Bachelor Thesis Econometrics and Operations Research

June 21, 2020

Generating non-dominated solution sets for the bi-objective shortest path problems with the ACO and HUMANT algorithm[1]

Name student: Annelot Bosman

Student ID number: 472467

Supervisor: Y.N. Hoogendoorn

Second accessor: dr. S. Sharif Azadeh

Abstract

This research considers bi-objective shortest path (BSP) problems. In order to find a set of non-dominated solutions for this problem, an heuristic needs to be employed. Such a heuristic is the an Ant Colony Optimization algorithm, which will be employed in this research. The idea of such a non-dominated set is that a decision maker has a wide array of incomparable options to choose from. To make this decision easier a combination of a decision making method PROMETHEE II, and the ACO can be used to form the HUManoid Ant (HUMANT) algorithm. This algorithm uses preference weights and preference functions within the algorithm and sorts the solution set based on the preference functions. We find that the ACO works sufficiently for the BSP problem for the networks used in this thesis, with overall reasonable computation time. The HUMANT outperforms the ACO with respect to most aspects, such as the average error distance and the uniformity of the non-dominated solutions in the solution space.

[1]The views stated in this thesis are those of the author and not necessarily those of the supervisor, second accessor, Erasmus School of Economics or Erasmus University Rotterdam.

Contents

1 Introduction

This thesis considers the shortest path (SP) problem, whose objective is to find the shortest possible path between a origin and destination within a network. Throughout this paper we assume non-negative arc lengths. The path which is considered the shortest is the path with minimum total cost over all the arcs in the path. Shortest path problems are widely researched and they have numerous applications. Besides, shortest path problems might be, as stated by Lawler (2001), amongst the most fundamental of all combinatorial optimization problems, as various more convoluted problems in combinatorial optimization can be reformulated as a shortest path problem and solved in this way. Single objective shortest path problems can be solved exactly in polynomial time by the well known shortest path algorithm by Dijkstra et al. (1959). This algorithm generates optimal solutions and it has been shown that it is the most robust in practice. Nevertheless, Dijkstra's algorithm is outperformed in respect to the running time by other algorithms (Goldberg & Tarjan, 1996).

Often more then one criterion is of importance in choosing a shortest path between origin and destination, such as travel time and travel cost for road transport or the cost of building and maintaining a route versus the damage to environment when a new road needs to be build (Andersen et al., 1996). Choosing an efficient path for all criteria remains of importance. The type of problem where the shortest path should be based on more then one criteria is called a multi-objective shortest path problem. In this research the *bi-objective shortest path* (BSP) problem with two (conflicting) criteria will be investigated.

It often turns out that the path which minimises one objective is not the same path for which the other objective is minimal. Andersen et al. (1996) state that a large set of solutions for multi-objective problems should be generated, such that the decision maker can find a solution among this set. Andersen et al. (1996) also argue that the question of efficiency of the solutions is of inconsequential importance when generating the set. Such a set of solutions consists of Pareto optimal paths. Pareto optimal paths are paths for which one objective solution has a higher value than other paths, but the other objective solution has a lower value, in such a way that no better solution can be chosen unambiguously. The purpose of the BSP, and other multi-objective problems, is to find a set of Pareto optimal paths, as stated by Ghoseiri & Nadjari (2010), rather than to find an optimal solution.

The first goal of this research is to find sets of Pareto optimal shortest paths based on two objectives between an origin and destination node in a network. To reach this goal, multiple research questions have been stated: *"Which algorithm is appropriate to use in solving the bi-objective shortest path problem?"* and *"How does this algorithm perform compared to an appropriate benchmark?"*. In order to find an answer to the first question we will perform a literature review and the second question will be answered by implementing an algorithm and testing it.

The second goal of this research is to order the set of Pareto optimal shortest paths based on preferences, in order for a decision maker to find a good solution amongst all solutions in the set. To reach this goal another research question is stated: *"In what way should a set of Pareto optimal solutions be ordered?"* The answer to this question will be discussed in the literature review and resulting from this literature a method to order the solution set will be developed.

Due to the relevance of BSP and similar problems in many practical fields, much research has been done into heuristics solving multi-objective problems. An extensive overview is given by García-Martínez et al. (2007). There are four classical methods in solving multi-objective problems, Objective weighting, distance functions, Min-Max formulation and Lexicographic approach. These classical methods can be incorporated in exact formulations as well as heuristical methods, but these all have several consequential drawbacks (Coello et al., 2007). In this research the Ant Colony Optimization (ACO) algorithm is considered since several different ACO algorithms have been shown to perform decent in multi-objective combinatorial problems (García-Martínez et al., 2007). This ACO algorithm is a form of Ant Colony Systems, which are heuristic methods in solving a and not one of the above mentioned classical methods.

The first ACO algorithm, called the ant system, was developed by Dorigo et al. (1996) and is based on the way ant colonies function in reality. The Ant System is used to solve single objective combinatorial problems and was tested with the *Travelling Salesman Problem* (TSP). The intuition of the algorithm is as follows. Ants manage to find the shortest path from their colony to some destination and back by leaving and following pheromone trails (Deneubourg & Goss, 1989). When an ant follows a path, it leaves a trail of pheromones which other ants can pick up on. When another ant follows the same path, the pheromone trail is reinforced. If there is a pheromone trail, an ant is very likely to take that path, while if there are no trails it will move practically at random (Dorigo et al., 1996). This process is self-reinforcing and in time leads to all ants following the same path.

The original Ant System was found to have a long runtime for problem instances with more than 30 nodes for the TSP by Dorigo & Gambardella (1997). To improve the Ant System, three aspects of the algorithm were changed to form the new *Ant Colony System* (ACS) algorithm. The ACS seemed to work quite well and had an excellent speed, which is why the algorithm is well-suited to extend to multi-objective problems, as stated by Ghoseiri & Nadjari (2010). Instead of one type pheromone trail, Iredi et al. (2001) employed two types of pheromone trail, one for each objective. They combined the information of the two trails as well as force some ants to search different paths than the rest of the colony with the goal to find even shorter paths. This algorithm was used to solve bi-objective vehicle routing problems. The Ant Colony optimization will generate an approximate set of non-dominated solutions that form the Pareto frontier, but will likely not generate the exact one.

Together with the ACO a multi-criterion decision-making (MCDM) method can be used to order all the Pareto-optimal solutions according to some preference. A big disadvantage to some existing MCDM methods in general, where preferences are known beforehand, is that they only perform well if the set of Pareto-optimal solutions is relatively small, so with 10-30 options (Mladineo et al., 2015). An option is to incorporate a part of the decision-making problem into the meta-heuristic, in this case the ACO algorithm. A successful integration of a MDCM into an ACO algorithm was made for the Partner Selection problem by Mladineo et al. (2017), where they combined the PROMETHEE method and the ACO algorithm.

The PROMETHEE (Preference Ranking Organization METHod for Enrichment Evaluations) method is an outranking method for multi-objective optimization problems (Brans et al.,1986). This method is used for a wide array of different decisions, such as choosing between different projects in a company.

Mladineo et al. (2015) give their ants the ability to have preferences, which is a human trait. Because of this trait, this type of ants are called *humanoid ants* and the algorithm is called HUMANT. The HUMANT algorithm works well for the partner selection problem and is even able to find better solutions than the traditional ACO algorithm in some complex cases. The HUMANT will be adjusted to fit the BSP problem and implemented in this research.

This report will continue as follows. In Section 2 the problem description of the BSP will be given, in Section 3 the ACO and HUMANT algorithm are described in detail as well as the performance measures used to asses these algorithm, in Section 4 the data used in this research will be described and the results of the algorithms are analysed. Finally in Section 5 our research questions will be answered.

2 Problem description

The bi-objective shortest path problem is defined as follows. Define a directed graph $G = (N, A)$ with $N = \{1, ..., n\}$ the set of nodes and $A = \{(i, j) : i, j \in N, i \neq j\}$ the set of directed arcs among the nodes in N. Each edge $(i, j) \in A$ has two coefficients $C^1_{i,j}$ and $C^2_{i,j}$. These two costs are represented by a cost vector formulated as $(C^1_{i,j}, C^2_{i,j})$. For now it is assumed that $C^1_{i,j}$ denotes the cost of travelling over arc $(i, j) \in A$ and $C^2_{i,j}$ denotes the travel time from node $i \in N$ to node $j \in N$ over edge $(i, j) \in A$. It is assumed those costs are both non-negative. Furthermore, let $s \in N$ denote the source node and node $t \in N$ the destination node. Define decision variable $x_{i,j}$, $\forall(i, j) \in A$ as a binary variable, where $x_{i,j}$ equals 1 if arc (i, j) is in included in the final path and equals 0 if arc (i, j) is not included. The formulation of the BSP looks as follows:

$$\begin{aligned}
&min \quad f^1(x) = \sum_{(i,j)\in A} C^1_{i,j} x_{i,j} \\
&min \quad f^2(x) = \sum_{(i,j)\in A} C^2_{i,j} x_{i,j} \\
&s.t. \\
&\sum_{j|(i,j)\in A} x_{i,j} - \sum_{j|(j,i)\in A} x_{j,i} = \begin{cases} 1, & \text{if } i = s, \quad \forall i \in N \\ 0, & \text{if } i \neq s, t, \quad \forall i \in N \\ -1, & \text{if } i = t, \quad \forall i \in N \end{cases} \\
&x_{i,j} \in \{0, 1\}, \quad \forall (i, j) \in A.
\end{aligned} \tag{1}$$

As stated before, the purpose of the BSP is to find a set of Pareto optimal paths. This set of Pareto optimal paths consists of paths with non-dominated total cost vectors. The total cost vector of a path consists of the total of travel costs of all arcs in the path and the total of travel time of all arcs in the path.

Formally a cost vector x_1 dominates a cost vector x_2, if the following two conditions hold (Mishra & Harit,2010):

1. The values for both objectives in x_1 are no worse (not higher) than the values in x_2.

2. At least one of the values in x_1 is strictly better than that in x_2 in at least one objective.

The non-dominated set of solutions exists of those that are not dominated by any of the solutions generated.

Figure 1 illustrates what non-dominated solutions entail. For the first objective, the shortest path goes from node s to t directly, resulting in a cost vector of $(3, 5)$. The shortest path for objective two is s-a-b-c-t, resulting in a cost of 4 and a cost vector of $(4, 4)$. No best path exists in this case, which makes both paths non-dominated solutions to the problem.

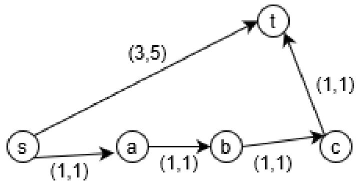


Figure 1: Example of two solutions to the BSP, with two non-dominated solutions.

3 Methodology

In this section the Ant Colony Algorithm as used by Ghoseiri & Nadjari (2010) and the HUManoid ANT algorithm as introduced by Mladineo et al. (2015) is adjusted and described, after which all the parameters will be explained and the performance measures that will be used to evaluate the algorithm are introduced and explained.

3.1 Ant colony algorithm

In Algorithm 1 the ACO algorithm is laid out and in this section we will explain the steps in the algorithm in detail.

Colonies of artificial ants are generated to find a path in a network between the source and destination nodes. The colony of ants consists of H ants in total. Each ant in the colony is labeled with a number $h \in \{1, ..., H\}$. Each ant in the colony, one by one and from the ant with the smallest number to the ant with number H, *walks* over the graph to find a path from the source to the destination node.

Each arc in the network has two pheromone trails, one for each objective. Arcs which contain a larger amount of pheromone trails are more likely to be chosen by the ant. Besides the pheromones, the ants also take into account heuristic information that depicts the preference of an ant to walk over an arc in the network. Some ants only take into account the pheromones and heuristic parameters from the first objective, while some only from the second and some ants use a combination of information of both objectives. In this research, there are two types of heuristic parameters, which will be explained later in this section. The next node should be connected to the current node with an arc directing towards that next node.

Every time an ant moves over an arc, a local update is performed. In the original Ant System ants deposited pheromone on the trails they walk on. In the ACO in this research a bit of the pheromone trail evaporates when an ant walks over an arc. This local update makes that the next ant is a bit less likely to choose the same arc, which forces the colony to search in many directions within a network.

A set of non-dominated solutions is kept and when an ant reaches the destination node t, the cost of the path for both objectives is analysed and the set of non-dominated solutions is updated. When all ants in the colony have reached the destination node, a *global update of pheromone trail* is performed. This global update consists of some evaporation of pheromone trails throughout the network as well as extra pheromone deposit on all non-dominates paths found by the current colony. In order for the algorithm not to converge to a local optimum too fast the amount of pheromone trails on an arc is not allowed to exceed a certain number. As long as some predetermined stopping condition has not been met, a new colony of ants is generated and the above steps are repeated with the new colony. The stopping conditions of the algorithm will be explained in Section 3.1.4.

Algorithm 1 Pseudo-code for ant colony algorithm

```
1:  Compute second heuristic parameter via Algorithm 2
2:  while stopping condition not true do
3:    Prepare a new colony
4:    while not all ants in colony generated a solution do
5:      next ant $h$ moves from node $s$
6:      while ant $h$ has not reached node $t$ do
7:        Ant $h$ moves to next node using transition rule ( (9) and (10))
8:        Local update ( (3) and (4))
9:      end
10:     Update non-dominated solutions set
11:   end
12:   Global update ( (5) through (8))
13: end
14: Print non-dominated set
```

3.1.1 Heuristic parameters of the Ant Colony Optimization

In the ACO suggested by Ghoseiri & Nadjari (2010) two heuristic parameters are used. The heuristic parameters are used to help ants choose a more efficient path.

The first heuristic parameter, $\eta^{1k}_{i,j} \quad \forall i, j \in A, \forall k \in \{1, 2\}$, helps an ant, that has not reached node t, choose the arc with minimum cost among all feasible arcs as the next arc. The value of this parameter is always between 0 and 1. The first heuristic parameter is defined as follows,

$$\eta_{i,j}^{1k} = min\left(1, \frac{C_{max}^k - C_{i,j}^k}{C_{max}^k - C_{min}^k} + \varepsilon\right). \tag{2}$$

Where C_{max}^k and C_{min}^k are the arcs with maximum and minimum cost values for objective k in the network respectively, ε is a small non-zero value and $C_{i,j}^k$ is the cost of arc (i, j) for objective k.

The second heuristic parameter, $\eta_i^2, \forall i \in N$, influences the ant to find nodes closest to the destination node t. This parameter is for every node $i \in N$ equals 1 divided by the minimum number of nodes of that is needed to create a path between node i and node t. The algorithm that is used to find the minimum number of nodes for a path between node i and node t and calculate the second heuristic parameter is described in Algorithm 2.

When an ant has not yet reached t, it is most likely to choose the subsequent arc as the one with the lowest cost when heuristic parameter two is not used (Dorigo & Gambardella (1997)). This may result in an inefficient path, as illustrated in Figure 2. Using both these parameters helps all the ants make a trade-off between the cost and length of a path. Ghoseiri & Nadjari (2010) finds that using both these parameters results in more results within a shorter period of time.

Algorithm 2 Pseudo-code for generating the second heuristic parameter

```
1: $nodes \leftarrow N$
2: $label \leftarrow$ list of length $|N|$
3: $label(i) \leftarrow \infty, \quad \forall i \in N$
4: $label(t) \leftarrow 0$
5: $list \leftarrow t$
6: $next \leftarrow \emptyset$
7: while $nodes \neq \emptyset$ do
8:   $nodes \leftarrow nodes$ minus all elements in $list$
9:   while $list \neq \emptyset$ do
10:    $j \in list$ is the first element from $list$    ▷ This first element is the node with the lowest index name.
11:    $In(j) \leftarrow$ all nodes that have arcs directed towards node $j$
12:    foreach $i \in In(j)$ do
13:      $label(i) = min(label(j) + 1, label(i))$
14:    end
15:    $next \leftarrow$ all elements from $In(j)$
16:    $In(j) \leftarrow \emptyset$
17:    $list \leftarrow list \setminus \{j\}$
18:  end
19:  $list \leftarrow$ all elements from $next$
20: end
21: foreach $i \in N \setminus t$ do
22:   $\eta_i^2 = \frac{1}{label(i)}$
23: end
24: $\eta_t^2 = 1$
```

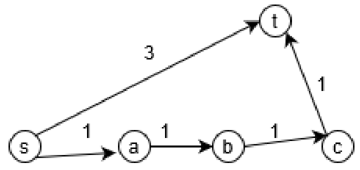


Figure 2: Example of an inefficient path for a problem with one objective, when the next arc is the one with the lowest cost from among the feasible arcs. The optimal path goes from s to t directly and yields a cost of 3, while the path s-a-b-c-t yields a cost of 4.

3.1.2 Local and global updates

In this research a distinction is made between local updates and global updates of the pheromone trails. The amount of pheromone trail over an arc $(i, j), \forall i, j \in A$ for each objective is represented by the parameter $\tau^k_{i,j}, k \in \{1, 2\}$. Local update of the pheromone trails will prevent the algorithm from converging to a local optimum. Some pheromone evaporates for both objectives every time an ant moves over an arc in the network in the following manner:

$$\tau^1_{i,j} = \tau^1_{i,j} \cdot \varphi. \tag{3}$$

$$\tau^2_{i,j} = \tau^2_{i,j} \cdot \varphi. \tag{4}$$

Here, parameter φ assumes a value from the interval $(0, 1)$.

The global update of the pheromone trails is done each time an entire colony of ants has produced paths from node s to node t. In other words, if one iteration of the algorithm is finished, the global update is performed. Evaporation as described by (5) and (6) is used on all arcs in the networks, while updating as described by (7) and (8) is used on all arcs that are in the paths in the non-dominated solution set from the current iteration. Updating the pheromone matrices using all non-dominated solutions found in the current iteration was proposed by Iredi et al. (2001) and has been shown by Ghoseiri & Nadjari (2010) to work well for the problem at hand.

$$\tau^1_{i,j} = \tau^1_{i,j} \cdot \rho. \tag{5}$$

$$\tau^2_{i,j} = \tau^2_{i,j} \cdot \rho. \tag{6}$$

$$\tau^1_{i,j} = \min\left(1, \tau^1_{i,j} + \frac{|N|}{C^1_{i,j}}\right). \tag{7}$$

$$\tau^2_{i,j} = \min\left(1, \tau^2_{i,j} + \frac{|N|}{C^2_{i,j}}\right). \tag{8}$$

Where $|N|$ equals the number of nodes in the problem instance used and parameter ρ assumes a value from the interval $(0, 1)$.

3.1.3 Transition procedure

Which node is chosen by an ant as the subsequent one is determined by a probability distribution, this selecting of the next node will be called the transition procedure. The probability with which ant h chooses the next node j when currently on node i is depicted by the parameter $p_{i,j}, \forall i, j \in N$. To determine the probabilities, the pheromone trail of arc (i,j) for an objective is multiplied by the first heuristic parameter for that objective. Lastly, the two objectives are weighted and multiplied with the second heuristic parameter over arc (i,j). The distribution used in the transition procedure is also described in the research by Dorigo & Gambardella (1997). When $q \leq q_0$ the ant will choose the arc with the most pheromone trail and otherwise the ant will choose one of the feasible arc with a probability determined by (10). Where q is a random uniformly distributed number in $[0,1]$ and q_0 is a parameter.

- if $q \leq q_0$:

$$p_{i,j} = \begin{cases} 1, & \text{if } j = \text{argmax}_{u \in \omega(i)} \{ (\tau_{i,u}^{1^{\alpha}} \eta_{i,u}^{11^{\beta}})^{\lambda} \ (\tau_{i,u}^{2^{\alpha}} \eta_{i,u}^{12^{\beta}})^{(1-\lambda)} \ \eta_u^{2^{\delta}} \} \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

- else:

$$p_{i,j} = \begin{cases} \dfrac{(\tau_{i,j}^{1^{\alpha}} \eta_{i,j}^{11^{\beta}})^{\lambda} \ (\tau_{i,j}^{2^{\alpha}} \eta_{i,j}^{12^{\beta}})^{(1-\lambda)} \ \eta_j^{2^{\delta}}}{\sum_{u \in \omega(i)} (\tau_{i,u}^{1^{\alpha}} \eta_{i,u}^{11^{\beta}})^{\lambda} \ (\tau_{i,u}^{2^{\alpha}} \eta_{i,u}^{12^{\beta}})^{(1-\lambda)} \ \eta_u^{2^{\delta}}}, & \text{if } j \in \omega(i) \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

Here, $\omega(i)$ is the set of outgoing arcs from current node i that have not been visited by ant h. Furthermore, α is the preference weight of the pheromone trails, β is the weight of the first heuristic parameter and δ the weight of the second heuristic parameter. These weights determine the relative importance of those properties in the choice for a certain arc of an ant.

Since ants favor arcs with strong pheromone trails, some arcs and paths may go unexplored. To counter this, the weight λ imposes the ant to choose a different path than the ants before them, this was also used by Iredi et al. (2001) and Doerner et al. (2004). This works as follows. if $0 < \lambda < 1$ holds, a combination of both pheromone trails is used in determining the probabilities for the arc in the transition procedure combined with the first heuristic parameter for both objectives. When λ equals 1, only the pheromone trail for the first objective is used combined with the first heuristic parameter and similarly when λ equals 0 the pheromone trail for the second objective is used in determining the probabilities. The definition of λ is as follows:

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

Where a and b are determined on the size of the instance used.

3.1.4 Parameter settings

The stopping condition that will be used in the research is the following: If no new non-dominated solutions are found by the colony in the previous iteration, the algorithm will stop, else stop after 100 iterations. In other words, stop after 100 colonies have been employed. This stopping conditions are the same as in the research by Ghoseiri & Nadjari (2010). The rest of the parameter settings will be similar to the ones in that research as well.

The number of ants per colony will be based on the number of nodes in the network, because bigger networks have more possible paths that can be explored in order to find better solutions. To determine the number of ants to be used in the ACO algorithm for the bi-objective shortest path problem, the number of ants used in the research by Ghoseiri & Nadjari (2010) was analysed. There seemed to be a relation between the number of nodes in the network of an instance and the number of ants in the colony. Ghoseiri & Nadjari (2010) show the algorithm to be robust with respect to the number of ants. Analysing their approach it seems that for every interval of nodes a certain number of ants is employed and when the number of nodes exceeds that interval, more ants are used. This approach will be followed as well in this research.

Parameters a and b that are used to calculate λ in Formula (11) are determined as follows: Parameter a will be the number of ants in the colony only using the pheromone trail and heuristic information for the second objective. The number of ants resulting from $b - a$ will be the number of ants employing the pheromone trails and heuristic information of both objectives. This number of ants will be smaller than a, since these ants will search in a more central area of the solution space, where paths are less complex to find. The number $h - (b - a) = a$ will be the number of ants only using the pheromone trail and heuristic information from the first objective. How these numbers are determined is not completely clear, but Ghoseiri & Nadjari (2010) state that the ACO is not sensitive to parameters a, b and the number of ants.

The other parameters q_0, α, β, δ, φ and ρ are determined experimentally, this will be explained in Section 4.3.1. The initial amount of pheromone trail for each arc in the network, for both objectives will be 0.5.

3.2 Humanoid ant algorithm

In this section the incorporation of the HUMANT algorithm into the ACO algorithm is explained. The HUMANT algorithm exists of a combination of the PROMETHEE II method and the ACO algorithm as explained in the previous section.

The PROMETHEE II method uses a complete a priori ranking of all possible *actions* (Brans et al.,1986). In this research such an *action* is selecting a path.

All paths found by ants are given two different scores. These scores are evaluated with respect to the other paths that are found, so they are always relative. The paths with the highest scores is the most preferred.

An important feature of the HUMANT proces is the fact that the ACO and PROMETHEE II are not executed separately, but the PROMETHEE II is used within the ACO. The HUMANT algorithm used in this research is based on the HUMANT as proposed by Mladineo et al. (2015), but we changed one important aspect. They incorporated the PROMETHEE II parameters in the decision of the next arc as well the pheromone update. In our version of the HUMANT algorithm, we will only incorporate the PROMETHEE II in the pheromone update since our decision making processes is already more extended than the one used by Mladineo et al. (2015). In Appendix E we explained which transition procedure was proposed by Mladineo et al. (2015) and why we choose to not use this.

3.2.1 Scoring methods

The scoring method $\theta'_{i,j}$, which is called the net-scoring method, is used for two separate procedures. The net-score is used to give a score to all the paths in the non-dominated solution set. The function works as follows:

$$\theta'_{i,j} = \frac{\frac{1}{h-1}\sum_{k=1}^{h}(\Phi(x_{i,j}, x_{i,k}) + (1 - \Phi(x_{i,k}, x_{i,j})))}{2}. \tag{12}$$

With h the number of the current ant, $x_{i,j}$ is the arc between node i and node j and $\Phi(a, b)$ the aggregated preference index for arc a and arc b. This index is defined as follows:

$$\Phi(a, b) = \frac{\sum_{j=1}^{2} \psi_j P_j(a, b)}{\sum_{j=1}^{2} \psi_j}. \tag{13}$$

With ψ_j the preference weight for objective j and $P_j(a, b)$ the preference function between arc a and arc b for objective j.

The preference function used in this research is the following:

$$P_k(a, b) = \begin{cases} 0, & \text{if } C_a^k \geq C_b^k. \\ 1 - \dfrac{C_a^k}{C_b^k}, & \text{if } C_a^k < C_b^k. \end{cases} \tag{14}$$

With C_a^k being the cost for objective k for arc a.

Besides the net-score, every path is also given a net flow. This flow shows how much a path is preferred compared to other paths. It works as follows: First it is calculated how many times the path is preferred compared to other paths in a set. Over this, an average is taken, see (16).Next it is calculated how much the other paths in a set are preferred over the above mentioned path. The average is taken over this, see (17). Finally the *net flow* is calculated by subtracting the negative flow $\theta^-(a)$ from the positive flow $\theta^+(a)$.

$$\theta(a) = \theta^+(a) - \theta^-(a). \tag{15}$$

$$\theta^+(a) = \frac{1}{|A| - 1} \sum_{x \in A} \Phi(a, x). \tag{16}$$

$$\theta^-(a) = \frac{1}{|A| - 1} \sum_{x \in A} \Phi(x, a). \tag{17}$$

Where A is the set of non-dominated solutions and a is one path from the set.

3.2.2 Local and global updates

The local and global pheromone evaporation as described in Section 3.1.2 will stay the same, but the pheromone update will be different. Mladineo et al. (2017) use a preference based pheromone update that will now be described. When every ant from a colony has constructed a path in the network, a global pheromone update will be done for all the arcs that are in paths in the global non-dominated set as follows:

$$\tau^1_{i,j} = \min\left(1, \tau^1_{i,j} + \Delta\tau^1(i,j)\right). \tag{18}$$

$$\tau^2_{i,j} = \min\left(1, \tau^2_{i,j} + \Delta\tau^2(i,j)\right). \tag{19}$$

$$\Delta\tau^k_{i,j} = 2(\theta'(x)) = 2\left(\frac{\Phi(x, s^{id}) + (1 - \Phi(s^{id}, x))}{2}\right). \tag{20}$$

Here, $\Delta\tau^k(i,j)$ is derived from (12), x is the obtained value for objective k and s^{id} is either the optimal solution for the objective or the ideal value, which is a lower bound.

For the global pheromone update, only the arcs in the non-dominated global solution set are compared to each other.

3.3 Performance measures

The ACO algorithm described in Section 3 will be compared to the label correcting algorithm based on three criteria: closeness of the solution generated by the ACO, uniformity of the solutions over the solution space and the rate of extension, which measures the extreme points of the found non-dominated solutions are spread in comparison with the Pareto front. The label correcting algorithm is gives the exact Pareto front, but such an algorithm is often time consuming (Zitzler et al.,2000). This label correcting algorithm finds non-dominated paths for every node in the network and gives an exact Pareto front for the non-dominated paths created for the destination node t.

Firstly the set of non-dominated paths generated by the ACO will be assessed on the distance, D_{Ave}, to the Pareto frontier generated by the label correcting algorithm. The distance for each element from the set of solutions from ACO will be the distance to the closest point on the label correcting Pareto frontier. This distance, as proposed by Zitzler et al. (2000), is calculated as follows:

$$D_{Ave} = \frac{1}{|Y'|}\sum_{a \in Y'} \min\{||a - b||; b \in \tilde{Y}\}. \tag{21}$$

Here Y' are the solutions generated by ACO and $\tilde{Y}$ the Pareto frontier generated by the label correcting algorithm. $||\cdot||$ is the Euclidean distance.

Performance measure E_{Ave} then gives the average distance of the error between the ACO solutions and the real Pareto frontier. This function was introduced by Jaszkiewicz (2004). This function favours regions of the solution space where many non-dominated solutions are found.

$$E_{Ave} = \frac{1}{|Y'|} \sum_{a \in Y'} \frac{\min\{||a-b||; b \in \tilde{Y}\}}{||a||}. \tag{22}$$

The performance measure W represents the worst case distance between the set generated by the ACO algorithm and the Pareto frontier.

$$W = \max\ \min\{||a-b||; b \in \tilde{Y}, a \in Y'\}. \tag{23}$$

Performance measure U shows the ratio between the worst case performance U and the average performance D_{Ave} of the ACO. The desired value of U is 1, because in this case all non-dominated solutions generated by the ACO have the same distance to the Pareto frontier.

$$U = \frac{W}{D_{Ave}}. \tag{24}$$

Performance measure SP shows how uniform the non-dominated solutions generated by the ACO are distributed. It poses as the standard deviation of the distance of the solutions generated by the ACO to the Pareto frontier. A lower SP means more uniformity in the distance, which is desired. This function is introduced by Schott (1995).

$$SP = \sqrt{\frac{1}{|Y'|-1} \sum_{a \in Y'} (\tilde{d} - d(a))^2}. \tag{25}$$

Where $d(a) = \min_{a' \in Y' \backslash a}\{\sum_{i=1}^{I} |a_i - a'_i|\}$, $\tilde{d}$ is the mean of all $d(a)$ and I is the set of objectives.

The uniformity of the solutions generated by the ACO can be shown by M with an outcome within the interval $[0, |Y'|]$. A high value of M is desired, since it shows that the non-dominated solutions are spread throughout the network.

$$M = \frac{1}{|Y'-1|} \sum_{a \in Y'} |\{b \in Y'; ||a-b|| > \sigma\}|. \tag{26}$$

Where σ is equal to the distance between the two most extreme points in Y' divided by $|Y'|$.

M_{norm} gives the normalized uniformity. The closer M_{norm} is to 1, the better the distribution of solutions generated by the ACO.

$$M_{norm} = \frac{M}{|Y'|}. \tag{27}$$

Performance measure EX shows how close the extreme solutions generated by the ACO are to the extreme points in the Pareto frontier. When EX equals 1, the extreme points of both match.

$$EX = \frac{\sum_{i=1}^{2} \min\{\max(a_i), \max(b_i)\}}{\sum_{i=1}^{2} \max(b_i)}. \tag{28}$$

With $i \in K$, $a \in Y'$ and $b \in \tilde{Y}$.

4 Results

In this section the data used in this research and the results of the ACO algorithm will be described, how we deal with an unforeseen problem is discussed and the results of the ACO and HUMANT algorithm are analysed.

In this research we use Java to implement the Ant Colony algorithm as shown in Section 3. To visualize the results and calculate the performance measures, as explained in Section 3.3, Matlab r2020a was used. The computer on which the programs where run has 8.00 GB RAM-memory and a 1.99 GHz processor. To perform the Label Correcting algorithm C++ was used, this was run on another computer.

4.1 Data

In order to investigate shortest path problems, instances consisting of networks are needed. These networks can be seen as connected graphs consisting of edges and vertices. One of those vertices is the source node and another one the destination node between which paths need to be found. All vertices in the graph have two coefficients which depict travel time and transportation cost over that arc.

The networks used in this research have a grid structure, see Figure 3 for an example. A width (w) and and a height (h) is specified to indicate the number of rows and columns of nodes in the network. The source and destination node are outside of this grid, one at each end, such that the number of nodes in a network equals $hw+2$. The number of arcs in the network is $2w(2h-1)$. There are arcs going forwards and backwards between the nodes, but the values of the coefficients are not symmetric.

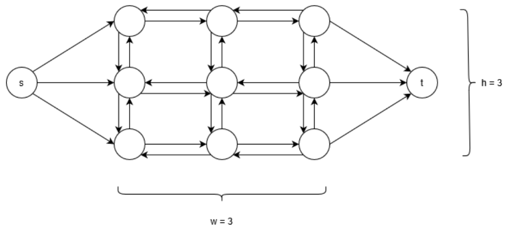


Figure 3: Depiction of a grid network where $w = 3$ and $h = 3$ such that the number of nodes is 11 and the number of arcs is 30.

The method of generating the coefficients for the arcs is the same as the approach used in research by Andersen, Jörnsten, & Lind (1996). Instead of choosing both coefficients for each arc to be from one interval, they create two intervals, namely $\{1, ..., 33\}$ and $\{66, ..., 100\}$. The first coefficient is drawn uniformly with equal probability from either the *low* interval $\{1, ..., 33\}$ or the *high* interval $\{66, ..., 100\}$. The second coefficient is then drawn uniformly from the other interval, such that if the first coefficient is drawn from *low* the second coefficient will be drawn from *high* and vice versa. Andersen et al. (1996) find that this method results in more non-dominated solutions. This method is advised by Hooker (1994) and is used to give insight in the way an algorithm is depended on the characteristics of the network.

In this research a total of 21 instances is used, the smallest of which has a width of 50 and the largest a width of 250 nodes and between each instance there is a 10 node difference in the width. Each network has a height of 100 nodes. For each arc in the network the head and tail nodes are known and the coefficient values are known. Besides this information for each instance the size of the Pareto frontier is known and the (cost, time)- pairs of which the Pareto frontier exists. A description of the networks can be found in Appendix B, Table 3.

4.1.1 Lower bound

The lower bound used in this research will be based on the data. Since the shortest a path can be for an objective is 1, the lower bound on the problem is $1(w+1)$. This is correct since $w+1$ is the minimum number of arcs between node s and node t. The probability a path for one objective has the actual value of $w+1$ is very small, but there is a possibility still, so this must be the lower bound.

We tested to influence of a non-trivial lower bound, see Appendix D, and concluded that a higher lower bound gave better results on average.

4.2 Practical problems and solutions

After implementing the ACO algorithm it seemed that some ants would get stuck somewhere on the network when, for instance, there was no possible next arc because it had visited all adjacent nodes already. We explored two possible solutions for this problem. The first being *killing* the stuck ant and setting the cost of their path to infinity for both objectives. The second solution makes the ant walk back one arc when it gets stuck and make it choose another arc.

The first solution of killing the ants seemed to work really well for the smaller instances in our data set, but for the larger instances the vast majority of ants would be killed leading to no solutions at all. Thus we decided to choose the second method of walking back over the arcs in the path until a new next arc can be chosen.

4.3 Results of the ACO and HUMANT algorithm

In this section the final setting of the parameters of both algorithm will be named, the runtime and actual performance of both algorithms will be analysed, we will experiment with some parameter setting and the way the non-dominated paths are spread over the network are analysed.

4.3.1 Parameters

Parameters q_0, α, β, δ, φ and ρ are initially set to 0.99, 2, 4, 1, 0.999 and 0.99 respectively in the research by Ghoseiri & Nadjari (2010). In consequential runs of the algorithm these parameters are adjusted in some unknown way. When examining the final settings used per instance in the paper by Ghoseiri & Nadjari (2010), it became clear that q_0 was set to 0.99 in 21 out of 23 cases. Therefore q_0 was set to 0.99 initially in our research as well. We later found out that 0.9 was more suitable for q_0. Parameter δ was equal to 0.1 in 19 out of 23 cases, this is why this value is used as well in this research for δ.

For the other parameters there was no value that was used in the vast majority of the cases. Therefore an experiment in selecting those parameters was done. Which values were considered depended on the values which were used multiple times for one parameter in the research by Ghoseiri & Nadjari (2010). The values considered for the remaining parameters are, $\alpha \in \{2, 3, 4\}$, $\beta \in \{3, 4\}$, $\varphi \in \{0.9, 0.99, 0.999\}$ and $\rho \in \{0.9, 0.99, 0.999\}$. There are 108 possible combinations of values for the four remaining parameters.

The idea behind is that tuning the parameters might result in better solutions. The experiment worked as follows, the algorithm was run ten times for every possible combination of the values for the parameters, for each instance. Multiple runs had to be done due to the stochastic nature of the ACO algorithm and the value ten was chosen because Ghoseiri & Nadjari (2010) use ten runs in their research. We assume that a solution with more non-dominated paths is better than a solution with less non-dominated paths, following the paper by Andersen et al. (1996), a large set of solutions should be generated and the quality of the solutions is of subordinate importance.

We will use the solution for each instance that has the biggest set of non-dominated solutions of all the runs. For each possible combination of parameter setting for q_0, α, β, φ and ρ. For parameters q_0, β, φ and ρ the setting for which the best solution was found were the same for each instance, namely 0.9, 4, 0.9 and 0.999 respectively. Except for two instances, namely instance 0 and 17, where the value for ρ was 0.99. The value of α varied between 2,3 and 4. The exact values for α for each instance can be found in Table 3.

The number of ants per colony was predetermined and no analysis was done, since Ghoseiri & Nadjari (2010) have shown the model to be robust. The way a and b were determined is as follows. Once the number of ants n was determined, b was set equal to one third of n, rounded to an integer and a was subsequently set to $n - b$. The number of ants and parameters a and b used for each instance can be found in Appendix B, Table 3.

For the HUMANT algorithm, the same parameter values were used as in the ACO algorithm. Ten runs were done for the HUMANT as well and the solution set with the most paths was choosen for the analysis. The HUMANT has some additional parameters, which are the preference weight ψ which we set to 0.5 for both objectives and the s^{id} which is the lower bound explained in Section 4.1.1. Furthermore we let the algorithm run a hundred times instead of using the stopping condition that was used for the ACO, since preliminary research showed that the HUMANT algorithm converged for all instances after only few runs and the results were not good.

4.3.2 Performance

In Figure 4 a plot for the runtime of the ACO, HUMANT and label correcting algorithm can be found. The runtime of the ACO and HUMANT cannot be compared directly with the label correcting algorithm, due to the use of other software and hardware, but we can get an idea how the running times grow when the size of the network grows. The running time of the label correcting algorithm grows exponentially with the size of the network, while the running time for the other two algorithms seem to be fairly stable. The HUMANT algorithm is a bit slower then the ACO algorithm. The exact running times for the ACO and HUMANT algorithm can be found in Appendix B, Table 4.

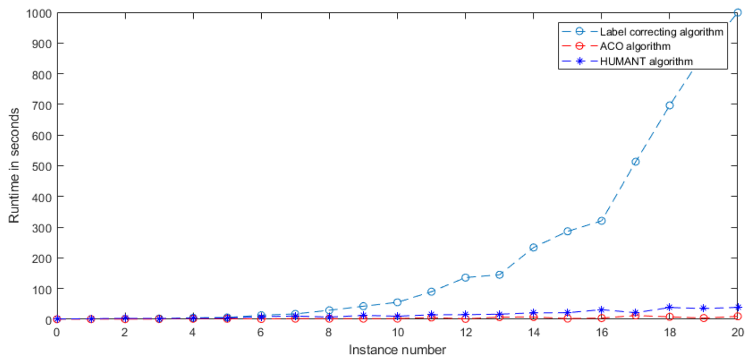
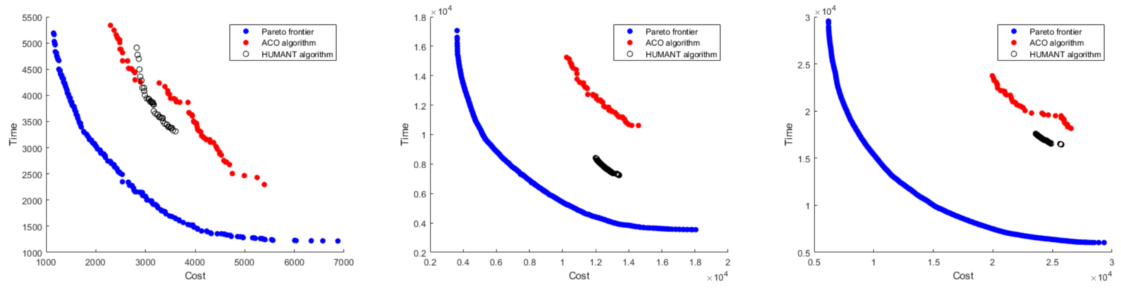


Figure 4: Runtime of the ACO and HUMANT algorithm compared to the label correcting algorithm.

For each analysed instance a plot was made to visualise the Pareto sets. All these plots can be found in Appendix A and three of these can we found in Figure 5. The plots of the results for the smallest, largest and middle instance are shown.



(a) Instance 0 (b) Instance 10 (c) Instance 20

Figure 5: Pareto frontiers from the label correcting algorithm and non-dominated set of the ACO and HUMANT algorithm.

Comparing the performance measures of the ACO algorithm in this research with the results in the research by Ghoseiri & Nadjari (2010), some differences stand out. Firstly, as can be seen from Table 1, the performance measure for the average error distance in this research are all very low, especially in comparison with the results from Ghoseiri & Nadjari (2010). The results are not of the same magnitude. The reason for this in unknown.

The uniformity U of the solutions seems to be quite good. The majority of values for this measure nears 1.0, which is desired. The only solution set that seems to perform not well is the one for instance 4. The conclusion of uniformity can also be drawn form the values for M_{norm}, the uniformity of the results seems good for most instances. For all instances the value for M_{norm} is above 0.9, except instance 12 and 20.

The measure for the efficient set spacing as described in (25) is high for every instance that was analysed. In this paper SP ranged from 13.190 to 120.163, while in the research by Ghoseiri & Nadjari (2010) the SP ranged from 0.87 to 7.03. There are two possible explanations for this difference. The first reason is that the solutions in the Pareto optimal set in this research were indeed not at all evenly spaced out. The other reason could be that Ghoseiri & Nadjari (2010) actually used another formula than they described. Nevertheless, as Schott (1995) stated, the closer the SP is to 0 the better the distribution over the solution space. Some solution sets seem to be spaced far worse than others, especially for instances 7, 8, 18 and 20. Looking at the plots for those four instances this does not become clear immediately.

The ability of the ACO in this research to find Pareto efficient paths in the exteriors of the solution space, so paths were one objective has a high value and the other a low value, is not as good as was found in the research by Ghoseiri & Nadjari (2010). They report values of EX for all instance above 0.95, while we found values between 0.608 and 0.829. They conclude that the ACO is capable of finding solutions in outer areas. We suspect that the ACO does actually not perform well with respect to this measure in networks with structures similar to those used in this research.

Instance	Closeness	Uniformity			Extension
	E_{ave}	U	SP	M_{norm}	EX
0	0.284	1.255	65.557	0.975	0.637
1	0.237	1.312	68.119	0.976	0.700
2	0.222	1.215	51.372	0.976	0.600
3	0.303	1.148	52.356	0.976	0.744
4	0.280	1.400	65.632	0.980	0.679
5	0.290	1.242	65.501	0.972	0.640
6	0.299	1.057	51.575	0.980	0.626
7	0.369	1.211	105.643	0.984	0.795
8	0.356	1.167	120.163	0.982	0.705
9	0.292	1.047	34.625	0.973	0.622
10	0.385	1.031	94.865	0.977	0.718
11	0.306	1.032	39.039	0.974	0.608
12	0.496	1.016	60.845	0.957	0.799
13	0.376	1.039	49.536	0.972	0.708
14	0.370	1.109	67.873	0.980	0.719
15	0.463	1.043	53.404	0.933	0.828
16	0.443	1.047	13.190	0.957	0.778
17	0.439	1.036	13.692	0.990	0.743
18	0.428	1.055	107.684	0.975	0.803
19	0.373	1.017	15.564	0.966	0.671
20	0.408	1.032	140.578	0.964	0.759

Table 1: Performance measures for the ACO algorithm all 21 instances

The HUMANT is used to order a set of Pareto optimal solutions. Such an ordering based on the net score and netweight is depicted in Table 5, Appendix C. Table 2 shows the performance measures as described in Section 3.3 for the HUMANT algorithm for all 21 instances. For the majority of the instances (13 of 21) the average error distance E_{ave} is smaller for the HUMANT algorithm then the ACO algorithm. In the cases that the value for E_{ave} are lower for the ACO, the values are not majorly different from the values for the HUMANT, except for instance 2. On average the HUMANT works 12.06% better with respect to the average distance error. The HUMANT thusly works better or comparable to the ACO algorithm with regard to the average error distance.

Instance	Closeness	Uniformity			Extension
	E_{ave}	U	SP	M_{norm}	EX
0	**0.274**	**1.159**	**39.380**	**0.965**	0.574
1	0.274	**1.202**	**27.754**	0.947	0.684
2	0.333	1.264	**16.724**	**0.981**	**0.706**
3	**0.239**	**1.071**	**11.322**	**0.982**	0.608
4	0.323	**1.098**	**18.457**	0.952	**0.697**
5	0.291	**1.120**	**43.677**	0.931	**0.686**
6	0.301	**1.024**	**29.750**	0.977	0.611
7	**0.349**	**1.018**	**19.923**	**0.989**	0.664
8	0.360	**1.130**	**34.853**	0.970	0.633
9	0.310	**1.030**	**11.645**	**0.991**	**0.630**
10	**0.225**	1.079	**11.752**	0.930	0.589
11	0.410	1.050	**0.746**	0.943	**0.772**
12	**0.302**	1.048	**23.716**	0.846	0.572
13	**0.342**	1.059	**25.716**	**0.973**	0.651
14	**0.329**	**1.010**	**12.112**	0.956	0.705
15	**0.416**	1.045	**49.628**	0.910	0.772
16	**0.306**	1.154	38.514	0.921	0.667
17	**0.314**	1.048	59.951	0.966	0.600
18	**0.315**	**1.009**	**5.342**	**0.991**	0.631
19	**0.357**	**1.039**	20.086	**0.982**	0.663
20	**0.350**	**1.029**	**14.076**	0.889	0.716

Table 2: Performance measures for the HUMANT algorithm all 21 instances. The numbers pressed in bold are performance measures that outperform the ACO.

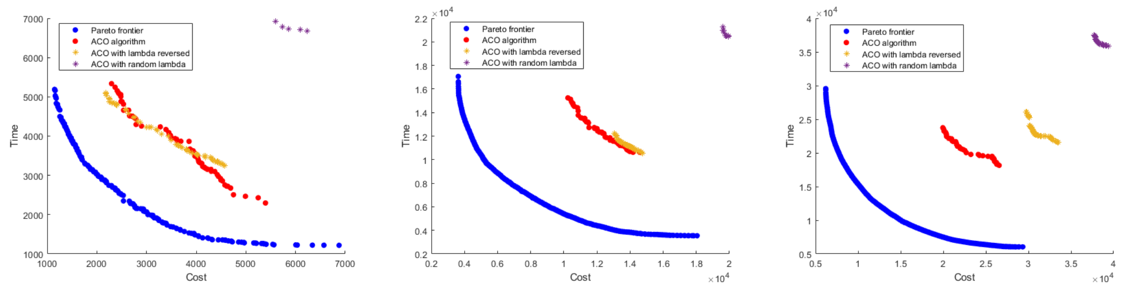
When regarding the uniformity of the solutions from the Pareto efficient set for the HUMANT algorithm, the HUMANT outperforms the ACO in the vast majority of the cases. Especially for the SP measure the difference is quite big. For example the value of SP for instance 18 is 107.7 using the ACO, while 5.342 using the HUMANT algorithm. The reason the HUMANT performs this much better in finding uniform solution is because of pheromone updates. These pheromone updates take the preference weights into account and solutions which are closer to adhering to the weights are preferred in the pheromone update. The HUMANT incorporates preferences into their pheromone updates. Solutions which have a better preference score are assigned proportionally more pheromones. This way the solutions are bound to lie closer together. Since in this case equal weights are used, the solutions all are in the middle of the Pareto set compared to the solutions of the ACO, as can be seen from Figure 5, especially for instances 10 and 20. This can also be seen in Appendix A. On average the HUMANT works 393% better on the SP measure, 3.65% better with respect to the U measure and 0.4% worse with respect to M_{norm}.

This above reason for greater uniformity is the same reason why the ability to find solutions in outer regions of the solution space is worse than that of the ACO on average. Though in 5 of the 21 instances the HUMANT values for EX where higher, on average the HUMANT performs 7% worse. This is not what was expected, since we expected the HUMANT to perform far worse at this measure.

4.3.3 Parameter differentiation

Some parameters were fixed beforehand, such as λ in both the ACO and HUMANT algorithm, and the preference weights ψ_k, k is one of the two objectives, for the HUMANT algorithm. To test the influence of these parameters, some tests were done. In these tests all other parameters were kept constant, as used in the ACO and HUMANT algorithm, except the parameter that is tested. Ten runs are performed with the same parameter input and the solution with the largest set of non-dominated solutions is analysed. We do this analyses on three different instances, namely instance 0, 10 and 20. While analysing these experiments we keep in mind that the algorithm is stochastic and these are only results from 10 runs.

Firstly, while all other parameters were kept the same, we tested what happened when λ as described in (11) was. So 0 when $h \leq a$ and 1 when $h \geq b$, the fraction for when $a < h < b$ stayed the same". We will call this the "reversed lambda". Furthermore, we tested what happened when λ was a random number in the interval $[0, 1]$ for every ant.

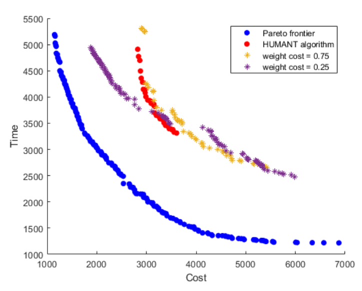


(a) Instance 0 (b) Instance 10 (c) Instance 20

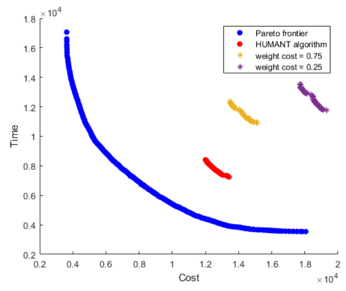
Figure 6: Pareto frontiers and non-dominated set of the ACO with different lambda values.

The results for the three different instances are presented in Figure 6. We only tested the difference with the ACO algorithm. As can be seen from the figure, there seems to not be much difference between the Pareto set when the original λ was used and when the reversed λ was used, except that the original set seems to have more solutions and those are more spread out. The random λ performs way worse in all three cases, which suggests that the way λ is defined is of importance for the algorithm.

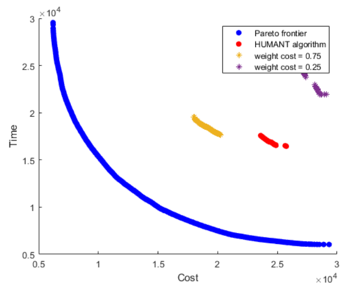
Secondly, the preference weights for the HUMANT were analysed. Previously the weights were set equal for both objectives. To see the influence of these preferences on the results, we set the weight for the cost to 0.25 and for time to 0.75 and vice versa. We expect the non-dominated solution set to be more off center, with lower values for the objectives with the higher preference.



(a) Instance 0



(b) Instance 10



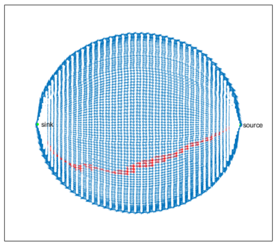
(c) Instance 20

Figure 7: Pareto frontiers and non-dominated set with different weights HUMANT algorithm.

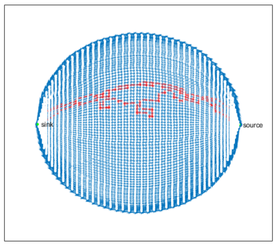
In Figure 7 the results of the experiment regarding the differentiation in weights are presented. For instance 0 we surprisingly see results that are opposite of what was expected. We expected that when the weight for the cost objective was higher, the lowest values for this objective would be found. This is not actually the case, since the lowest values were found for the cost objective, when there was more weight on the time objective. This could have two explanations. Firstly, when more emphasis is put on time, shorter paths for both objectives can be found in this network. Secondly, this results could just be due to the stochastic nature of the algorithm and in another run the results would be different. For instance 10 and 20 the results did turn out as expected, namely when the higher weight was put on the cost objective, the resulting Pareto front was to the left of the original HUMANT Pareto set and to the right when less weight was put on the cost objective. This suggests that the weight parameter has a larger influence when the network is bigger.

4.3.4 Spread over the network

In Figure 8 a visualisation of instance 0 is made. It can be seen that the arcs that are in the non-dominated solution set are mostly in the same region of the network, even though the number of paths is quite high and all these paths are unique. A possible reason of this is quick convergence. This convergence can be either due to the method of pheromone update or method of picking the next arc. Nevertheless, the networks in Figure 8 suggest that a set of non-dominated paths are grouped together.



(a) ACO, 46 paths



(b) HUMANT, 63 paths

Figure 8: Non-dominated set for the HUMANT and ACO algorithm for instance 0. The green arcs are the unexplored arcs, the blue arcs are the explored arcs and the red arcs are all in the final global non-dominated set.

5 Conclusion and discussion

This thesis has two aims, namely finding sets of Pareto optimal shortest paths based on two objectives in networks and ordering such a set of shortest paths in order to make the decision making process easier.

First, we performed a literature review and found that the Ant Colony optimization algorithm would be appropriate to use in the light of our shortest path problem. Furthermore, we found that a combination of the PROMETHEE method and the Ant Colony optimization would be appropriate in ordering the Pareto optimal set. To analyse the performance of the ACO algorithm, we compared the results with the labelling algorithm. This algorithm finds the exact paths but its runtime increases exponentially with the size of the network. The runtime of the ACO algorithm on the other hand increases in proportion with the problem size. We found that the algorithm works well with respect to some measures and very poorly with respect to others. The algorithm seems to work well but not excellent for the problem instances used in this research.

Next we implemented the HUMANT algorithm and compared the results with the label correcting algorithm. This algorithm seems to work better on average compared to the ACO algorithm. The only aspect on which the ACO performs significantly better is the wideness of search range in the solution space. We found the HUMANT to be a relatively simple way of incorporating preferences in the ACO algorithm and since the results are overall better we would advice to use it, since it simplifies decision making in practical situations.

On average the HUMANT algorithm performed 80% better, when all 5 performance measures a taken into account. This seems a major improvement, but this results from one of the performance measures SP, in which the HUMANT performs 393% better. For 3 of the 5 performance measures analysed, the HUMANT outperforms the ACO.

The ACO and HUMANT algorithm both seem to be converging to a certain set of paths very quickly. We did not research the speed of convergence in this research, which is very important in the analysis of the algorithm. This should be investigated in further research, since slower convergence may also find more paths in extreme regions of the solution space.

For the ACO we decided to use the non-dominated set with the largest number of paths, following Andersen et al. (1996). This might not be the solution set with the best values for the performance measures. In further research, other approaches of selecting the final solution set that will be used could be explored. Furthermore, when two solution sets had the same cardinally we used the first found solution set, while the second found set might have performed better.

In this research a trivial lower bound was used for one of the procedures for the HUMANT algorithm, we found that using a non-trivial lower bound yields better results with respect to the analysed performance measures. Further research can be done in finding a non-trivial lower bound without the use of the Label Correcting algorithm.

References

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

Brans, J.-P., Vincke, P., & Mareschal, B. (1986). How to select and how to rank projects: The promethee method. *European journal of operational research*, *24*(2), 228–238.

Coello, C. A. C., Lamont, G. B., Van Veldhuizen, D. A., et al. (2007). *Evolutionary algorithms for solving multi-objective problems* (Vol. 5). Springer.

Deneubourg, J.-L., & Goss, S. (1989). Collective patterns and decision-making. *Ethology Ecology & Evolution*, *1*(4), 295–311.

Dijkstra, E. W., et al. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, *1*(1), 269–271.

Doerner, K., Gutjahr, W. J., Hartl, R. F., Strauss, C., & Stummer, C. (2004). Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection. *Annals of operations research*, *131*(1-4), 79–99.

Dorigo, M., & 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., & 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.

García-Martínez, C., Cordón, O., & 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.

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

Goldberg, A. V., & Tarjan, R. E. (1996). Expected performance of dijkstra's shortest path algorithm. *NEC Research Institute Report*.

Hooker, J. N. (1994). Needed: An empirical science of algorithms. *Operations research*, *42*(2), 201–212.

Iredi, S., Merkle, D., & Middendorf, M. (2001). Bi-criterion optimization with multi colony ant algorithms. In *International conference on evolutionary multi-criterion optimization* (pp. 359–372).

Jaszkiewicz, A. (2004). Evaluation of multiple objective metaheuristics. In *Metaheuristics for multiobjective optimisation* (pp. 65–89). Springer.

Lawler, E. L. (2001). *Combinatorial optimization: networks and matroids.* Courier Corporation.

Mishra, K., & Harit, S. (2010). A fast algorithm for finding the non dominated set in multi objective optimization. *International Journal of Computer Applications*, *1*(25), 35–39.

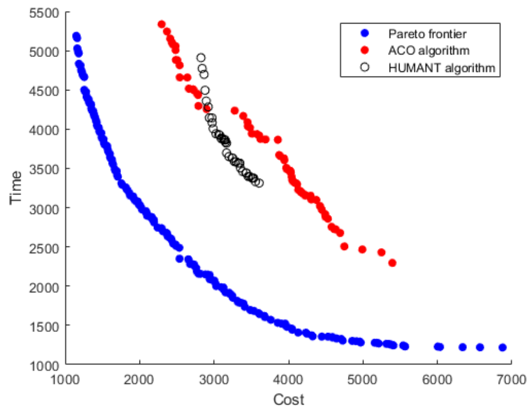
Mladineo, M., Veža, I., & Gjeldum, N. (2015). Single-objective and multi-objective optimization using the humant algorithm. *Croatian Operational Research Review*, *6*(2), 459–473.

Mladineo, M., Veza, I., & Gjeldum, N. (2017). Solving partner selection problem in cyber-physical production networks using the humant algorithm. *International Journal of Production Research*, *55*(9), 2506–2521.

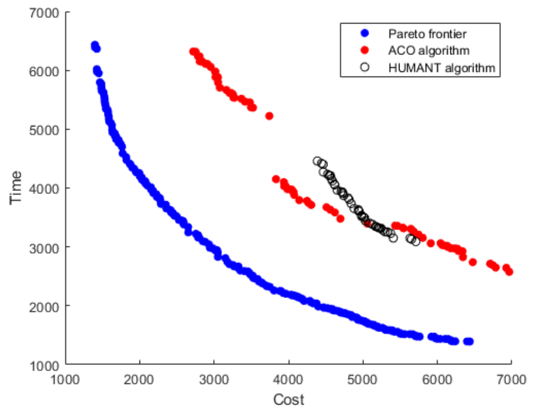
Schott, J. R. (1995). *Fault tolerant design using single and multicriteria genetic algorithm optimization.* (Tech. Rep.). Air force inst of tech Wright-Patterson afb OH.

Zitzler, E., Deb, K., & Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, *8*(2), 173–195.

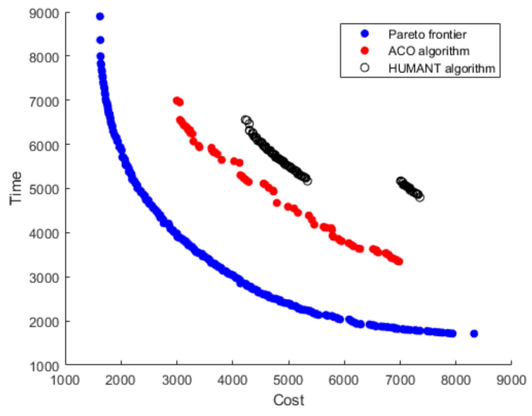
A Visualisation of Pareto frontiers



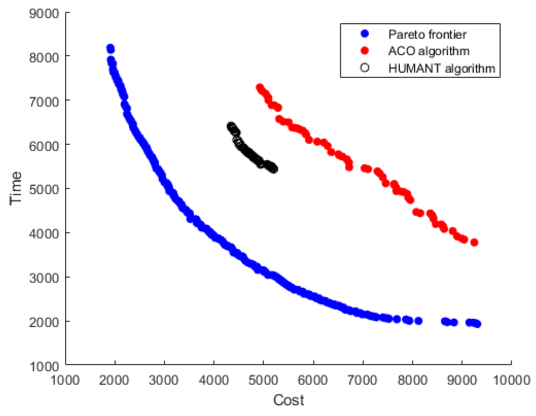
(a) Instance 0



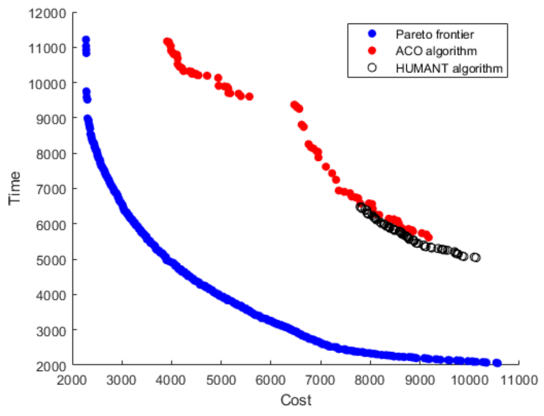
(b) Instance 1



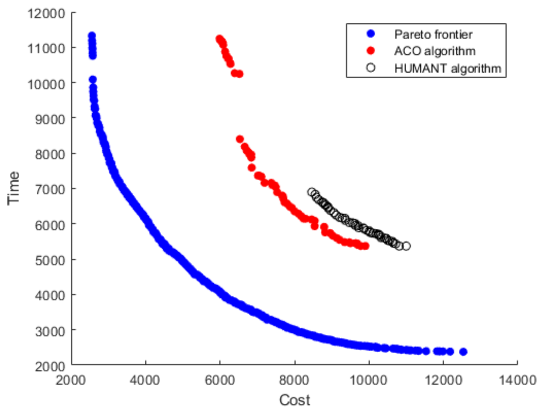
(c) Instance 2



(d) Instance 3

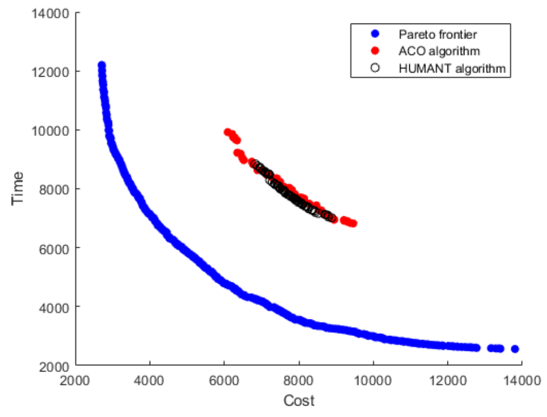


(e) Instance 4

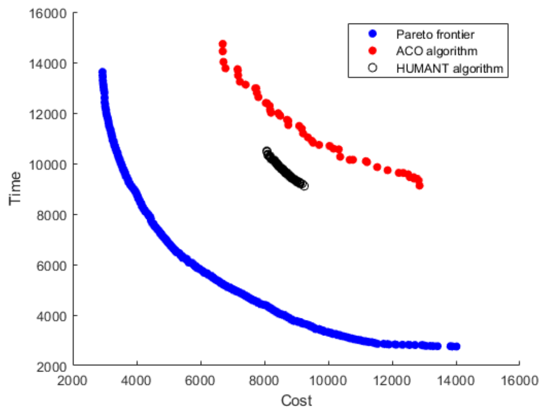


(f) Instance 5

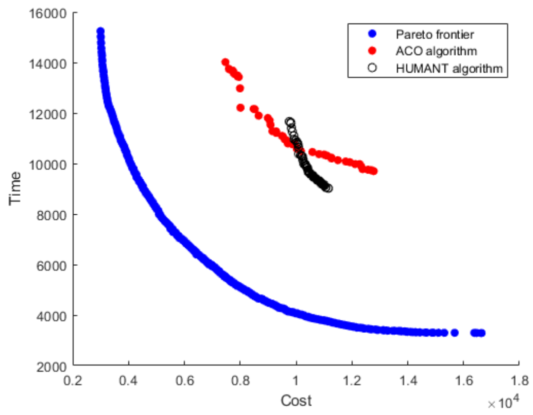
Figure 9: Results from the ACO and HUMANT algorithm compared with the Pareto frontier for instances 0 to 5.



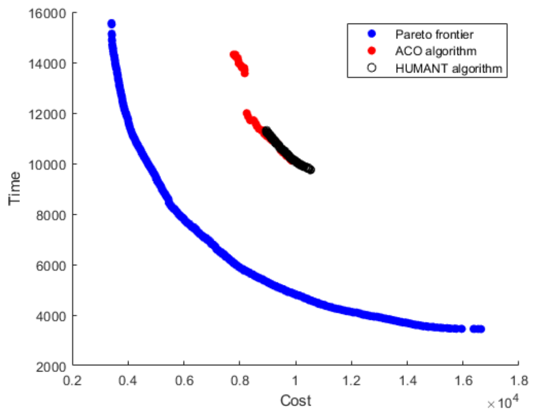
(a) Instance 6



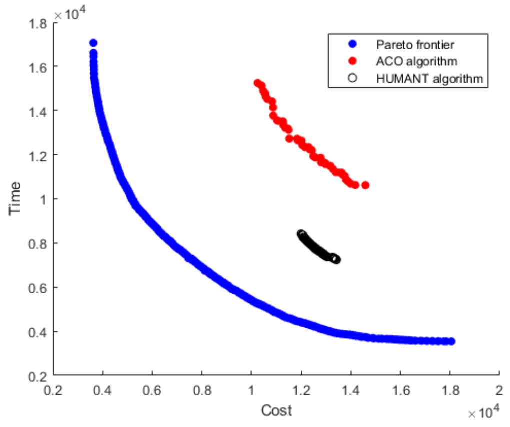
(b) Instance 7



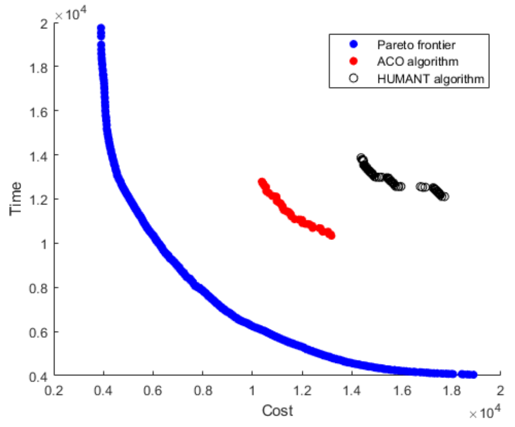
(c) Instance 8



(d) Instance 9

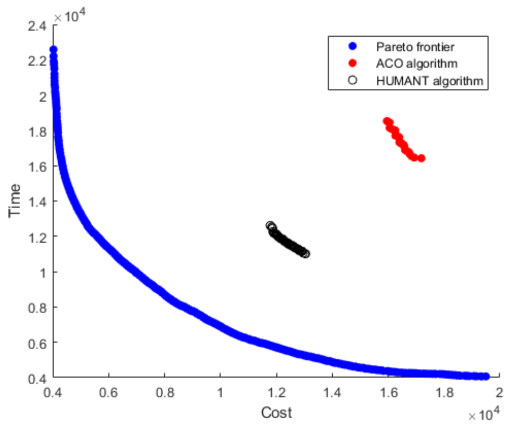


(e) Instance 10

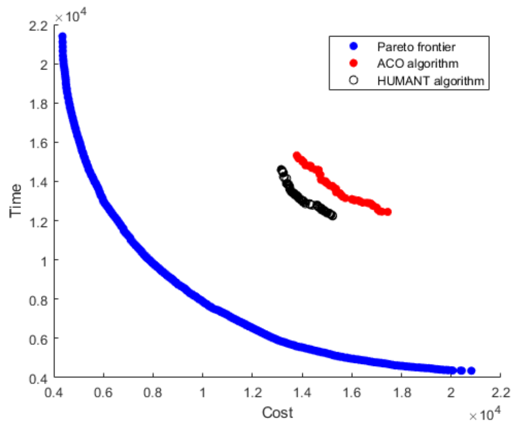


(f) Instance 11

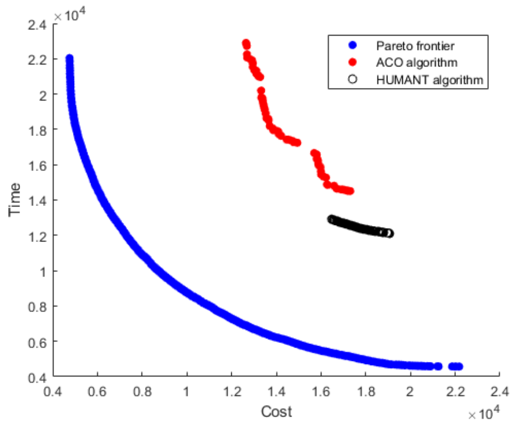
Figure 10: Results from the ACO and HUMANT algorithm compared with the Pareto frontier for instances 6 to 11.



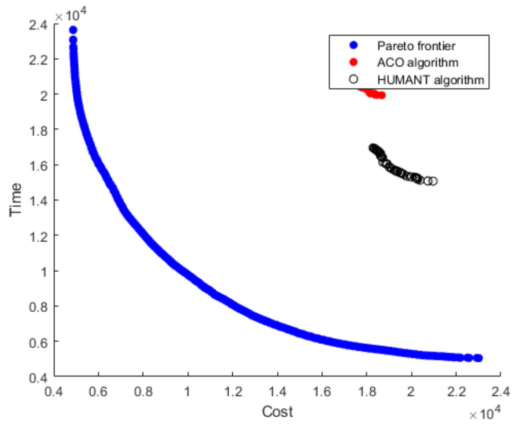
(a) Instance 12



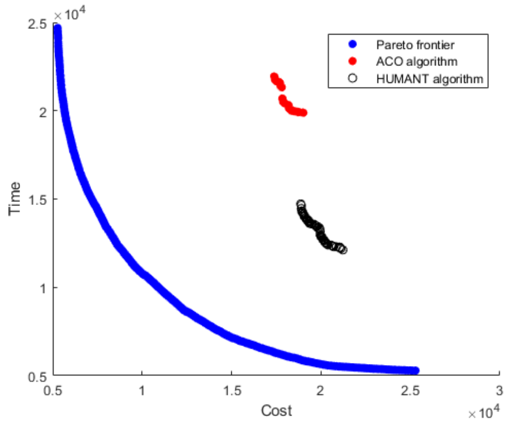
(b) Instance 13



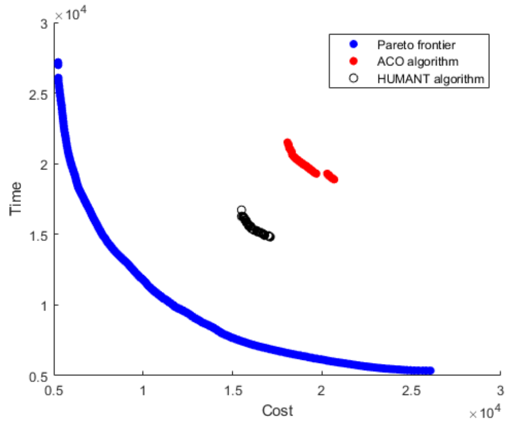
(c) Instance 14



(d) Instance 15

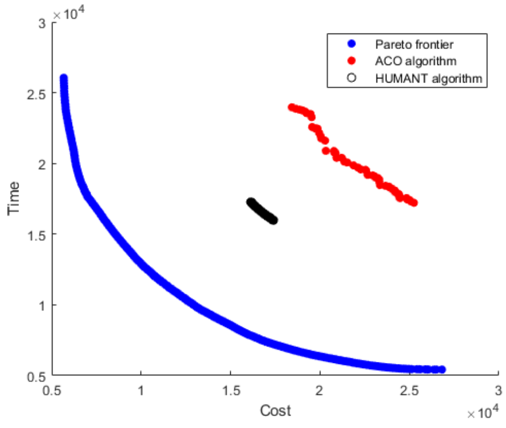


(e) Instance 16

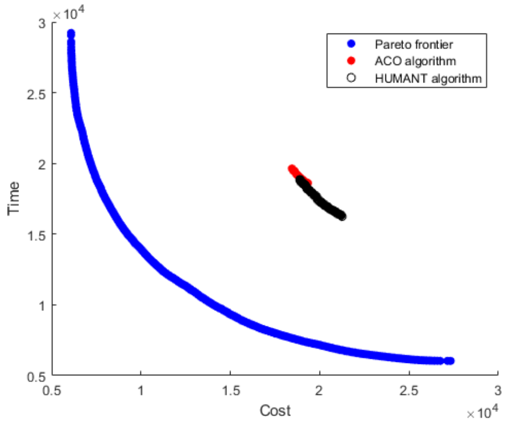


(f) Instance 17

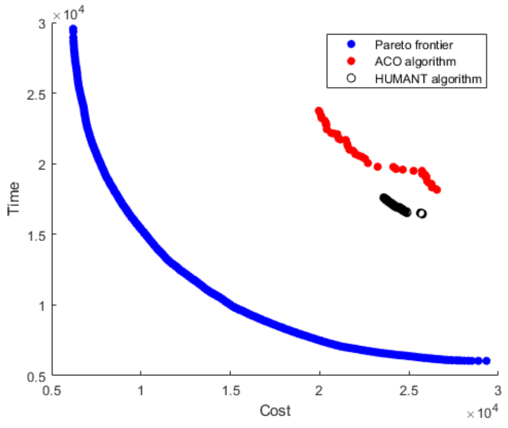
Figure 11: Results from the ACO and HUMANT algorithm compared with the Pareto frontier for instances 12 to 17.



(a) Instance 18



(b) Instance 19



(c) Instance 20

Figure 12: Results from the ACO and HUMANT algorithm compared with the Pareto frontier for instances 18 to 20.

B Results of the ACO and HUMANT algorithm

Problem instance	Network specification		parameters			
	number of nodes	number of arcs	ants	a	b	α
0	5002	19900	200	50	150	3
1	6002	23880	250	63	187	4
2	7002	27860	300	75	225	3
3	8002	31840	350	88	262	2
4	9002	35820	400	100	300	4
5	10002	39800	450	113	337	4
6	11002	43780	500	125	375	3
7	12002	47760	550	138	412	4
8	13002	51740	600	150	450	2
9	14002	55720	650	163	487	4
10	15002	59700	700	175	525	2
11	16002	63680	750	188	562	4
12	17002	67660	800	200	600	2
13	18002	71640	850	213	637	2
14	19002	75620	900	225	675	4
15	20002	79600	950	238	712	4
16	21002	83580	1000	250	750	2
17	22002	87560	1050	263	787	3
18	23002	91540	1100	275	825	3
19	24002	95520	1150	288	862	2
20	25002	99500	1200	300	900	2

Table 3: Network specification and parameter settings

Problem instance	ACO			HUMANT	
	runtime (s)	iterations	# solutions	runtime (s)	# solutions
0	0.439	22	73	0.907	40
1	0.577	37	80	2.033	45
2	0.675	20	85	3.042	127
3	0.947	29	62	2.743	74
4	1.796	38	82	3.44	73
5	2.066	36	64	4.264	47
6	1.161	18	57	8.567	74
7	2.298	24	50	10.86	86
8	2.238	25	49	7.335	83
9	2.192	29	54	12.821	132
10	2.347	22	56	10.192	94
11	5.823	39	58	14.636	64
12	1.686	37	8	14.998	112
13	7.306	25	57	16.226	86
14	7.621	31	78	20.976	112
15	2.462	7	23	21.521	68
16	4.111	13	57	31.304	59
17	12.463	32	118	20.978	54
18	8.192	24	54	38.593	163
19	4.148	14	50	35.327	124
20	9.671	28	49	38.92	97

Table 4: Results of the ACO algorithm and HUMANT algorithm

C Ranking of a Pareto optimal set of paths

Rank	Cost criterion	Time criterion	Net score θ'	Net flow θ
1	3199	3650	0,508343	0,016686
2	3168	3697	0,507676	0,015353
3	3384	3460	0,507363	0,014725
4	3273	3586	0,506993	0,013985
5	3458	3396	0,506553	0,013106
6	3353	3507	0,506419	0,012837
7	3242	3633	0,506218	0,012437
8	3250	3632	0,505688	0,011377
9	3427	3443	0,50551	0,011019
10	3296	3585	0,505373	0,010745
11	3481	3395	0,505066	0,010132
12	3435	3442	0,505026	0,010052
13	3555	3331	0,504514	0,009029
14	3324	3568	0,50446	0,008919
15	3509	3378	0,504346	0,008692
16	3025	3942	0,503642	0,007284
17	3524	3378	0,503356	0,006711
18	3532	3377	0,502898	0,005796
19	3347	3567	0,502874	0,005749
20	2988	4007	0,502747	0,005493
21	3606	3313	0,502479	0,004958
22	3093	3885	0,501735	0,00347
23	3099	3878	0,501694	0,003388
24	3062	3932	0,501324	0,002649
25	3076	3924	0,500706	0,001413
26	3152	3834	0,500293	5,87E+11
27	3122	3877	0,499968	-6,36E+10
28	3167	3821	0,499948	-1,04E+12
29	2974	4081	0,499616	-7,69E+11
30	3136	3868	0,499436	-0,00113
31	2937	4146	0,498941	-0,00212
32	3144	3867	0,49888	-0,00224
33	3150	3860	0,498847	-0,00231
34	2972	4144	0,496254	-0,00749
35	2921	4283	0,49287	-0,01426
36	2895	4358	0,491101	-0,0178
37	2879	4495	0,485634	-0,02873
38	2866	4698	0,47733	-0,04534
39	2840	4773	0,476157	-0,04769
40	2824	4910	0,471723	-0,05655

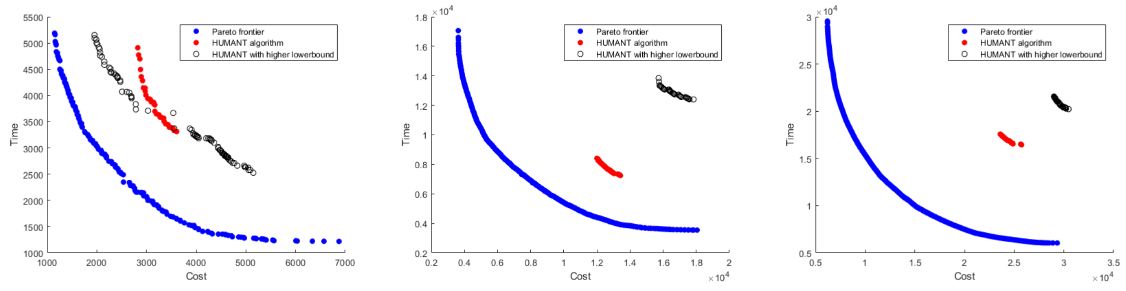
Table 5: Ranking of 40 possible solutions for instance 0 using the HUMANT algorithm using equal weights for both criterion.

D Sensitivity lower bound for the HUMANT

In order to test the influence of the value of s^{id} in (20) we used a higher, non-trivial lower bound . Originally we use a lower bound of $(w + 1)$, which is very low. This might result in low values of $\Delta\tau_{i,j}^{k}$ and this means that the pheromone update would have little to no influence.

The non-trivial lower bound that will be used is the lowest possible value found for the cost and time objectives using the Label Correcting algorithm, i.e. the two most extreme points in the solution space using the Label Correcting algorithm. The reason we did not use this originally in our research is because computing the Pareto Frontier with the Label Correcting algorithm can be time consuming as shown in Figure 4 and computing both the Label Correcting algorithm and the HUMANT algorithm in practice, would make the use of the HUMANT algorithm obsolete.

Nevertheless, it is interesting to see the influence of this lower bound on the algorithm. In Figure 13 the results for the HUMANT algorithm with the original lower bound as explained in Section 4.1.1 and for the HUMANT algorithm with the non-trivial lower bound for three different instances is shown. For instance 10 and 20 it seems that the lower bound actually results in the algorithm finding longer paths, for instance 0 the non-dominated solution set of the algorithm with non-trivial lower bound seems to perform better in the spread of the solutions. To verify this, we also computed the performance measures explained in Section 3.3.



(a) Instance 0 (b) Instance 10 (c) Instance 20

Figure 13: Pareto frontiers from the label correcting algorithm and non-dominated set of the HUMANT algorithm and HUMANT with a higher lowerbound.

In Table 6 the performance measures of the HUMANT with original and non-trivial lower bound are presented. Except for the uniformity measure U, where the HUMANT with original lower bound outperforms the HUMANT with non-trivial lower bound for all three instances, and spacing measures SP, where the opposite is true, we cannot see directly which version performs better. With respect to the closeness, the original lower bound performs 17% worse on average. This is odd, since both instance 10 and 20 seem to perform far worse in closeness when the non-trivial lower bound is used. With respect with approximation of uniformity distribution measure M_{norm}, the HUMANT algorithm with original lower bound performs 12.5% better and with respect to the extension EX the original lower bound performs 9.6 % worse.

	Algorithm specification	Closeness	Uniformity			Extension
		E_{ave}	U	SP	M_{norm}	EX
Instance 0	original lowerbound	0.284	**1.255**	65.557	0.975	**0.637**
	non-trivial lowerbound	**0.241**	1.308	**54.848**	**0.976**	0.636
Instance 10	original lowerbound	**0.385**	**1.031**	94.865	**0.977**	0.718
	non-trivial lowerbound	0.442	1.090	**58.494**	0.889	**0.861**
Instance 20	original lowerbound	0.408	**1.032**	140.758	**0.964**	0.759
	non-trivial lowerbound	**0.407**	1.052	**32.892**	0.754	**0.842**

Table 6: Performance meausures for HUMANT algorithm compared with the label correcting approach and HUMANT with non-trivial lowerbound. The bold pressed are the better values.

We can conclude that on average the HUMANT with the non-trivial lower bound outperforms the HUMANT with original lower bound based on the used performance measures. We therefore expect it to be useful to perform further research towards the use of a non-trivial lower bound for the problem at hand.

E HUMANT transition procedure

The HUMANT used in this research was inspired by Mladineo et al. (2015 & 2017). They use a combination of the PROMETHEE II method and ACO algorithm to solve the multi-objective partner selection problem.

The transition procedure used in their research incorporates the preference net-score to give the ants a preference while choosing the next arc. This net-score replaces the original heuristic parameter. The transition procedure is the following:

$$p_{i,j} = \begin{cases} \dfrac{[\tau_{i,j}(t)]^{\alpha}[\theta'_{i,j}]^{\beta}}{\sum_{u\in\omega(i)}[\tau_{i,u}(t)]^{\alpha}[\theta'_{i,u}]^{\beta}}, & \text{if } j \in \omega(i), \\ 0, & \text{otherwise.} \end{cases} \tag{29}$$

Where all notation remains as explained in Section 3.2. We ran some tests with this procedure but found that it performed poorly compared with the ACO used in this thesis. We also tried several combinations of the transition procedure explained in Section 3.1.3 and (29), but the transition procedure as in Section 3.1.3 remained the best functioning. Which is why we did not incorporate the net-score or any other preference functions into the HUMANT algorithm transition procedure and it remained as in the ACO.