A Population Based ACO Algorithm for the Combined Tours TSP Problem

Martin Clauß
Fraunhofer FKIE
Fraunhoferstraße 20
D-53343
Wachtberg-Werthhoven, Germany
martin.clauss@fkie.fraunhofer.de

Lydia Lotzmann
University of Leipzig
Dept. of Computer Science
Augustusplatz 10
D-04109 Leipzig, Germany
mam10crs@studserv.uni-leipzig.de

Martin Middendorf
University of Leipzig
Dept. of Computer Science
Augustusplatz 10
D-04109 Leipzig, Germany
middendorf@informatik.uni-leipzig.de

ABSTRACT

In this paper we apply a Population based Ant Colony Optimization (PACO) algorithm for solving the following new version of the Traveling Salesperson problem that is called the Combined Tours TSP (CT-TSP). Given are a set of cities, for each pair of cities a cost function and an integer k. The aim is to find a set of k (cyclic) tours, i.e., each city is contained exactly once in each tour and each tour returns to its origin city, which have minimum total costs. In this paper the case of finding two tours is studied where the costs of one tour depends on the other tour. Each pair of cities has a distance and a weight which influence the costs of the tours. The weight is used to define if it is advantageous or disadvantageous when the corresponding pair of cities is contained, i.e., neighbouring, in both tours. Different heuristics that the ants of the PACO use for the construction of the tours are compared experimentally. One result is that it is (often) advantageous when the heuristic for the second tour is different from the heuristic for the first tour such that the former heuristic uses knowledge about the first tour.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*

Keywords

Ant colony optimization, population based ACO, traveling salesperson problem, metaheuristic

1. INTRODUCTION

A variant of the Traveling Salesperson problem is introduced in this paper where the aim is to find for a given set of cities several (cyclic) tours such that each of them connects all given cities, i.e., each tour contains each city exactly once and returns to its origin city. The difficulty is that the costs of each tour depend on the other tours and the total costs of all tours should be minimized. In the classical Traveling Salesperson problem (TSP) the aim is to find a single shortest, i.e., cheapest, (cyclic) tour. The TSP is one of the most investigated combinatorial optimization problems. It is an NP-hard problem and one of the standard benchmark problems that are used for testing metaheuristics. Many variants of the TSP have been studied in the literature (for an overview see Section 2).

In this paper we use the Population based Ant Colony Optimization metaheuristic (PACO) that was proposed in [7] to solve the new variant of the TSP. This variant is called Combined Tours TSP (CT-TSP). An instance of the CT-TSP consists of a set of cities, for each pair of cities a cost function (distance function), and an integer $k \geq 1$. The aim of the CT-TSP is to find k (cyclic) tours which have minimum total costs. The cost function for a pair of cities (i, j) assigns for each tour that includes a drive from i to j a corresponding cost. We are interested in particular in cost functions where the costs that a tour has for driving from i to j depend on how many other tours include a drive from i to j. The notation k-CT-TSP is used for the CT-TSP with fixed k. In this initial study on the CT-TSP we restrict us to the 2-CT-TSP, i.e., to the case of finding $k = 2$ tours. Note, that 1-CT-TSP is the standard TSP problem. Observe, that the special case where it holds for every pair of cities (i, j) that the cost for driving from i to j is always the same fixed value d_{ij} can be reduced to the standard TSP problem. Simply, choosing k times the cheapest TSP tour is the optimal solution. It follows that k-CT-TSP is an NP-hard problem for every $k \geq 1$.

As an application of the CT-TSP consider a scenario where companies have to pay fees for using the roads between a given set of cities. The administration that defines the road fees uses them to regulate the traffic on the different roads. If the administration wants to keep the traffic on the road from a city i to a city j low it might define increasing costs for each use of this road during a day by a company. The first use of the road might cost a company d_{ij}, the second use might cost $w_{ij}d_{ij}$ where $w_{ij} > 1$, the third use might cost $2w_{ij}d_{ij}$, and so on. If in this application d_{ij} is the distance between city i and city j then the costs for the first use are equal to the distance between i and j. If, on the other hand, $w_{ij} < 1$ then the average costs for using the road from i to j by the company decrease with an increasing number of tours that use the road. Now consider a company that has

.

BICT 2015, December 03-05, New York City, United States
Copyright © 2016 ICST
DOI 10.4108/eai.3-12-2015.2262573

to plan k tours for one day such that the total costs for all k tours are minimal. Then the corresponding tour planning problem is an instance of the k-CT-TSP.

As another application of the CT-TSP consider a scenario with set of robots in a fabrication site. Each robot has to drive a tour that connects all machines in the fabrication site. The fabrication site is setup in the morning so that all lanes that the robots can use are clean and safe. When more than r robots have used a lane between two machines it is necessary to make a security check and to clean the lane. Each such check of a lane might lead to fixed costs $c > 0$. Then the costs for each of the first $r - 1$ robots that uses a lane might be d_{ij}. For example, d_{ij} could be the length of the lane or the time it takes a robot to drive along the lane. The costs for the rth robot that uses the lane might then be $d_{ij} + c$, i.e., the additional costs for the check of the lane have to be paid. Analogously, the costs for the mrth use of the lane are $d_{ij} + c$ and for all other uses the costs are d_{ij}, $m \geq 2$. The problem to plan k tours for the robots such that the total costs are minimum is an instance of the k-CT-TSP.

In Section 2 we present background information on the TSP and its variants, on the Population based ACO metaheuristic (PACO), and on related literature. The CT-TSP problem is defined formally in Section 3. How the PACO is applied to the 2-CT-TSP is described in Section 4. The experimental setup is described in Section 5. The experimental results are presented in Section 6 and conclusions are given in Section 7.

2. BACKGROUND AND RELATED LITERATURE

Formally, the TSP can be defined in graph theoretic terms as follows. Given are a set of vertices $V = \{1, \dots, n\}$, a set of edges $E = V^2$, and for each edge $(i, j) \in E$ a length $d_{ij} \geq 0$. The elements of V are also called cities and value d_{ij} is called the distance between city i and city j. A (cyclic) tour such that each city is contained exactly once in the tour is described as a permutation $\pi = (\pi_1, \pi_2, \dots, \pi_n)$ of $\{1, \dots, n\}$ such that π_{i+1} is the successor of π_i for $i \in \{1, \dots, n-1\}$ and π_1 is the successor of π_n. For a permutation $\pi = (\pi_1, \pi_2, \dots, \pi_n)$ of $\{1, \dots, n\}$ we define $(i, j) \in \pi$ iff there exists an $h \in 1, \dots, n-1$ such that $i = \pi_h$ and $j = \pi_{h+1}$ or $i = \pi_n$ and $j = \pi_1$, i.e., i and j are neighbouring in the corresponding tour. The problem is to find a shortest (cyclic) tour, i.e., a permutation $\pi = (\pi_1, \pi_2, \dots, \pi_n)$ of $\{1, \dots, n\}$, such that the total length $l(\pi) := \sum_{(i,j)\in\pi} d_{ij}$ is minimum. Several variants of the TSP have been studied in the literature. A short overview is given in the following (for more information see, e.g., [8, 12]).

Some TSP variants exists where not all given cities have to be visited. One example is the generalized TSP (GTSP), also known as the set TSP, where the set of cities is partitioned into clusters [5]. Then the problem is to find a shortest tour that contains exactly one city from every cluster. In a variant of this problem the tour should contain at least one city from each cluster. Note, that this variant is different from the GTSP when the triangle equation does not hold. Another example is the Prize Collecting Traveling Salesman Problem where a salesperson receives a reward or a penalty in every city [6]. The aim is that the salesperson visits a subset of the cities such that the tour length is minimum under the restriction that a given minimum total reward has to be earned.

Other TSP variants consider scenarios where tours have to be planned for several days. In the period TSP one tour must be planned for each of $m \geq 2$ days such that city i is visited on $r_i \geq 1$ tours for given numbers r_i, $i \in \{1, 2, \dots, n\}$ (e.g., [21]). The aim is to minimize the total length of the tours. A specific variant is the Single-Double Traveling Salesman Problem where some cities have to be visited every day and some cities have to be visited only every second day. Hence, the aim is to find two tours — one for each of two successive days.

Other variants of the TSP have different objective functions. One example is the minimax version (bottleneck TSP, BTSP) where the aim is to find a tour where the largest distance between two neighbouring cities in the tour is minimal (e.g., [9]). Another example is the maximin version (maximum scatter TSP, MSTSP) where the aim is to find a tour where the smallest distance between two neighbouring cities in the tour is maximal (e.g., [9]).

Several variants of the TSP exist with several salespersons. It is typical for these variants that the cities have to be partitioned such that each salesperson visits exactly the cities in one set of the partition. In the multiple TSP (mTSP) there exist $m \geq 2$ salespersons that start from the same city $v_0 \in V$ which is called depot ([1]). The goal is to find for each salesperson a tour that starts at the depot and ends there such that each city is contained in one of the tours and such that the sum of all tours is minimal. There exist several variants of the mTSP. For example, it could be required that the number of cities in the different tours is as equal as possible or that each tour contains at least one city that is different from the depot. Another variant has several depots and each salesperson starts from a different depot.

It should be noted, that a large class of TSP variants is motivated by applications in tour planning and vehicle routing. The double TSP (DTSP) is an example that is somewhat related to the TSP variant considered in this paper. A salesman has to find two tours for a set of cities: one pick up tour and one delivery tour. The aim is to minimize the total lengths of both tours. However, there exist restrictions for packing and unloading the items to the car (e.g., the car has a certain number of stacks) such that the sequence of pick-ups leads to restrictions for the possible sequences for delivering the items (see, e.g., [14]). For the area of planning and vehicle routing it is typical that several tours are sought which are done by several vehicles, each of them might only visit a subset of the cities. Typically, there exist additional information and restrictions that have to be considered. An example is that time windows for the different cities are given which define the time intervals when the corresponding city can be visited. Another example is that capacities for the edges are given which restrict the maximum load that can be transported along the corresponding edges.

Clearly, as for many combinatorial optimization problems, also for the TSP and its variants there exist also dynamic versions, probabilistic versions, noisy versions, online versions, and multi-criteria versions.

The ACO metaheuristic is, in general, very successful for solving the TSP and its variants. Therefore, ACO algorithms have been proposed for most of the mentioned TSP variants. Examples are ACOs for the following problems:

the GTSP [20, 17], the Prize Collecting TSP [18], the period TSP [21], the bottleneck TSP [10], the multiple TSP [2, 3, 22], and for many vehicle routing problems (see, e.g., [15] for an overview).

The PACO metaheuristic was proposed in [7]. It is a variant of ACO [4]. Different from ACO, PACO keeps a small population of solutions P which is used to generate the m new solutions of the next iteration. Each new solution is generated by an artificial ant which uses artificial pheromone information. The pheromone information in the standard PACO algorithm for the TSP is given as a pheromone matrix $[\tau_{ij}]$, $i, j \in [1, n]$ with $\tau_{ij} = \tau_{init} + n_{ij} \times \tau_{solution}$ where n_{ij} is the number of solutions in population P that have (i, j) in their tour and where $\tau_{init} > 0$ and $\tau_{solution} > 0$ are parameters. In some PACO algorithms the best solution that has been found so far - the so called elitist solution - influences the pheromone information. If that is the case and (i, j) is included in the elitist solution, then τ_{ij} is defined as $\tau_{ij} = \tau_{init} + \tau_{elite} + n_{ij} \times \tau_{solution}$ where $\tau_{elite} > 0$ is a parameter. For symmetric TSP instances, i.e., where $d_{ij} = d_{ji}$ for all cities i, j, we set $\tau_{ij} = \tau_{ji}$.

To find a new solution an ant in PACO for the TSP starts at a randomly chosen city. An ant that is located at city i uses the following probabilistic rule — as in classic ACO [4] — to decide which city should be chosen as next city:

$$p_{ij} = \frac{\tau_{ij}^{\alpha} \eta_{ij}^{\beta}}{\sum_{h \in S} \tau_{ih}^{\alpha} \eta_{ih}^{\beta}} \tag{1}$$

where η_{ij} is heuristic information, S is the set of cities that are still selectable, and $\alpha > 0$, $\beta \geq 0$ are parameters.

When all m solutions of the current iteration have been generated population P is updated as follows. The best of the solutions — which is called iteration best solution — is added to P and the oldest solution in the population is removed from P (see [7] for alternative population update strategies).

Several studies have shown that PACO is a competitive metaheuristics that is suitable for solving the TSP. For example, it was shown in [7] that PACO is competitive to standard ACO. In [13] it was shown that PACO is competitive to the state-of-the-art ACO algorithms for TSP and PACO has the advantage that it can find good solutions in a shorter computation time. In that study a variant of PACO was considered that uses local search and a restart mechanism since both principles are also used by the other state-of-the-art algorithms for the TSP. An open source framework for evaluating and comparing TSP solvers was developed recently in [19]. In this study several evolutionary computation methods have been benchmarked and it was concluded that PACO is the method of choice. Several variants of PACO for the TSP have been investigated recently in [11].

3. THE CT-TSP PROBLEM

In the TSP variant that is studied in this paper the aim is to find several tours. However, the cost (or length) of a tour is not independent from the other tours. We call this TSP variant the *Combined Tour TSP* (*CT-TSP*). To the best of our knowledge this variant of the TSP has not been studied before in the literature.

Formally, the CT-TSP is defined as follows. An instance of the CT-TSP consists of a set of cities $V = \{1, \ldots, n\}$, for each pair of cities $(i, j) \in V$, $i \neq j$ a cost function (distance function) c_{ij}, and an integer $k \geq 1$. Let $\pi^{(1)}, \ldots, \pi^{(k)}$ be k (cyclic) tours, i.e., permutations of V. Then $c_{ij}(\pi^{(1)}, \ldots, \pi^{(k)})(m)$ are the costs assigned to π_m for driving from i to j, $m \in [1, k]$. For the cost function it is assumed in this paper that $c_{ij}(\pi^{(1)}, \ldots, \pi^{(k)})(m) = 0$ when $(i, j) \notin \pi_m$, i.e., when i and j are not neighbours in the tour π_m. Otherwise, $c_{ij}(\pi^{(1)}, \ldots, \pi^{(k)})(m) \geq 0$. The total costs of tour π_m are defined as $\sum_{(i,j) \in \pi_m} c_{ij}(\pi^{(1)}, \ldots, \pi^{(k)})(m)$ and are denoted by $c(\pi^{(1)}, \ldots, \pi^{(k)})(m)$. The total costs of k tours $\pi^{(1)}, \ldots, \pi^{(k)}$ are defined as $\sum_{m \in [1:k]} c(\pi^{(1)}, \pi^{(2)}, \ldots, \pi^{(k)})(m)$ and are denoted by $c(\pi^{(1)}, \ldots, \pi^{(k)})$. The CT-TSP problem is to find k tours, i.e., k permutations $\pi^{(1)}, \ldots, \pi^{(k)}$ of $\{1, \ldots, n\}$, such that that the total costs $c(\pi^{(1)}, \ldots, \pi^{(k)})$ are minimal. If k is fixed we call the problem the k-CT-TSP.

In the following we consider the CT-TSP only for the case of $k = 2$, i.e., the 2-CT-TSP. The particular cost function that we study is defined in the following. To define the cost function, for each pair of cities (i, j) are given a distance $d_{ij} \geq 0$ and a weight $w_{ij} \geq 0$. The cost for two permutations π_1, π_2 and each pair of cities (i, j), $i \neq j$ is defined by

$$c_{ij}(\pi_1, \pi_2)(1) := d_{ij} \tag{2}$$

and

$$c_{ij}(\pi_1, \pi_2)(2) := \begin{cases} d_{ij} * w_{ij} & \text{if } (i, j) \in \pi_1 \\ d_{ij} & \text{else} \end{cases} \tag{3}$$

Observe, that weight $w_{ij} < 1$ indicates that it is an advantage for the second tour π_2 when the edge (i, j) is used by the first tour π_1 (in the sense that $w_{ij} < 1$ reduces the cost of driving from i to j for the second tour). Correspondingly, if $w_{ij} > 1$ it is a disadvantage for the second tour when (i, j) is included also in the first tour.

For comparison with the standard TSP problem we define for a tour π that $l(\pi) := \sum_{(i,j) \in \pi} d_{ij}$, i.e., $l(\pi)$ is the total length of the tour π. Note, that if $w_{ij} = 1$ for all pairs of cities (i, j) that are part of both tours, i.e., for which $(i, j) \in \pi_1$ and $(i, j) \in \pi_2$, then $c(\pi_1, \pi_2) = l(\pi_1) + l(\pi_1)$. This equation holds also if there does not exist a pair of cities that is included in both tours. Note also, that it is possible that there exist tours π_1 and π_2 with $c(\pi_1, \pi_2) < 2 \times l(\pi^\star)$ where $\pi^\star$ is a shortest tour, i.e., $\pi^\star$ is the optimal TSP solution. Clearly, this is possible only if at least one pair of cities (i, j) exists that is part of both tours and for which $w_{ij} < 1$ holds.

4. THE PACO FOR THE 2-CT-TSP

In this paper we apply the Population based Ant Colony Optimization (PACO) metaheuristic for solving the 2-CT-TSP. The proposed PACO uses two pheromone matrices $\tau = [\tau_{ij}]$ and $\tau' = [\tau'_{ij}]$ with $i, j \in [1, n]$ and n is the number of cities in the 2-CT-TSP instance that has to be solved. For the construction of the first tour (second tour) pheromone matrix $[\tau_{ij}]$ (respectively $[\tau'_{ij}]$) is used as in the PACO for the standard TSP. Pheromone value $\tau_{ij} := \tau_{init} + x_{ij} \times \tau_{elite} + n_{ij} \times \tau_{solution}$ if n_{ij} solutions in the population P have the pair of cities (i, j) in their first tour and where x_{ij} is an indicator variable that is 1 if (i, j) is in first tour of the elitist solution and 0 otherwise. Here, $\tau_{init} > 0$ and $\tau_{solution} > 0$ are two parameters. Similarly, $\tau'_{ij} := \tau_{init} +$

$x'_{ij} \times \tau_{elite} + n'_{ij} \times \tau_{solution}$ if n'_{ij} solutions in the population P have the pair of cities (i, j) in their second tour and where x'_{ij} is an indicator variable that is 1 if (i, j) is in second tour of the elitist solution and 0 otherwise.

To find a new solution an ant starts at a randomly chosen city and constructs the first tour iteratively as follows. When the ant is at city i it uses the probabilistic rule in Formula 1. When the first tour π_1 is finished the ant constructs the second tour π_2. Again it starts at some randomly chosen city and proceeds analogously as for the first tour. The difference to the construction of the first tour is that pheromone matrix τ' and heuristic η' are used.

As heuristics η and η' one of the three heuristics $\eta^{(1)}$, $\eta^{(2)}$, and $\eta^{(3)}$ that are defined in the following are used in this paper. Heuristic $\eta^{(3)}$ can only be used for the construction of the second tour π_2 since it assumes that the first tour π_1 of an ant is already known. For a pair of cities (i, j) define

$$\eta_{ij}^{(1)} := 1/d_{ij} \tag{4}$$

$$\eta_{ij}^{(2)} := 1/(d_{ij} * w_{ij}) \tag{5}$$

$$\eta_{ij}^{(3)} := \begin{cases} 1/(d_{ij} * w_{ij}) & \text{if } (i,j) \in \pi_1 \\ \\ 1/d_{ij} & \text{else} \end{cases} \tag{6}$$

where π_1 denotes the first tour that has been constructed by the ant.

Note, that $\eta^{(1)}$ is the standard TSP heuristic which is typically used in ACO for the TSP ([4]). Heuristic $\eta^{(2)}$ prefers a city j that has a small distance to the city i where the ant is located and where the corresponding weight is small. Heuristic $\eta^{(3)}$ is only used for the construction of tour 2 since it assumes that the first tour is already known. Heuristic $\eta^{(3)}$ considers the weight w_{ij} of a pair of cities (i, j) only if (i, j) is already contained in the first tour of the ant.

5. EXPERIMENTAL SETUP

Pairs of cities (i, j) which have a weight $w_{ij} > 1$ are interesting when solving the 2-CT-TSP because ideally only one of the solutions includes such an edge (unless $w_{ij} * d_{ij}$ is relatively small compared the alternative pairs of cities during the construction of the second tour). The case of pairs of cities (i, j) with weight $w_{ij} < 1$ is mainly only interesting when there exist other pairs of cities that are similarly close and attractive for constructing the tours. In that case, ideally both tours should include the same pair of cities. However, in most (real world metric) TSP instances the shortest tour is unique. In that case the ants can basically simply try to find a shortest tour for both solutions and then profit automatically from edges with $w < 1$. Therefore, in the experiments the main focus is on problem instances where all edges have a weight of at least one.

For the experiments symmetric 2-CT-TSP instances have been used, i.e., $d_{ij} = d_{ji}$ and $w_{ij} = w_{ji}$ for all $i, j \in 1, \dots, n$, $i \neq j$. For the first experiment 2-CT-TSP instances were created where some pairs of cities have weight 5 and all other pairs of cities have weight 1. Which pairs of cities have weight 5 was chosen randomly with uniform probability. Test instances with different number of pairs of cities with weight 5 have been created. The second experiment is a variant of the first experiment. The difference is that each pair of cities has weight 0.2 or weight 5. For the third experiment 2-CT-TSP instances were created where the weight for each pair of cities was chosen randomly with uniform probability from the interval $[1, 5]$.

For defining the distances d_{ij} we used the TSP test instances berlin52 and eil101 from the standard benchmark library TSPLIB [16]. Note, that the total number of cities (pairs of cities) is 52 (respectively 1326) for berlin52 and 101 (respectively 5050) for eil101. The test parameter values that were used are (if not stated otherwise): 10 ants per iteration, population size $|P| = 5$, $\alpha = 1$, $\beta = 5$, $\tau_{init} = 1/(n-1)$, $\tau_{solution} = 0.3$, and $\tau_{elite} = 0.2$. Note, that parameter values $\alpha = 1$ and $\beta = 5$ are typically used in ACO algorithms for the TSP (e.g., [4]) and it has been shown that a small population size works well for PACO and the TSP ([7]). Each test run was done over 1000 iterations and has been repeated 50 times. The results given in Section 6 are averages over these 50 runs.

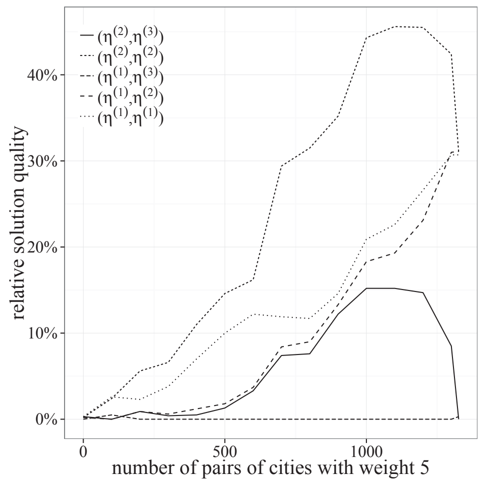
The influence of the different heuristics was tested. In particular, it was tested how important it is to use a heuristic for the construction of the second tour that takes the first tour into account. Therefore, heuristics $\eta^{(1)}$ and $\eta^{(2)}$ have been tested for the construction of the first tour of an ant and all three heuristics $\eta^{(1)}$, $\eta^{(2)}$, and $\eta^{(3)}$ have been tested for the construction of the second tour. We use notation $(\eta^{(i)}, \eta^{(j)})$ for a test run where heuristic $\eta^{(i)}$ is used for the construction of the first tour π_1 and heuristic $\eta^{(j)}$ is used for the construction of the second tour π_2, $i, j \in \{1, 2, 3\}$.

Table 1: Experiment 1: Average solution quality and standard deviation (in brackets) of PACO with different heuristics for berlin52 instance with different number of pairs of cities with weight 5

Heuristics	# pairs of cities with weight 5		
	100	700	1300
	mean	mean	mean
$(\eta^{(1)}, \eta^{(1)})$	16135.2	19696.7	24278.2
	(201.0)	(467.5)	(488.2)
$(\eta^{(1)}, \eta^{(2)})$	15733.5	19083.2	24349.1
	(122.8)	(252.8)	(459.3)
$(\eta^{(1)}, \eta^{(3)})$	15812.9	17608.5	18580.6
	(117.0)	(133.4)	(226.3)
$(\eta^{(2)}, \eta^{(2)})$	16113.9	22793.2	26456.8
	(152.9)	(285.3)	(622.8)
$(\eta^{(2)}, \eta^{(3)})$	15734.5	18907.7	20150.8
	(93.1)	(199.9)	(217.9)

6. RESULTS AND DISCUSSION

Table 1 shows the results of the PACO for the 2-CT-TSP instances of Experiment 1 when using the different heuristics and for different number of pairs of cities with weight 5. The relative quality of PACO with the different pairs of heuristics can also be seen in Figure 1. The pair of heuristics $(\eta^{(1)}, \eta^{(3)})$ is best for nearly all different number of pairs of cities with weight 5 that have been tested. An exception is the case with a small number of only 100 pairs of cities with weight 5. In this case the pairs of heuristics $(\eta^{(1)}, \eta^{(1)})$ and $(\eta^{(1)}, \eta^{(2)})$ perform best. Thus, most often it is best when the first tour π_1 is constructed with the standard TSP heuristic and



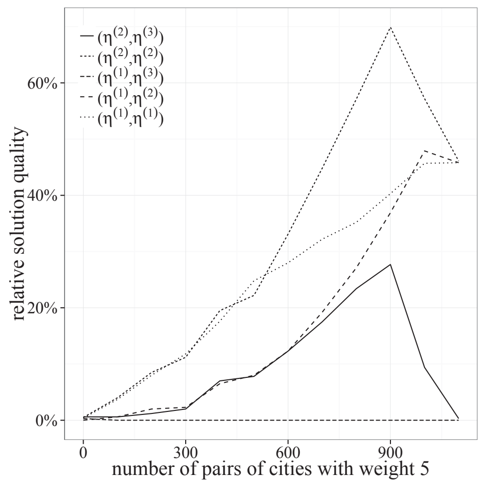


Figure 1: Experiment 1: relative average solution quality for different number of pairs of cities with weight 5, top: berlin52, bottom: eil101

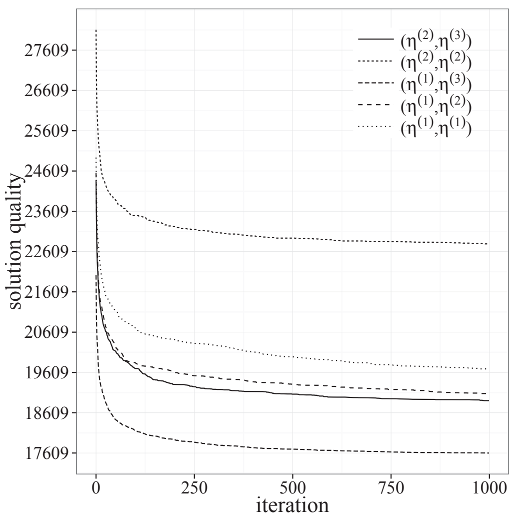


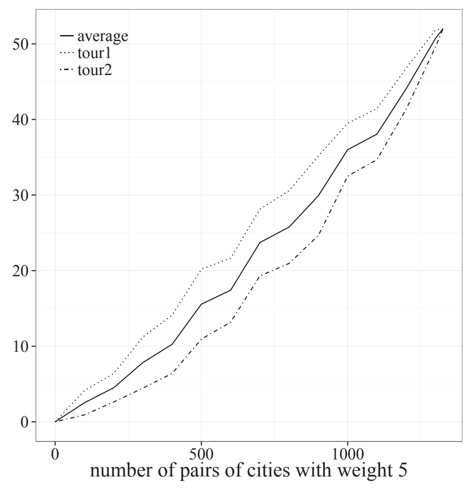
Figure 2: Experiment 1: convergence behaviour of PACO for berlin52 instance with 700 pairs of cities with weight 5

for the second tour π_2 the heuristic $\eta^{(3)}$ which considers the weights and considers also which pairs of cities are already included in tour π_1. Clearly, in the extreme cases when all edges have the same weight heuristics $\eta^{(1)}$ and $\eta^{(2)}$ become equal. In the experiments using $\eta^{(1)}$ instead of $\eta^{(2)}$ for the first tour π_1 is always better. The solution quality difference between using the two heuristics for π_1 is particularly large when only few pairs of cities with weight 1 exist. When using $\eta^{(1)}$ for tour π_1 it is nearly always better to use $\eta^{(3)}$ for π_2 instead of using $\eta^{(2)}$. The results for the eil101 instances are very similar to the corresponding results for berlin52 instances (see Figure 1).

Figure 2 gives an example that shows the convergence behaviour of PACO for the case of 700 pairs of cities with weight 5. It can be seen that the quality differences between the different PACO versions occur already after a few iterations and that the algorithms have mostly converged after 1000 iterations.

The number of pairs of cities of weight 5 that are included in the solutions of Experiment 1 are shown in Figure 3 for PACO with heuristics $(\eta^{(1)}, \eta^{(3)})$ and $(\eta^{(2)}, \eta^{(3)})$. Note, that in the extreme cases of instances with only pairs of cities with weight 1 and with only pairs of cities with weight 5 each tour includes zero, respectively 52, pairs of cities with weight 5. Since heuristic $\eta^{(1)}$ does not consider the weights, the number of pairs of cities with weight 5 in the first tour π_1 increases linearly with the total number of pairs of cities with weight 5 in the problem instance. The effect of using heuristic $\eta^{(3)}$ for the second tour π_2 is that the second tour contains less pairs of cities with weight 5. When heuristic $\eta^{(2)}$ is used for tour π_1 there are clearly fewer pairs of cities with weight 5 in tour π_1. This holds in particular for instances with a medium total number of pairs of cities with weight 5. In these cases it is relatively easy to avoid using pairs of cities with weight 5 for the second tour.

Table 2 shows the quality of the following variants of PACO: 1) $\beta = 2$, i.e., the influence of the heuristic is re-



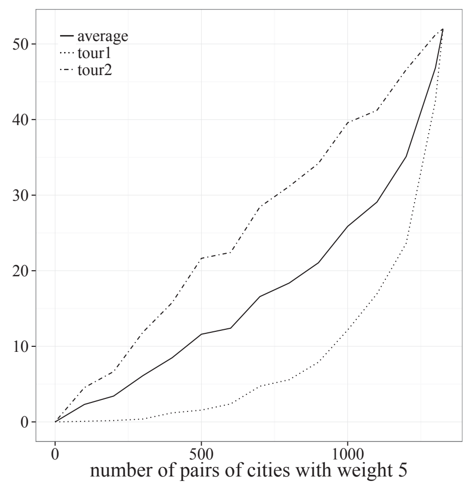


Figure 3: Experiment 1: average number of pairs of cities with length 5 in the first tour π_1 and the second tour π_2, for PACO with $(\eta^{(1)},\eta^{(3)})$ (top) and $(\eta^{(2)},\eta^{(3)})$ (bottom)

Table 2: Experiment 1: average solution quality of PACO for berlin52 instance with different number pairs of cities with weight 5, for each version of PACO also the best combination of heuristics is shown

Algorithm	# pairs of cities with weight 5		
version	100	700	1300
$\tau_{elite} = 0.2$	16620	18771	19843
$\beta = 2$	$(\eta^{(1)},\eta^{(3)})$	$(\eta^{(1)},\eta^{(3)})$	$(\eta^{(1)},\eta^{(3)})$
$\tau_{elite} = 0.2$	16054	18201	19408
$\beta_1 = 5$, $\beta_2 = 2$	$(\eta^{(2)},\eta^{(3)})$	$(\eta^{(1)},\eta^{(3)})$	$(\eta^{(1)},\eta^{(3)})$
$\tau_{elite1} = 0.2$	15824	17658	18644
$\tau_{elite2} = 0$	$(\eta^{(1)},\eta^{(2)})$	$(\eta^{(1)},\eta^{(3)})$	$(\eta^{(1)},\eta^{(3)})$

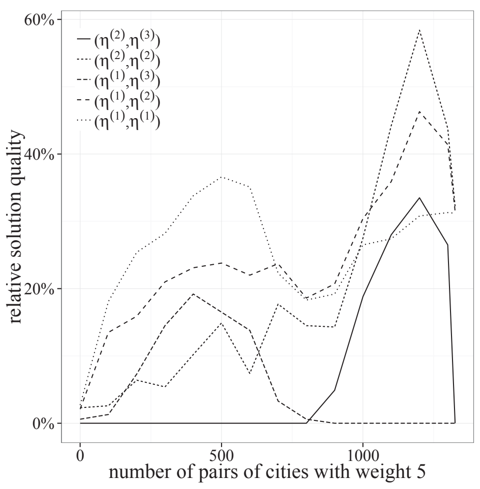


Figure 4: Experiment 2 (each pair of cities has weight $w = 0.2$ or $w = 5$): relative average solution quality for different number of pairs of cities with weight 5

duced, 2) $\beta = 5$ is used for the construction of tour π_1 and $\beta = 2$ is used for the construction of tour π_2, i.e., the influence of the heuristic is smaller for the second tour, and 3) the elitist solution is only used for the construction of the first tour π_1 but not for the second tour π_2 (i.e., a different value for parameter τ_{elite} was used for π_1 ($\tau_{elite1} = 0.2$) and π_2 ($\tau_{elite2} = 0$)). One hypothesis why PACO variants (2) and (3) might be good is that the construction of the second tour should depend on the first tour that has already been constructed by the ant. The reason is that some pairs of cities should be avoided for the second tour that are included already in the first tour. In these cases a strong influence of a heuristic or of the elitist solution might lead the ants in the wrong direction. However, the results show that this hypothesis does not hold. None of the three PACO variants (1) - (3) results in a better solution quality.

The results of Experiment 2 where each pair of cities has either weight 0.2 or 5 are shown in Figure 4. For small and medium numbers of pairs of cities with weight 5 the relative solution quality of the PACO versions shows differences to the results of Experiment 1. For less than 800 pairs of cities with weight 5 heuristics $(\eta^{(2)},\eta^{(3)})$ are better than $(\eta^{(1)},\eta^{(3)})$. Heuristics $(\eta^{(2)},\eta^{(2)})$ is the worst combination for each problem instance in Experiment 1. However, in Experiment 2 $(\eta^{(2)},\eta^{(2)})$ is the second best combination for small and medium numbers (200 - 600) of pairs of cities with weight 5. This shows, that the existence of a large enough number of pairs of cities with a small weight (<1) is useful for a heuristic that prefers pairs of cities with small weight for tour π_2 when they are already included in the first tour π_1.

For the evaluation of Experiment 3 all pairs of cities where classified with respect to their weight. Four classes are considered with weights in one of the following intervals: $[1.0, 2.0)$, $[2.0, 3.0)$, $[3.0, 4.0)$, and $[4.0, 5.0]$. Figure 5 shows the fraction of pairs of cities in the different classes that are included in the first tour π_1 and in the second tour π_2. It can be seen that the percentage of pairs of cities that are used by tour π_1 is nearly the same in all weight classes when heuristic $\eta^{(1)}$ is used for the construction of π_1. This is different when heuristic $\eta^{(2)}$ is used for the construction of π_1. In that case, pairs of cities with small distance are preferred and a higher percentage of pairs of cities in class $[1.0, 2.0)$ is contained in π_1 than for pairs of cities in class $[4.0, 5.0]$ (the percentage for the former class is approximately 15 times higher). This holds also for tour π_2 when heuristic $\eta^{(2)}$ is used and this result is relatively independent from the heuristic that is used for the first tour.

The results are very different when $\eta^{(3)}$ is used for the construction of the second tour π_2. If in that case $\eta^{(2)}$ is used for tour π_1, the percentage of pairs of cities in tour π_2 is much higher for class $[4.0, 5.0]$ than for class $[1.0, 2.0)$. The reason is that heuristic $\eta^{(3)}$ can differentiate if a pair of cities with large weight is included in π_1 or not. If not, such a pair of cities can be chosen for tour π_2 without the disadvantage of increased costs. When heuristic $\eta^{(3)}$ is used for the second tour π_2 together with heuristic $\eta^{(1)}$ for tour π_1 the percentage of pairs of cities in π_2 is smaller for class $[4.0, 5.0]$ than for class $[1.0, 2.0)$. Here, tour π_1 includes already many pairs of cities with small distance and large weight. These pairs of cities should not be chosen for tour π_2.

7. CONCLUSIONS

A new version of the Traveling Salesperson problem (TSP) which is called the Combined Tours TSP (CT-TSP) was defined in this paper. The aim of the CT-TSP is to find several (cyclic) tours that connect given cities such that each city is contained exactly once in each tour. The total costs of all tours should be minimum. The problem is that the costs of one tour may depend on the other tours. In this paper the particular version of CT-TSP where two tours are sought was studied (2-CT-TSP). For the studied cost function each pair of cities has a length and a weight. The quality of a solution of the corresponding 2-CT-TSP depends on the total lengths of both tours but also on the weights of the included pairs of cities. The weights are used to indicate if it is advantageous or disadvantageous when a pair of cities is contained in both tours. A Population based Ant Colony Optimization (PACO) algorithm was applied for solving the 2-CT-TSP. It was investigated how different heuristics that the ants use for the construction of the tours influence the solution quality. It was shown that it can be important that an ant uses a different heuristic for the construction of the first tour and the second tour. In particular, the heuristic for the construction of the second tour should use the knowledge about which pairs of cities are included in the first tour.

8. ACKNOWLEDGMENTS

This work has been supported by the European Social Fund (ESF) and the Free State of Saxony within the Nachwuchsforschergruppe "Schwarm-inspirierte Verfahren zur Optimierung, Selbstorganisation und Ressourceneffizienz".

9. REFERENCES

[1] T. Bektas: The multiple traveling salesman problem: an overview of formulations and solution procedures. OMEGA: The International Journal of Management Science 34(3): 209-219, 2006.

[2] Y.J. Costa Salas, R.A. Ledón, N.I. Coello Machado, A. Nowé: Multi-type ant colony system for solving the

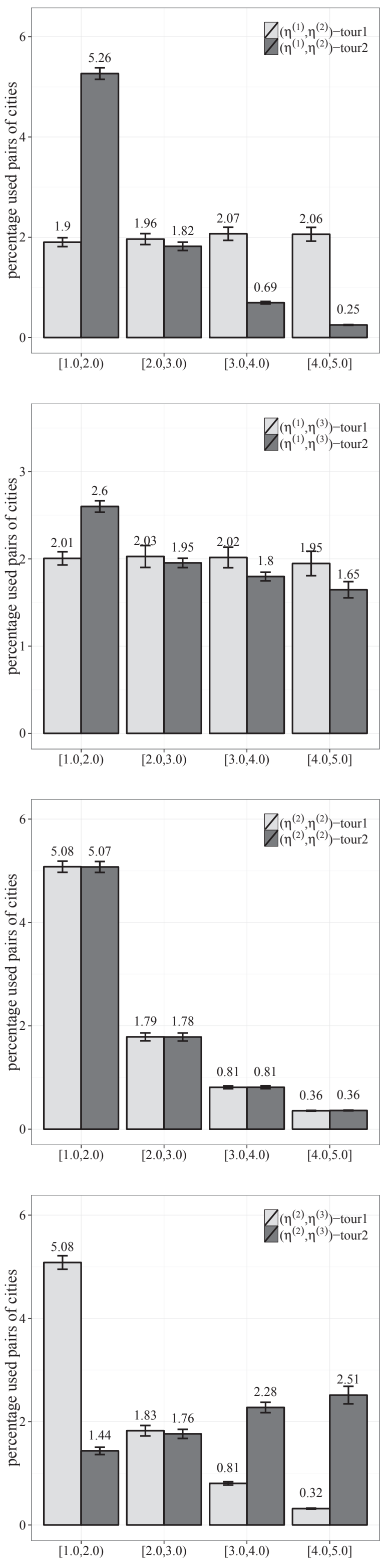


Figure 5: Experiment 3 (each pair of cities has a weight in $[1, 5]$): average percentage of pairs of cities used in the different weight classes by the first tour π_1 and the second tour π_2 (from top to bottom): PACO with $(\eta^{(1)}, \eta^{(2)})$, PACO with $(\eta^{(1)}, \eta^{(3)})$, PACO with $(\eta^{(2)}, \eta^{(2)})$, PACO with $(\eta^{(2)}, \eta^{(3)})$

multiple traveling salesman problem. Rev. Téc. Ing. Univ. Zulia., 35(3): 311 - 320, 2012.

[3] Y.J. Costa Salas, W.A. Sarache Castro: An alternative solution for the repair of electrical breakdowns after natural disasters based on ant colony optimization. DYNA, 81(186): 304-310, 2014.

[4] M. Dorigo, L.M. Gambardella: Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation, 1:53-66, 1997.

[5] M. Fischetti, J.J. Salazar-Gonzalez, P. Toth: A Branch and Cut Algorithm for the Symmetric Generalized Traveling Salesman Problem. Operations Research, 45(3):378-394, 1997.

[6] M. Fischetti, P. Toth: An Additive Approach for the Optimal Solution of the Prize-Collecting Traveling Salesman Problem. In Vehicle Routing: Methods and Studies. B.L. Golden, A.A. Assad (eds.). North-Holland, Amsterdam, 319-343, 1988.

[7] M. Guntsch, M. Middendorf: A Population Based Approach for ACO. Proc. 2nd European Workshop on Evolutionary Computation in Combinatorial Optimization (EvoCOP-2002), LNCS 2279, 72-81, 2002.

[8] G. Gutin, A.P. Punnen (Eds): The Traveling Salesman Problem and Its Variations. Springer, Combinatorial Optimization Series, Vol. 12, 2007.

[9] J. LaRusic: The bottleneck traveling salesman problem and some variations. *M.Sc. thesis*, Department of Mathematics, Simon Fraser University , 2010.

[10] M.A. Liang: Ant Colony Optimization for Bottleneck TSP. Computer Engineering, 9: 24-25, 2001.

[11] Y.-C. Lin, M. Clauß, M. Middendorf: Simple Probabilistic Population Based Optimization. IEEE Transactions on Evolutionary Computation, early access article, DOI 10.1109/TEVC.2015.2451701, 2015.

[12] R. Matai, S.P. Singh, M.L. Mittal:Traveling Salesman Problem: An Overview of Applications, Formulations, and Solution Approaches. In: D. Davendra (ed.), Traveling Salesman Problem, Theory and Applications, 2010.

[13] S. Oliveira, M. S. Hussin, T. Stützle, A. Roli, M. Dorigo: A detailed analysis of the population-based ant colony optimization algorithm for the TSP and the QAP. Proc. Genetic and Evolutionary Computation Conference (GECCO-2011), 13-14, 2011.

[14] A. Plebe, A.M. Anile: A Neural-Network-Based Approach to the Double Traveling Salesman Problem. Neural Computation, 14(2): 437-471, 2002.

[15] J.-Y. Potvin: A Review of Bio-Inspired Algorithms for Vehicle Routing. In: F.B. Pereira, J. Tavares (eds.), Bio-inspired Algorithms for the Vehicle Routing Problem, Studies in Computational Intelligence, Vol. 161, 1-34, 2009.

[16] G. Reinelt: TSPLIB. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95.

[17] M. Reihaneh, D. Karapetyan: An Efficient Hybrid Ant Colony System for the Generalized Traveling Salesman Problem. Algorithmic Operations Research, 7: 21-28, 2012.

[18] X. Shi, L. Wang, Y. Zhou, Y. Liang: An Ant Colony Optimization Method for Prize-collecting Traveling Salesman Problem with Time Windows. Proc. Fourth International Conference on Natural Computation (ICNC ’08), 480 - 484, 2008.

[19] T. Weise, R. Chiong, J- Lässig, K. Tang, S. Tsutsui, W. Chen, Z. Michalewicz, X. Yao: Benchmarking Optimization Algorithms: An Open Source Framework for the Traveling Salesman Problem. IEEE Computational Intelligence Magazin, 9(3):40-52, 2014.

[20] J. Yang, X. Shi, M. Marchese, Y. Liang: An ant colony optimization method for generalized TSP problem. Progress in Natural Science, 18:1417-1422, 2008.

[21] B. Yu , Z.Z. Yang: An ant colony optimization model: The period vehicle routing problem with time windows. Transportation Research Part E: Logistics and Transportation Review, 47(12):166-181, 2011.

[22] W. Zhou, P. Yao: An Improved Ant Colony Optimization Algorithm for Multiple Traveling Salesman Problem. Advances in Information Sciences and Service Sciences (AISS), 5(10):637-644, 2013.