universität
wien

Diplomarbeit

Titel der Diplomarbeit

„Heuristic Solution Approaches for the Covering Tour Problem“

Verfasser

Patrick Kubik

angestrebter akademischer Grad

Magister der Wirtschaftswissenschaften

Wien, im Dezember 2007

Studienkennzahl lt. Studienblatt:	A 157
Studienrichtung lt. Studienblatt:	Internationale Betriebswirtschaft
Betreuer:	Univ.-Doz. Dr. Karl Dörner

Für meine Eltern,
die mich immer unterstützt und
stets an mich geglaubt haben.

Weiters möchte ich mich bei Univ.-Doz. Dr. Karl Dörner für die hervorragende Betreuung bedanken.

Besonders möchte ich mich bei Kerstin für Ihr Verständnis und Ihre Unterstützung bedanken. Ohne Dich hätte ich es nicht geschafft.

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Wien, im Dezember 2007

Patrick Kubik

Table of Content

OVERVIEW OF FIGURES 6

OVERVIEW OF TABLES 7

LIST OF ABBREVIATIONS 9

LIST OF SETS, PARAMETERS AND VARIABLES 10

1 INTRODUCTION 13

1.1 LOGISTICS AND TRANSPORTATION 13
1.2 COMBINATORIAL OPTIMIZATION 14
1.3 THE COVERING TOUR PROBLEM 14
1.4 LAYOUT 15

2 THE COVERING TOUR PROBLEM 15

2.1 DESCRIPTION 15
2.2 BRIEF LITERATURE REVIEW 19
2.3 THE CTP MODEL 19
2.4 APPLICATIONS 21
2.5 COMPONENTS OF THE COVERING TOUR PROBLEM 22
2.5.1 THE TRAVELING SALESMAN PROBLEM 22
2.5.2 THE SET COVERING PROBLEM 23

3 SOLUTION APPROACHES 25

3.1 ANT COLONY OPTIMIZATION 25
3.1.1 REAL ANTS 25
3.1.2 ARTIFICIAL ANTS AND ACO ALGORITHMS FOR THE TSP 27
3.1.3 ANT SYSTEM 29
3.1.4 ANT COLONY SYSTEM 31
3.1.5 OTHER ACO ALGORITHMS 32
3.2 GENIUS ALGORITHM 33
3.2.1 GENI 33
3.2.1.1 TYPE I INSERTION 34
3.2.1.2 TYPE II INSERTION 36
3.2.1.3 GENI ALGORITHM 37
3.2.2 US 37
3.2.2.1 TYPE I UNSTRINGING 38
3.2.2.2 TYPE II UNSTRINGING 39
3.2.2.3 STRINGING 39
3.2.2.4 US ALGORITHM 40
3.3 PRIMAL1 SET COVERING HEURISTIC 40

3.4 SOLVING THE CTP 41
3.4.1 H-1-CTP HEURISTIC 41
3.4.2 ACS FOR THE CTP 42
3.4.2.1 GACS 43
3.4.2.2 SCACS 45
3.4.2.3 CTACS 48

4 COMPUTATIONAL RESULTS 49

4.1 TESTS ON THE TSP PART OF THE PROBLEM 49
4.1.1 NEIGHBORHOOD SIZE FOR GENI AND GENIUS 49
4.1.2 PARAMETER ANALYSIS FOR GACS 50
4.1.3 GENI VARIANTS COMPARISON 52
4.2 TESTS ON THE SCP PART OF THE PROBLEM 54
4.2.1 PARAMETER ANALYSIS FOR SCACS 54
4.2.2 HEURISTIC INFORMATION IN SCACS 55
4.2.3 PRIMAL1 VS. SCACS 56
4.3 TESTS ON THE CTP 57

5 CONCLUSION 62

APPENDIX A 64

APPENDIX B 72

APPENDIX C 80

C.1 GERMAN ABSTRACT 80
C.2 ENGLISH ABSTRACT 82
C.3 CURRICULUM VITAE 83

6 REFERENCES 85

Overview of figures

Figure 1: A possible solution to the CTP 18
Figure 2: CTP with $c = 0$ reduces to TSP 18
Figure 3: Double bridge experiment: (a) equal length and (b) double length.. 26
Figure 4: The effect of stigmergy during food foraging 27
Figure 5: ACO pseudo-code 28
Figure 6: Decision making in ACO 30
Figure 7: Type I insertion procedure 35
Figure 8: Type II insertion procedure 36
Figure 9: Type I unstringing of vertex v_i from the tour 38
Figure 10: Type II unstringing of vertex v_i from the tour 39
Figure 11: GACS algorithm 45
Figure 12: SCACS algorithm 48

Overview of Tables

Table 1: Importance of neighborhood size for GENI.................................. 49
Table 2: Importance of neighborhood size for GENIUS............................... 50
Table 3: Influence of parameter ρ on GACS solution quality....................... 51
Table 4: Influence of parameter α on GACS solution quality....................... 51
Table 5: Influence of parameter γ on GACS solution quality 51
Table 6: Influence of parameter q_0 on GACS solution quality 52
Table 7: GACS results with best parameters 52
Table 8: Comparison of GACS, mGENI and GENIUS 53
Table 9: Comparison of GACS and mGENI 53
Table 10: Influence of parameter β on SCACS solution quality.................... 54
Table 11: Influence of parameter q_0 on SCACS solution quality 55
Table 12: Influence of parameter ρ on SCACS solution quality.................... 55
Table 13: Influence of parameter α on SCACS solution quality.................... 55
Table 14: SCACS tests on heuristic information 56
Table 15: PRIMAL1 vs. SCACS 56
Table 16: Results for CTACS and H-1-CTP with $V \subset W = \{1000\}$ 58
Table 17: Results for CTACS and H-1-CTP with $V \subset W = \{2000\}$ 59
Table 18: Results for CTACS and H-1-CTP with $V \subset W = \{1000\}$ and $c = 20$ 60
Table 19: Results for CTACS and H-1-CTP with $V \subset W = \{2000\}$ and $c = 20$ 61
Table 20: Tests on q_0 for KubLE25 (GACS).. 64
Table 21: Tests on γ for KubLE25 (GACS)... 64
Table 22: Tests on α for KubLE25 (GACS) 65
Table 23: Tests on ρ for KubLE25 (GACS) 65
Table 24: Tests on q_0 for Berlin52 (GACS) 66
Table 25: Tests on γ for Berlin52 (GACS) 66
Table 26: Tests on α for Berlin52 (GACS).. 66
Table 27: Tests on ρ for Berlin52 (GACS).. 67
Table 28: Tests on q_0 for st70 (GACS).. 68
Table 29: Tests on γ for st70 (GACS)... 68
Table 30: Tests on α for st70 (GACS) 68

Table 31: Tests on ρ for st70 (GACS) 69
Table 32: Tests on q_0 for pr107 (GACS) 70
Table 33: Tests on γ for pr107 (GACS) 70
Table 34: Tests on α for pr107 (GACS) 70
Table 35: Tests on ρ for pr107 (GACS) 71
Table 36: Tests on α for SCP41 (SCACS) 72
Table 37: Tests on β for SCP41 (SCACS) 72
Table 38: Tests on ρ for SCP41 (SCACS) 73
Table 39: Tests on q_0 for SCP41 (SCACS) 73
Table 40: Tests on α for SCP43 (SCACS) 74
Table 41: Tests on β for SCP43 (SCACS) 74
Table 42: Tests on ρ for SCP43 (SCACS) 74
Table 43: Tests on q_0 for SCP43 (SCACS) 75
Table 44: Tests on α for SCP44 (SCACS) 76
Table 45: Tests on β for SCP44 (SCACS) 76
Table 46: Tests on ρ for SCP44 (SCACS) 76
Table 47: Tests on q_0 for SCP44 (SCACS) 77
Table 48: Tests on α for SCP57 (SCACS) 78
Table 48: Tests on β for SCP57 (SCACS) 78
Table 50: Tests on ρ for SCP57 (SCACS) 78
Table 51: Tests on q_0 for SCP57 (SCACS) 79

List of abbreviations

ACO	Ant Colony Optimization
ACS	Ant Colony System
AS	Ant System
CTP	Covering Tour Problem
CTACS	Covering Tour Ant Colony System
GACS	GENI Ant Colony System
GENI	General Insertion
M&S	Marchiori and Steenbeek
mGENI	multi-start GENI
SCACS	Set Covering Ant Colony System
SCP	Set Covering Problem
TSP	Traveling Salesman Problem
US	Unstring and String

List of sets, parameters and variables

a_{ij}	$\begin{cases} 1 & \textit{if column } j \text{ covers } \textit{row } i, \\ 0 & \textit{otherwise.} \end{cases}$
$A = (a_{ij})$	$m \times n$ 0-1 matrix
b_k	Number of rows that can be covered by vertex k
c	Covering distance parameter
c_{ij}	Cost of an edge (i, j)
c'_{ij}	Modified cost of an edge (i, j)
c_j	Cost of a column j
c_k	Cost of inserting vertex k
$C = (c_{ij})$	Distance matrix
C^{bs}	Length of the best-so-far tour T^{bs} (ACS)
C^{GENI}	Cost of a GENI tour
C^{nn}	Cost of a nearest-neighbor tour
$C^{SCP} = (c_j)$	n-dimensional cost vector
$draw(J)$	Random variable determining next insertion (GACS)
E	Set of edges on a graph
$f(c_k, b_k)$	Function determining the covering criterion
G	Graph
H, H', H^*	Final, intermediate and best-so-far tour (GENIUS)
I	Number of iterations (GACS,SCACS)
$i \in M = \{1,.....,n\}$	Set of n rows
J	Random variable determining the next move (ACS)
$j \in N = \{1,.....,m\}$	Set of m columns
m_a	Number of ants
M^k	Memory of ant k
N_i^k	Neighborhood of ant k at vertex i
$N_p(v)$	Neighborhood of vertex v with p-closest neighbors
O	Effect of the problem size on an algorithm's usage

	of computational resources
p	Sets the size of $N_p(v)$ (GENIUS)
p_{ij}^k	Probability of ant k choosing the edge from vertex i to j
p_j^k	Probability of ant k choosing column j
S	Subset of an unfinished solution
S^{bs}	Best-so-far solution (ACS SCP)
S^{GENI}	Number of ranks (GACS)
S_k	Partial solution obtained by ant k (ACS SCP)
S_ℓ	Covering set of a vertex $v_\ell \in W$
T^{bs}	Best-so-far tour in ACS
T^{GENI}	Random variable determining next move (GACS)
T^k	Tour generated by ant k
q	Random variable uniformly distributed in $[0,1]$
q_0	Sets the bar for the best choice (ACO)
$v_k \in T$	Set of vertices that must be visited
$v_k \in V$	Set of vertices on a graph
$v_\ell \in W$	Set of vertices that must be covered
v_0	Starting point on a tour
x_{ij}	$\begin{cases} 1 & \text{if edge } (i,j) \text{ is used}, \\ 0 & \text{otherwise.} \end{cases}$
y_k	$\begin{cases} 1 & \text{if vertex } k \text{ is on the tour}, \\ 0 & \text{otherwise.} \end{cases}$
$z,\ z',\ z^*$	Cost of the final, intermediate and best-so-far tour (GENIUS)
z^{bs}	Best-so-far cost (ACS SCP)
z_{GR}	Cost of a greedy solution
α	Controls pheromone deposit in global update (ACS)
β	Controls heuristic influence (ACO)

$\delta_{\ell k}$	$\begin{cases} 1 & if\ vertex\ \ell\ is\ \mathrm{cov}ered\ by\ k, \\ 0 & otherwise. \end{cases}$
γ	Modifies the relative influence of pheromone values η_{ij}
	Heuristic value of edge (i, j)
η_j	Heuristic value of column j
ρ	Controls pheromone deposit in local update (ACS)
τ_0	Initial pheromone value
τ_{ij}	Pheromone trail on edge (i, j)
$\Delta\tau_{ij}^{bs}$	Pheromone deposited in global update (ACS)
$\Delta\tau_{ij}^{k}$	Pheromone deposited in global update (AS)
τ_{ij}^{R}	Relative amount of pheromone on edge (i, j)
τ_j	Pheromone trail of column j
$\Delta\tau_j^{bs}$	Pheromone deposited in global update (ACS SCP)
ζ	Controls pheromone influence (AS)

1 Introduction

This section outlines the importance of logistics and combinatorial optimization in a profit-orientated society and very briefly discusses the relevant literature. Finally, it gives a short outlook on the remaining contents of the diploma thesis.

1.1 Logistics and transportation

The globalization of world economy, increasing dynamics of global markets and of customer requirements as well as the rapid development of Asian economies have awarded logistics and its associated costs a completely new economical importance. Companies have to monitor these costs with increasing precaution because "*distribution costs account for almost half of the total logistics costs and in some industries, such as food and drink business, distribution costs can account for up to 70% of the value added costs of goods*" [5]. Only those companies taking advantage of global cost synergies while improving their customer service and therefore increasing their logistical capacity are able to remain competitive.

One example, to what extent the importance and value of business logistics - especially in terms of transportation - has grown during the last few years, is shown by the fact that Maersk Line, the largest container shipping company worldwide, operates up to 11.5 million containers with a total value of approximately 250 million US-Dollars per year. The economic wealth realized through these transports exceeds the worldwide budget for foreign aid about five times. During the last year, 100 million containers were shipped from seaports around the world and forecasts for the next ten years lead to the assumption that this number will at least double. Three billion ton-kilometers by train, road and air transport EU wide in 2005 document the increasing importance of cargo transportation [22].

On the other side, the role of logistics - particularly of transportation - in the regional sector is becoming increasingly important. The key role of public transportation in a nation's economy used to result in governmental ownership of public transportation companies, e.g. Deutsche Bahn in Germany. The same can be said about postal delivery services. However, recent developments, especially in Central Europe, have shown a trend for these companies to go public, reducing the social component of their services by closing down non-profitable branches and service lines mainly in rural regions. Still, the companies' interest lies in maintaining a certain service level which leads to a trade-off between customer satisfaction and profit.

Another interesting field of logistic and transportation is the effective service distribution of non-profit health care organizations in industrial countries as well as in third world countries and disaster areas.

1.2 Combinatorial optimization

The majority of problems in logistics and transportation are difficult and can be modeled as combinatorial problems. They usually deal with maximizing or minimizing an objective under certain constraints. One of the most important factors in fields like logistics, operations research or applied mathematics is decision making. Algorithmic approaches and computational complexity theory help to improve and optimize these decisions. When dealing with NP-hard[1] problems, combinatorial optimization offers three possible solution techniques to solve the problem: enumerative methods that lead to guaranteed optimal solutions but require a lot of resources, approximation algorithms running in polynomial time and heuristics with some a priori uncertainty concerning solution quality and processing time [1]. All these methods examine the normally large solution spaces of a combinatorial optimization problem and reduce it by effective exploration.

1.3 The Covering Tour Problem

The Covering Tour Problem (CTP) is one of the combinatorial optimization problems that can be applied to these real world problems. There is a given set of vertices (e.g., cities) that have to be visited. Further vertices exist that can be visited. A third set of vertices may not be visited but must be covered by a city that is visited. Covering means that a vertex that is visited is within a predefined distance of a city to be covered. The objective is to find the shortest

[1] Problems that may not be solved to optimality in polynomial time.

tour so that all covering and visiting requirements are met. Literature and solution methods on this problem are scarce. I filled a part of this gap by applying heuristic and meta-heuristic approaches to the problem. I performed extensive tests to find optimal parameter settings and to determine the best solution approach.

1.4 Layout

The diploma thesis is organized as follows. Section 2 describes in detail the CTP along with its model, applications and components. Section 3 is dedicated to the chosen solution approaches for the CTP such as Ant Colony Optimization (ACO) and a combination of a general insertion and post-optimization algorithm (GENIUS) and a set covering algorithm (PRIMAL1). Test problems and their computational results are discussed in section 4 while section 5 summarizes the findings. Appendix A and B illustrate test results.

2 The Covering Tour Problem

Section 2 presents the CTP with a small example and states the model. Then some real world applications of the CTP are described. In addition, the relationship of the CTP with the Traveling Salesman Problem (TSP) and the Set Covering Problem (SCP) is emphasized.

2.1 Description

First an example[2]:

The national postal service has decided to cut costs by reducing the number of local post offices in the countryside. Only those post offices in more populated towns should remain and operate as distribution centers for rural villages without post offices. In order to sustain the present service level at lower costs, the logistic department of the national post company has decided to assign

[2] CTP elements are written in bold font.

each town with a number of small villages that used to have a post office and a number of even smaller villages that used to be serviced by those offices. A town office should use ***a vehicle*** *to maintain postal service for the appointed region. This vehicle should be loaded with post destined for the region as well as with postal goods (stamps, envelopes, etc.) needed by the population of the rural villages* ***at the town's post office*** *every morning.* ***Then the vehicle must visit a certain number of villages (e.g.: villages with population above a certain limit or where frequent need of postal service is known from the past). It can visit some additional villages to ensure that the rural population is able to reach the vehicle without too much effort. However every village is only visited once each day.*** *At each stop the population can collect their post and purchase goods needed. They also hand in their mail.* ***After the vehicle has visited all mandatory destinations it should return to the post office.***

A number of transportation problems can be found in this example. First, the decision to close some and continue other offices is a location problem. Loading the vehicle is a Bin Packing problem and delivering and collecting post is a Pick-up and Delivery problem (in our example assembly with time windows). However, in order to focus on the CTP, we neglect the vehicle's capacity constraints and focus on the objective of covering all obliged targets at minimum cost.

The CTP is defined on a complete undirected graph $G = (V \cup W, E)$ with a set of vertices $V \cup W$ where $V = \{v_0, ..., v_n\}$ is a set of vertices that can be visited, W defines the set of vertices that have to be covered by the tour and $E = \{(v_i, v_j): v_i, v_j \in V \cup W, i < j\}$ is the set of edges. "Covered by the tour" means that any vertex $v_\ell \in W$ has to lie within a predefined distance of a vertex on the tour. The set V includes the subset T. The subset $T \subset V$ determines the set of vertices whose visit is obligatory. Vertex v_0 represents the depot and belongs to the set $T \subset V (v_0 \in T)$. The distance or travel time matrix $C = (c_{ij})$ indicates the edge length between all vertices $(V \cup W)$ in the edge set $E = \{(v_i, v_j): v_i, v_j \in V \cup W, i < j\}$ while satisfying the triangle inequality. The triangle inequality theorem states that for any triangle, the length of a given side must be shorter than the sum of the other two but greater than the difference between these two. This theorem holds for all Euclidean spaces.

The parameter c specifies the allowed maximum covering distance or in other words the maximum length of an edge between an unvisited vertex of set W and the nearest visited vertex of set V.

The solution to the CTP is a minimum length tour or Hamiltonian cycle [14]. The tour starts and ends at the depot $(v_0 \in T)$. The tour is defined by a certain subset (often referred to as S) of V so that all vertices of the subset T (all vertices that have to be visited) are visited by the tour and each vertex of set W (all vertices that have to be covered) lies within a predetermined distance c of a vertex belonging to the tour. The assumption that the depot $(v_0 \in T)$ does not cover all vertices of set W must also hold. Consequently, if the covering distance c equals zero, the CTP reduces to a TSP because then naturally every vertex from the set W becomes a member of the vertex subset T and has to be visited directly. Determining the Hamiltonian path or minimum length tour is classified as an NP-hard problem and a feasible solution can not always be found.

Figure 1 shows a possible solution to the CTP. Note that the coverage circles around each vertex of set V all have the same radius c which is the predetermined covering distance.

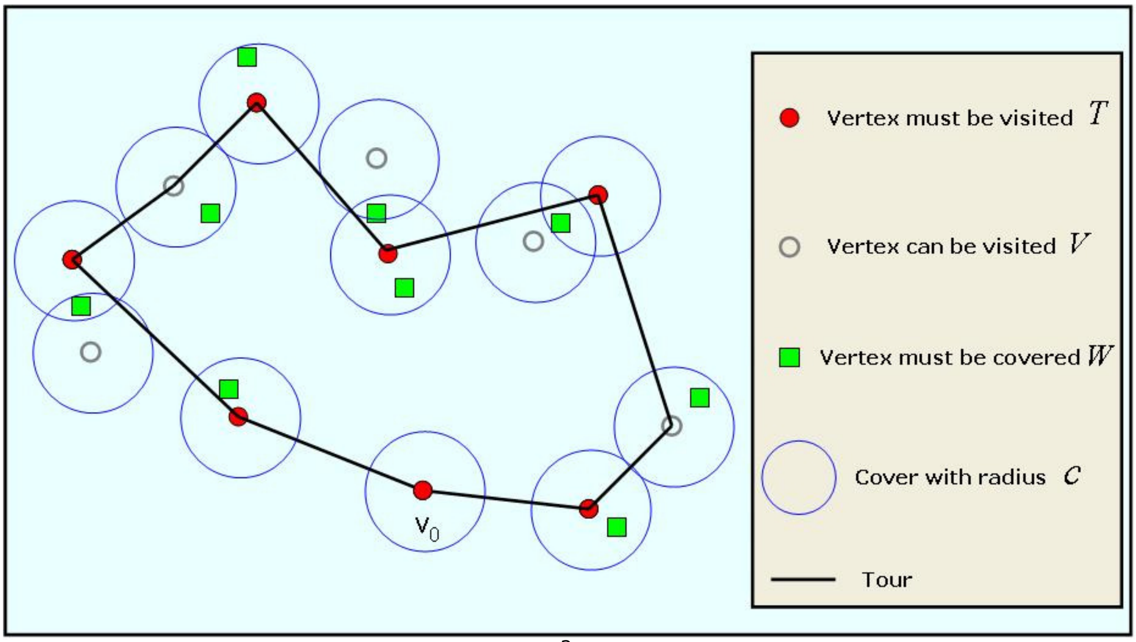

Figure 1: A possible solution to the CTP[3]

If the predetermined covering distance c of every vertex equals zero, every vertex of set W corresponds to a vertex of set V. The CTP then reduces to a TSP (Figure 2).

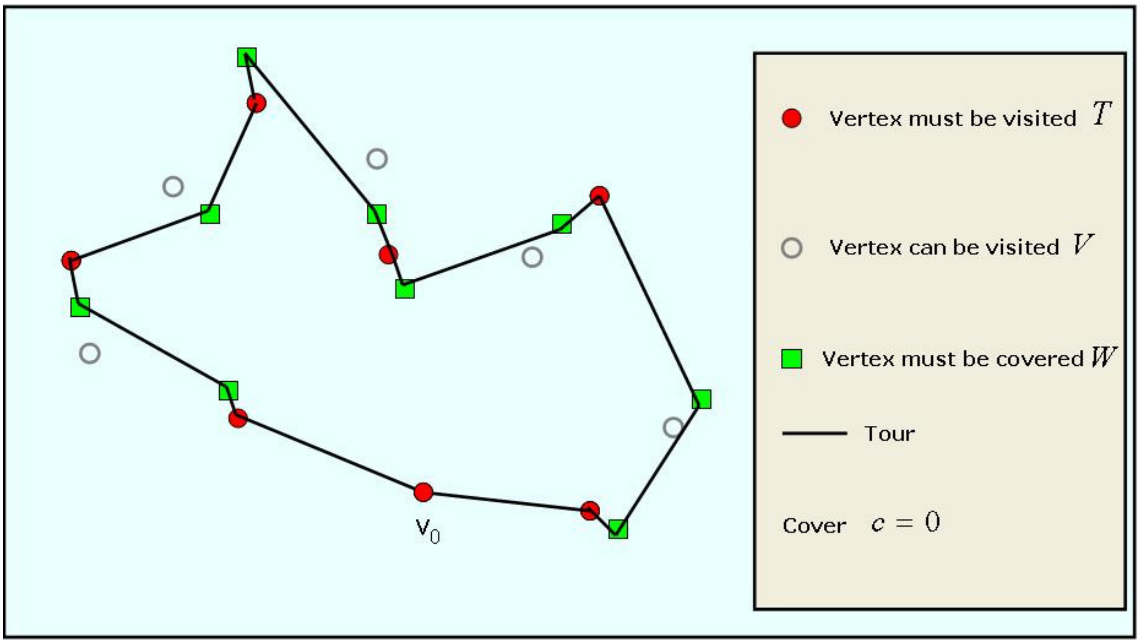

Figure 2: CTP with $c = 0$ reduces to TSP

[3] Note that this is only a graphical example and not necessarily an optimal solution.

2.2 Brief literature review

Not a lot of literature concerning the CTP exists today. Gendreau et al. [14] give a good discussion of papers related to the problem before 1997 including the first actual formulation by Current and Schilling [6] under the name Covering Salesman Problem. Gendreau et al. [14] are also the first to formulate a model and an exact algorithm in order to solve the problem. Hachida et al. [16] introduce the multi-vehicle Covering Tour Problem (m-CTP) and apply the heuristic used in [14]. They also present modified versions of the sweep and savings algorithms. Jozefowiez et al. [18] tackle the bi-objective CTP by combining a multi-objective evolutionary algorithm with a branch-and-cut algorithm.

2.3 The CTP model

The CTP can be formulated as a linear integer program. To start with, some binary variables have to be defined.

For $v_k \in V$ the binary variable y_k equals 1, if a vertex v_k of the vertex set V is visited. Otherwise the variable y_k equals 0. Of course, if $v_k \in T$, then y_k must always equal 1.

For $v_i, v_j \in V$ and $i < j$, the binary variable x_{ij} equals 1 for every edge (v_i, v_j) visited by the tour. Otherwise the variable x_{ij} equals 0.

The binary coefficient $\delta_{\ell k}$ equals 1 if and only if $v_\ell \in W$ can be covered by $v_k \in V$. This means that the distance $c_{\ell k}$ between $v_\ell \in W$ (the vertex that has to be covered) and $v_k \in V$ (the vertex that covers $v_\ell \in W$) is smaller than the predetermined covering distance c. Otherwise the coefficient $\delta_{\ell k}$ equals 0.

The subset $S_\ell = \{v_k \in V | \delta_{\ell k} = 1\}$ detects all vertices of the set V capable of covering a vertex $v_\ell \in W$ within the predetermined covering distance for every $v_\ell \in W$. The condition $|S_\ell| \geq 2$ for all $v_\ell \in W$ and the infeasibility of the degenerate tour (v_0) are also necessary assumptions.

The CTP can be stated as:

Minimize $$\sum_{i=0}^{n-1} \sum_{j=i+1}^{n} c_{ij} x_{ij} \tag{2.1}$$

Subject to

$$\sum_{v_k \in S_\ell} y_k \geq 1 \qquad \forall v_\ell \in W, \tag{2.2}$$

$$\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2 y_k \qquad \forall v_k \in V, \tag{2.3}$$

$$\sum_{\substack{v_i \in S, v_j \in V \setminus S \\ or\ v_j \in S, v_i \in V \setminus S}} x_{ij} \geq 2 y_k \qquad (S \subset V, 2 \leq |S| \leq n-2, T \setminus S \neq \emptyset, v_k \in S), \tag{2.4}$$

$$x_{ij} \in \{0,1\} \qquad \forall 1 \leq i < j \leq n, \tag{2.5}$$

$$y_k \in \{0,1\} \qquad \forall v_k \in V \setminus T, \tag{2.6}$$

$$y_k = 1 \qquad \forall v_k \in T. \tag{2.7}$$

The objective function (2.1) minimizes the total distance traveled to reach all $v_k \in T$ and to cover all $v_\ell \in W$.

The first constraint (2.2) demands coverage for each vertex $v_\ell \in W$ by the tour. Constraint (2.3) ensures that each vertex $v_k \in V$ is visited only once and that it is entered and left again while constraint (2.4) eliminates sub-tours by making sure that, for every subset S of V there are at least two edges between any subset S and the set of vertices $V \setminus S$ (set V without vertices of subset S) such that subset $T \setminus S \neq \emptyset$ and subset S contains a vertex $v_t \in S$.

Constraints (2.5), (2.6) and (2.7) ensure that variables x_{ij} and y_k are binary, the model is integer and y_k always equals 1 if $v_k \in T$.

2.4 Applications

One application of the CTP occurs in the health care sector concerning the deployment of a mobile medical facility in developing countries [17]. Traveling health care teams can only access a limited number of villages. This may be due to infrastructural restrictions like non-existing roads, resource restrictions like the tank size of the vehicle or governmental rule setting. Of course, the cost factor is always a barrier for non-profit organizations too. However, the routes of the health care teams have to be chosen in such a way that every person in need of medical service has the possibility to reach one of the villages integrated on the team's tour by foot. Solving the CTP enables the construction of efficient routes for these health care teams, reducing costs by minimizing traveling distances and therefore petrol consumption, minimizing traveling time and therefore increasing the time for medical service as well as maximizing the patient coverage.

The design of bi-level transportation networks is another common application where the tour chosen to reach all $v_k \in T$ and to cover all $v_\ell \in W$ represents the route of any primary vehicle and all $v_\ell \in W$ are within covering distance [14]. One example would be to locate a number of regional distribution centers from a set of candidates for an express delivery service (such as DHL or UPS) in order to minimize the cost of distributing the objects to every region from a central distribution centre and vice versa collecting objects from the regional centers. The covering tour chosen represents the tour of a primary vehicle (e.g.: large truck) with the central distribution centre v_0 as depot and regional centers as vertices (all $v_k \in T$ and possibly some $v_k \in V \setminus T$) on the tour). On the secondary level the CTP does not consider how to distribute efficiently but ensures that the end customers of each regional centre lie within a reasonable covering distance (in the sense that a small delivery truck can reach all of them in one day and at minimum cost). The problem on the secondary level could then be solved as a separate TSP or vehicle routing problem.
Other real world applications are the postal service example in 2.1, the routing of aircrafts for overnight delivery systems where only cities with airports are visited and other cities within a maximal covering distance are supplied by ground transportation [6] or the design of computer networks where servers

are the vertices and the tour is a ring network to increase the reliability. The covering part should minimize the cost of connecting personal computers with their nearest server [7].

2.5 Components of the Covering Tour Problem

In order to solve the CTP, Gendreau, Laporte and Semet [14] classified it as a combination of the TSP and SCP. I chose to adopt this approach but used additional algorithms for these two problems which I combined in order to solve the CTP. In the next two sections a brief overview of these two problems follows.

2.5.1 The Traveling Salesman Problem

The TSP deals with the following problem:

A salesman wants to visit a number of clients at different locations, starting from his hometown. He wants to visit every client once and then return to his starting point. What sequence should he choose in order to minimize his total traveling distance?

The TSP is the most common form of all combinatorial optimization problems and qualifies as an NP-hard problem. The importance of the TSP is not due to the fact that millions of salesmen need a solution to their business problem but that a TSP can be applied to a great number of variations of combinatorial optimization and "every day" problems. There are symmetric and asymmetric formulations of the TSP but as the focus of this chapter lies on showing the components of the CTP which is discussed only for symmetric problems, the symmetric TSP will be introduced. An example for an asymmetric TSP would be route optimization with one-way streets.

The objective is to find a sequence of vertices $V = \{v_0, ..., v_n\}$ on a weighted graph $G = (V, E)$ that results in the shortest tour or Hamiltonian cycle by visiting each vertex of V exactly once and then returning to the starting point. $E = \{(v_i, v_j): v_i, v_j \in V, i < j\}$ determines the edge set. Edges are used to connect the vertices and their weights are given by a distance or travel time matrix $C = (c_{ij})$.

As already mentioned the TSP is one of the most important problems in combinatorial optimization. Therefore, in the past three decades numerous papers on various solution methods to the TSP have been published. Exact, heuristic and meta-heuristic approaches have been developed. Some will be introduced later on when solution approaches for the CTP are described.

The TSP can be stated as [8]:

Minimize $$\sum_{i<j} c_{ij} x_{ij} \tag{2.8}$$

Subject to

$$\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2 \qquad (k \in V), \tag{2.9}$$

$$\sum_{v_i, v_j \in S} x_{ij} \le |S| - 1 \qquad S \subset V, 3 \le |S| \le n-3, \tag{2.10}$$

$$x_{ij} \in \{0,1\} \qquad (i \in V, j \in V). \tag{2.11}$$

The objective function (2.8) minimizes the tour length under the condition that vertices $V = \{v_0, \ldots., v_n\}$ are only visited once and that each vertex is entered and left (2.9). Equation (2.10) eliminates subsets (sub-tours). The binary variable x_{ij} equals 1 if edge (v_i, v_j) belongs to the tour. Otherwise the variable x_{ij} equals 0 (2.11).

2.5.2 The Set Covering Problem

Just like the TSP, the SCP qualifies as an NP-hard combinatorial optimization problem with applications in facility location and vehicle routing. The objective is to cover a number of rows with a set of columns at minimum cost.
It can be defined by a $m \times n$ 0-1 matrix $A = (a_{ij})$. To solve the SCP, a subset of columns $j \in N = \{1, \ldots\ldots, m\}$ that covers all the rows $i \in M = \{1, \ldots\ldots, n\}$ in A with

minimal total costs has to be derived. Covering costs are given by an n-dimensional cost vector $C^{SCP} = (c_j)$, where c_j is the cost of selecting column j in matrix A. A row $i \in M = \{1,.....,n\}$ is covered by a column $j \in N = \{1,.....,m\}$ if a_{ij} equals 1.

A good example is the problem of assigning factories producing goods to customers in order to satisfy their demands with minimal costs. Another example is airline crew scheduling.

The SCP can be stated as:

Minimize $\sum_{j \in N} c_j x_j$, (2.12)

Subject to

$$\sum_{j \in N} a_{ij} x_j \geq 1 \qquad \forall i \in M, \qquad (2.13)$$

$$x_j \in \{0,1\} \qquad \forall j \in N. \qquad (2.14)$$

The objective function (2.12) minimizes the total cost of covering all the rows in N. (2.13) ensures that every row is covered by at least one column and (2.14) ensures integrality.

Again, solution methods will be introduced later when tackling the CTP but to highlight the relation between SCP and CTP, some adjustments have to be made. The objective function (2.12) changes from $\sum_{j \in N} c_j x_j$ to $\sum_{v_k \in V} c_k y_k$. The set $v_k \in V = \{v_0,....,v_n\}$ with subset $T \subseteq V$ from the CTP replaces the set of columns $j \in N = \{1,....,m\}$ in the SCP. The binary variable x_j becomes y_k. In the SCP the binary variable x_j equals 1 if a column is chosen to cover one or more rows. In the CTP y_k equals 1 if a vertex $v_k \in V$ is included into the tour. The variable a_{ij} is the cost of choosing column x_j in the SCP. This is the cheapest cost c_k of inserting vertex $v_k \in V$ in the tour in the CTP. Constraint (2.13) substitutes $\sum_{j \in N} a_{ij} x_j \geq 1$ with $\sum_{v_k \in S_\ell} y_k \geq 1$ for the set $v_\ell \in W$ which stands for the set of rows $i \in M = \{1,....,n\}$ in the SCP. The parameter S_ℓ equals the covering set $S_\ell = \{v_k \in V | \delta_{\ell k} = 1\}$ for every $v_\ell \in W$ of the CTP.

In addition, the value of the binary variable y_k for all vertices associated with subset $T \subseteq V$ (determining the set of obligatory vertices) equals 1 which means that some columns are always chosen.

3 Solution approaches

This section introduces the concept of ACO and describes in detail the idea of the Ant Colony System (ACS) metaheuristic. Furthermore, the GENIUS algorithm for solving TSPs is specified. After that follows a presentation of the set covering heuristic PRIMAL1. Finally, the pieces are put together in order to solve the CTP and the H-1-CTP heuristic, a combination of GENIUS and PRIMAL1, as well as CTACS, a combination of GENI Ant Colony System (GACS) [19] and an ACS for the SCP (SCACS) [20] are introduced.

3.1 Ant colony optimization[4]

ACO is a nature inspired metaheurisitic for solving computational and combinatorial problems that deal with finding the shortest path on graphs. The solution strategy is based on the swarm-like behavior of real ants foraging for food.

3.1.1 Real ants

The lack of vision that characterizes the majority of ant species forces the individual insect to communicate with its colony by producing chemicals called pheromones. Ants can sense these pheromones and use them as a form of indirect communication called stigmergy [12]. An individual ant may only perform simple tasks. However, a whole colony of ants - a highly structured social organization - is able to fulfill complex tasks by coordinating their

[4] This section is based on [12] Dorigo, M., Stützle, T. (2004)

activities by modifying their environment. A particularly important chemical is the trail pheromone that helps ants to move in the surrounding area of their nest. Experiments like the double bridge experiment [9] and [15] show the behavior of ants foraging for food.

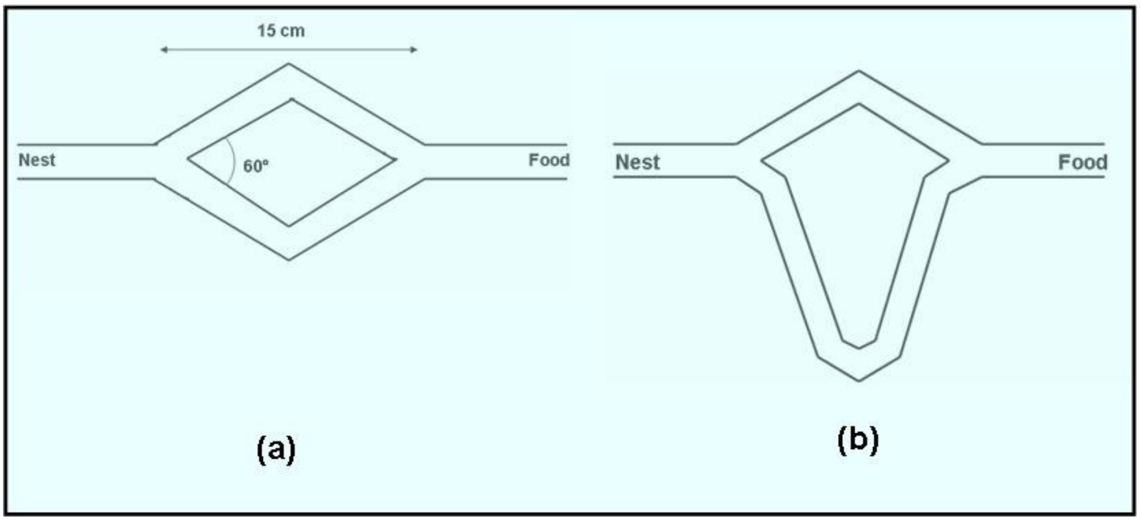

Figure 3: Double bridge experiment: (a) equal length and (b) double length

As figure 3 shows, the nest and the food source are connected by two bridges equally long in instance (a) and one longer than the other in instance (b). Initially, no pheromone trails are laid. Ants proceed from their nest to the first intersection and, in both cases (a) and (b), randomly choose one of the two bridges with nearly the same probability while searching for food. Still, the number of ants on each connection differs due to random fluctuation. Ants cross the bridges laying pheromone trails on the ground. On the way back from the food source to the nest, the amount of chemicals produced depends on the quality of the food, consisting of food quantity and the distance between nest and source. Over time, the pheromones laid in this manner evaporate. When other ants search for food, they will follow the pheromone trails and therefore abandon their random behavior more and more. In (a), one connection's pheromones dominate the other's due to initial fluctuation and after some time, all ants choose the same path to the food source. In (b), the pheromone trail on the shorter path becomes stronger than on the longer one because ants using the shorter branch arrive earlier at the food source. The usage is more frequent and the laying exceeds evaporation by far. More ants follow the most attractive path leading to less pheromone deposit on the longer path. After some time, the trails on the longer path disappear and all ants eventually choose the shorter path (see figure 4).

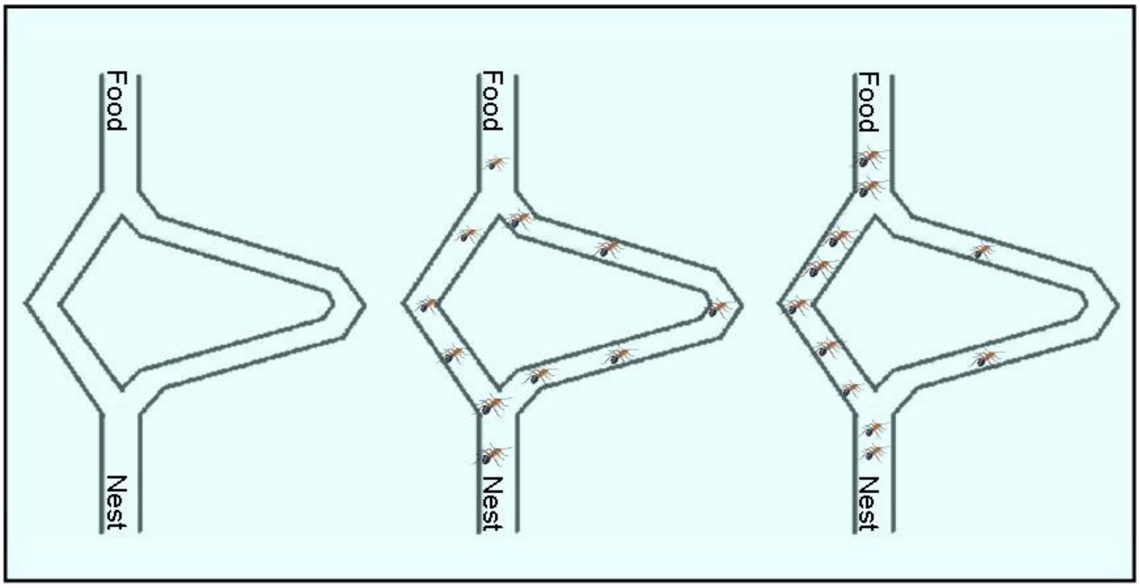

Figure 4: The effect of stigmergy during food foraging

3.1.2 Artificial ants and ACO algorithms for the TSP

In ACO, artificial ants simulate the trail laying and following procedure of real ants in order to build solutions to an optimization problem. Dorigo and Stützle [12] call ants stochastic constructive procedures that incrementally build solutions by performing a randomized walk on a completely connected graph and by adding opportunely defined solution components to a partial solution under construction. m_a ants construct solutions to a problem which can be defined on a completely connected construction graph $G=(C,L)$. $C=\{c_0,c_1,.....,c_n\}$ represents the components and L is a set of connections between these components on the graph. An ant $k's$ move from one component c_i to another c_j is subject to a probabilistic decision depending on heuristic information η_{ij} and pheromone trail τ_{ij}. After finishing a move on the graph, ant k stores the found solution in its memory M^k. M^k can be used to build feasible solutions, compute heuristic values η_{ij} and to evaluate the solution found by updating the pheromone τ_{ij} on the connections visited depending on their quality. If a pre-specified termination condition e^k is met, ant k ends the construction process.

Figure 5 demonstrates the general framework of ACO algorithms with a pseudo-code. *ConstructAntsSolution* controls the construction moves of the colony. *UpdatePheromones* manages the value of new pheromones and evaporation. *DaeomonActions* are optional measures including local search and global pheromone update.

```
procedure ACOMetaheuristic
        ScheduleActivities
                ConstructAntsSolutions
                UpdatePheromones
                DaemonActions               //optional
        end-ScheduleActivities
end-procedure
```

Figure 5: ACO pseudo-code

The difference between global and local update will become clear in the next section, where the functionality of ACO algorithms applied to the TSP is demonstrated. The focus lies on the Ant System (AS) and especially the ACS algorithm. Also, other important ACO algorithms, namely Elitist Ant System, Ant-based Ant System and Max-Min Ant System will be addressed. Since I will implement ACO for the TSP (as part of the CTP), I will refer to components $C = \{c_1, c_2, \ldots\ldots, c_n\}$ as vertices (or cities) $V = \{v_0, \ldots., v_n\}$ and to the connection set L as edge set $E = \{(v_i, v_j) : v_i, v_j \in V, i < j\}$. In all ACO algorithms, each edge is assigned a pheromone trail τ_{ij} and a heuristic value η_{ij} (e.g. $\eta_{ij} = 1/c_{ij}$ the reciprocal of the distance between two cities i and j) during solution construction. The initial pheromone value τ_0 is set to $1/nC^{nn}$ with n being the number of cities and C^{nn} the length of a nearest-neighbor tour. Following the construction process, each ant is placed at an initial city based on some criterion, then uses τ_{ij} and η_{ij} in the probabilistic manner described above to iteratively visit all the vertices and finally returns to the starting vertex. Afterwards, the ant passes through the found solution in the opposite order to assign the edges used with pheromone values. Furthermore, daemon actions may be executed.

3.1.3 Ant System

The first algorithm imitating the foraging behavior was AS, introduced in [10] and [11]. At first there were three versions, two with pheromone updates directly after a move from one city to the next (ant-density and ant-quantity) which performed rather poorly in comparison to the third one where pheromone updates were related to the tour quality and executed after all ants had finished constructing (ant-circle). The latter is now known as AS. It consists of solution construction and pheromone update. An iteration draws the following pattern: m_a ants are randomly positioned at different starting points. An ant k moves from city i to j according to the probabilistic state transition rule:

$$p_{ij}^k = \frac{\left[\tau_{ij}\right]^{\zeta}\left[\eta_{ij}\right]^{\beta}}{\sum l \in N_i^k \left[\tau_{il}\right]^{\zeta}\left[\eta_{il}\right]^{\beta}}, \qquad \forall j \in N_i^k, \tag{3.1}$$

p_{ij}^k is the probability of ant k choosing edge (i, j) to move from city i to j. As mentioned above, $\eta_{ij} = 1/c_{ij}$ is the heuristic information value, ζ and β are parameters determining the influence of pheromone values τ_{ij} and heuristic information on the decision which city to visit next. N_i^k is the feasible neighborhood of ant k defined by not yet visited cities available at city i. If $\zeta = 0$, only the heuristic information and therefore the closest city is taken into account. If $\beta = 0$, only pheromone values determine the move and stagnation may occur.

Before every move of ant k, the probability p_{ij}^k has to be calculated for all candidate edges and is then added up to a cumulative probability. Then the so called Roulette Wheel selection is performed by generating a random number between 0 and the probability sum of all possible moves ($\sum p_{ij}^k$) and selecting a move if the corresponding cumulative probability range contains that number. Figure 6 demonstrates this procedure with a small example:

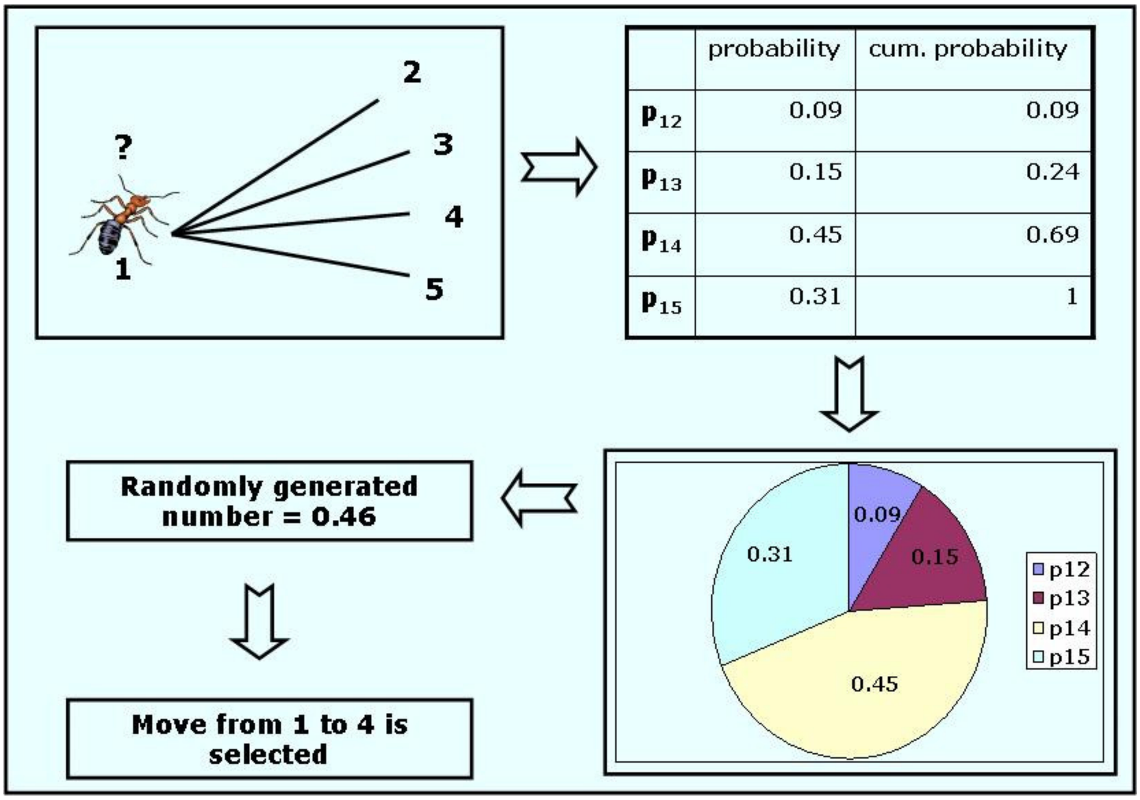

Figure 6: Decision making in ACO

In addition, each ant k possesses a memory M^k. It stores the list of cities already visited in the relevant order. This serves as a basis to determine the feasible neighborhoods N_i^k. Furthermore, it enables the ant to compute the length C^k of its tour T^k as well as to follow the tour in the opposite way to deposit pheromones. The pheromone trail update includes on the one hand a phase of evaporation by a factor α and on the other an update of an edge (i, j) by all ants $\sum_{k=1}^{m} \Delta\tau_{ij}^k(t)$.

$$\tau_{ij}(t+1) = (1-\alpha) * \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^k(t) \qquad \forall (i, j) \qquad (3.2)$$

A single ant k reinforces the edge with $\Delta\tau_{ij}^k = 1/C^k$, if edge (i, j) belongs to T^k and not at all otherwise. Consequently, the quality (shortness) of a tour and ant frequency on an edge increases its pheromone level.

Whether ants construct their solutions sequentially or in parallel doesn’t influence the quality of algorithmic output significantly. AS did not turn out to be competitive with other solution approaches to the TSP but the idea was adapted and modified.

3.1.4 Ant Colony System

When comparing ACS to AS, the main differences are based on exploration and exploitation. To start with, the construction phase uses a different, more aggressive state transition rule. Ant k chooses vertex j after vertex i according to

$$j = \arg\max_{l \in N_i^k} \left\{ \tau_{il} \left[\eta_{il} \right]^{\beta} \right\}, \qquad \text{if } q \leq q_o;$$

$$j = J, \qquad \text{otherwise.} \tag{3.4}$$

q is a random variable uniformly distributed in $[0,1]$ and q_0 $(0 \leq q_0 \leq 1)$ is a parameter. If $q \leq q_0$, the move with the highest state transition value is performed. Otherwise the next step is assigned by J. J is a random variable that is determined by the probabilistic state transition rule in (3.1) with $\zeta = 1$ and the roulette wheel decision method. Exploitation of existing knowledge (memorized pheromone trails and heuristic information) and therefore concentration on the best-so-far tour occurs with probability q_0 while exploration of other tours is performed with probability $(1 - q_0)$.

Another difference lies in the pheromone updating rule. In ACS, global and local updating procedures occur. Every ant performs local modification of the pheromone level immediately after traversing an edge. Hence, the updating process is partly executed during the tour construction phase for each edge. The local update rule is

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \rho\tau_0, \tag{3.5}$$

with parameters τ_0 (initial pheromone value $1/nC^{nn}$) and $0 \leq \rho \leq 1$. ρ regulates the amount of evaporation and pheromone deposit during the updating procedure. Local update has the effect that high frequency on an edge (i, j)

leads to decreasing pheromone level τ_{ij}. Other ants are less likely to cross this edge which in turn favors the exploration of new edges and avoids stagnation.
During global update, only the ant that constructed the best-so-far tour may add pheromone after each iteration:

$$\tau_{ij} = (1-\alpha)\tau_{ij} + \alpha\Delta\tau_{ij}^{bs}, \qquad \forall(i,j) \in T^{bs}. \tag{3.6}$$

Naturally, now $\Delta\tau_{ij}^{bs} = 1/C^{bs}$ where C^{bs} is the length of the best-so-far tour T^{bs}. Only edges on this tour are affected by pheromone deposit as well as evaporation.
So far, ACS implementations have shown that, using parallel construction by all ants does not exceed solution quality of sequential construction. Using the iteration best tour instead of the best-so-far tour in global updating leads to worse results solving larger TSP instances. For all further use of ACO, I will apply sequential construction and global updating according to the best-so-far tour.

3.1.5 Other ACO algorithms

Elitist Ant System

The first update of AS was Elitist Ant System which uses stronger pheromone trail laying on the best-so-far tour T^{bs} with length C^{bs} constructed by an ant. In addition to the pheromone update applied in AS (equation (3.2)), edges belonging to T^{bs} receive $e\Delta\tau_{ij}^{bs}$ additional pheromone. Parameter e is a weight for T^{bs} and $\Delta\tau_{ij}^{bs}$ equals $1/C^{bs}$.

Rank-Based Ant System

Rank-Based Ant System sorts ants according to the quality of their solutions constructed and only the $(w-1)$ best-ranked ants as well as the best-so-far ant (with rank w) may deposit pheromone weighted according to their rank:

$$\tau_{ij}(t+1) = (1-\rho) * \tau_{ij}(t) + \sum_{r=1}^{w-1} (w-r)\Delta\tau_{ij}^{r}(t) + w\Delta\tau_{ij}^{bs} \qquad \forall (i,j) \tag{3.7}$$

<u>MAX-MIN Ant System</u>

In MAX-MIN Ant System, only the best ant (either the best-so-far or the iteration-best) lays pheromone trails. In order to prevent stagnation by following only one ant, limits $[\tau_{\min}, \tau_{\max}]$ for the amount of deposit are introduced. First, the trails are initialized with $\tau_{\max}$ and evaporation is small. If signs of stagnation emerge after some time, trails are reset to $\tau_{\max}$.

3.2 GENIUS algorithm

GENIUS, a two phase heuristic composed of the GENI phase (abbr. for General Insertion) and the US phase (abbr. for Unstringing and Stringing), first constructs and then re-optimizes a tour [13].
This two-phase heuristic consists of an iterative insertion heuristic GENI and a post-optimization procedure US and was first applied to the TSP [13]. In the following sections, both heuristics will be described separately. When looking at a set of vertices that should belong to a tour, GENI iteratively includes them one by one until all vertices are visited. Afterwards, US improves the tour also vertex by vertex.

3.2.1 GENI

"*Generalized insertion can be described as an insertion procedure which uses a limited form of incremental local search*" [3].
The insertion procedure GENI adds a vertex v, currently not on the tour, between two vertices already belonging to the tour. Initially, these two vertices need not appear in consecutive order along the tour. However, after vertex v was inserted into the tour, the two vertices will be the preceding and succeeding neighbor of v. This procedure combines local optimization and insertion steps.

In general, any vertex v_h on any tour has a predecessor v_{h-1} and a successor v_{h+1}. As stated above, vertex v should be integrated in the tour between any two vertices v_i and v_j.

In order to limit the search space for any vertex $v \in V$ waiting to be inserted, GENI checks a set of p vertices already on the tour belonging to the p-neighborhood $N_p(v)$, including only those vertices closest to vertex v (based on the distance or travel time matrix $C = (c_{ij})$). The parameter p is usually set to a relative small number somewhere between 4 and 7. If a tour consists of less than p vertices, all members of this tour belong to the neighborhood $N_p(v)$ of a vertex $v \in V$. GENI will investigate insertions for a given parameter p.

Gendreau et al. describe two different types of insertion possibilities [13].

3.2.1.1 Type I Insertion

Vertex v_k lies on the path between v_j and v_i for a clockwise orientation of the tour. Vertices v_i and v_j must be chosen such that $v_i, v_j \in N_p(v)$ and vertex v_k such that $v_k \in N_p(v_{i+1})$. Also, $v_k \neq v_i$ and $v_k \neq v_j$ has to be taken into account.

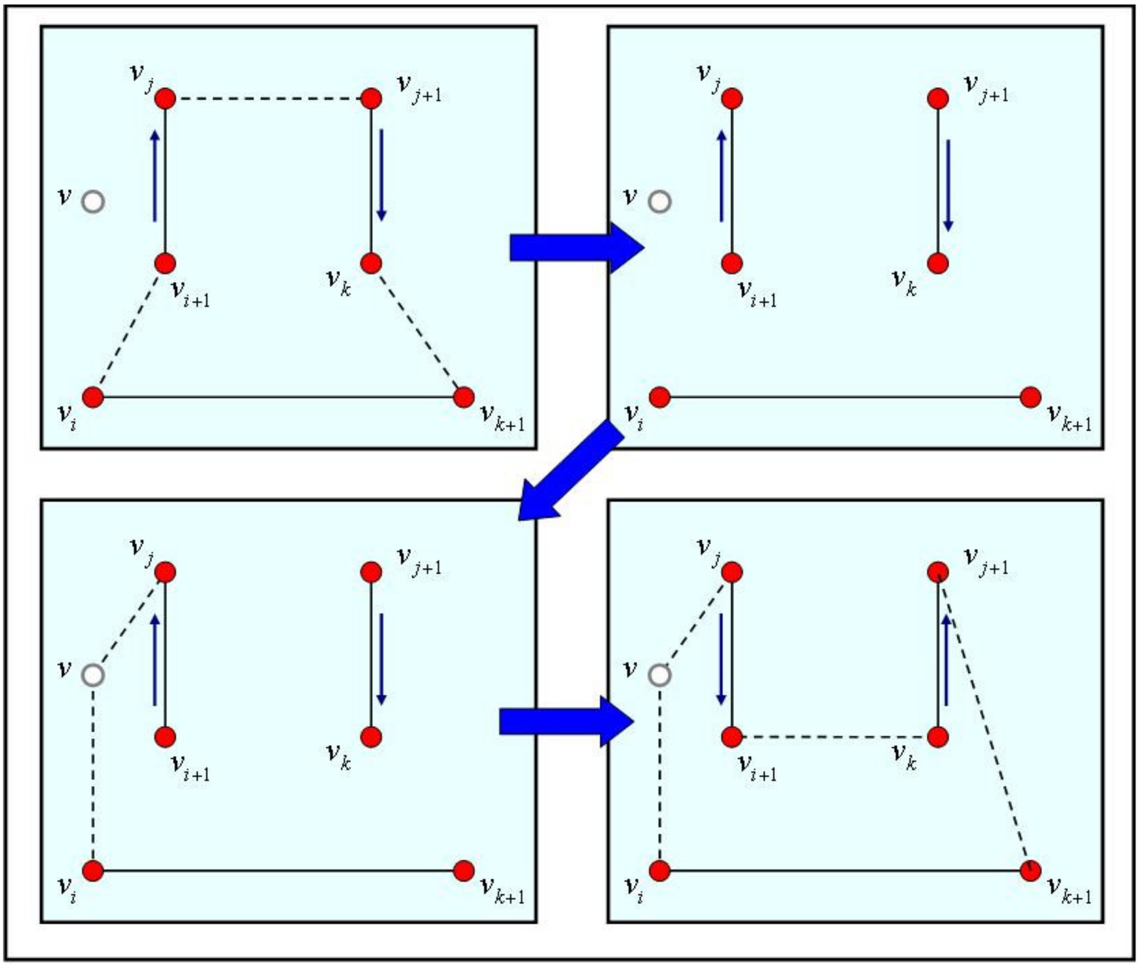

Figure 7: Type I insertion procedure

Each move demands an insertion (limited to $N_p(v)$) and one 3-opt exchange. Figure 7 shows that after choosing vertices v_i, v_j and v_k, inserting vertex v leads to the replacement of old edges $(v_i, v_{i+1}), (v_j, v_{j+1})$ and (v_k, v_{k+1}) by new edges $(v_i, v), (v, v_j), (v_{i+1}, v_k)$ and (v_{j+1}, v_{k+1}) in order to construct the best possible GENI tour. One of the latter four edges is constructed due to the insertion of v and three are substitutes for the first three edges (3-opt local re-optimization). Furthermore, paths (v_{i+1}, v_j) and (v_{j+1}, v_k) are reversed. The objective of type I insertion procedure is to choose the best of all possible moves for $v_i, v_j \in N_p(v)$ and $v_k \in N_p(v_{i+1})$.

3.2.1.2 Type II Insertion

Again, vertex v_k lies on the path between v_j and v_i. Furthermore, vertex v_l is located on the path from v_i to v_j for a clockwise orientation of the tour. Vertices v_i and v_j must be chosen such that $v_i, v_j \in N_p(v)$, vertex v_k such that $v_k \in N_p(v_{i+1})$ and vertex v_l such that $v_l \in N_p(v_{j+1})$. Also, $v_l \neq v_i, v_{i+1}$ and $v_k \neq v_j, v_{j+1}$ has to be taken into account.

A type II insertion of vertex v results in the deletion of old edges $(v_i, v_{i+1}), (v_{l-1}, v_l), (v_j, v_{j+1})$ and (v_k, v_{k-1}). The following figure shows that they are replaced by new edges $(v_i, v), (v, v_j), (v_l, v_{j+1}), (v_{k-1}, v_{l-1})$ and (v_{i+1}, v_k) to obtain the best possible GENI constructed tour. Paths (v_{i+1}, v_{l-1}) and (v_l, v_j) are inverted. The difference to type I is that the local search and re-optimization is achieved by running a 4-opt algorithm instead of a 3-opt.

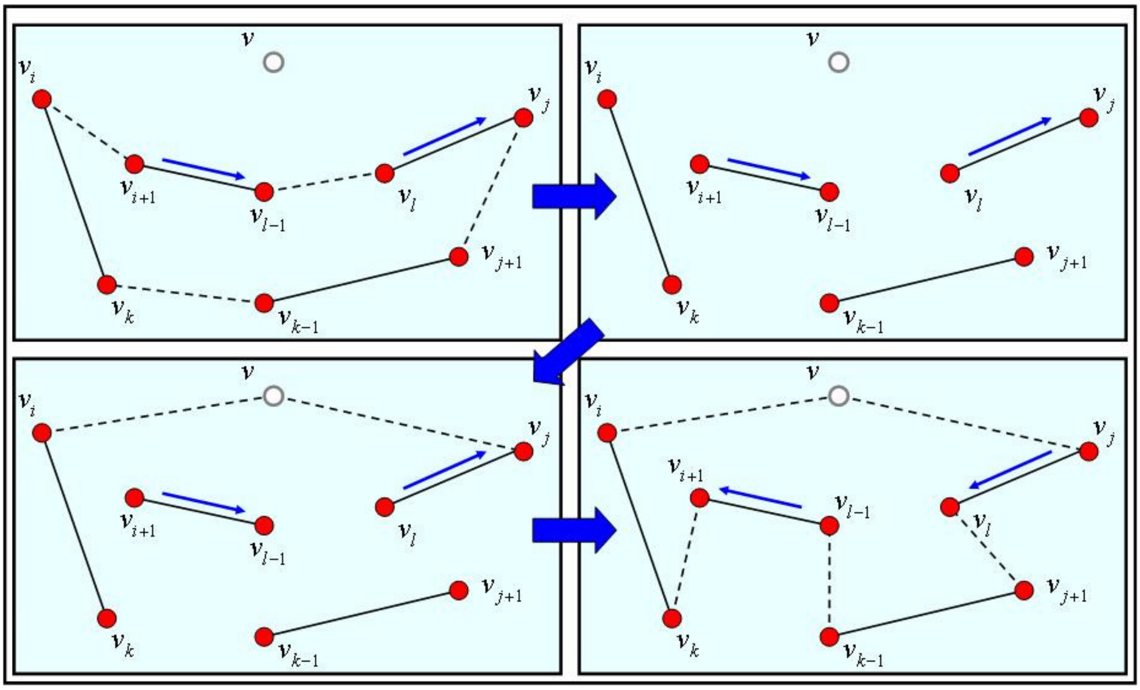

Figure 8: Type II insertion procedure

Both types of insertion are considered likewise for a clockwise and a counter-clockwise orientation of the tour which leads to four different types of insertions. Moreover, for each type of insertion, the potential number of choices for v_i, v_j, v_k and v_l is n^4, where n is the number of vertices in total. The introduction of neighborhoods to narrow the search space reduces the complexity to $O(p^4)$, where O describes the effect of the problem size on the algorithm's usage of computational resources. If $v_i \in N_p(v)$, an examination of the insertion of $v \in V$ between two consecutive vertices v_i and v_{i+1} will also be executed. Finally, the best overall insertion will be executed.

3.2.1.3 GENI algorithm

The GENI algorithm passes through the following iterations:

Iteration 1:
An initial tour is created by a random subset selection containing three vertices (one of them the depot v_0).
The p-neighborhoods for every vertex are initialized.

Iteration 2:
Random selection of any vertex $v \in V$ not yet inserted in the tour. The least cost insertion of the chosen vertex $v \in V$ with respect to all possible insertions of type I and II is selected.
The p-neighborhoods of all remaining vertices are updated due to the insertion of vertex $v \in V$ on the tour.

Iteration 3:
If all vertices have been inserted, END. Else go to iteration 2.

Inserting vertex $v \in V$ and updating the tour requires $O(n)$ time. As iteration 2 has to be executed $n-3$ times, the overall complexity for the GENI algorithm is $O(np^4 + n^2)$.

3.2.2 US

The post-optimization algorithm US [13] can be operated on tours produced by any algorithm. The main feature of US is to remove a vertex (U - unstring) from a feasible tour and reinsert (S - string) it. While the stringing process is

identical with iteration 2 of the GENI algorithm, unstringing a given tour simply reverses the insertion procedure used by the GENI algorithm.
Again, there are two possible options of reconnecting the members of the tour after the removal of any vertex v_i.

3.2.2.1 Type I Unstringing

Vertices v_j and v_k are chosen such that $v_j \in N_p(v_{i+1})$ and $v_k \in N_p(v_{i-1})$ is a vertex on the path $(v_{i+1},\ldots,v_{j-1})$. Figure 9 demonstrates an US iteration:

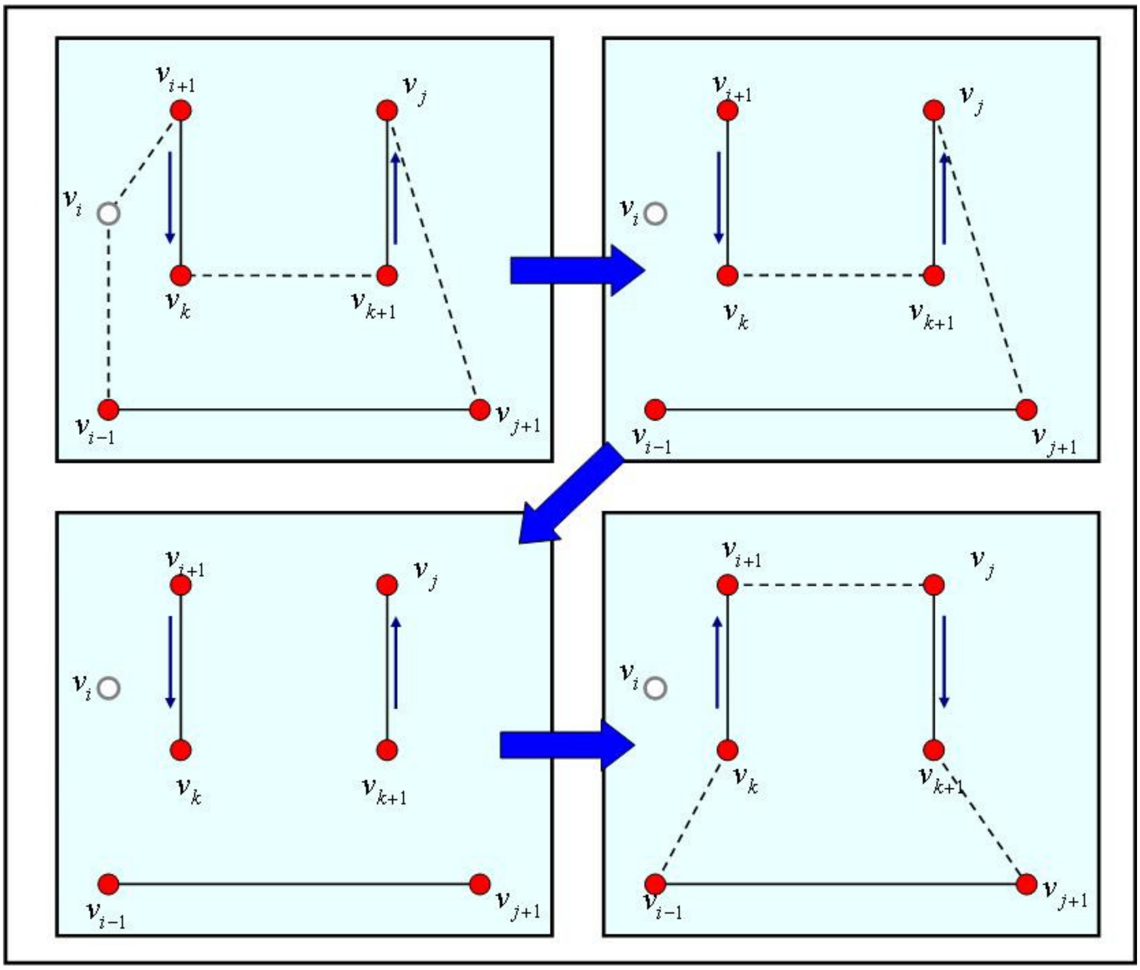

Figure 9: Type I unstringing of vertex v_i from the tour

The old edges $(v_{i-1},v_i),(v_i,v_{i+1}),(v_k,v_{k+1})$ and (v_j,v_{j+1}) are removed and replaced by edges $(v_{i-1},v_k),(v_{i+1},v_j)$ and (v_{k+1},v_{j+1}). Additionally, paths (v_{i+1},v_k) and (v_{k+1},v_{j+1}) are reversed.

3.2.2.2 Type II Unstringing

As before, vertices v_j and v_k are chosen such that $v_j \in N_p(v_{i+1})$ and $v_k \in N_p(v_{i-1})$ is a vertex on the path $(v_{i+1},....,v_{j-1})$. Additionally, vertex v_l is selected so that $v_l \in Np(v_{k+1})$ on the path $(v_j,....,v_{l+1})$. Figure 10 demonstrates an US iteration:

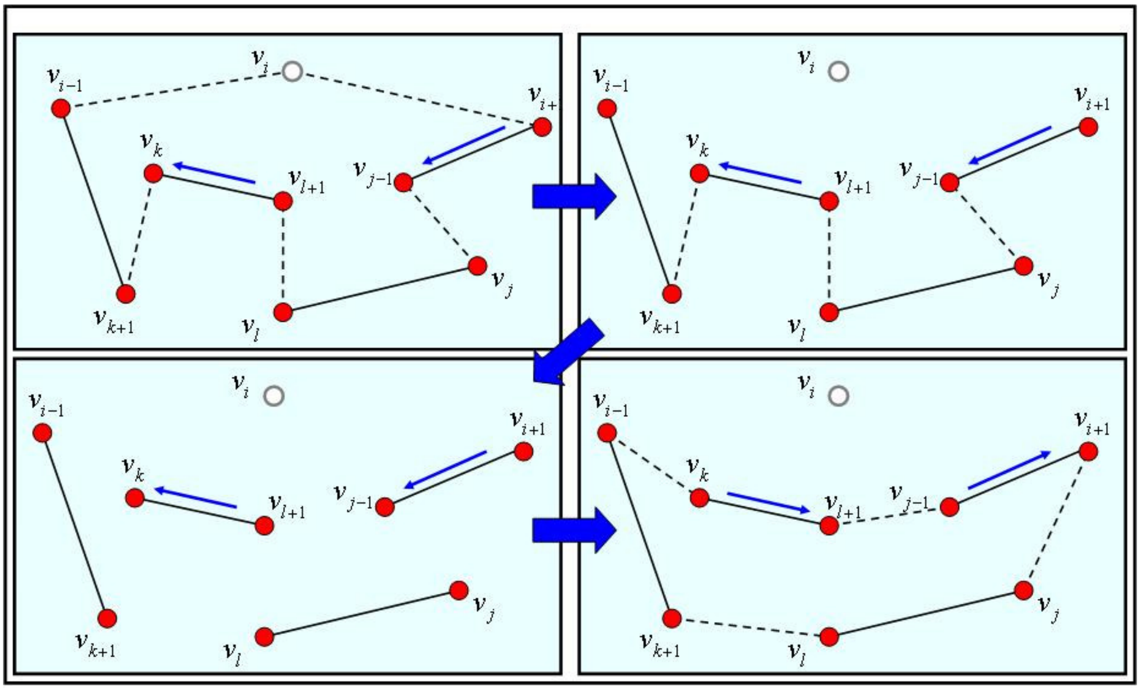

Figure 10: Type II unstringing of vertex v_i from the tour

Then, old edges $(v_{i-1},v_i),(v_i,v_{i+1}),(v_{j-1},v_j),(v_l,v_{l+1})$ and (v_k,v_{k+1}) are removed and replaced by $(v_{i-1},v_k),(v_{i+1},v_{j-1}),(v_{i+1},v_j)$ and (v_l,v_{k+1}). Again, two paths, (v_{i+1},v_{j-1}) and (v_{l+1},v_k), are inverted.

3.2.2.3 Stringing

Stringing works just like a GENI insertion but now different neighborhood structures possibly lead to new re-insertion positions and therefore to a changed vertex sequence.

3.2.2.4 US algorithm

The following iterations demonstrate the work flow of the US algorithm:

Iteration 1:
Use an initial tour H of cost z created by any algorithm.
Set the best-so-far tour $H^* := H$ and the best-so-far cost of the tour $z^* := z$ and $t := 1$;

Iteration 2:
Randomly select a vertex v_i that has not been considered yet. First, unstring and string using both types and possible tour orientations for vertex v_i of the current tour.
The resulting tour H' has cost z'.

- If $z' < z^*$, set $H^* := H'$, $z^* := z'$ and $t := 1$; repeat Iteration 2;
- If $z' \geq z^*$, set $t := t + 1$;repeat Iteration 2;
- If $t = n + 1$, STOP. The best available tour is H^* with costs z^*.

3.3 PRIMAL1 set covering heuristic

The PRIMAL1 set covering heuristic [2] was developed to solve SCPs.
In order to keep track of what is supposed to happen during solving the CTP, I will adapt the formulations used in the SCP model of section 2.4.2 to the ones used in the CTP model in 2.2 in the next section.
PRIMAL1 first sets $y_k := 1$ for all $v_k \in T$ and then iteratively adds the remaining vertices (columns) v_k following a greedy criterion that minimizes the function $f(c_k, b_k)$. For each individual vertex (column) v_k with $y_k = 0$, the parameter b_k sums up uncovered vertices (rows) $v_\ell \in W$ (the set that has to be covered) with a binary coefficient $\delta_{\ell k} = 1$. This means that all vertices (rows) $v_\ell \in W$ covered by a vertex (column) $v_k \in V \setminus S$ but not by the temporary solution are added up to b_k. Three different versions of the function $f(c_k, b_k)$ are considered and applied to the set covering problem:

$f(c_k, b_k) = c_k / \log_2 b_k$, (i)

$f(c_k, b_k) = c_k / b_k$, (ii)

$f(c_k, b_k) = c_k$. (iii)

Vertices (columns) $v_k \in V$ are sorted according to the version of the function $f(c_k, b_k)$ currently in use and the cheapest insertion is performed. At the beginning, criterion (i) is applied until all rows $v_\ell \in W$ are covered. If at least one vertex (row) $v_\ell \in W$ with $\delta_{\ell k} = 1$ is covered by more than one vertex (column) $v_k \in V$, the associated vertices (columns) that overcover the row are deleted from the partial solution and sorted again, now according to criterion (ii). Once more, overcovering vertices (columns) are removed, criterion (iii) is applied and the final solution of the first run is obtained.
The heuristic is run a second time with criteria sequence (i), (iii) and (ii). The best sequence from both runs is kept.

3.4 Solving the CTP

After an introduction of the CTP, of the components it can be separated into and of possible solution techniques for these components, this section focuses on solving the problem itself. I combine solution methods for the TSP and the SCP in order to find a good solution for the CTP. The first attempt is the same heuristic approach as applied by Gendreau et al. [14] which uses GENIUS and PRIMAL1.
The second attempt applies ACO with GACS for the TSP and SCACS for the SCP. I named the combination of these two methods ACS for the CTP (CTACS). The algorithms created are described below.

3.4.1 H-1-CTP heuristic

The combination of PRIMAL1 and GENIUS results in the approximate algorithm H-1-CTP [16]. It passes through the following iterations twice, considering the same covering criteria sequence as in PRIMAL1. H is the set of vertices belonging to the current TSP tour under construction, z the cost of this tour,

$H*$ the local optimum tour, $z*$ its cost and $f(c_k, b_k)$ the current covering criterion.

Iteration 1 - Initialization

Set $H := T$ and $z* := \infty$, $f(c_k, b_k) = (i)$ (PRIMAL1);

Iteration 2 – Construction

Using GENIUS, construct a Hamiltonian cycle over H where z represents the length of the tour;

Iteration 3 – Termination

If one vertex $v_\ell \in W$ is not yet covered by the tour over H, go to iteration 4.

Else, if $z \leq z*$, set $z* := z$ and $H* := H$.

If the covering criterion (PRIMAL1) is the last one, the local optimum is given by $H*$ with cost $z*$.

Else remove all vertices from H associated with over-covered vertices of W and move to the next covering criterion (PRIMAL1).

Iteration 4 – Selection

A coefficient c_k representing the cheapest insertion of v_k in the current tour H is calculated for every $v_k \in V \setminus H$. The best vertex v_k with respect to the current covering criterion is inserted into H (PRIMAL1).

Set $H := H \cup \{v_k\}$and go to iteration 2.

The better of the two runs then delivers the final solution of the CTP.

3.4.2 ACS for the CTP

The metaheuristic approach CTACS uses the idea of the COVTOUR Covering Salesman Problem heuristic [7] where the SCP was solved first and this solution was then used to formulate the TSP instance which was then solved separately. Here, I use ACS to solve the SCP and then GACS to solve the resulting TSP problem. The following sections introduce the two solution approaches in detail.

3.4.2.1 GACS

The classical ACS algorithm uses a nearest neighbor approach to choose the next city to be visited. The next vertex to be inserted is selected according to the probabilistic state transition rule which incorporates the pheromone trails and the heuristic information. Also, the vertex will always be positioned at the end of a tour under construction. Without the degree of probability, ACS would deliver identical vertex sequences and therefore equal results for the same starting point. Consequently, solutions generated by the classical ACS strongly depend on the selection order of the cities. GACS introduces the GENI heuristic. Here, the next vertex to be inserted is chosen in a random fashion. However, now the insertion procedure is more accurate because the position of the vertex on the tour is chosen very carefully and is more important than the assigned vertex.

Two adjustments concerning the cost of an edge and the state transition rule have to be made.

First, the cost of an edge (i, j) now depends on its length c_{ij} as well as on the amount of pheromone τ_{ij} stored on it.

The modified cost of an edge is:

$$c'_{ij} = \frac{c_{ij}}{1+\gamma \cdot \tau_{ij}^{R}} \qquad \forall (i, j) \in E \qquad (3.7)$$

with the relative amount of pheromone τ_{ij}^{R}:

$$\tau_{ij}^{R} = \frac{\tau_{ij}}{\max_{(k,l)\in E}(\tau_{kl})}. \qquad (3.8)$$

The original cost c_{ij} of edge (i, j) is taken from the distance or travel time matrix $C = (c_{ij})$. τ_{ij}^{R} is the relative amount of pheromone on edge (i, j) where the original pheromone value τ_{ij} is normalized between 0 and 1 on every edge. If $\max_{(k,l)\in E}(\tau_{kl}) = 0$, which is the case when no pheromone has been distributed on the edges, $\tau_{ij}^{R} = 0$ for every edge. Parameter γ modifies the relative influence of pheromone values on an edge. Equation (3.7) assigns fewer costs to edges with higher pheromone values. In addition, it provides a lower and upper bound on the adjusted edge costs c'_{ij}. On the one hand it can not exceed the original cost c_{ij} and on the other it never declines to less than half of them.

Second, the state transition rule has to be modified with respect to the GENI insertion method. As already mentioned, without probabilistic decision making, the classical ACS would produce identical selection orders for the same starting point, which in turn results in equivalent solutions. GACS may use different selection orders that still result in the same solutions. Consequently, GACS uses a new probabilistic state transition rule with a rank-based approach to alter the search space and to decide on the GENI insertion type used for the next city. In every iteration, the available moves consisting of all possible GENI insertions are reduced to a parameter S^{GENI} in order to prevent bad choices. They are then ranked from the cheapest insertion with rank S^{GENI} to the most expensive insertion with rank 1. An insertion with rank t will then be selected between 1 and S^{GENI} according to the following state transition rule:

$$t = \begin{cases} S^{GENI}, & if\ q \le q_0, \\ T^{GENI}, & otherwise. \end{cases} \qquad (3.9)$$

As in (3.4), q is a random variable uniformly distributed in $[0,1]$ and parameter q_0 is $(0 \le q_0 \le 1)$. T^{GENI} is a random variable with the following probability distribution:

$$p(T^{GENI} = t) = \frac{t}{\sum_{t'=1}^{S^{GENI}} t'} = \frac{2t}{S(S+1)}, \qquad t = 1,\ldots.., S^{GENI}. \qquad (3.10)$$

In this case, the roulette wheel decision process between ranks $1,\ldots,S^{GENI}$ appoints the next insertion.

The ranking system allows better distinctions between almost equal moves and prevents stagnation by setting the selection probability of the best insertion below 1. According to these two equations, the cheapest insertion with rank $t = S^{GENI}$ will be chosen with the highest and the most expensive insertion with rank $t = 1$ with the lowest probability. I have settled for $S^{GENI} = 5$, just like Le Louran et al. [19], in order to limit the search space and to prevent too intense diversification.

The GACS algorithm can be summarized in the following way:

Initialization of pheromones and parameters
For I iterations do:
    For ant 1 to m_a do:
        GENI algorithm
        Local pheromone update;
    Global pheromone update;
GACS best solution output;

Figure 11: GACS algorithm

3.4.2.2 SCACS

In general, ACO for the SCP assigns column j a pheromone value τ_j and a heuristic value η_j where τ_j represents the learned desirability and η_j the heuristic desirability of choosing column j. A single ant starts with an empty memory M^k and constructs a solution by probabilistically adding columns step by step until all rows are covered. Again, the probabilistic rule of column choice depends on the pheromone value τ_j and the heuristic value η_j. After all ants have constructed their solution, local search may be implemented and finally the pheromone trails are updated.

However, three main differences to other ACO applications such as the TSP appear when solving the SCP: ants do not need the same number of iterations to solve the problem, the order of including columns has no influence on the solution and possible redundant information in intermediate solutions may be eliminated by local search before updating.

For the solution of the SCP embedded in the CTP, I will use the ACS algorithm introduced by Lessing et al. [20] that more or less represents the ACS framework introduced in 3.1.2. The state transition rule for choosing the next column in the SCP is:

$$j = \begin{cases} \arg\max_{l \notin S_k} \left\{ \tau_l [\eta_l]^\beta \right\}, & \text{if } q \le q_0, \\ draw(J), & \text{otherwise.} \end{cases} \tag{3.11}$$

Again, q is a random variable uniformly distributed in $[0,1]$, q_0 $(0 \le q_0 \le 1)$ is a parameter, S_k is the partial solution obtained by ant k and $draw(J)$ is a random variable that equals the probabilistic state transition rule. β determines the influence of the heuristic information η on the decision. The corresponding probability p_j^k of ant k choosing column j equals:

$$p_j^k = \frac{\tau_j [\eta_j]^\beta}{\sum_{h \notin S_k} \tau_h [\eta_h]^\beta}, \qquad \text{if } j \notin S_k \tag{3.12}$$

If $q \ge q_0$, the next column will be chosen through roulette wheel just as in AS, ACS and GACS. Of course, if $j \in S_k$ then $p_j^k = 0$. In addition, redundant columns have to be deleted to ensure good solution quality.

Ant k proceeds with the local pheromone update in order to increase exploration after it has added a column j to its partial solution S_k according to

$$\tau_j = (1-\rho)\tau_j + \rho\tau_0, \tag{3.13}$$

with parameters $\tau_0 = 1/(n \cdot z_{GR})$ where z_{GR} is the cost of a greedy solution and $0 \le \rho \le 1$.

Finally, when the construction process of each ant has ended, the best-so-far ant updates the pheromone trails globally by

$$\tau_j = (1-\alpha)\tau_j + \alpha \cdot \Delta\tau_j^{bs}, \qquad \forall j \in S^{bs} \tag{3.14}$$

with the best-so-far solution S^{bs}, its cost z^{bs} and $\Delta\tau_j^{bs} = 1/z^{bs}$.

Another important factor of SCACS is to decide on the best kind of heuristic information to use for the state transition. Lessing et al. [20] list seven different types of possible heuristic information which they categorize as static and dynamic. I will focus on three approaches, namely column cost, cover cost and Marchiori and Steenbeek cover costs [21].

Column costs

This approach, using static costs, is very straight forward because the heuristic information η_j is defined as the column cost reciprocal $1/c_j$.

Cover costs

When looking back at the PRIMAL1 algorithm (3.3), the parameter b_k was the number of rows covered by vertex k but not by a partial solution. With cover costs, the heuristic information η_j equals b_j/c_j for all columns not yet part of the solution. Subscript k is replaced by j.

Marchiori and Steenbeek (M&S) cover costs

This is a variant of the cover costs above. $\mathrm{cov}(S)$ is the set of rows covered by the columns of a partial solution S. The set of rows covered by column j but not by any other column in S is $\mathrm{cov}(j,S)$. The minimum cost $c_{\min}(i)$ of all columns covering row i not yet part of the solution and member of $\mathrm{cov}(j,S)$ must be determined. In order to derive the cover value $cv(j,S)$ (of a column j with respect to a partial solution S), $c_{\min}(i)$ for all rows of $\mathrm{cov}(j,S)$ have to be summed up:

$$cv(j,S) = \sum_{i \in \mathrm{cov}(j,S)} c_{\min}(i) \tag{3.15}$$

The modified cover costs $\mathrm{cov\_}val(j,S)$ are:

$$\mathrm{cov\_}val(j,S) = \begin{cases} \infty, & \text{if } cv(j,S) = 0, \\ c_j / cv(j,S), & \text{otherwise.} \end{cases} \tag{3.16}$$

The heuristic information η_j equals the (M&S) cover costs $1/\mathrm{cov}\_val(j,S)$.

Three of the remaining four methods used in [20] apply normalized Lagrangean costs instead of column costs c_j in the above variants and the last one uses lower bounds. However, I decided not to include a Lagrangean approach to keep the scope of this thesis manageable.

The SCACS algorithm can be summarized as:

```
Initialization of pheromones and parameters
For I iterations do:
        For ant 1 to m_a do:
                Construct solution;
                Eliminate redundant columns;
                Local update;
        Global pheromone update;
SCACS best solution output;
```

Figure 12: SCACS algorithm

3.4.2.3 CTACS

I combined the two algorithms described above to construct CTACS and to solve the CTP. The set V accounts for the columns and the set W for the rows in the SCP. First, a tour is constructed with GENI over all vertices within the set $T \in V$. The column cost for every vertex of set $V \setminus T$ is determined by calculating the cost of a GENI insertion in the tour over T for every vertex. The column coverage naturally depends on the choice of the covering distance c. To find an initial pheromone level for SCACS, a greedy solution is created and then SCACS is run. Then I use the vertices obtained by SCACS to generate a GENI solution to derive the initial pheromone level for GACS. GACS then constructs the final CTP solution over the vertices obtained from SCACS.

4 Computational results

All algorithms described above - GENI, GENIUS, PRIMAL1, H-1-CTP, GACS, SCACS and CTACS – where implemented in C++ programming language. First, the influence of neighborhood size on GENI and GENIUS results was tested. Then GACS performance with different parameter values was observed. Furthermore, GENIUS, a multi-start GENI heuristic (mGENI) and GACS were compared while running on TSP instances. Parameter sensitivity of the SCACS was tested on a set of SCPs taken from the ORLIB [4]. I also tested the impact of the different types of heuristic information on solution quality. The algorithm was then compared to PRIMAL1.
Finally, H-1-CTP and CTACS where run on various stochastic CTPs and compared to each other.

4.1 Tests on the TSP part of the problem

4.1.1 Neighborhood size for GENI and GENIUS

In order to observe the influence of neighborhood size on the performance of GENI and GENIUS on TSP instances, the algorithms where tested on a set of four different problems. The Euclidean problems Berlin52, st70 and pr107 with 52, 70 and 107 vertices where taken from the TSPLIB library [23] and downloaded from [24]. In addition, KubLE25 with 25 vertices with x and y coordinates randomly distributed between 0 and 100 was generated to test how the algorithms react to different problem sizes. The neighborhood size range was set between $p=3$ and $p=9$. For each size and problem, the algorithms where run ten times. The average cost of a tour generated by GENI and GENIUS are shown in tables 1 and 2. The best result for each data set is highlighted with bold font.

	p						
Problem	**3**	**4**	**5**	**6**	**7**	**8**	**9**
KubLE25	477.85	450.69	454.53	449.77	450.16	448.39	**447.49**
Berlin52	8939.48	8474.93	8224.31	8116.54	8059.48	**7976.81**	8087.31
St70	788.71	718.96	699.79	700.93	693.19	**690.58**	691.31
pr107	51065.02	46691.33	46361.28	45499.44	45303.93	45512.43	**45091.05**

Table 1: Importance of neighborhood size for GENI

	p						
Problem	**3**	**4**	**5**	**6**	**7**	**8**	**9**
KubLE25	464.48	445.97	446.32	**437.26**	438.15	438.86	439.93
Berlin52	8285.78	8114.42	7939.27	7938.96	**7833.34**	7843.07	7872.23
St70	741.73	699.18	694.58	692.39	689.04	687.24	**686.95**
pr107	48721.34	46128.57	45568.39	45278.60	45163.04	45099.95	**45024.90**

Table 2: Importance of neighborhood size for GENIUS

These results indicate that a larger search space leads to better solution quality up to a certain neighborhood size. It seems that increasing neighborhood size to excessively large neighborhoods (e.g., $p > 7$) only has marginal benefit. From a neighborhood size of $p = 7$ onwards, there was a significantly higher increase in computation time than for smaller neighborhood sizes. Consequently, I settled for $p = 7$ for all the following algorithms including the GENI heuristic, which is also consistent with literature (e.g.: Gendreau et al. [13], Le Louran et al. [19]). The trade-off between resource cost and solution quality seems best with $p = 7$.

4.1.2 Parameter analysis for GACS

The impact of parameters of GACS on TSP instances was tested on the same set of problems. Just like Le Louran et al. [19], I took the average cost of 10 different runs for each problem to show the solution quality of each parameter value. I set the number of iterations I for each problem equal to the problem size (number of vertices) and the number of ants to $m_a = 10$. Parameters were tested one at a time and the others where fixed to the values $\rho = 0.5$, $\alpha = 0.5$, $\gamma = 0.5$ and $q_0 = 0.95$. I also chose to set the initial pheromone level $\tau_0 = 1/C^{GENI}$, where C^{GENI} is the cost of the tour acquired by one GENI run. As mentioned above, $p = 7$.

The results for parameter ρ - responsible for regulating the removal of pheromone from edges involved in the current tour during the local update (see equation (3.5)) - can be found in table 3. The lowest average costs over ten runs where obtained with values of $\rho \geq 0.75$ just as in the original article [19]. This implies that more search diversification leads to better solutions. Therefore, I set $\rho = 0.75$.

	ρ						
Problem	**0**	**0.1**	**0.25**	**0.5**	**0.75**	**0.9**	**1**
KubLE25	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
Berlin52	7561.66	7550.02	7551.98	7547.80	7546.40	**7544.43**	7554.55
St70	678.32	693.29	678.29	678.35	**677.32**	678.69	679.64
pr107	44727.77	44482.26	44516.17	**44425.87**	44474.28	44473.74	44477.48

Table 3: Influence of parameter ρ on GACS solution quality

Table 4 shows that parameter α - used to control the amount of pheromone deposited on edges of the best-so-far tour during global update - provides the best solutions when set to $\alpha = 0.25$. Consequently, the search does not focus too intensively on a certain solution that may only be a local optimum. This rather low value of α stimulates further search diversification.

	α						
Problem	**0**	**0.1**	**0.25**	**0.5**	**0.75**	**0.9**	**1**
KubLE25	398.06	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
Berlin52	7549.83	7551.57	**7544.37**	7547.80	7555.22	7556.81	7643.01
St70	681.41	679.73	**678.54**	678.81	679.59	679.52	686.70
pr107	44613.28	44656.33	44483.13	**44425.87**	44527.71	44528.90	45172.80

Table 4: Influence of parameter α on GACS solution quality

Parameter γ coordinates the importance of pheromone in the evaluation of edge costs. Table 5 illustrates that GACS with $\gamma = 0.5$ produces the best solutions and confirms the positive influence of pheromone trails on solution quality.

	γ					
Problem	**0**	**0.25**	**0.5**	**0.75**	**1**	**5**
KubLE25	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
Berlin52	7557.16	7559.91	**7547.80**	7551.01	7557.99	7555.04
St70	678.48	**677.30**	678.35	678.63	677.51	680.53
pr107	44426.96	44471.70	**44425.87**	44467.34	44481.94	44595.87

Table 5: Influence of parameter γ on GACS solution quality

Solutions for different settings of q_0 can be found in table 6. It is remarkable that $q_0 = 1$ - the best GENI insertion is always performed – does not necessarily lead to the highest solution quality. Deviation from the best insertion provides better results which is why I settled for $q_0 = 0.98$.

Problem	q_0 0.9	0.95	0.98	1
KubLE25	**395.64**	**395.64**	**395.64**	**395.64**
Berlin52	7550.25	7547.80	**7544.40**	7544.95
St70	679.96	678.44	677.24	**677.17**
pr107	44593.02	44425.87	**44425.77**	44426.19

Table 6: Influence of parameter q_0 on GACS solution quality

After identifying the parameter settings as $q_0 = 0.98$, $\gamma = 0.5$, $\alpha = 0.25$ and $\rho = 0.75$, I re-ran the problems to obtain results shown in table 7. In 2 instances, I found better solutions than in all the proceeding tests by combining the optimal settings for all parameters. In the other 2 instances, the solution was equal to the best solution already obtained in the earlier tests.

Problem	Solution
KubLE25	**395.64**
Berlin52	**7544.37**
St70	**677.11**
pr107	**44337.40**

Table 7: GACS results with best parameters

A detailed summary of the test runs on each parameter can be found in appendix A.

4.1.3 GENI variants comparison

GENIUS, mGENI and GACS where tested on Euclidean problems from the TSPLIB with less than 300 vertices ($n \leq$300). mGENI runs through the GENI algorithm m times without using already obtained results during further solution finding. For problems of a size up to 100 vertices, each algorithm was given the time needed to run 2.5n GENI iterations. GACS and mGENI both produced quite similar results while GENIUS lagged behind in most problems. Table 8 shows this comparison between GACS, mGENI and GENIUS, where column "Opt." holds the best known solution from the TSPLIB, column "Total runtime" is the time needed for 2.5n GENI iterations (in seconds), column "Gap" presents the percentage gap to "Opt." and column "Time" shows the runtime (in seconds) in which the different algorithms reached their best solution.

Problem	Opt.	GACS	Time	Gap	mGENI	Time	Gap	GENIUS	Time	Gap	Total Runtime
eil51	426	**428.87**	367	**0.67%**	428.98	42	0.70%	429.12	257	0.73%	1238
eil76	538	**548.43**	270	**1.94%**	551.73	155	2.55%	553.28	341	2.84%	1638
kroA100	21282	**21285.4**	417	**0.02%**	**21285.4**	257	**0.02%**	**21285.4**	809	**0.02%**	3058
kroB100	22141	22197.3	517	0.25%	22197.3	1411	0.25%	**22191.3**	1328	**0.23%**	3065
kroC100	20749	20771.3	1100	0.11%	**20750.8**	944	**0.01%**	20852.3	2750	0.50%	3060
kroD100	21294	21337	2194	0.20%	**21307.1**	1395	**0.06%**	21404.1	521	0.52%	3061
kroE100	22068	**22117**	1946	**0.22%**	22139.8	1006	0.33%	22162.7	2327	0.43%	3055
pr76	108159	**108183**	710	**0.02%**	108234	533	0.07%	108589	1678	0.40%	2506
rat99	1211	**1219.86**	2124	**0.73%**	1224.85	577	1.14%	1236.52	1229	2.11%	2997
rd100	7910	7918.94	490	0.11%	**7911.35**	628	**0.02%**	7944.35	2969	0.43%	3045

Table 8: Comparison of GACS, mGENI and GENIUS

Therefore, I ran only GACS and mGENI on larger instances. The next table indicates the positive influence of pheromones on the solution quality of most of the problems, as GACS beats mGENI in 8 out of 10 problems. This time, problems up to a size of 150 were given 3 hours runtime and larger ones 4 hours.

Problem	Opt.	GACS	Time	Gap	mGENI	Time	Gap	Runtime
a280	2579	**2662.63**	11354	**3.24%**	2688.04	7569	4.23%	14400
bier127	118282	**119888**	6734	**1.36%**	120426	5051	1.81%	10800
ch150	6528	**6581.02**	7476	**0.81%**	6588.07	5607	0.92%	10800
kroA150	26524	**26626.2**	6224	**0.39%**	26647.3	4668	0.46%	10800
kroA200	29368	**29698.7**	10606	**1.13%**	29774.8	7071	1.39%	14400
kroB150	26130	**26231**	3010	**0.39%**	26253	2258	0.47%	10800
kroB200	29437	**29648.6**	6454	**0.72%**	29670	4303	0.79%	14400
pr144	58537	**59710.9**	6274	**2.01%**	59788.7	4706	2.14%	10800
tsp225	3916	3981.04	10378	1.66%	**3969.11**	6919	**1.36%**	14400
u159	42080	42600.5	6347	1.24%	**42466.5**	4760	**0.92%**	14400

Table 9: Comparison of GACS and mGENI

These results suggest that, with increasing problem size, the importance of using pheromones grows and GACS delivers better solution quality than mGENI and GENIUS. However, pheromones occupy a lot of resources and GACS therefore takes longer to produce its best solution.

4.2 Tests on the SCP part of the problem

4.2.1 Parameter analysis for SCACS

I tested the parameters' sensitivity of SCACS algorithm on a set of 4 problems from the ORLIB. Problems SCP41, SCP43, and SCP44 are problems with $j = 1000$ columns and $i = 200$ rows. SCP57 consists of $j = 2000$ columns and $i = 200$ rows. Just as in my analysis of GACS, I averaged the costs of 10 different runs for every problem to show the impact of each parameter setting. The heuristic information chosen for these tests is cover costs because I first wanted to determine the ideal ACS parameters before examining heuristic information. The number of iterations I for each problem equaled the number of rows. I used $m_a = 5$ ants. While one parameters was tested, the others were kept constant at $\rho = 0.1$, $\alpha = 0.1$, $\beta = 1$ and $q_0 = 0.9$. The initial pheromone level was set to $\tau_0 = 1/z_{GR}$, where z_{GR} is the cost found by a greedy solution.

Table 10 shows that the influence of the heuristic information η_j on the choice of the next column to be included in the solution is of great importance. In accordance, I set β to 5 for further testing.

	β		
	1	**3**	**5**
SCP41	570.9	491.7	**466.1**
SCP43	732.4	613.4	**590.1**
SCP44	620.5	574.5	**557.4**
SCP57	589.6	509.6	**498.3**

Table 10: Influence of parameter β on SCACS solution quality

Although solutions in table 11 show that, with $q_0 = 0.99$, the algorithm performs best, I fixed q_0 to 0.98. This ensures that a certain amount of diversification is created. Also, the gap between values with $q_0 = 0.99$ and $q_0 = 0.98$ was – with the exception of SCP57 – not very large.

	q_0			
	0.9	**0.95**	**0.98**	**0.99**
SCP41	570.9	506.3	**462.9**	467
SCP43	732.4	672.9	592	**590**
SCP44	620.5	602.8	556.9	**552.9**
SCP57	589.6	371.9	351.7	**332.9**

Table 11: Influence of parameter q_0 on SCACS solution quality

Results for the parameters for local and global pheromone update, ρ and α, in tables 12 and 13 suggest that the SCACS algorithm tends to associate those columns which produce good solution quality with high pheromone levels from the beginning. In order for the algorithm to perform well, these columns should be kept attractive for the further construction process. ρ and α limit the level of evaporation. Therefore, I set $\rho = 0.1$ and $\alpha = 0.2$.

	ρ		
Problem	**0.1**	**0.2**	**0.3**
SCP41	**570.9**	613.1	682.9
SCP43	**732.4**	785.3	830.2
SCP44	**620.5**	698.5	753
SCP57	**589.6**	649.5	672.1

Table 12: Influence of parameter ρ on SCACS solution quality

	α		
	0.1	**0.2**	**0.3**
SCP41	570.9	**504.2**	534
SCP43	732.4	**694.1**	730.1
SCP44	**620.5**	629.8	623.2
SCP57	589.6	**539.1**	543

Table 13: Influence of parameter α on SCACS solution quality

Consequently, the best parameter settings are: $\rho = 0.1$, $\alpha = 0.2$, $\beta = 5$ and $q_0 = 0.98$. I used them in all further applications of SCACS.

4.2.2 Heuristic information in SCACS

After determining the best parameter settings, I compared the three types of heuristic information - column cost, cover costs and M&S cover costs. I ran SCACS only three times for each information type because a strong trend towards M&S cover costs developed immediately. They outperformed column costs and cover costs by far (table 14).

Problem	Column Cost	Cover Cost	M&S Cover Cost
SCP41	1758.67	450.00	**434.67**
SCP43	3214	575.67	**543.33**
SCP44	2818.67	529.67	**501**
SCP57	1340.33	318.33	**308.67**

Table 14: SCACS tests on heuristic information

4.2.3 PRIMAL1 vs. SCACS

PRIMAL1 was tested on the four problems used before to determine the optimal parameter settings and heuristic information as well as on some additional problems. In the following table the values are compared to those produced by SCACS and to optimal solutions. Column "Opt." holds the best known solution from the ORLIB, column "Gap" presents the percentage gap to "Opt." and column "Time" shows the runtime (in seconds) in which the different algorithms reached their best solution.

Problem	Opt.	SCACS	Gap	Time	PRIMAL1	Gap	Time
SCP41	429	**432**	0.70%	310	466	8.62%	15
SCP42	512	**535**	4.49%	1703	556	8.59%	15
SCP43	516	**541**	4.84%	401	561	8.72%	15
SCP44	494	**495**	0.20%	482	538	8.91%	15
SCP45	512	**516**	0.78%	2829	542	5.86%	15
SCP48	492	**542**	10.16%	1771	556	13.01%	15
SCP52	302	**314**	3.97%	3000	335	10.93%	29
SCP54	242	**247**	2.07%	1230	249	2.89%	30
SCP56	213	**221**	3.76%	1798	245	15.02%	49
SCP57	293	**304**	3.75%	25	316	7.85%	30

Table 15: PRIMAL1 vs. SCACS

The results show a clear dominance of SCACS. However, this appears to be quite obvious as PRIMAL1 only runs through the problem six times. Nevertheless, SCACS delivers solutions with significantly higher quality. The gap between SCACS results and the optimum is acceptable in most cases but could be improved (for example with local search) while solution quality of PRIMAL1 is rather poor.

4.3 Tests on the CTP

Unfortunately, unlike the TSP and the SCP, no test problems for the CTP exist. Therefore, I randomly created 5 problem instances with $V \subset W = \{1000\}$ - CTP1k1 to CTP1k5 - and 5 with $V \subset W = \{2000\}$ - CTP2k1 to CTP2k5. The smaller problems were tackled in the following fashion: the size of T was set to 3, 5 and 10 and the size of V to 0.1, 0.15 and 0.2 times $V \subset W = \{2000\}$ and $W = (V \subset W) - (T + V)$ (e.g.: for $T = \{3\}$, 3 instances were run with $V = \{100; 150; 200\}$ and $W = \{897; 847; 797\}$). Sets for the larger instances were chosen in a similar way: again, the size of T was set to 3, 5 and 10 and $W = (V \subset W) - (T + V)$, but now the size of V was 0.06, 0.08 and 0.1 times $V \subset W = \{2000\}$ (e.g.: for $T = \{3\}$, 3 instances were run with $V = \{120; 160; 200\}$ and $W = \{1877; 1837; 1797\}$). 9 different types of each problem were created.

In my first approach, the covering distance c was set to the minimum distance at which the CTP is still feasible. I applied CTACS and H-1-CTP to the problems. The tables below outline the results.

As in previous tables, the cost of a tour written in bold letters represents the best obtained solution for an instance. In all CTP tables, column "Problem" specifies the test instance, "Cover distance" gives the used constant cover distance c, "Tour size" shows the number of tour stops and "Time" the runtime of the individual algorithm. The column "Gap (%)" refers to the difference in percentage between CTACS and H-1-CTP where a negative number implies that CTACS provides a better solution. Finally, "Total runtime" refers to the time allowed for both algorithms on an instance. This is the time needed by CTACS to run through the problem. The number of iterations for the SCACS part of CTACS equalled the size of V while GACS iterations equalled the number of columns produced by SCACS (which refers to the CTACS tour size column in the tables).

Problem	Cover distance	CTACS			H-1-CTP			Gap	Total Runtime
		Tour size	Time	Cost	Tour size	Time	Cost		
1k1 t3 v0.1	17.62	20	397	**434.61**	29	228	454.19	-4.31%	615
1k1 t3 v0.15	14.68	31	615	529.97	40	353	**522.78**	1.37%	726
1k1 t3 v0.2	14.68	30	993	**518.27**	42	571	551.95	-6.10%	1098
1k1 t5 v0.1	17.62	24	547	**444.33**	30	314	458.27	-3.04%	655
1k1 t5 v0.15	14.68	34	860	522.35	42	494	**521.64**	0.14%	1053
1k1 t5 v0.2	14.68	32	1540	**524.71**	46	885	547.26	-4.12%	1828
1k1 t10 v0.1	17.62	28	769	**466.01**	32	442	479.92	-2.90%	933
1k1 t10 v0.15	14.68	33	2178	**522.35**	43	1252	534.07	-2.19%	2640
1k1 t10 v0.2	14.68	32	2636	**524.71**	46	1515	528.01	-0.62%	2754
1k2 t3 v0.1	18.43	20	545	**442.88**	29	313	475.70	-6.90%	615
1k2 t3 v0.15	18.43	22	603	454.04	25	346	**432.99**	4.86%	726
1k2 t3 v0.2	13.33	40	980	**570.67**	45	563	577.09	-1.11%	1098
1k2 t5 v0.1	18.43	27	515	467.00	27	296	**456.65**	2.27%	655
1k2 t5 v0.15	18.43	24	729	**447.44**	27	419	459.71	-2.67%	1053
1k2 t5 v0.2	13.33	46	1402	**585.88**	48	806	613.67	-4.53%	1828
1k2 t10 v0.1	18.43	24	674	**456.85**	29	387	488.33	-6.45%	933
1k2 t10 v0.15	18.43	30	2128	**459.12**	28	1223	481.78	-4.70%	2640
1k2 t10 v0.2	13.33	51	2522	**594.01**	48	1449	613.96	-3.25%	2754
1k3 t3 v0.1	19.33	17	429	426.24	26	411	**425.30**	0.22%	615
1k3 t3 v0.15	13.07	38	529	**515.01**	47	304	567.89	-9.31%	726
1k3 t3 v0.2	12.07	41	1005	**546.89**	49	578	575.15	-4.91%	1098
1k3 t5 v0.1	19.33	20	381	**442.52**	27	219	448.33	-1.30%	655
1k3 t5 v0.15	13.07	39	778	**537.36**	44	447	565.76	-5.02%	1053
1k3 t5 v0.2	12.07	43	1498	580.75	50	861	**572.40**	1.46%	1828
1k3 t10 v0.1	13.68	40	623	**559.33**	44	358	573.25	-2.43%	933
1k3 t10 v0.15	13.07	44	2329	**532.98**	48	1338	577.51	-7.71%	2640
1k3 t10 v0.2	12.07	51	2521	**572.70**	49	1449	573.41	-0.12%	2754
1k4 t3 v0.1	17.86	21	608	**420.02**	28	350	422.16	-0.51%	615
1k4 t3 v0.15	17.77	23	599	437.08	28	344	**415.51**	5.19%	726
1k4 t3 v0.2	9.97	55	994	**695.12**	74	572	711.54	-2.31%	1098
1k4 t5 v0.1	17.86	23	402	**423.97**	31	231	452.30	-6.27%	655
1k4 t5 v0.15	17.77	29	782	**450.80**	32	450	453.62	-0.62%	1053
1k4 t5 v0.2	9.97	64	1630	**710.81**	79	937	745.74	-4.68%	1828
1k4 t10 v0.1	17.86	28	531	**452.78**	37	305	489.56	-7.51%	933
1k4 t10 v0.15	17.77	30	2410	**467.30**	37	1385	488.81	-4.40%	2640
1k4 t10 v0.2	9.97	71	2563	**720.15**	76	1473	751.08	-4.12%	2754
1k5 t3 v0.1	16.91	25	487	**446.49**	34	280	492.27	-9.30%	615
1k5 t3 v0.15	13.25	33	550	**557.78**	45	316	603.10	-7.51%	726
1k5 t3 v0.2	10.58	47	957	**674.86**	64	550	699.83	-3.57%	1098
1k5 t5 v0.1	16.91	32	381	481.97	34	219	**472.67**	1.97%	655
1k5 t5 v0.15	12.10	49	690	**609.90**	55	396	686.71	-11.18%	1053
1k5 t5 v0.2	10.58	58	1352	705.06	64	777	**704.59**	0.07%	1828
1k5 t10 v0.1	16.91	32	358	**480.33**	32	206	525.98	-8.68%	933
1k5 t10 v0.15	12.10	52	1992	**618.09**	56	1145	664.31	-6.96%	2640
1k5 t10 v0.2	10.58	59	2415	**690.10**	61	1388	703.55	-1.91%	2754

Table 16: Results for CTACS and H-1-CTP with $V \subset W = \{1000\}$

	Cover	CTACS			H-1-CTP				Total
Problem	distance	Tour size	Time	Cost	Tour size	Time	Cost	Gap	Runtime
2k1 t3 v0.06	14.43	34	1281	**530.21**	50	736	563.76	-5.95%	2043
2k1 t3 v0.08	13.45	37	1838	**563.27**	49	1056	579.86	-2.86%	2751
2k1 t3 v0.1	13.45	36	2825	**554.55**	53	1624	604.37	-8.24%	3713
2k1 t5 v0.06	14.43	40	1013	**536.54**	48	582	596.99	-10.13%	2095
2k1 t5 v0.08	13.45	37	2720	**540.80**	58	1563	642.61	-15.84%	3560
2k1 t5 v0.1	13.45	41	3421	**551.88**	54	1966	619.56	-10.93%	5150
2k1 t10 v0.06	14.43	41	1460	**531.88**	50	839	592.51	-10.23%	2847
2k1 t10 v0.08	13.45	45	4402	**576.19**	56	2530	618.26	-6.80%	4906
2k1 t10 v0.1	13.45	45	6186	**572.97**	56	3555	614.74	-6.79%	8505
2k2 t3 v0.06	13.04	39	959	582.07	52	551	**578.36**	0.64%	2043
2k2 t3 v0.08	12.61	38	1646	591.72	51	946	**579.50**	2.11%	2751
2k2 t3 v0.1	12.61	39	2538	**575.69**	54	1459	622.59	-7.53%	3713
2k2 t5 v0.06	13.04	43	1276	**566.35**	48	733	589.17	-3.87%	2095
2k2 t5 v0.08	12.61	47	1907	**571.76**	53	1096	618.83	-7.61%	3560
2k2 t5 v0.1	12.61	50	2748	**577.82**	51	1579	614.78	-6.01%	5150
2k2 t10 v0.06	12.61	49	953	**581.64**	54	548	630.22	-7.71%	2847
2k2 t10 v0.08	12.61	50	2556	**588.46**	55	1469	657.38	-10.48%	4906
2k2 t10 v0.1	12.61	54	7705	**590.58**	59	4428	658.64	-10.33%	8505
2k3 t3 v0.06	15.53	30	1599	**550.55**	42	919	557.59	-1.26%	2043
2k3 t3 v0.08	15.27	32	2210	**539.35**	44	1270	561.51	-3.95%	2751
2k3 t3 v0.1	15.27	32	3249	**534.68**	44	1867	576.24	-7.21%	3713
2k3 t5 v0.06	15.53	40	1107	**551.16**	44	636	567.42	-2.87%	2095
2k3 t5 v0.08	15.27	44	1552	565.26	39	892	**543.66**	3.97%	3560
2k3 t5 v0.1	15.27	43	3565	553.69	39	2049	**536.18**	3.27%	5150
2k3 t10 v0.06	15.27	39	1591	**558.46**	46	914	587.62	-4.96%	2847
2k3 t10 v0.08	15.27	46	2286	563.14	45	1314	**552.39**	1.95%	4906
2k3 t10 v0.1	15.27	45	6104	**550.91**	42	3508	555.63	-0.85%	8505
2k4 t3 v0.06	15.16	32	1540	**542.11**	45	885	570.80	-5.03%	2043
2k4 t3 v0.08	12.87	42	1394	**602.69**	58	801	640.98	-5.98%	2751
2k4 t3 v0.1	12.87	39	2569	**597.27**	57	1476	660.58	-9.58%	3713
2k4 t5 v0.06	15.16	39	1230	**558.60**	46	707	582.00	-4.02%	2095
2k4 t5 v0.08	12.87	46	1640	**596.06**	58	943	637.42	-6.49%	3560
2k4 t5 v0.1	12.87	47	3739	**591.35**	57	2149	640.92	-7.74%	5150
2k4 t10 v0.06	15.16	40	1811	**549.47**	46	1041	581.57	-5.52%	2847
2k4 t10 v0.08	12.87	50	2473	**635.06**	56	1421	650.11	-2.31%	4906
2k4 t10 v0.1	12.87	53	5593	**635.84**	56	3214	662.06	-3.96%	8505
2k5 t3 v0.06	18.52	24	1907	**438.35**	29	1096	474.72	-7.66%	2043
2k5 t3 v0.08	15.85	31	1957	491.78	33	1125	**472.87**	4.00%	2751
2k5 t3 v0.1	15.85	31	3471	499.48	36	1995	**497.77**	0.34%	3713
2k5 t5 v0.06	18.52	29	1516	**417.56**	32	871	481.20	-13.23%	2095
2k5 t5 v0.08	15.85	36	2681	503.65	35	1541	**498.16**	1.10%	3560
2k5 t5 v0.1	15.85	39	4344	**496.49**	38	2497	501.21	-0.94%	5150
2k5 t10 v0.06	18.52	36	2048	**459.55**	38	1177	507.38	-9.43%	2847
2k5 t10 v0.08	15.85	40	3733	**537.09**	45	2145	547.95	-1.98%	4906
2k5 t10 v0.1	15.85	40	7222	**531.04**	42	4151	539.34	-1.54%	8505

Table 17: Results for CTACS and H-1-CTP with $V \subset W = \{2000\}$

The tests confirm that CTACS dominates H-1-CTP in the majority of the problems. The main reason is the far better performance of SCACS compared to PRIMAL1. The importance of the tour construction components GACS and GENIUS in these smaller problems seems to be rather low but, as I have demonstrated earlier (4.1.3), CTACS would also outperform H-1-CTP with larger tour sizes. However, H-1-CTP finds its best solutions faster than CTACS.

In addition, the results show an obvious inverse relationship between cover distance and tour length. Although the minimum distance choice allows a good comparison of the two algorithms, it is not very helpful for comparing the effects of different sizes of T and V (inter-problem comparison).

Therefore I set the covering distance c to 20 for all problems and variants in order to focus on inter-problem comparison and ran both algorithms a second time (tables 18 and 19).

Problem	Cover distance	CTACS Tour size	CTACS Time	CTACS Cost	H-1-CTP Tour size	H-1-CTP Time	H-1-CTP Cost	Gap	Total Runtime
1k1 t3 v0.1	20.00	18	226	**398.70**	26	130	405.93	-1.78%	350
1k1 t3 v0.15	20.00	17	459	408.19	30	264	**396.16**	3.04%	542
1k1 t3 v0.2	20.00	16	740	423.60	26	425	**404.89**	4.62%	818
1k1 t5 v0.1	20.00	20	359	**388.20**	31	206	432.27	-10.20%	430
1k1 t5 v0.15	20.00	19	575	398.62	26	330	**381.07**	4.61%	704
1k1 t5 v0.2	20.00	20	1016	384.34	22	584	**357.91**	7.39%	1206
1k1 t10 v0.1	20.00	21	528	**394.88**	29	303	434.70	-9.16%	641
1k1 t10 v0.15	20.00	23	1664	**398.57**	27	956	430.81	-7.48%	2017
1k1 t10 v0.2	20.00	21	1967	**388.89**	26	1130	427.66	-9.07%	2055
1k2 t3 v0.1	20.00	17	310	**400.42**	29	178	429.22	-6.71%	350
1k2 t3 v0.15	20.00	17	450	**383.62**	25	259	401.39	-4.43%	542
1k2 t3 v0.2	20.00	17	730	**400.02**	27	419	430.56	-7.09%	818
1k2 t5 v0.1	20.00	22	338	406.63	23	194	**394.23**	3.14%	430
1k2 t5 v0.15	20.00	22	487	**384.80**	25	280	394.51	-2.46%	704
1k2 t5 v0.2	20.00	23	925	**380.13**	26	531	393.14	-3.31%	1206
1k2 t10 v0.1	20.00	25	463	**422.38**	27	266	479.50	-11.91%	641
1k2 t10 v0.15	20.00	24	1626	**413.53**	28	934	484.12	-14.58%	2017
1k2 t10 v0.2	20.00	23	1882	**409.14**	27	1081	459.79	-11.02%	2055
1k3 t3 v0.1	20.00	18	244	**420.20**	26	140	420.96	-0.18%	350
1k3 t3 v0.15	20.00	19	395	404.45	21	227	**372.13**	8.68%	542
1k3 t3 v0.2	20.00	16	749	**365.89**	23	430	377.61	-3.11%	818
1k3 t5 v0.1	20.00	22	250	**445.39**	28	144	458.21	-2.80%	430
1k3 t5 v0.15	20.00	20	520	**408.03**	27	299	425.25	-4.05%	704
1k3 t5 v0.2	20.00	22	988	**390.44**	26	568	419.39	-6.90%	1206
1k3 t10 v0.1	20.00	21	428	**439.47**	28	246	451.87	-2.75%	641
1k3 t10 v0.15	20.00	21	1779	**427.10**	27	1022	439.23	-2.76%	2017
1k3 t10 v0.2	20.00	21	1881	**428.65**	25	1081	446.93	-4.09%	2055
1k4 t3 v0.1	20.00	18	346	**410.46**	25	199	416.91	-1.55%	350
1k4 t3 v0.15	20.00	16	447	385.05	24	257	**375.63**	2.51%	542
1k4 t3 v0.2	20.00	16	741	**386.62**	26	426	412.42	-6.26%	818
1k4 t5 v0.1	20.00	20	264	**414.46**	24	152	422.58	-1.92%	430
1k4 t5 v0.15	20.00	21	523	**396.76**	24	301	409.72	-3.16%	704
1k4 t5 v0.2	20.00	19	1075	**387.91**	31	618	406.17	-4.50%	1206
1k4 t10 v0.1	20.00	23	365	**401.88**	29	210	467.82	-14.09%	641
1k4 t10 v0.15	20.00	21	1841	**406.81**	29	1058	471.92	-13.80%	2017
1k4 t10 v0.2	20.00	24	1913	**403.19**	31	1099	469.00	-14.03%	2055
1k5 t3 v0.1	20.00	16	277	**387.98**	25	159	414.95	-6.50%	350
1k5 t3 v0.15	20.00	19	411	**383.40**	25	236	411.68	-6.87%	542
1k5 t3 v0.2	20.00	18	713	**378.11**	27	410	397.18	-4.80%	818
1k5 t5 v0.1	20.00	21	250	**411.68**	24	144	418.66	-1.67%	430
1k5 t5 v0.15	20.00	23	461	**409.53**	27	265	421.78	-2.90%	704
1k5 t5 v0.2	20.00	24	892	**415.53**	24	513	419.66	-0.98%	1206
1k5 t10 v0.1	20.00	27	246	**441.38**	24	141	456.33	-3.28%	641
1k5 t10 v0.15	20.00	25	1522	**431.64**	25	874	462.66	-6.71%	2017
1k5 t10 v0.2	20.00	27	1802	**432.00**	24	1035	445.84	-3.10%	2055

Table 18: Results for CTACS and H-1-CTP with $V \subset W = \{1000\}$ and $c = 20$

Problem	Cover distance	CTACS			H-1-CTP				Total Runtime
		Tour size		Cost	Tour size		Cost	Gap	
2k1 t3 v0.06	20.00	19	1101	**433.79**	36	633	440.37	-1.49%	1162
2k1 t3 v0.08	20.00	18	1931	**436.13**	35	1110	476.52	-8.48%	2055
2k1 t3 v0.1	20.00	17	2662	**426.40**	34	1530	470.88	-9.45%	2766
2k1 t5 v0.06	20.00	22	1369	**422.16**	41	787	490.61	-13.95%	1376
2k1 t5 v0.08	20.00	25	2006	**452.42**	45	1153	474.60	-4.67%	2379
2k1 t5 v0.1	20.00	24	3026	**421.89**	45	1739	463.98	-9.07%	3397
2k1 t10 v0.06	20.00	28	1782	**426.23**	31	1024	449.44	-5.16%	1956
2k1 t10 v0.08	20.00	27	3245	**430.97**	35	1865	455.13	-5.31%	3748
2k1 t10 v0.1	20.00	27	6036	**415.92**	42	3469	447.29	-7.01%	6346
2k2 t3 v0.06	20.00	17	1109	**411.80**	26	637	439.58	-6.32%	1162
2k2 t3 v0.08	20.00	17	2021	429.03	26	1161	**421.19**	1.86%	2055
2k2 t3 v0.1	20.00	16	2731	**405.74**	25	1570	420.86	-3.59%	2766
2k2 t5 v0.06	20.00	23	1227	**390.62**	25	705	444.22	-12.07%	1376
2k2 t5 v0.08	20.00	26	2194	**392.84**	25	1261	418.29	-6.09%	2379
2k2 t5 v0.1	20.00	25	3210	**369.51**	26	1845	399.29	-7.46%	3397
2k2 t10 v0.06	20.00	24	1782	**424.77**	27	1024	454.22	-6.48%	1956
2k2 t10 v0.08	20.00	25	3538	**408.44**	29	2033	462.29	-11.65%	3748
2k2 t10 v0.1	20.00	27	6092	**409.89**	33	3501	466.65	-12.16%	6346
2k3 t3 v0.06	20.00	19	1031	**460.90**	34	593	463.89	-0.65%	1162
2k3 t3 v0.08	20.00	18	2020	460.70	34	1161	**443.70**	3.83%	2055
2k3 t3 v0.1	20.00	19	2729	**429.96**	29	1568	459.56	-6.44%	2766
2k3 t5 v0.06	20.00	26	1726	**460.08**	32	992	480.46	-4.24%	1376
2k3 t5 v0.08	20.00	38	1372	**476.50**	32	789	489.39	-2.63%	2379
2k3 t5 v0.1	20.00	29	3198	**427.46**	34	1838	457.84	-6.64%	3397
2k3 t10 v0.06	20.00	28	1403	**469.12**	35	806	474.03	-1.04%	1956
2k3 t10 v0.08	20.00	30	3090	**447.69**	37	1776	463.80	-3.47%	3748
2k3 t10 v0.1	20.00	30	6029	**429.75**	35	3465	442.29	-2.84%	6346
2k4 t3 v0.06	20.00	18	1129	**411.23**	31	649	481.11	-14.53%	1162
2k4 t3 v0.08	20.00	18	2046	**432.81**	34	1176	438.84	-1.37%	2055
2k4 t3 v0.1	20.00	18	2667	**398.49**	37	1533	435.26	-8.45%	2766
2k4 t5 v0.06	20.00	25	971	**430.93**	32	558	490.87	-12.21%	1376
2k4 t5 v0.08	20.00	27	1912	**418.28**	37	1099	465.66	-10.17%	2379
2k4 t5 v0.1	20.00	28	2924	**425.08**	38	1680	486.61	-12.64%	3397
2k4 t10 v0.06	20.00	31	1198	**453.34**	32	689	480.82	-5.72%	1956
2k4 t10 v0.08	20.00	29	3109	**430.72**	33	1787	467.03	-7.77%	3748
2k4 t10 v0.1	20.00	28	6081	**438.90**	30	3495	466.32	-5.88%	6346
2k5 t3 v0.06	20.00	19	1111	440.28	27	639	**436.58**	0.85%	1162
2k5 t3 v0.08	20.00	18	2036	**398.98**	28	1170	437.11	-8.72%	2055
2k5 t3 v0.1	20.00	17	2721	**383.14**	27	1564	417.45	-8.22%	2766
2k5 t5 v0.06	20.00	24	1209	**390.05**	25	695	433.07	-9.93%	1376
2k5 t5 v0.08	20.00	22	2370	**382.82**	28	1362	429.42	-10.85%	2379
2k5 t5 v0.1	20.00	25	3326	**368.11**	28	1911	424.98	-13.38%	3397
2k5 t10 v0.06	20.00	20	1685	**402.43**	33	968	453.36	-11.23%	1956
2k5 t10 v0.08	20.00	26	3035	**402.44**	34	1744	430.45	-6.51%	3748
2k5 t10 v0.1	20.00	28	6214	422.27	33	3571	**402.18**	4.99%	6346

Table 19: Results for CTACS and H-1-CTP with $V \subset W = \{2000\}$ and $c = 20$

Again, CTACS dominates H-1-CTP but the interesting finding when dealing with a constant covering distance was that a greater pool of possible tour stops can lead to better solutions, even if more stops are made. On the other hand the solution quality may suffer if vertices of set T that would not have been chosen had they been members of set V, increase the length of the tour.

5 Conclusion

The CTP is an important problem with highly relevant issues in the public and private sector. It is an NP-hard combinatorial optimization problem. The objective is to determine a minimum length tour over a subset of vertices while covering another set of vertices. Good examples for CTP applications are the design of bi-level transportation networks or the deployment of a mobile medical facility in developing countries.

This thesis shows two methods for solving the CTP. I divided the problem into two other optimization problems, namely the TSP and the SCP, both also NP-hard problems. The objective of the TSP is to construct the shortest tour over a set of vertices and return to the starting point while in the SCP a set of columns that covers a set of rows at minimum cost has to be determined. I combined solution approaches for them in order to solve the CTP. I created the following algorithms with C++ programming language:

The first approach, an approximation algorithm called H-1-CTP created by Gendreau et al. [14], delivers good solutions but seems to get caught in local optima as it only considers the best insertions. Nevertheless, especially the GENIUS heuristic, responsible for constructing the TSP tour, leads to better solutions than other heuristics due to a random choice of the next vertex to be inserted.

I created the second approach called CTACS, a combination of two ACS algorithms for the TSP and the SCP, myself. This method outperforms the first one in over 82 percent of the instances because it also allows inferior steps during construction with a certain amount of probability.

The main barrier of evaluating the solution quality of both algorithms was that no model problems exist for comparison. Consequently, I first tested the individual methods used to solve the TSP (namely GENI, GENIUS and GACS) and the SCP (namely PRIMAL1 and SCACS) and compared them to the best known solutions obtained from the TSPLIB [24] and ORLIB [4]. All solution methods using ACS and GENI as well as US returned very good results so I assume that, at least for CTACS, the solution quality, when dealing with CTPs, will be able to bear comparison with other approaches to come.

However, the time needed to generate solutions for the CTP must be seen critically and can surely be improved with more C++ experience. In addition, the introduction of local search especially to SCACS should improve solution quality a bit more, leading to optimal results.

Appendix A

Appendix A shows detailed GACS results for every one of the four problems analyzed (KubLE25, Berlin52, st70 and pr107). Four tables for each problem for parameters q_0, γ, α and ρ with the results for ten runs on each parameter setting follow. The values which are lower or equal to the best average cost of each parameter are written in bold letters.

KubLE25:

KubLE25 Parameter q_0				
Runs	**0.9**	**0.95**	**0.98**	**1**
1	**395.64**	**395.64**	**395.64**	**395.64**
2	**395.64**	**395.64**	**395.64**	**395.64**
3	**395.64**	**395.64**	**395.64**	**395.64**
4	**395.64**	**395.64**	**395.64**	**395.64**
5	**395.64**	**395.64**	**395.64**	**395.64**
6	**395.64**	**395.64**	**395.64**	**395.64**
7	**395.64**	**395.64**	**395.64**	**395.64**
8	**395.64**	**395.64**	**395.64**	**395.64**
9	**395.64**	**395.64**	**395.64**	**395.64**
10	**395.64**	**395.64**	**395.64**	**395.64**
Average	**395.64**	**395.64**	**395.64**	**395.64**
Minimum	395.64	395.64	395.64	395.64
Maximum	395.64	395.64	395.64	395.64

Table 20: Tests on q_0 for KubLE25 (GACS)

KubLE25 Parameter γ						
Runs	**0**	**0.25**	**0.5**	**0.75**	**1**	**5**
1	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
2	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
3	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
4	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
5	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
6	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
7	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
8	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
9	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
10	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
Average	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
Minimum	395.64	395.64	395.64	395.64	395.64	395.64
Maximum	395.64	395.64	395.64	395.64	395.64	395.64

Table 21: Tests on γ for KubLE25 (GACS)

	KubLE25 Parameter α						
Runs	**0**	**0.1**	**0.25**	**0.5**	**0.75**	**0.9**	**1**
1	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
2	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
3	398.23	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
4	400.57	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
5	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
6	401.11	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
7	400.98	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
8	401.50	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
9	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
10	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
Average	398.06	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
Minimum	395.64	395.64	395.64	395.64	395.64	395.64	395.64
Maximum	401.50	395.64	395.64	395.64	395.64	395.64	395.64

Table 22: Tests on α for KubLE25 (GACS)

	KubLE25 Parameter ρ						
Runs	**0**	**0.1**	**0.25**	**0.5**	**0.75**	**0.9**	**1**
1	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
2	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
3	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
4	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
5	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
6	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
7	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
8	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
9	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
10	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
Average	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**	**395.64**
Minimum	395.64	395.64	395.64	395.64	395.64	395.64	395.64
Maximum	395.64	395.64	395.64	395.64	395.64	395.64	395.64

Table 23: Tests on ρ for KubLE25 (GACS)

Berlin52:

	Berlin52 Parameter q_0			
Runs	**0.9**	**0.95**	**0.98**	**1**
1	**7544.37**	7572.85	**7544.37**	**7544.37**
2	7549.89	**7544.37**	**7544.37**	**7544.37**
3	7548.99	**7544.37**	**7544.37**	**7544.37**
4	**7544.37**	7549.89	**7544.37**	**7544.37**
5	**7544.37**	**7544.37**	**7544.37**	**7544.37**
6	7544.66	**7544.37**	**7544.37**	**7544.37**
7	7563.69	**7544.37**	**7544.37**	**7544.37**
8	**7544.37**	**7544.37**	7544.66	7550.19
9	7550.19	**7544.37**	**7544.37**	**7544.37**
10	7567.62	7544.66	**7544.37**	**7544.37**
Average	7550.25	7547.80	**7544.40**	7544.95
Minimum	7544.37	7544.37	7544.37	7544.37
Maximum	7567.62	7572.85	7544.66	7550.19

Table 24: Tests on q_0 for Berlin52 (GACS)

	Berlin52 Parameter γ					
Runs	**0**	**0.25**	**0.5**	**0.75**	**1**	**5**
1	**7544.37**	**7544.37**	7572.85	**7544.37**	**7544.37**	**7544.37**
2	**7544.66**	**7544.37**	**7544.37**	**7544.37**	7583.09	**7544.37**
3	7585.20	7571.62	**7544.37**	7571.62	7567.33	**7544.37**
4	**7544.37**	7565.87	7549.89	**7544.37**	**7544.37**	7576.25
5	**7544.37**	**7544.37**	**7544.37**	**7544.37**	**7544.37**	7585.20
6	**7544.37**	7549.29	**7544.37**	**7544.37**	**7544.37**	**7544.37**
7	7567.33	7567.33	**7544.37**	**7544.66**	7597.07	**7544.37**
8	7567.33	7548.99	**7544.37**	**7544.37**	**7544.37**	7555.40
9	7565.87	7567.33	**7544.37**	**7544.66**	7566.17	**7544.37**
10	7563.69	7595.58	**7544.66**	7582.95	**7544.37**	7567.33
Average	7557.16	7559.91	**7547.80**	7551.01	7557.99	7555.04
Minimum	7544.37	7544.37	7544.37	7544.37	7544.37	7544.37
Maximum	7585.2	7595.58	7572.85	7582.95	7597.07	7585.2

Table 25: Tests on γ for Berlin52 (GACS)

	Berlin52 Parameter α						
Runs	**0**	**0.1**	**0.25**	**0.5**	**0.75**	**0.9**	**1**
1	**7544.37**	**7544.37**	**7544.37**	7572.85	**7544.37**	**7544.37**	7624.60
2	7544.66	**7544.37**	**7544.37**	**7544.37**	7567.33	**7544.37**	7651.32
3	**7544.37**	**7544.37**	**7544.37**	**7544.37**	**7544.37**	7549.89	7616.25
4	**7544.37**	**7544.37**	**7544.37**	7549.89	**7544.37**	**7544.37**	7567.62
5	7549.89	7567.33	**7544.37**	**7544.37**	**7544.37**	7548.99	7791.17
6	**7544.37**	7571.92	**7544.37**	**7544.37**	**7544.37**	**7544.37**	7670.96
7	**7544.37**	**7544.37**	**7544.37**	**7544.37**	**7544.37**	**7544.37**	7648.29
8	**7544.37**	7565.87	**7544.37**	**7544.37**	7629.59	**7544.37**	7687.40
9	7565.87	**7544.37**	**7544.37**	**7544.37**	**7544.37**	7620.37	7605.12
10	7571.62	**7544.37**	**7544.37**	7544.66	7544.66	7582.66	7567.33
Average	7549.83	7551.57	**7544.37**	7547.80	7555.22	7556.81	7643.01
Minimum	7544.37	7544.37	7544.37	7544.37	7544.37	7544.37	7567.33
Maximum	7571.62	7571.92	7544.37	7572.85	7629.59	7620.37	7791.17

Table 26: Tests on α for Berlin52 (GACS)

	Berlin52 Parameter ρ						
Runs	**0**	**0.1**	**0.25**	**0.5**	**0.75**	**0.9**	**1**
1	7567.33	7566.83	7568.32	7572.85	**7544.37**	**7544.37**	**7544.37**
2	7566.83	**7544.37**	**7544.37**	**7544.37**	**7544.37**	**7544.37**	**7544.37**
3	7616.03	**7544.37**	7548.99	**7544.37**	**7544.37**	**7544.37**	**7544.37**
4	**7544.37**	**7544.37**	**7544.37**	7549.89	**7544.37**	7544.66	**7544.37**
5	**7544.37**	7555.40	**7544.37**	**7544.37**	7555.40	**7544.37**	**7544.37**
6	7565.87	7567.33	7548.99	**7544.37**	7548.99	**7544.37**	**7544.37**
7	**7544.37**	**7544.37**	**7544.37**	**7544.37**	**7544.37**	**7544.37**	7596.54
8	**7544.37**	**7544.37**	7571.62	**7544.37**	**7544.37**	**7544.37**	7555.40
9	7567.33	**7544.37**	7548.99	**7544.37**	**7544.37**	**7544.37**	7571.95
10	7555.70	**7544.37**	7555.40	7544.66	7548.99	7544.66	7555.40
Average	7561.66	7550.02	7551.98	7547.80	7546.40	**7544.43**	7554.55
Minimum	7544.37	7544.37	7544.37	7544.37	7544.37	7544.37	7544.37
Maximum	7616.03	7567.33	7571.62	7572.85	7555.40	7544.66	7596.54

Table 27: Tests on ρ for Berlin52 (GACS)

st70:

	st70 Parameter q_0			
Runs	**0.9**	**0.95**	**0.98**	**1**
1	677.83	677.52	**677.11**	**677.11**
2	680.15	681.38	677.19	677.19
3	679.86	678.51	677.19	677.20
4	680.75	677.19	677.19	677.19
5	680.99	677.19	677.20	**677.11**
6	681.00	677.88	677.19	677.19
7	682.66	682.41	**677.11**	677.19
8	678.51	677.91	677.19	677.19
9	680.66	677.19	**677.11**	**677.11**
10	677.19	677.20	677.88	677.19
Average	679.96	678.44	677.24	**677.17**
Minimum	677.19	677.19	677.11	677.11
Maximum	682.66	682.41	677.88	677.20

Table 28: Tests on q_0 for st70 (GACS)

	st70 Parameter γ					
Runs	**0**	**0.25**	**0.5**	**0.75**	**1**	**5**
1	**677.11**	**677.11**	682.15	**677.19**	**677.19**	682.90
2	681.00	**677.19**	**677.19**	**677.19**	**677.20**	679.82
3	**677.19**	**677.11**	**677.19**	**677.20**	**677.11**	**677.11**
4	677.52	**677.19**	678.99	**677.19**	678.51	682.00
5	**677.19**	**677.11**	**677.19**	679.07	**677.19**	682.77
6	678.51	677.88	678.26	677.91	678.51	677.82
7	679.08	**677.11**	679.80	678.54	677.82	**677.11**
8	677.53	677.88	**677.11**	682.66	**677.19**	682.58
9	**677.11**	**677.19**	677.52	682.18	**677.19**	680.62
10	682.54	**677.19**	678.12	**677.19**	**677.19**	682.58
Average	678.48	**677.30**	678.35	678.63	677.51	680.53
Minimum	677.11	677.11	677.11	677.19	677.11	677.11
Maximum	682.54	677.88	682.15	682.66	678.51	682.90

Table 29: Tests on γ for st70 (GACS)

	st70 Parameter α						
Runs	**0**	**0.1**	**0.25**	**0.5**	**0.75**	**0.9**	**1**
1	**677.19**	682.58	679.82	**677.20**	678.86	**677.82**	688.91
2	681.83	681.28	**677.19**	682.66	**677.52**	682.77	685.12
3	681.18	**677.19**	**677.19**	**677.19**	682.58	**678.20**	688.29
4	681.83	683.34	**677.11**	**677.79**	**677.83**	**677.19**	687.25
5	682.58	**677.79**	**678.26**	681.82	**677.87**	**677.11**	684.42
6	680.48	682.77	**677.19**	**677.11**	**677.88**	**677.44**	685.50
7	681.83	**677.53**	681.66	**677.19**	682.77	683.08	686.15
8	681.92	**677.44**	**677.20**	**677.11**	681.29	680.01	688.65
9	682.58	679.49	682.58	**677.88**	681.26	680.38	685.64
10	682.66	**677.87**	**677.19**	682.19	**678.03**	681.19	687.06
Average	681.41	679.73	**678.54**	678.81	679.59	679.52	686.70
Minimum	677.19	677.19	677.11	677.11	677.52	677.11	684.42
Maximum	682.66	683.34	682.58	682.66	682.77	683.08	688.91

Table 30: Tests on α for st70 (GACS)

	st70 Parameter ρ						
Runs	**0**	**0.1**	**0.25**	**0.5**	**0.75**	**0.9**	**1**
1	**677.11**	695.07	677.52	682.15	**677.19**	682.15	682.58
2	682.58	692.37	681.92	**677.19**	**677.11**	681.29	**677.11**
3	**677.11**	700.26	**677.19**	**677.19**	**677.19**	**677.11**	682.58
4	**677.11**	695.74	**677.11**	678.99	677.83	**677.11**	677.82
5	**677.20**	691.06	**677.19**	**677.19**	**677.20**	678.62	**677.19**
6	678.12	692.35	677.88	678.26	**677.19**	**677.19**	**677.11**
7	**677.20**	689.99	**677.19**	679.80	**677.20**	**677.19**	**677.11**
8	**677.11**	696.37	**677.11**	**677.11**	**677.19**	682.01	680.31
9	**677.11**	691.26	**677.11**	677.52	677.91	**677.11**	681.97
10	682.58	688.39	682.66	678.12	**677.19**	**677.11**	682.58
Average	678.32	693.29	678.29	678.35	**677.32**	678.69	679.64
Minimum	677.11	688.39	677.11	677.11	677.11	677.11	677.11
Maximum	682.58	700.26	682.66	682.15	677.91	682.15	682.58

Table 31: Tests on ρ for st70 (GACS)

pr107:

	pr107 Parameter q_0			
Runs	**0.9**	**0.95**	**0.98**	**1**
1	44589.00	**44387.80**	**44417.40**	**44301.70**
2	44555.80	**44337.40**	44429.30	**44337.40**
3	44746.90	**44393.80**	**44379.20**	44484.30
4	44545.40	44433.50	**44387.80**	**44324.80**
5	44681.50	44440.70	44442.00	44516.20
6	44575.20	44516.80	44438.10	**44346.20**
7	44486.70	44434.00	**44390.30**	44516.20
8	44551.90	**44379.70**	44498.80	44507.00
9	44742.60	44436.20	44432.80	44537.80
10	44455.20	44498.80	44442.00	**44390.30**
Average	44593.02	44425.87	**44425.77**	44426.19
Minimum	44455.20	44337.40	44379.20	44301.70
Maximum	44746.90	44516.80	44498.80	44537.80

Table 32: Tests on q_0 for pr107 (GACS)

	pr107 Parameter γ					
Runs	**0**	**0.25**	**0.5**	**0.75**	**1**	**5**
1	**44352.1**	44436.2	**44387.8**	**44406.4**	**44396.6**	44690.4
2	**44418.8**	44481.2	**44337.4**	**44403.6**	44656.6	44443.1
3	**44390.3**	**44390.3**	**44393.8**	44486.6	44637.8	44733.7
4	**44397.7**	44516.2	44433.5	44539.4	44528.5	44478.8
5	44480.1	44521.2	44440.7	44558.7	44436.2	44572.4
6	44498.4	**44385.2**	44516.8	44429.3	44553.2	44697.6
7	**44301.7**	44512.2	44434	44571.3	44566	44583.8
8	44473.2	**44363.8**	**44379.7**	**44381.7**	**44396.6**	44500.8
9	44454	44436.2	44436.2	**44404.8**	**44301.7**	44558.9
10	44503.3	44674.5	44498.8	44491.6	**44346.2**	44699.2
Average	44426.96	44471.7	**44425.87**	44467.34	44481.94	44595.87
Minimum	44301.7	44363.8	44337.4	44381.7	44301.7	44443.1
Maximum	44503.3	44674.5	44516.8	44571.3	44656.6	44733.7

Table 33: Tests on γ for pr107 (GACS)

	pr107 Parameter α						
Runs	**0**	**0.1**	**0.25**	**0.5**	**0.75**	**0.9**	**1**
1	44710.80	44750.50	44536.00	**44387.80**	**44324.80**	44532.80	44968.40
2	44589.40	44742.60	44530.10	**44337.40**	44522.30	44506.80	45037.10
3	44577.70	44491.10	**44381.70**	**44393.80**	44722.20	44496.80	45297.10
4	44487.80	44482.80	44599.60	44433.50	44560.00	**44337.40**	45218.40
5	44557.70	44654.60	**44301.70**	44440.70	44522.00	44553.40	45253.70
6	44838.00	44598.70	44520.70	44516.80	44455.20	44601.30	45210.30
7	44850.80	44923.30	44459.40	44434.00	44532.10	44487.60	45272.10
8	44656.60	44440.00	44536.00	**44379.70**	44688.20	44611.40	45252.30
9	44459.20	44696.80	44490.90	44436.20	44429.60	44479.70	45384.90
10	**44404.80**	44782.90	44475.20	44498.80	44520.70	44681.80	44833.70
Average	44613.28	44656.33	44483.13	**44425.87**	44527.71	44528.90	45172.80
Minimum	44404.80	44440.00	44301.70	44337.40	44324.80	44337.40	44833.70
Maximum	44850.80	44923.30	44599.60	44516.80	44722.20	44681.80	45384.90

Table 34: Tests on α for pr107 (GACS)

	pr107 Parameter ρ						
Runs	**0**	**0.1**	**0.25**	**0.5**	**0.75**	**0.9**	**1**
1	44700.6	44504.9	44647.2	**44387.8**	44562.5	44459.4	44638.8
2	44832.7	44486	44503.3	**44337.4**	**44324.8**	**44337.4**	44518.2
3	44657.1	**44376.4**	44491.7	**44393.8**	44509.3	44539.4	**44301.7**
4	44718.8	44430.1	44524.6	44433.5	44535.6	44440.7	44471.1
5	44603.7	44635.2	44455.2	44440.7	44440.2	44470.3	**44396.6**
6	44783.7	44648.1	44574.4	44516.8	**44346.2**	44516.2	44487.1
7	44551.8	44453.1	44491.1	44434	**44400.5**	**44404.8**	44608.7
8	44834.1	44575	44553.2	**44379.7**	44618.6	44561.3	44507
9	44816.8	**44337.4**	**44375.2**	44436.2	44545.9	44553.4	44465.9
10	44778.4	**44376.4**	44545.8	44498.8	44459.2	44454.5	**44379.7**
Average	44727.77	44482.26	44516.17	**44425.87**	44474.28	44473.74	44477.48
Minimum	44551.8	44337.4	44375.2	44337.4	44324.8	44337.4	44301.7
Maximum	44834.1	44648.1	44647.2	44516.8	44618.6	44561.3	44638.8

Table 35: Tests on ρ for pr107 (GACS)

Appendix B

Appendix B lists SCACS results for the four problems analyzed (SCP41, SCP43, SCP44, SCP57). Four tables for each problem for parameters q_0, γ, α and ρ with the results of ten runs on each parameter setting follow. The values lower or equal to the best average cost of each parameter are written in bold letters.

SCP41:

SCP41 Parameter α			
Runs	**0.1**	**0.2**	**0.3**
1	585	537	**467**
2	520	512	577
3	541	**503**	586
4	689	**477**	516
5	528	**481**	537
6	508	**487**	**500**
7	539	507	598
8	**490**	536	531
9	706	**464**	554
10	603	538	**474**
Average	570.9	**504.2**	534
Minimum	490	464	467
Maximum	706	538	598

Table 36: Tests on α for SCP41 (SCACS)

SCP41 Parameter β			
Runs	**1**	**3**	**5**
1	585	**455**	491
2	520	482	**448**
3	541	523	468
4	689	472	**451**
5	528	481	**466**
6	508	552	**458**
7	539	475	474
8	490	505	470
9	706	**457**	472
10	603	515	**463**
Average	570.9	491.7	**466.1**
Minimum	490	455	448
Maximum	706	552	491

Table 37: Tests on β for SCP41 (SCACS)

SCP41 Parameter ρ			
Runs	**0.1**	**0.2**	**0.3**
1	585	**511**	643
2	**520**	**516**	799
3	**541**	719	**460**
4	689	653	870
5	**528**	**551**	766
6	**508**	587	600
7	**539**	760	885
8	**490**	687	635
9	706	598	662
10	603	**549**	**509**
Average	**570.9**	613.1	682.9
Minimum	490	511	460
Maximum	706	760	885

Table 38: Tests on ρ for SCP41 (SCACS)

SCP41 Parameter q_0				
Runs	**0.9**	**0.95**	**0.98**	**0.99**
1	585	**457**	**453**	**452**
2	520	508	475	485
3	541	546	463	467
4	689	514	**447**	**450**
5	528	659	480	467
6	508	**445**	**462**	**458**
7	539	465	**452**	470
8	490	495	**457**	470
9	706	500	480	**458**
10	603	474	**460**	493
Average	570.9	506.3	**462.9**	467
Minimum	490	445	447	450
Maximum	706	659	480	493

Table 39: Tests on q_0 for SCP41 (SCACS)

SCP43:

SCP43 Parameter α			
Runs	**Cost**	**Cost**	**Cost**
1	709	749	884
2	703	**616**	902
3	**662**	**661**	**634**
4	777	789	722
5	977	716	734
6	726	799	**614**
7	769	**641**	707
8	745	**620**	**657**
9	**583**	**650**	708
10	**673**	700	739
Average	732.4	**694.1**	730.1
Minimum	583	616	614
Maximum	977	799	902

Table 40: Tests on α for SCP43 (SCACS)

SCP43 Parameter β			
Runs	**1**	**3**	**5**
1	709	**564**	**566**
2	703	603	**577**
3	662	**585**	606
4	777	660	**584**
5	977	620	**575**
6	726	633	594
7	769	**583**	602
8	745	630	608
9	**583**	**581**	**588**
10	673	675	601
Average	732.4	613.4	**590.1**
Minimum	583	564	566
Maximum	977	675	608

Table 41: Tests on β for SCP43 (SCACS)

SCP43 Parameter ρ			
Runs	**0.1**	**0.2**	**0.3**
1	**709**	**640**	770
2	**703**	**714**	769
3	**662**	737	889
4	777	**700**	1060
5	977	976	852
6	**726**	823	877
7	769	865	**662**
8	745	894	**615**
9	**583**	**661**	1014
10	**673**	843	794
Average	**732.4**	785.3	830.2
Minimum	583	640	615
Maximum	977	976	1060

Table 42: Tests on ρ for SCP43 (SCACS)

SCP43 Parameter q_0				
Runs	**0.9**	**0.95**	**0.98**	**0.99**
1	709	669	**574**	**589**
2	703	684	617	**578**
3	662	641	592	**589**
4	777	611	624	595
5	977	645	**583**	614
6	726	689	**566**	**584**
7	769	**589**	**578**	**561**
8	745	765	606	612
9	**583**	741	620	609
10	673	695	**560**	**569**
Average	732.4	672.9	592	**590**
Minimum	583	589	560	561
Maximum	977	765	624	614

Table 43: Tests on q_0 for SCP43 (SCACS)

SCP44:

SCP44 Parameter α			
Runs	**0.1**	**0.2**	**0.3**
1	**595**	**580**	707
2	**613**	625	681
3	**562**	650	**614**
4	623	698	**616**
5	**584**	**615**	656
6	696	675	622
7	**540**	631	624
8	732	**601**	**603**
9	713	640	**561**
10	**547**	**583**	**548**
Average	**620.5**	629.8	623.2
Minimum	540	580	548
Maximum	732	698	707

Table 44: Tests on α for SCP44 (SCACS)

SCP44 Parameter β			
Runs	**1**	**3**	**5**
1	595	561	563
2	613	576	**546**
3	562	572	**542**
4	623	562	573
5	584	578	**554**
6	696	601	**548**
7	**540**	565	572
8	732	**553**	568
9	713	565	**538**
10	**547**	612	570
Average	620.5	574.5	**557.4**
Minimum	540	553	538
Maximum	732	612	573

Table 45: Tests on β for SCP44 (SCACS)

SCP44 Parameter ρ			
Runs	**0.1**	**0.2**	**0.3**
1	**595**	698	777
2	**613**	**593**	845
3	**562**	674	932
4	623	665	750
5	**584**	737	636
6	696	667	771
7	**540**	858	693
8	732	762	675
9	713	733	755
10	**547**	**598**	696
Average	**620.5**	698.5	753
Minimum	540	593	636
Maximum	732	858	932

Table 46: Tests on ρ for SCP44 (SCACS)

SCP44 Parameter q_0				
Runs	**0.9**	**0.95**	**0.98**	**0.99**
1	595	**551**	586	**524**
2	613	736	572	**536**
3	562	611	**552**	**529**
4	623	620	**521**	604
5	584	589	598	576
6	696	585	**542**	**542**
7	**540**	613	**548**	**545**
8	732	588	589	586
9	713	582	**527**	559
10	**547**	553	**534**	**528**
Average	620.5	602.8	556.9	**552.9**
Minimum	540	551	521	524
Maximum	732	736	598	604

Table 47: Tests on q_0 for SCP44 (SCACS)

SCP57:

SCP57 Parameter α			
Runs	**0.1**	**0.2**	**0.3**
1	612	**427**	570
2	**336**	**457**	**494**
3	626	625	596
4	813	624	575
5	742	615	551
6	**508**	601	683
7	638	548	**431**
8	540	**518**	**538**
9	590	589	**463**
10	**491**	**387**	**529**
Average	589.6	**539.1**	543
Minimum	336	387	431
Maximum	813	625	683

Table 48: Tests on α for SCP57 (SCACS)

SCP57 Parameter β			
Runs	**1**	**3**	**5**
1	612	**438**	521
2	**336**	532	534
3	626	529	536
4	813	538	536
5	742	586	**351**
6	508	**353**	**429**
7	638	627	544
8	540	**429**	550
9	590	616	521
10	**491**	**448**	**461**
Average	589.6	509.6	**498.3**
Minimum	336	353	351
Maximum	813	627	550

Table 48: Tests on β for SCP57 (SCACS)

SCP57 Parameter ρ			
Runs	**0.1**	**0.2**	**0.3**
1	612	686	787
2	**336**	**572**	811
3	626	**548**	**583**
4	813	806	731
5	742	**469**	743
6	**508**	**458**	**552**
7	638	627	872
8	**540**	886	608
9	590	638	607
10	**491**	805	**427**
Average	**589.6**	649.5	672.1
Minimum	336	458	427
Maximum	813	886	872

Table 50: Tests on ρ for SCP57 (SCACS)

SCP57 Parameter q_0				
Runs	**0.9**	**0.95**	**0.98**	**0.99**
1	612	370	355	**327**
2	336	**330**	350	**327**
3	626	357	**327**	**327**
4	813	**329**	369	**328**
5	742	455	**327**	347
6	508	354	358	**327**
7	638	**327**	375	**331**
8	540	363	371	360
9	590	453	358	**328**
10	491	381	**327**	**327**
Average	589.6	371.9	351.7	**332.9**
Minimum	336	327	327	327
Maximum	813	455	375	360

Table 51: Tests on q_0 for SCP57 (SCACS)

Appendix C

C.1 German Abstract

Diese Arbeit beschäftigt sich mit dem Covering Tour Problem (CTP) und verschiedenen heuristischen Lösungsmethoden. Dieses Problem der Tourenplanung zählt zu den kombinatorischen Optimierungsproblemen, welche sehr oft im Bereich der Distributionslogistik international agierender Großunternehmen auftreten und durch deren Lösung man entsprechend Kosten einsparen und Gewinne maximieren kann. Im Zuge der Globalisierung der Weltwirtschaft rückt das Problem der Distributionskosten immer mehr in den Mittelpunkt.

Das CTP kann auf einem ungerichteten Graphen $G = (V \cup W, E)$ definiert werden. $V \cup W$ ist eine Menge von Knoten. $V = \{v_0, ..., v_n\}$ sind jene Knoten, die von der zu konstruierenden Tour besucht werden können. $T \subset V$ ist eine Teilmenge von V und beinhaltet jene Knoten, die von der Tour besucht werden müssen. W ist die Menge jener Knoten, welche von der Tour abgedeckt werden müssen, also in einer vorgegebenen Entfernung zur Tour liegen müssen. Das Kantenset $E = \{(v_i, v_j) : v_i, v_j \in V \cup W, i < j\}$ beinhaltet die Verbindungen zwischen sämtlichen Knoten. Ziel ist es nun, eine möglichst kurze Tour zu finden, die im Punkt v_0 beginnt, alle Knoten aus $T \subset V$ besucht, sämtliche Knoten aus W abdeckt und wieder in v_0 endet.

Um das Problem zu lösen, wurde das CTP gemäß einer bereits angewandten Methode[14] in zwei Subprobleme, nämlich das Traveling Salesman Problem (TSP) und das Set Covering Problem (SCP) unterteilt und diese wurden vorgestellt. Nach einer kurzen Einführung der Ant Colony Optimierung wurden die Algorithmen GENI, GENIUS und GENI Ant Colony System für den TSP Teil und PRIMAL1 sowie ein Set Covering Ant Colony System für den SCP Teil detailliert beschrieben. In weitere Folge wurde erklärt, wie man die Algorithmen kombinieren kann, um das CTP zu lösen.

Sämtliche Algorithmen wurden mit Hilfe der Programmiersprache C++ simuliert und getestet. Zunächst wurden die Algorithmen an Instanzen einer Datenbank getestet und mit bereits vorhandenen Lösungen verglichen, um ihre Funktionalität und Konkurrenzfähigkeit zu überprüfen. Da für das CTP keine

Vergleichsinstanzen vorhanden sind, wurden stochastische Probleme entworfen und mit dem H-1-CTP Algorithmus [14] und der von mir entworfenen Metaheuristik Covering Tour Ant Colony System bestehend aus GENI Ant Colony System und Set Covering Ant Colony System gelöst und die Ergebnisse verglichen, um dann die beiden Lösungsansätze zu bewerten.

C.2 English Abstract

This thesis deals with the Covering Tour Problem (CTP) and different heuristic solution approaches. It can be classified as a combinatorial optimization problem. Logistics and distribution departments of economic global players have to handle this sort of problems to reduce costs and maximize profit. Distribution costs enjoy increasing importance due to the globalization of world economy.

The CTP is defined on a complete undirected graph $G = (V \cup W, E)$ with a set of vertices $V \cup W$ where $V = \{v_0, ..., v_n\}$ is a set of vertices that can be visited, W defines the set of vertices that have to be covered by the tour and $E = \{(v_i, v_j) : v_i, v_j \in V \cup W, i < j\}$ is the set of edges. "Covered by the tour" means that any vertex $v_\ell \in W$ has to lie within a predefined distance of a vertex on the tour. The set V includes the subset T which includes the vertices that have to be visited by the tour. The solution to the CTP is a minimum length tour. The tour starts and ends at the depot and is defined by a certain subset so that all vertices that have to be visited are visited by the tour and all vertices that have to be covered lie within a predetermined distance of a vertex belonging to the tour.

In order to solve the problem, it was classified as a combination of the Traveling Salesman Problem (TSP) and the Set Covering Problem (SCP) and the components were introduced. After a short description of Ant Colony Optimization, algorithms GENI, GENIUS and GENI Ant Colony System for the TSP part and PRIMAL1 as well as Set Covering Ant Colony System for the SCP part were introduced in detail. Then the combinations of these algorithms for solving the CTP were described.

All algorithms were simulated and tested with the help of C++ programming language. First, algorithms were tested individually on instances from data libraries to ensure their functionality and competitiveness. Then stochastic instances were developed for the CTP because no comparable benchmarks exist and the H-1-CTP algorithm as well as the Covering Tour Ant Colony System, that I created myself, were run on these instances and results were compared.

C.3 Curriculum vitae

Personal Details

Name:	Patrick Kubik
Date of Birth:	7 January 1981
Nationality:	Austrian
Marital Status:	Single

Education

University:	**International Business Management** at the University of Vienna (masters degree)
	October 2000 – December 2007
	Fields of specialization: operations management, corporate finance and transportation logistics
High School:	**Bundesgymnasium Stockerau** (1991 – 1999)
	Matura (equivalent of A-Level) passed with **distinction** (grade average 2)

Professional Experience

Since Sept 2005	**CUBE Consult Unternehmensberatungs Gmbh**
	Junior Analyst
July – Aug 2004	**Boehler Uddeholm GmbH Germany, Duesseldorf**
	Internship in the **accounting department** (participation in preparing the quarterly statement and in creating forecasts, corporate group reporting with SAP and Hyperion, cost centre planning) and the **sales department** (sales management and budget planning)
July 2003	**VA Tech Finance GmbH & Co, Vienna**
	Internship assisting the managing director in writing a paper (“**Anticipating Credit Risks”**) for a textbook for Austrian Universities
July 2001,2002	**Boehler Uddeholm AG, Vienna**
	Internship in the **treasury department** (participation in hedging activity, cash pooling and credit insurance management)

Language Skills

German (native language)

English (bilingual/excellent - mother comes from England)

French (basic knowledge - second foreign language at university)

Computer Skills

Microsoft Office (Word, Excel, Power Point) experienced

Programming (Pascal, Java) basic knowledge, (C++) advanced

Arena (simulation software) advanced

SAP basic knowledge

Further activities

Handball (since 1988, semi-professional from 1999 until 2002/Club: UHC Stockerau)

Catholic youth group (organising charity work and weekly events, leader of a group of 14 – 16 year olds with weekly meetings including discussions and creative activities, organisation of the cocktail bar at the yearly ball, organisation of other activities like football tournaments or Christmas parties)

6 References

[1] **Aarts, E., Lenstra, J.K. (1997):** Local Search in Combinatorial Optimization, John Wiley & Sons, Chichester.

[2] **Balas, E., Ho, A. (1980):** Set Covering Algorithms Using Cutting Planes, Heuristics and Subgradient Optimization: A Computational Study, Mathematical Programming Vol.12, 37-60.

[3] **Balaprakash, P., Birattari, M., Stützle, T., Dorigo, M. (2006):** Incremental Local Search in Ant Colony Optimization: Why it fails for the Quadratic Assignment Problem, TR/IRIDIA/2006-011.

[4] **Beasley, J.E. (1990):** OR-Library: distributing test problems by electronic mail, Journal of the Operational Research Society 41(11), 1069-1072.

[5] **Bräysy, O., Gendreau, M. (2005):** Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms, Transportation Science Vol.39:1, 104-118.

[6] **Current, J.R., Schilling, D.A. (1989):** The Covering Salesman Problem, Transportation Science Vol.23, No.3, 208 – 213.

[7] **Current, J.R., Schilling, D.A. (1994):** The median tour and maximal covering tour problems: Equations and heuristics, European Journal of Operations Research Vol.73, 114 – 126.

[8] **Dantzig, G.B., Fulkerson, D.R., Johnson, S.M. (1954):** Solution of a large-scale traveling salesman problem, Operations Research, Vol. 2, 929 – 410.

[9] **Deneubourg, J.L., Goss, S., Aron, S., Paseels, J.M. (1990):** The self-organizing exploratory pattern of the Argentine ant, Journal of Insect Behavior Vol.3, 159.

[10] **Dorigo, M. (1992):** Optimization, Learning and Natural Algorithms, PhD thesis, Politecnico di Milano, Italy.

[11] **Dorigo, M., Maniezzo, V., Colorni, A.(1991):** Ant System: An autocatalytic optimizing process, working paper No. 91-016 Revised, Politecnico di Milano, Italy.

[12] **Dorigo, M., Stützle, T. (2004):** Ant colony optimization, MIT-Press, Cambridge.

[13] **Gendreau, M., Hertz, A., Laporte, G. (1992):** New Insertion and Postoptimization Procedures for the Traveling Salesman Problem, Operations Research Vol.40, No.6, 1086 – 1094.

[14] **Gendreau, M., Laporte, G., Semet, F. (1997):** The Covering Tour Problem, Operations Research Vol.45, No.4, 568 – 576.

[15] **Goss, S., Aron, S., Deneubourg, J.L., Paseels, J.M. (1989):** Self-organized shortcuts in the Argentine ant, Naturwissenschaften Vol.76, 579 – 581.

[16] **Hachida, M., Hodgson, M.J., Laporte, G., Semet, F. (2000):** Heuristics for the multi-vehicle covering tour problem, Computers & Operations Research Vol.27, 29-42.

[17] **Hodgson M.J., Laporte G., Semet F. (1998):** A covering tour model for planning mobile health care facilities in Suhum district, Ghana, Journal of Regional Science Vol.38, 621-628.

[18] **Jozefowiez, N., Semet, F., Talbi, E. (2007):** The bi-objective covering tour problem, Computers & Operations Research Vol. 34, 1929-1942.

[19] **Le Louran, F-X., Gendreau, M., Potvin, J-Y. (2004):** GENI Ants for the Traveling Salesman Problem, Annals of Operations Research 131, 187-201.

[20] **Lessing, L., Dumitrsecu, I., Stützle, T. (2004):** A comparison between ACO algorithms for the Set Covering Problem, Lecture Notes in Computer Science, v3172, 1-12.

[21] **Marchiori, E., Steenbeek, A. (2000):** An evolutionary algorithm for large scale set covering problems with application to airline crew scheduling, In Real World Applications of Evolutionary Computing, Springer Verlag, 367-381.

[22] **McKinsey & Company, Inc. (2006):** McK Wissen 16 – Logistik, brand eins Verlag, Hamburg.

[23] **Reinelt, G. (1991):** TSPLIB – A Traveling Salesman Problem library, ORSA Journal on Computing 3, 376-384.

[24] **Reinelt, G. (2005):** http://www.iwr.uni-heidelberg.de/groups/ comopt/software/TSPLIB95/tsp/ (07.06.2007).