can be avoided.

The transition rule indicates the probability distribution for moving to any of the subsequent nodes, weighing broad searching for new optima and thorough searching in the neighbourhood of

known optima. The original transition probability for moving from node i to j as defined by Dorigo et al. (1996) was based on a ratio between the pheromone level τ and a heuristic parameter η:

$$p_{ij}(t) = \begin{cases} \dfrac{\tau_{ij}(t)^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{u \in \omega_i} [\tau_{iu}(t)^{\alpha} \cdot \eta_{iu}^{\beta}]} & \text{if } j \in \omega_i, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

At iteration t, $\tau_{ij}(t)$ represents the pheromone value at edge $(i, j) \in E$, while the heuristic parameter η_{ij} was originally defined as $1/c_{ij}$. The former encourages ant to choose commonly used arcs, while the latter causes them to prefer cheaper arcs. Furthermore, the values of α and β indicate the importance of the pheromone level of edge $(i, j) \in E$ relative to its cost. Finally, $\omega_i \subset N$ represents the set of nodes adjacent to i that have not been previously visited by the ant. At the end of every iteration the pheromone level is updated for every arc $(i, j) \in A$ by summing the pheromone deposition τ_{ij}^{h} of every ant $h \in A$ (step 3). Moreover, evaporation rate $\rho \in (0, 1)$ is applied according to the following formula:

$$\tau_{ij}(t+1) = (1 - \rho) \cdot \tau_{ij}(t) + \sum_{h \in A} \Delta\tau_{ij}^{h}(t) \tag{6}$$

When ACS was introduced in 1997, the transition rule changed into a so-called *pseudo-random proportional rule* (Dorigo and Gambardella, 1997). The chance of moving from node i to node j during iteration t, $p_{ij}(t)$, since also depends on a uniformly distributed random variable q and a pre-specified parameter q_0. If $q \leq q_0$, the edge with the highest probability according to (5) is selected, i.e. node j is chosen such that $\tau_{ij}(t) \cdot \eta_{ij}^{\beta}$ is maximized. Otherwise, the same probability distribution as in (5) is applied. Moreover, ACS led to the introduction of a local pheromone update on top of the aforementioned global update. After traversing an edge, every ant performs the following update to the pheromone concentration on that edge:

$$\tau_{ij}(t) = (1 - \phi) \cdot \tau_{ij}(t) + \phi \cdot \tau_{ij}(0) \tag{7}$$

This local update is generally applied to encourage the ants to choose exploration over exploitation. By setting $\phi \in (0, 1)$ at a high value, the search is diversified as the nodes already visited before become less attractive.

3.2 Procedures as used by Ghoseiri and Nadjari

As mentioned before, different ways have been used to deal with the multiple objectives. The first multi-objective algorithms which used ACO employed multiple colonies, one for every objective

(Gambardella and Taillard, 1999; Mariano and Morales, 1999). Iredi et al. (2001) in contrast, used a different pheromone matrix for every objective, while Barán and Schaerer (2003) employed several heuristic parameters instead, one for every objective. Yet another way is proposed by Doerner et al. (2003) who introduced the so-called COMPETants algorithm, which does all three for a bi-objective transportation problem. It uses two colonies of ants, two pheromone matrices and two heuristic parameters, one for each objective. The 'competing' element of this algorithm is that the number of ants each colony gets is dependent on the quality of the solutions they generate. For a comparison between ten different multi-objective ACO algorithms, the reader is referred to García-Martínez et al. (2007). This taxonomy was very recently updated by Falcón-Cardona et al. (2020).

Ghoseiri and Nadjari use a transition rule similar to the one defined by Dorigo and Gambardella (1997), although an extra heuristic parameter is added for the nodes, which will be referred to as θ. Like Iredi et al., for the pheromone level and the first heuristic parameter an extra index k is included to account for the different objectives. The transition probability distribution is based on the following formula:

$$p_{ij}(t) = \begin{cases} \dfrac{([\tau^1_{ij}(t)]^\alpha \cdot [\eta^1_{ij}]^\beta)^\lambda \cdot ([\tau^2_{ij}(t)]^\alpha \cdot [\eta^2_{ij}]^\beta)^{1-\lambda} \cdot [\theta_j]^\delta}{\sum_{u \in \omega_i} ([\tau^1_{iu}(t)]^\alpha \cdot [\eta^1_{iu}]^\beta)^\lambda \cdot ([\tau^2_{iu}(t)]^\alpha \cdot [\eta^2_{iu}]^\beta)^{1-\lambda} \cdot [\theta_u]^\delta} & \text{if } j \in \omega_i, \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

Again, this probability only applies if $q > q_0$. Otherwise, the node with the highest probability is selected. Put differently, if $q \leq q_0$:

$$p_{ij}(t) = \begin{cases} 1 & \text{if } j = arg\max_{u \in \omega_i} \{([\tau^1_{iu}(t)]^\alpha \cdot [\eta^1_{iu}]^\beta)^\lambda \cdot ([\tau^2_{iu}(t)]^\alpha \cdot [\eta^2_{iu}]^\beta)^{1-\lambda} \cdot [\theta_u]^\delta\}, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

The first heuristic parameter η as defined by Ghoseiri and Nadjari is different from the classic $1/c_{ij}$, using the costs corresponding to the minimum and maximum cost edges, respectively c^k_{min} and c^k_{max}, instead:

$$\eta^k_{ij} = \min\left(1, \frac{c^k_{max} - c^k_{ij}}{c^k_{max} - c^k_{min}} + \epsilon\right) \tag{10}$$

In this equation, ϵ is a small parameter preventing η from being zero, while the $\min(\cdot)$ function guarantees η not to grow more than one.

The second heuristic parameter θ_i on the other hand is equal to the inverse of the number of nodes on the shortest path from node i to the sink node and is meant to encourage the ants moving

to the nodes closer to the sink node. Ghoseiri and Nadjari show that the inclusion of θ leads to a larger Pareto-optimal set in less computation time.

Two other parameters that are new to the transition rule are δ and λ. The former is used to set the weight of the second heuristic parameter, while latter decides the weight of the first objective versus the second. The value of λ is determined using two experimental parameters, a and b, and the ant index number h:

$$\lambda = \begin{cases} 0 & \text{if} \quad h \leq am \\ \dfrac{h}{(b-a)m} - \dfrac{a}{1-b-a} & \text{if} \quad am < h < 1 - bm \\ 1 & \text{if} \quad h \geq bm \end{cases} \tag{11}$$

As follows from this equation, a represents the fraction of ants that are entirely dedicated to finding the cheapest path in terms of cost, while b is the fraction of ants that are dedicated to finding the cheapest path in terms of time. Taking formulas (8) - (11) into account, we implement the algorithm in Appendix 1 into Java.

3.3 Another ACO algorithm

As shown in Section 3.2 many ways have been used to approach the multiple objectives using ACO. García-Martínez et al. (2007) assess the different ACO algorithm that have been developed and compare them by applying them to different instances of a traveling salesman problem (TSP). They point out a method as proposed by Iredi et al. (2001) as one of the methods producing the best results. As this method uses two ant colonies instead of one, we will apply this procedure as an alternative way of applying ant colony optimization to the instances. I will then compare their results to those as achieved by Ghoseiri and Nadjari.

As a consequence of the existence of multiple colonies, this method uses one pheromone variable τ and one heuristic parameter η for every colony. The transition probability distribution now looks as follows.

$$p_{ij}(t) = \begin{cases} \dfrac{([\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta})^{\lambda} \cdot ([\tau'_{ij}(t)]^{\alpha} \cdot [\eta'_{ij}]^{\beta})^{1-\lambda} \cdot [\theta_j]^{\delta}}{\sum_{u \in \omega_i} ([\tau_{iu}(t)]^{\alpha} \cdot [\eta_{iu}]^{\beta})^{\lambda} \cdot ([\tau'_{iu}(t)]^{\alpha} \cdot [\eta'_{iu}]^{\beta})^{1-\lambda} \cdot [\theta_u]^{\delta}} & \text{if } j \in \omega_i, \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

τ refers to the pheromone level of the colony the ant under consideration belongs to, while τ' refers to the pheromone level of the other colony. The same cis true for respectively η and η'.

Although θ is not part of the original formula, it is still added for reasons that will become clear in Section 5. Finally, λ is now defined as h/m, where h is the index number of the ant and m is the total number of ants in both colonies. Both colonies therefore prefer a different objective, although every ant takes both objectives into account, except for the first ant in colony 1 and the last ant in colony 2. These ants are entirely focused on the first and the second objective, respectively.

4 Performance measures

Ghoseiri and Nadjari (2010) use three types of performance measures to assess the quality of the ACO approximation set Y' relative to the real Pareto-optimal set Y, generated using the label correcting algorithm. Firstly, performance measure E indicates the closeness of the generated approximation set to the real Pareto-optimal frontier. Every solution a from the approximated front is compared to the closest solution in the real set.

$$E_a = \frac{\min_{b \in Y}(||a - b||)}{||a||}$$

$\bar{E}$, the average of E_a for every vector $a \in Y'$, takes a value between 0 and 1, where a lower value indicates a closer approximation of the real set. Secondly, the quality of the ACO solutions is assessed with regard to its extent. Ghoseiri and Nadjari (2010) compare the worst solution in terms of every objective k offered by the approximation set to the worst solution in the real Pareto-optimal front. We will however use a slightly different approach, comparing the best solution in terms of both objectives instead.

$$EX = \frac{\sum_k \min\{\min_{a \in Y'}(a_k), \min_{b \in Y}(b_k)\}}{\sum_k \min_{a \in Y'}(a_k)}$$

By definition, EX is at most 1, which would indicate that the approximation front covers the entire range of Y, both in terms of cost and in terms of time. Finally, three performance measures are used to indicate to what extent the solutions of the approximated front are close to each other

and uniformly distributed over the entire front.

$$U = \frac{\max_{a \in Y'}(\min_{b \in Y}(||a-b||))}{\sum_{a \in Y'} \min_{b \in Y}(||a-b||)} \tag{13}$$

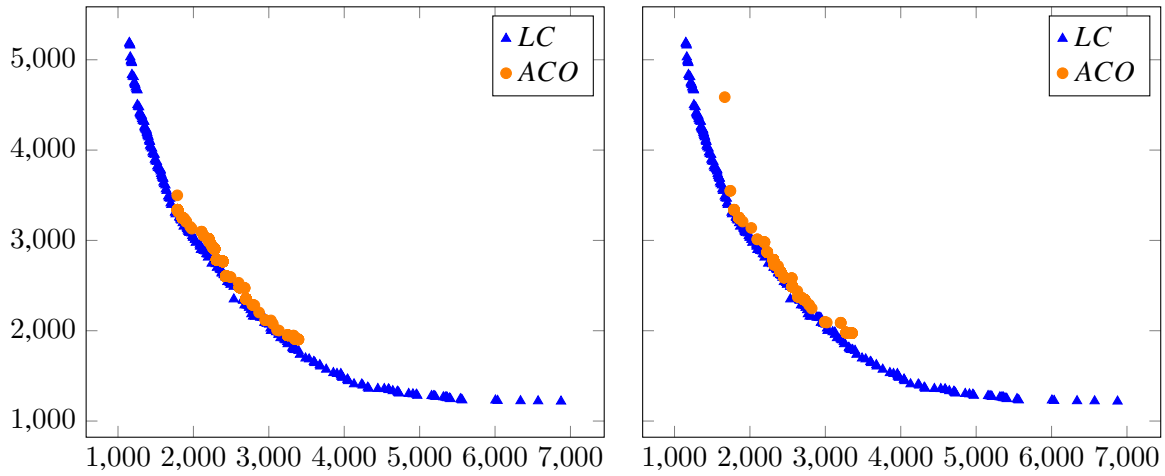
$$SP = \sqrt{\frac{1}{|Y'-1|} \sum_{a \in Y'} (\overline{d}(a) - d(a))^2} \tag{14}$$

$$M = \frac{1}{|Y'| \cdot |Y'-1|} \sum_{a \in Y'} \sum_{b \in Y'} \mathbb{1}_{||a-b|| > \sigma} \tag{15}$$

U is the ratio between the worst case distance and the average distance to the real front, i.e. it indicates the uniformity of the closeness of the ACO solutions to the real Pareto-optimal front. SP indicates the spread of the solutions offered by the ACO algorithm. This performance measure is strongly correlated with the number of solutions. Finally, M gives an indication of the uniformity of the distribution of the solutions on the Pareto-optimal frontier. It is calculated using neighbourhood parameter σ which is equal to the distance between the outer solutions in the approximation set Y' divided by the size of this set. A value close to 1 shows the solutions are located not only in a limited area. This performance measure was introduced by Zitzler et al. (2003).

As fas as U is concerned it has to be pointed out that this performance measure is not always as informative. This can be shown using the graphs in Figure 4. In case of the front shown to the left, U is equal to 1.05, while it is equal to a staggering 7.25 in case of the front shown to the right. This difference is solely caused by the left upper observation of the ACO front in the graph to the right, whose distance to the LC front is significantly larger than the distance of the other observations.

Figure 2: Pareto-optimal frontiers resulting in different values of U.



5 Results

We tested the algorithms on 21 instances, based on an idea from Andersen et al. (1996). These networks have a grid structure, i.e. given a width w and a height h, the structure of the instances is such as is shown in Figure 3:

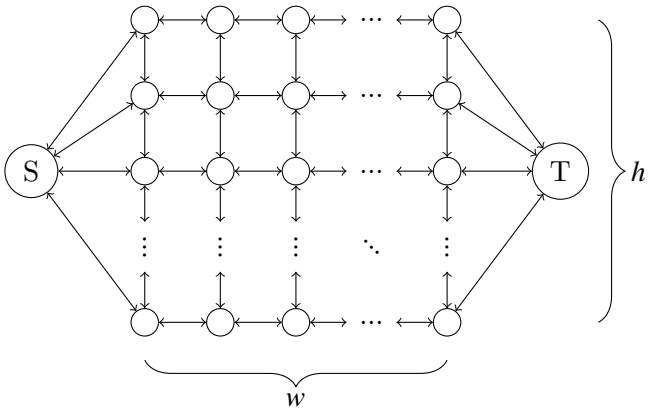


Figure 3: Instance format

The number of nodes is $wh+2$, while the number of edges equals $2w(2h-1)$. Important to note, this results in almost four edges per node if a two-way edge counts as two, whereas this fraction for the instances used by Ghoseiri and Nadjari (2010) is slightly more than 15 edges per node. The width of the 21 instances is equal to $[50, 60, ..., 250]$, while the height is always 100. The instances are numbered from `0` to `20`, starting with the smallest instance.

In order to determine the cost and time associated with every edge $(i, j) \in E$, two independent random positive integers a and b are uniformly drawn from respectively [1, 33] and [68, 100]. Then, with probability $\frac{1}{2}$ we assign $(c(i,j),\ t(i,j)) = (a,\ b)$ and with probability $\frac{1}{2}$ we assign $(c(i,j),\ t(i,j)) = (b,\ a)$. This results in a negative correlation between cost and time Andersen et al. (1996).

The relatively low edge/node ratio of our networks poses a problem to the ACO algorithm. As shown in the transition rule formula (8) an ant never visits a node twice during the same iteration. This means that there is a probability that an ant gets stuck, i.e. $\omega_i = \emptyset$ if an ant is at node i. For obvious reasons, this probability increases with lower edge/node ratios. In the following section we explore how we deal with this.

5.1 Effects of different values of the parameters

A natural way to prevent the ants of getting stuck is discouraging them to move left, that is, moving away from the sink node (see Figure 3). This can be achieved by increasing the value of δ, which gives parameter θ more weight. As shown in the Table 1, the extent to which ants get stuck is strongly dependent on the value of δ.

δ	0.1	0.5	1	5	10	20
EX	0.47	0.44	0.48	0.69	0.73	0.65
$\bar{E}(\%)$	27.18	29.24	24.56	10.07	3.87	1.58
U	1.05	1.28	1.36	1.58	2.39	4.36
SP	32.40	46.18	28.35	34.71	32.19	1.10
M	0.72	0.91	0.88	0.86	0.88	0.92
# return moves (%)	20.2	20.2	20.2	6.9	0.3	0.0
# solutions	415	164	223	236	306	1847
CPU time (s)	63	69	42	20	12	196

Table 1: Performance measures for instance `0` with varying values of δ

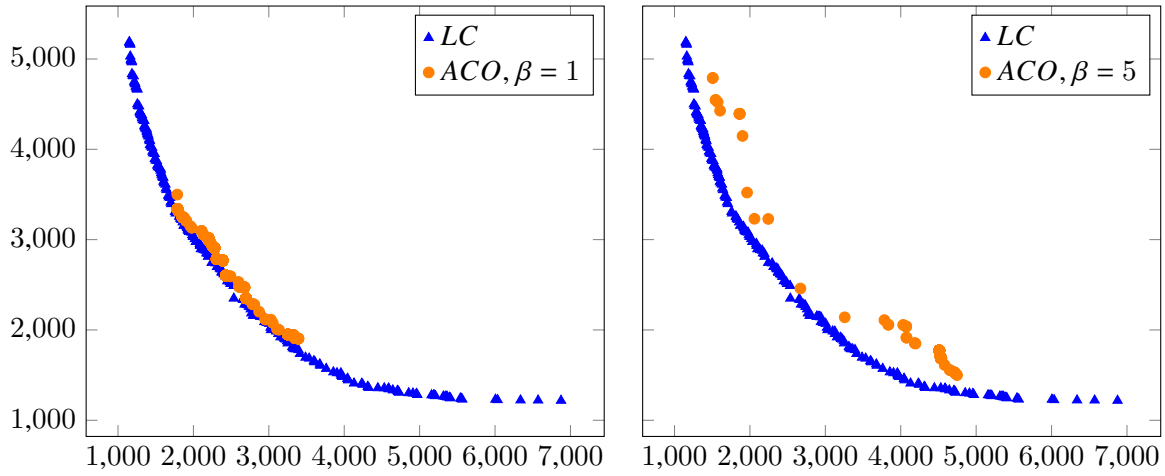
The seventh row shows the percentage of moves that an ant is returning to the former node. As can be observed from the table, this percentage falls as the value of δ becomes larger than 1. The CPU time also decreases, except it is much higher for $\delta = 20$. This is caused by the larger number of generated Pareto-optimal solutions, which results in better performance in terms of spread (SP). It now takes significantly more time to determine the Pareto-optimal set at the end of every iteration. Performance measure EX shows an optimal value for $\delta = 10$, while E, SP and M obtain a better value for $\delta = 20$. Only the value of U is optimal for $\delta = 0.1$. As the CPU time is lowest for $\delta = 10$, we will however proceed with $\delta = 10$.

Another parameter that will receive attention is β. As follows from (8), the value of β indicates the relative weight of the first heuristic parameter. In other words, the higher the value of β, the more likely an ant is to choose the cheapest path instead of the path containing the highest level of pheromone or the path leading to the node closest to the sink node. The table and graph below show the outcome of the behaviour of the ants under different values of β.

β	0.5	1	2	3	4	5	10
EX	0.64	0.64	0.68	0.74	0.79	0.79	0.75
$\bar{E}(\%)$	1.43	1.78	3.11	7.89	7.59	7.40	19.19
U	3.87	2.04	3.67	1.30	2.24	1.52	1.41
SP	10.22	10.07	24.61	64.43	52.11	89.79	72.20
M	0.89	0.89	0.92	0.91	0.92	0.64	0.86
# solutions	1042	643	330	87	158	178	87
CPU time (s)	30	15	7	5	6	6	7

Table 2: Performance measures for instance `0` with varying values of β

Figure 4: Pareto-optimal frontiers as a result of different values of β.



As Table 2 shows, increasing β from 0.5 to 3, results in an improvement of the running time of the algorithm and the extension (EX) of the generated Pareto-optimal front at the expense of its closeness (E) and spread (SP). Higher than 3, none of the performance measures improves anymore. Lower than 3, the optimal value of β depends on the decision-maker whether they prefer closeness or extension. For now we proceed with $\beta = 2$. If closeness is preferred over extension, it is better to set β equal to 1, while 3 should be considered in the opposite case.

5.2 Performance on small and large networks

Initially we fix both the number of iterations and the number of ants at 100. This led to the performance measures stated in Table 3. The parameters are set at $\alpha = 1,\ \beta = 2,\ \delta = 10,\ \phi = \rho = 0.9$

and $q_0 = 0.99$, while a and b are both equal to 45%. Except for δ these values are more or less the same as used by Ghoseiri and Nadjari (2010). The performance measures for the 21 instances are stated in Table 3.

Instance	#Solutions	Extension	Closeness	Uniformity		
		EX	$\bar{E}(\%)$	U	SP	M
0	454	0.71	3.04	3.26	20.20	0.90
1	313	0.72	3.96	2.01	32.03	0.90
2	366	0.79	4.36	1.84	44.07	0.93
3	235	0.72	4.21	2.08	49.72	0.95
4	461	0.69	3.96	1.88	40.60	0.89
5	278	0.71	4.89	1.91	38.66	0.95
6	239	0.68	5.54	1.50	45.11	0.91
7	131	0.68	5.66	1.55	97.07	0.95
8	120	0.69	4.73	1.88	98.07	0.97
9	72	0.62	6.63	1.68	134.39	0.90
10	156	0.68	6.47	1.41	92.46	0.94
11	71	0.67	5.17	1.32	139.75	0.96
12	54	0.63	4.38	2.08	199.54	0.95
13	137	0.64	6.83	1.71	102.72	0.93
14	77	0.64	7.77	1.35	131.05	0.97
15	109	0.65	6.20	1.39	149.26	0.96
16	93	0.67	8.69	1.48	155.77	0.93
17	99	0.61	5.10	1.54	98.54	0.96
18	93	0.65	6.38	1.73	183.82	0.86
19	52	0.63	7.78	1.41	337.58	0.97
20	84	0.59	8.33	1.24	141.95	0.90
Avg.	176	0.67	5.72	1.73	111.06	0.93
St. dev.	129	0.05	1.57	0.44	73.78	0.03

Table 3: Performance measures for all instances with a fixed number of ants.

Remarkably performance measures EX, E and M significantly worsen as the size of the networks increases. As far as U and M are concerned there is no significant effect. Therefore, in line with Ghoseiri and Nadjari (2010) we choose to let the number of ants increase linearly proportional to the number of nodes. Choosing the number of ants equal to the width of the network gives the results as shown in Table 4.

Instance	#Solutions	Extension	Closeness	Uniformity		
		EX	$\bar{E}(\%)$	U	SP	M
0	292	0.69	4.17	2.18	19.56	0.93
1	181	0.63	4.60	2.15	35.20	0.90
2	540	0.76	2.50	2.63	29.83	0.78
3	221	0.69	5.07	1.47	40.76	0.95
4	309	0.71	4.68	2.02	37.26	0.94
5	272	0.73	5.22	1.49	49.95	0.74
6	194	0.69	6.60	1.51	64.26	0.92
7	429	0.64	5.64	1.54	39.01	0.91
8	179	0.67	4.46	1.75	95.35	0.93
9	174	0.66	5.10	1.71	61.47	0.96
10	174	0.66	6.80	1.79	77.77	0.97
11	418	0.69	5.43	2.23	78.50	0.85
12	188	0.65	6.71	1.85	98.30	0.86
13	282	0.67	6.33	1.44	89.87	0.88
14	228	0.64	6.49	1.43	70.07	0.91
15	267	0.64	6.70	1.40	71.90	0.90
16	164	0.67	5.82	1.67	92.14	0.97
17	213	0.61	6.88	1.41	73.79	0.95
18	300	0.67	7.30	1.47	79.58	0.89
19	292	0.66	7.68	1.19	96.24	0.56
20	284	0.62	6.53	2.26	91.18	0.91
Avg.	267	0.67	5.75	1.74	66.29	0.89
St. dev.	97	0.04	1.25	0.37	24.68	0.09

Table 4: Performance measures for all instances with a variable number of ants.

Although there is no significant decline in the number of solutions anymore, surprisingly enough, the significant relationship between EX, E and SP and the size of the networks is still there, although the effect is now smaller. As the method and the values of the parameters are more or less the same as those used by Ghoseiri and Nadjari (2010), this has probably to do with the structure of the networks. For the same reason we do not compare the outcome of the performance measures with theirs. Two other points we would like to direct attention to are the mean and standard deviation shown in the last two rows of both tables. As far as the mean is concerned, only the spread has clearly changed for the better. The other performance measures differ less than 5%. The main gain of making the number of ants dependent on the size of the network has taken place in terms of the stability of the observations. Except for M, the standard deviation of every performance measure has decreased, although the deterioration in the stability of M should not be overlooked.

We also implement the method as described in Section 3.3. The main difference with the method used by Ghoseiri and Nadjari (2010) is that the former method uses multiple ant colonies instead of only one. We employed two ant colonies each half the size of the colonies used in table 4. Parameters $\alpha, \beta, \delta, \epsilon, \phi, \rho$ and q_0 each remain at the same value. The results of the ACO algorithm by Iredi et al. (2001) are stated in Table 5 below.

Instance	#Solutions	Extension	Closeness	Uniformity		
		EX	$\bar{E}(\%)$	U	SP	M
0	770	0.63	2.03	5.64	22.87	0.75
1	509	0.60	2.02	2.99	16.31	0.76
2	109	0.64	3.11	1.84	65.22	0.90
3	58	0.61	3.78	2.54	139.05	0.94
4	100	0.64	3.13	4.25	100.73	0.94
5	235	0.64	2.39	4.16	86.56	0.83
6	216	0.63	3.37	3.08	104.16	0.60
7	65	0.60	2.87	3.16	106.67	0.95
8	64	0.60	2.76	4.26	195.36	0.91
9	69	0.59	3.84	2.93	257.31	0.95

10	60	0.59	3.46	3.51	156.40	0.93
11	52	0.60	4.07	3.03	183.51	0.96
12	102	0.58	3.26	4.83	177.07	0.89
13	54	0.60	3.30	3.28	318.95	0.94
14	56	0.61	3.25	3.17	360.99	0.96
15	65	0.60	3.93	2.12	128.62	0.97
16	98	0.60	3.26	3.99	96.92	0.97
17	68	0.57	3.80	3.17	225.11	0.92
18	63	0.60	3.76	2.47	248.02	0.95
19	65	0.61	3.62	2.30	188.01	0.95
20	40	0.59	3.62	2.10	218.41	0.96
Avg.	139	0.61	3.27	3.28	161.72	0.90
St. dev.	179	0.02	0.58	0.96	90.02	0.09

Table 5: Performance measures for the method by Iredi et al. (2001).

We would like to draw a number of inferences from this table. First is that the closeness of the generated front to the real Pareto-optimal front has clearly improved, seemingly at the expense of its expansion. This has most likely to do with the structure of both methods. As can be observed from the transition probability distribution (12), both colonies prefer one objective, but they are not entirely focused on this objective. The transition probability function as used by Ghoseiri and Nadjari (2010) however, makes sure a fraction of $a + b$ ants is entirely focused on one objective (see (8) and (11)). As we choose $a = b = 45\%$, only 10% of the ants are concerned with both objectives. This results in a larger expansion of the generated front, whereas not many ants are dedicated to finding high quality solutions in the centre of the Pareto-optimal front. Also, the method as used by Iredi et al. (2001) performs worse in terms of spread, although the results are clearly more stable in case of both expansion and closeness. Finally, the performance of the algorithm is still declining with the size of the networks in terms of expansion, closeness and spread.

Figure 5: Ghoseiri and Nadjari (left) and Iredi et al. (right) Pareto-optimal fronts for instance `0`.

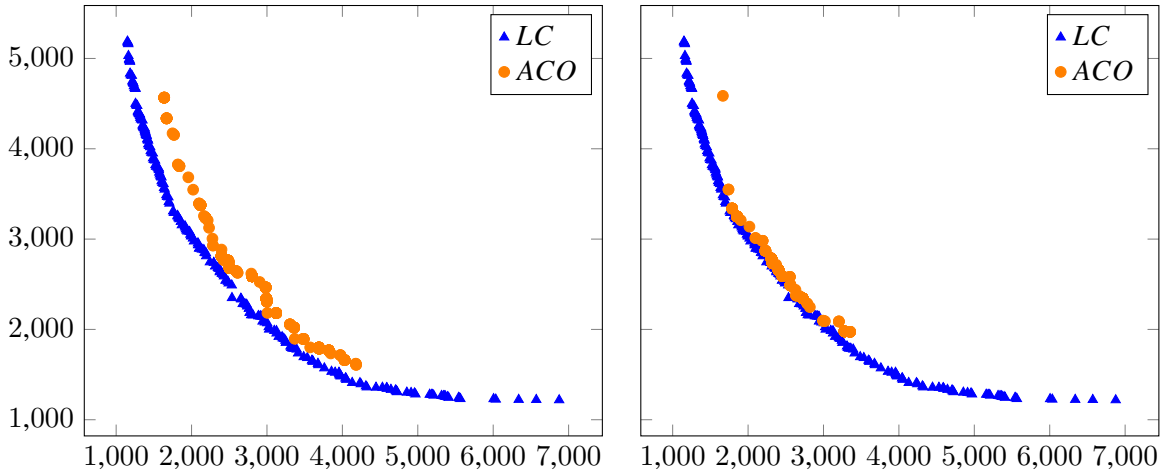
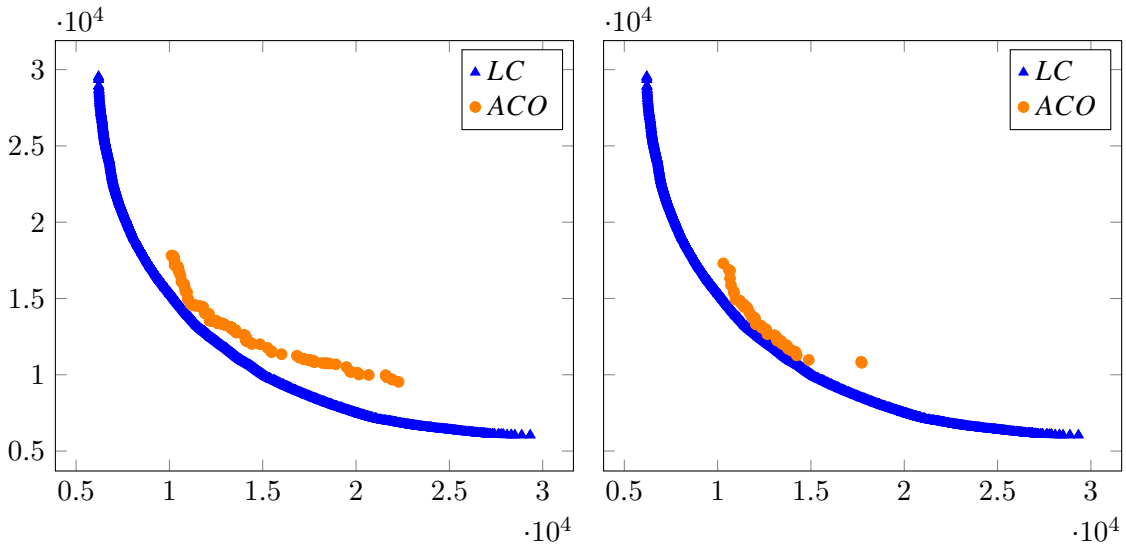


Figure 6: Ghoseiri and Nadjari (left) and Iredi et al. (right) Pareto-optimal fronts for instance `20`.



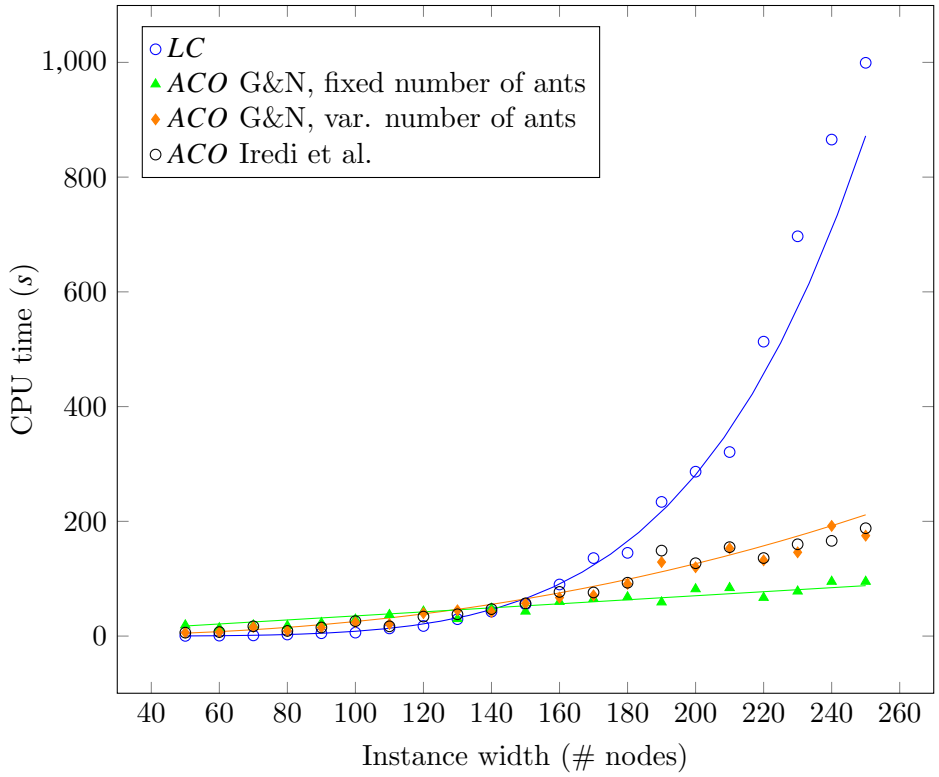
Comparing Figure 5 and Figure 6 indeed clearly shows that the extension has dropped for both methods in the case of the larger instance. For the closeness of the fronts generated using the Ghoseiri and Nadjari (2010) method this is not so easy to tell, although it is for the Iredi et al. (2001) method.

5.3 CPU solving time

Finally, we come to the main point offered by Ghoseiri and Nadjari (2010): the Ant Colony Optimization algorithm is time saving in computation of large-scale bi-objective path problems. This

is shown in the graph below. The running times of the label correcting algorithm are represented in blue, while the running times of the ACO algorithm with a fixed and a variable number of ants are shown in green and orange, respectively. It has to be noted however, that the results of the label correcting algorithm were generated on another computer in C++. Therefore, the running times of the algorithms are not one-to-one comparable, although a trend can still be observed. The running times of the algorithm by Iredi et al. (2001) do not significantly differ from those of the algorithm by Ghoseiri and Nadjari (2010).

Figure 7: The running time of the ACO algorithm compared to the LC algorithm.



Although the ACO algorithm is outperformed by label correcting algorithm in terms of CPU running time for half of the instances, the former performs significantly better when it comes to the larger networks. Important to note, only the running times of the ACO algorithm with a fixed number of ants are linearly correlated to the number of nodes. The trendline of the running times of the ACO algorithm with a variable number of ants as well as the one of the label correcting algorithm is of the form $c \cdot x^n$, where $n > 1$. As we have seen that the stability of the performance has decreased but not so much the performance itself, a fixed number of ants may be considered for very large networks.

6 Conclusion

We have shown that the Ant Colony Optimization algorithm is able to generate good solutions for a bi-objective shortest path problem that approach the real Pareto-optimal frontier generated by the label correcting algorithm. Especially for large networks, using the ACO algorithm would be worth considering as this significantly saves running time. It has to be noted however, that the quality of the solutions may decline as the number of nodes increases. This is the case for every ACO algorithm that we have implemented. We recommend further research to finding different ways or methods to make sure the algorithm is robust to changes in the size of the networks. We have shown that the ACO algorithm can also be applied to networks that are relatively sparse on arcs, although the ants should more strongly be encouraged to move to nodes closer to the sink node, in order to prevent the ants from getting stuck many times. We could have improved the performance of the algorithm in terms of closeness and uniformity by making this incentive even stronger, although this would have resulted in much longer running times (see Table 1).

For the method used by Ghoseiri and Nadjari (2010), we have also seen that the determination of the values of the parameters enable the decision-maker to choose closeness over expansion or vice versa. As the former comes at the expense of running time (see table 7), one might as well use the method as proposed by Iredi et al. (2001). We have seen that this method performs well on closeness, although it has to be taken into account that it worsens more as the size of the networks increases. We have also seen that this method has more stable results, although the method as used by Ghoseiri and Nadjari (2010) has more opportunities to influence the shape the front by changing the parameters. The latter method for instance has the opportunity to set the fraction of ants that is entirely dedicated to optimizing one objective, which the other method has not.

References

Afshar, M. (2005). A new transition rule for ant colony optimization algorithms: application to pipe network optimization problems. *Engineering Optimization*, 37(5):525–540.

Andersen, K. A., Jörnsten, K., and Lind, M. (1996). On bicriterion minimal spanning trees: an approximation. *Computers & Operations Research*, 23(12):1171–1182.

Barán, B. and Schaerer, M. (2003). A multiobjective ant colony system for vehicle routing problem with time windows. In *Applied informatics*, pages 97–102.

Beckers, R., Deneubourg, J.-L., and Goss, S. (1992). Trails and u-turns in the selection of a path by the ant lasius niger. *Journal of Theoretical Biology*, 159(4):397–415.

Doerner, K., Hartl, R., and Reimann, M. (2003). Ar e competants more competent for problem solving?–th e case of a multiple objective transportation problem. *Centra l European Journal of Operations Research*, 11(2):115–141.

Dorigo, M. and Blum, C. (2005). Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344(2-3):243–278.

Dorigo, M. and Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation*, 1(1):53–66.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41.

Dorigo, M. and Stützle, T. (2003). The ant colony optimization metaheuristic: Algorithms, applications, and advances. In *Handbook of Metaheuristics*, pages 250–285. Springer.

Falcón-Cardona, J. G., Leguizamón, G., Coello, C. A. C., and Tapia, M. G. C. (2020). Multi-objective ant colony optimization: An updated taxonomy and review of approaches. *Preprint available at www. researchgate. net*.

Gambardella, L. and Taillard, G. (1999). A multiple ant colony system for vehicle routing problems with time windows. In *New Ideas in Optimization*. McGraw-Hill. London.

García-Martínez, C., Cordón, O., and Herrera, F. (2007). A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria tsp. *European Journal of Operational Research*, 180(1):116–148.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability*, volume 174. Freeman San Francisco.

Ghoseiri, K. and Nadjari, B. (2010). An ant colony optimization algorithm for the bi-objective shortest path problem. *Applied Soft Computing*, 10(4):1237–1246.

Goss, S., Aron, S., Deneubourg, J.-L., and Pasteels, J. M. (1989). Self-organized shortcuts in the argentine ant. *Naturwissenschaften*, 76(12):579–581.

Hansen, P. (1980). Bicriterion path problems. In *Multiple criteria decision making theory and application*, pages 109–127. Springer.

Iredi, S., Merkle, D., and Middendorf, M. (2001). Bi-criterion optimization with multi colony ant algorithms. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 359–372. Springer.

Mariano, C. E. and Morales, E. M. (1999). Moaq an ant-q algorithm for multiple objective optimization problems. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1*, pages 894–901.

Modesti, P. and Sciomachen, A. (1998). A utility measure for finding multiobjective shortest paths in urban multimodal transportation networks. *European Journal of Operational Research*, 111(3):495–508.

Paixão, J. M. and Santos, J. L. (2013). Labeling methods for the general case of the multi-objective shortest path problem–a computational study. In *Computational Intelligence and Decision Making*, pages 489–502. Springer.

Sastry, V., Janakiraman, T., and Mohideen, S. I. (2003). New algorithms for multi objective shortest path problem. *Opsearch*, 40(4):278–298.

Serafini, P. (1987). Some considerations about computational complexity for multi objective combinatorial problems. In *Recent advances and historical development of vector optimization*, pages 222–232. Springer.

Skriver, A. J. and Andersen, K. A. (2000). A label correcting approach for solving bicriterion shortest-path problems. *Computers & Operations Research*, 27(6):507–524.

Vincke, P. (1975). *Problemes multicriteres*, volume 16. Cahiers du Centre d'Etudes de Recherche Operationnelle.

Zhan, F. B. and Noon, C. E. (2000). A comparison between label-setting and label-correcting algorithms for computing one-to-one shortest paths. *Journal of Geographic Information and Decision Analysis*, 4(2):1–11.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and Da Fonseca, V. G. (2003). Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on evolutionary computation*, 7(2):117–132.

A Appendices

A.1 Ant Colony Optimization Algorithm

Algorithm 1: Algorithm for Ant Colony Optimization

Data: $G = (N, E), \alpha, \beta, \delta, \epsilon, \phi, \rho, q_0, a, b$

Initialize ant colony A and pheromone levels τ.

Pareto-optimal solutions set $\leftarrow \phi$;

End condition $\leftarrow false$;

for $i \in N$ **do**

  Calculate θ_i;

  for $j \in N$ **do**

    Calculate η_{ij}^k;

  end

end

repeat

  repeat

    for $h \in A$ **do**

      Calculate λ;

      Determine next node using the transition rule;

      Perform local pheromone update;

    end

  until *Every ant has reached sink node* t;

  Update Pareto-optimal solutions set;

  Perform global pheromone update;

until *End condition = true*;

Result: *Pareto-optimal solutions set*