



**CTU**

**CZECH TECHNICAL  
UNIVERSITY  
IN PRAGUE**

**Faculty of Electrical Engineering  
Department of Computer Science**

**Master's thesis**

# **Lower Bound Estimates for Path Planning in Environment with Obstacles**

**Bc. Kristýna Kučerová**

**May 2023**

**Supervisor: Ing. Jindřiška Deckerová**



## I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kučerová** Jméno: **Kristýna** Osobní číslo: **474523**  
Fakulta/ústav: **Fakulta elektrotechnická**  
Zadávající katedra/ústav: **Katedra počítačů**  
Studijní program: **Otevřená informatika**  
Specializace: **Umělá inteligence**

## II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

**Odhady dolních mezí pro plánování cesty v prostředí s překážkami**

Název diplomové práce anglicky:

**Lower Bound Estimates for Path Planning in Environment with Obstacles**

Pokyny pro vypracování:

- Familiarize yourself with routing problems such as the Traveling Salesman Problem (TSP) [1] and its variants with neighborhoods [2] motivated by data collection planning. Familiarize yourself with the Branch-and-Bound (BnB) for computation of lower bound values for the Close Enough TSP [3,4] and methods addressing problems in an environment with obstacles [5,6,7].
- Propose problem formulation of the TSP with continuous neighborhoods in an environment with obstacles as a mathematical model.
- Implement an extension of the BnB method to address the studied problem using the proposed mathematical model.
- Evaluate the extended BnB and compare it with the baseline methods [5].

Seznam doporučené literatury:

- [1] Applegate, D. and Bixby, R. and Chvátal, V. and Cook, W., The Traveling Salesman Problem: A Computational Study, Princeton University Press, 2007.
- [2] Gulczynski, D. J. and Heath, J. W. and Price, C. C., The Close Enough Traveling Salesman Problem: A Discussion of Several Heuristics, Perspectives in Operations Research: Papers in Honor of Saul Gass' 80th Birthday, 2006; 271--283, 10.1007/978-0-387-39934-8\_16.
- [3] Coutinho, W. & Nascimento, R. & Pessoa, A. & Subramanian, A., A Branch-and-Bound Algorithm for the Close-Enough Traveling Salesman Problem. INFORMS Journal on Computing. 2016; 28. 752-765. 10.1287/ijoc.2016.0711.
- [4] Gentilini, I. and Margot, F. and Shimada, K., The travelling salesman problem with neighbourhoods: MINLP solution, Optimization Methods & Software, 2012; 10.1080/10556788.2011.648932.
- [5] Faigl, J. & Kulich, M. & Vonasek, V.h & Preucil, L. An application of the self-organizing map in the non-Euclidean Traveling Salesman Problem. Neurocomputing. 2011; 74. 671-679. 10.1016/j.neucom.2010.08.026.
- [6] Schulman J, Duan Y, Ho J, et al., Motion planning with sequential convex optimization and convex collision checking. The International Journal of Robotics Research. 2014; 33(9):1251-1270. doi:10.1177/0278364914528132.
- [7] Babel, L. Curvature-constrained traveling salesman tours for aerial surveillance in scenarios with obstacles. European Journal of Operational Research. 2017; 262. 10.1016/j.ejor.2017.03.067.

Jméno a pracoviště vedoucí(ho) diplomové práce:

**Ing. Jindřiška Deckerová katedra počítačů FEL**

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **24.09.2022**

Termín odevzdání diplomové práce: \_\_\_\_\_

Platnost zadání diplomové práce: **24.09.2024**

\_\_\_\_\_  
Ing. Jindřiška Deckerová  
podpis vedoucí(ho) práce

\_\_\_\_\_  
podpis vedoucí(ho) ústavu/katedry

\_\_\_\_\_  
prof. Mgr. Petr Páta, Ph.D.  
podpis děkana(ky)

### III. PŘEVZETÍ ZADÁNÍ

Diplomantka bere na vědomí, že je povinna vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

\_\_\_\_\_  
Datum převzetí zadání

\_\_\_\_\_  
Podpis studentky



## Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, May 26, 2023

.....  
Bc. Kristýna Kučerová





## Acknowledgment

I would like to thank Ing. Jindřiška Deckerová for supervising my thesis and for the valuable help. Her determination and drive for research are admirable. I would also like to thank prof. Ing. Jan Faigl, Ph.D. for the discussions and his expertise. Many thanks to my family, my caring boyfriend, and my friends for their love and support, it was a *prerequisite* to where I am today. Namely, I would like to thank Bc. Petra Fridrichová for sharing the struggles once again, while still being so joyful.





## Abstract

In the thesis, we address a variant of the well-known Traveling Salesman Problem called the Close-Enough Traveling Salesman Problem with obstacles in the polygonal domain. The problem is to visit given target regions within their disk neighborhoods, such that the length of the formed closed-loop tour is minimal, and the tour does not interfere with any polygonal obstacles. We formulate the problem as the Mixed Integer Non-Linear Program (MINLP) for a fixed sequence that includes the intermediate points between regions necessary for obstacle avoidance. We propose an extension of the sequence-based Branch-and-Bound (BnB) method to address the studied problem utilizing several lower bound estimates of the solution cost based on the Euclidean distance, the Visibility Graph, and the mathematical model disregarding the obstacles. Moreover, we propose two different methods to determine feasible upper-bound solutions: the **Sampled**-based method, where the target region borders are sampled and the problem becomes discretized; and the MINLP model, which includes obstacle constraints using half-plane separation of obstacle points.

The proposed BnB method is evaluated on randomly generated instances and compared with the existing heuristic methods. We examine the lower bound estimate influence on the solution quality and the performance of the upper bound solutions. Based on the results, all proposed lower bound estimates provide the same optimal sequence for all evaluated instances. The upper bound determined by the sampling-based method is dependent on the density of the sampling. The found solutions provided by the method with sparse sampling can be significantly improved by the proposed MINLP-based method as the post-optimization procedure. In comparison to the reference methods, the obtained results had better or comparable costs in a longer computational time, which is to be expected due to using exact methods. Furthermore, the post-optimization procedure can be used to improve the reference solutions as well.

**Keywords:** Close-Enough Traveling Salesman Problem with Obstacles, Polygonal Domain, Mixed Integer Non-Linear Program, Branch-and-Bound



## Abstrakt

V této práci se zabýváme variantou známého Problému obchodního cestujícího, nazvaného Problém obchodního cestujícího s kruhovým okolím v prostředí s překážkami v polygonálním prostoru. Cílem je navštívit všechny definované cílové regiony v jejich kruhovém okolí tak, aby délka vzniklé cesty tvořící uzavřený cyklus byla minimální a zároveň cesta neprotínala žádné polygonální překážky. Formulujeme problém jako Celočíslný nelineární program (MINLP) pro pevně danou sekvenci, která obsahuje body mezi cílovými regiony nutné pro vyhnutí se překážkám. Navrhujeme rozšíření Metody větví a mezí (BnB) pro větvení na sekvencích pro řešení studovaného problému, včetně různých druhů odhadů spodních limitů délky řešení na základě Euklidovských vzdáleností, Grafu viditelnosti, a pomocí matematického modelu neuvažujícího překážky. Dále navrhujeme dvě různé metody pro získání řešení problému: metoda vzorkující hranici kruhového regionu, která transformuje studovaný problém do vzorkovaného prostoru, a metoda využívající MINLP model, obsahující omezení pro rozdělení plochy na poloroviny, do kterých přísluší daná překážka. Navrhované řešení problému je horním ohraničením optimálního řešení.

Výsledky navrhovaného BnB algoritmu jsou napočítány na náhodně vygenerovaných instancích a porovnány s existujícími heuristickými metodami. Vyhodnocujeme jak vliv různých odhadů spodního limitu, tak kvalitu různých metod řešení. Z výsledků vyplývá že všechny typy odhadu spodního limitu řešení poskytují stejnou výslednou optimální sekvenci pro všechny testované instance. Navržené řešení problému pomocí vzorkující metody je závislé na hustotě vzorkování. Řešení nalezené touto metodou s malým množstvím vzorků je možné výrazně vylepšit použitím MINLP modelu jako dodatečnou optimalizaci. V porovnání s referenčními metodami mají výsledky lepší nebo porovnatelnou délku cesty získanou v delším výpočetním čase, což je očekávané kvůli exaktnímu přístupu. Navíc je možné použít dodatečnou optimalizaci i na vylepšení řešení nalezených pomocí referenčních metod.

**Klíčová slova:** Problém obchodního cestujícího s kruhovým okolím a překážkami, polygonální doména, Celočíslný nelineární program, Metoda větví a mezí




# Contents

Used Abbreviations	xv
Used Symbols	xvii
List of Figures	xix
List of Tables	xxi
List of Algorithms	xxiii
<b>1 Introduction</b>	<b>1</b>
<b>2 Related Work</b>	<b>5</b>
2.1 Traveling Salesman Problem . . . . .	5
2.2 Traveling Salesman Problem with Neighborhoods . . . . .	6
2.3 Multi Goal Path Planning in Environment with Obstacles . . . . .	7
2.4 Mathematical Modeling and Optimization Solvers . . . . .	8
<b>3 Problem Statement</b>	<b>11</b>
3.1 Polygonal domain . . . . .	11
3.1.1 Obstacle Representation . . . . .	12
3.2 Traveling Salesman Problem . . . . .	12
3.3 Traveling Salesman Problem with Neighborhoods . . . . .	13
3.4 Close-Enough Traveling Salesman Problem with Obstacles . . . . .	13
<b>4 Background</b>	<b>17</b>
4.1 Visibility Graph . . . . .	17
4.1.1 Shortest Path in the Visibility Graph . . . . .	18
4.2 Priority Queue . . . . .	18
4.3 Computation of Area Between Two Disks . . . . .	18
4.3.1 Equal Radii . . . . .	19
4.3.2 Non-equal Radii . . . . .	19

<b>5</b>	<b>Proposed Method</b>	<b>21</b>
5.1	Branch-and-Bound for the Sequence Optimization . . . . .	21
5.1.1	Initialization . . . . .	24
5.1.2	Lower Bound Estimates . . . . .	25
5.1.3	Upper Bound Computation . . . . .	25
5.1.4	Proposed Solution . . . . .	26
5.2	Sampled-Based Method . . . . .	27
5.3	Mixed Integer Non-Linear Program for the <b>CETSP<sub>obs</sub></b> . . . . .	28
5.3.1	Second-Order Cone Program for the <b>CETSP</b> . . . . .	28
5.3.2	Polygonal Obstacles Constrained in a Half-Plane . . . . .	29
5.3.3	Conditions for Adding Obstacle Constraints to Model . . . . .	31
5.3.4	Discussion of the Model Solution Quality . . . . .	34
5.3.5	The <b>MINLP</b> as Post-optimization Heuristic . . . . .	34
<b>6</b>	<b>Empirical Evaluation</b>	<b>37</b>
6.1	Instances . . . . .	37
6.2	Evaluation of the Lower Bound Estimates . . . . .	39
6.3	Evaluation of the Upper Bound Solutions . . . . .	41
<b>7</b>	<b>Conclusion</b>	<b>47</b>
	<b>Bibliography</b>	<b>49</b>
<b>A</b>	<b>Instances</b>	<b>53</b>

# Used Abbreviations

<b>ATSP</b>	Asymmetric Traveling Salesman Problem
<b>BnB</b>	Branch-and-Bound
<b>CETSP</b>	Close-Enough Traveling Salesman Problem
<b>CETSP<sub>obs</sub></b>	Close-Enough Traveling Salesman Problem with Obstacles
<b>CHOMP</b>	Covariant Hamiltonian Optimization for Motion Planning
<b>CTSPO</b>	Curvature-Constrained Traveling Salesman Problem (TSP) with Obstacles
<b>DoF</b>	Degrees of Freedom
<b>ETSP</b>	Euclidean Traveling Salesman Problem
<b>GLNS</b>	Large Neighborhood Search Heuristic for the Generalized Traveling Salesman Problem (GTSP)
<b>GLNS-CETSP</b>	Large Neighborhood Search Heuristic for the GTSP (GLNS) for the Close-Enough Traveling Salesman Problem (CETSP)
<b>GTSP</b>	Generalized Traveling Salesman Problem
<b>LP</b>	Linear Program
<b>MTP</b>	Multi-Goal Path Planning / Multi-Target Planning
<b>MILP</b>	Mixed Integer Linear Program
<b>MINLP</b>	Mixed Integer Non-Linear Program
<b>MISOCP</b>	Mixed Integer Second-Order Cone Program
<b>NLP</b>	Non-Linear Program
<b>NN</b>	Nearest Neighbor



<b>PQ</b>	Priority Queue
<b>RFID</b>	Radio-Frequency Identification
<b>SOCP</b>	Second-Order Cone Program
<b>SOM</b>	Self-Organizing Map
<b>TSP</b>	Traveling Salesman Problem
<b>TSPN</b>	Traveling Salesman Problem with Neighborhoods
<b>VG</b>	Visibility Graph



# Used Symbols

## Mathematical notation

$\mathbb{R}$	set of real numbers
$\mathbb{R}_0^+$	set of non-negative real numbers
$\mathbb{Z}$	set of integer numbers
$\mathbb{N}$	set of non-negative integer numbers, also denoted as $\mathbb{N}_0$
$\mathbb{N}^+$	set of positive integer numbers
$\{\bullet_1, \dots, \bullet_x\}$	set containing $x$ elements $\bullet$
$\{\bullet \mid c(\bullet)\}$	set containing elements $\bullet$ that fulfill the condition $c$
$(\bullet_1, \dots, \bullet_x)$	ordered list containing $x$ elements $\bullet$
$ \bullet $	number of elements (size) of the set/list $\bullet$
$\ \bullet\ $	length / Euclidean distance of vector $\bullet$
$\boldsymbol{v}$	vector in $\mathbb{R}^2$ denoted by the bold symbol
$\boldsymbol{v}_\bullet$	vector $\boldsymbol{v}$ with an identifier $\bullet$
$v_\bullet$	element of vector $\boldsymbol{v}$ at index $\bullet$ , note: in case of multiple subscripts, the index is always the last subscript
$\mathcal{O}(f(n))$	the computational complexity $\mathcal{O}$ growing at most as much as function $f$ with the input data size $n$
$\bullet \cup \blacktriangle$	unite sets $\bullet$ and $\blacktriangle$ into a set containing all elements from both sets
$\bullet \sqcup \blacktriangle$	append elements of lists $\blacktriangle$ after the elements of list $\bullet$ into a new list

## Named symbols

$\mathcal{S}$	set of $n$ target regions $S$ , $\{S_1, \dots, S_n\}$
$S_i$	$i$ -th target region with disk neighborhood, $S_i = (c_i, \delta_i)$

$\mathbf{c}_i$	$i$ -th target region center coordinates, $\mathbf{c}_i \in \mathbb{R}^2$
$\delta_i$	$i$ -th target region disk radius, $\delta_i \in \mathbb{R}_0^+$
$\Omega$	set of $m$ obstacles $O$ , $\Omega = \{O_1, \dots, O_m\}$
$O_i$	$i$ -th polygonal obstacle given by list of $k$ points $\mathbf{o}$ , $O = \{\mathbf{o}_1, \dots, \mathbf{o}_k\}$ , $\mathbf{o}_i \in \mathbb{R}^2$
$\Sigma$	visiting sequence to the target regions, $\Sigma = (\sigma_1, \dots, \sigma_n)$ , $\sigma_i \in \{1, \dots, n\}$
$\mathcal{LB}$	lower bound value
$\mathcal{UB}$	upper bound value
$\mathcal{C}$	cost (length) of the solution tour
$\mathcal{Q}$	set of sets of intermediate points between two subsequent target regions, $\mathcal{Q} = \{Q_1, \dots, Q_n\}$
$Q_i$	$i$ -th possibly empty set of intermediate points between two subsequent target regions
$\mathcal{M}$	mathematical model including objective function and constraints over a set of obstacles
$\mu$	a Branch-and-Bound node, containing the solution node lower bound denoted as $\mu.\mathcal{LB}$ , upper bound $\mu.\mathcal{UB}$ , sequence $\mu.\Sigma$ , and cost $\mu.\mathcal{C}$
$M$	Big-M constant for switching between the constraints.
$t$	computational time in seconds
$t_{\max}$	maximum computational time limit in seconds

# List of Figures

1.1	Example of the solutions produced by the proposed methods . . . . .	2
3.1	Example of 2D configuration spaces with obstacles and a path . . . . .	11
3.2	Example of convex and concave polygons . . . . .	12
3.3	Example of the instances for the $CETSP_{obs}$ . . . . .	14
3.4	Notation used for the $CETSP_{obs}$ . . . . .	14
4.1	Visibility Graph from a single vertex . . . . .	17
4.2	Area of direct visibility (cone) between two targets with same radii . . . . .	19
4.3	Area of possible solution space (cone) between two targets with different radii .	20
5.1	Branch-and-Bound example on a $CETSP$ instance . . . . .	22
5.2	Solution points on target disk . . . . .	26
5.3	Instance with sampled target regions . . . . .	27
5.4	Auxiliary graph with target samples . . . . .	28
5.5	Points in half-planes generated by a solution line segment . . . . .	30
5.6	Area of possible solutions between targets on a $CETSP_{obs}$ instance . . . . .	32
5.7	Example of a target region and obstacle scenario with obstacle inside of the disk	32
5.8	Example of a target region and obstacle scenario with a complex concave obstacle	33
5.9	Example solution where the intermediate solution points can improve its cost .	34
6.1	Process for random obstacle generation . . . . .	38
6.2	Example of randomly generated instances . . . . .	38
6.3	Different number of examined lower bound values for the <b>Sampled</b> -based method	39
6.4	Solutions found before timeout with different lower bounds . . . . .	41
6.5	The comparison of the solution cost and time for the <b>Sampled</b> -based method .	43
A.1	The randomly generated instances with proposed and reference solutions . . . .	53
A.2	The randomly generated instances with proposed and reference solutions contd.	54
A.3	The randomly generated instances with proposed and reference solutions contd.	55
A.4	The randomly generated instances with proposed and reference solutions contd.	56



# List of Tables

2.1	Supported mathematical optimization categories of selected solvers . . . . .	9
6.1	Comparison of the $\mathcal{LB}$ types for <b>Sampled</b> -based method with $k = 64$ . . . . .	40
6.2	<b>Sampled</b> -based method with different number of samples $k$ and $\mathcal{LB}_{\text{SOCP}}$ . . . . .	42
6.3	Comparison of solution cost $\mathcal{C}$ for proposed and reference methods . . . . .	44
6.4	Comparison of solution cost $\mathcal{C}$ with <b>Post-optimization</b> heuristic . . . . .	45



# List of Algorithms

1	Shortest paths using the Visibility Graph . . . . .	18
2	Proposed Branch-and-Bound algorithm for the $\text{CETSP}_{\text{obs}}$ . . . . .	24
3	Heuristic sequence insert of target regions . . . . .	26
4	Create optimization model with cone . . . . .	33
5	Pointify model solution . . . . .	35





# Introduction

Path planning is the process of finding a feasible path from a starting point to a destination. The path feasibility is based on the desired criteria, the capabilities of the vehicle, and also on the environment. In robotics and autonomous systems, path planning is crucial for navigating the movement and preserving resources, most often either energy or time. In the process of determining the best path usually a set of constraints needs to be considered, such as obstacles, speed limits, curvature limits, or remaining fuel or battery charge.

One of the most studied problems in Multi-Goal Path Planning / Multi-Target Planning (MTP) and optimization is the Traveling Salesman Problem (TSP) [1]. The problem is to find the shortest path visiting each location from the given set of locations exactly once, such that the path is a closed loop. It has many important applications in logistics and transportation, for example, scheduling a salesman's visits to different locations, planning delivery routes for logistics companies, or airline itineraries of a travel agency. Despite the simple formulation, the TSP is NP-hard [1]. The number of solutions for the TSP increases by factorial with the number of locations, making it difficult to solve for large instances. Because of the TSP usability and complexity, it has received extensive research interest in the past and it is still an active research topic today.

A variant of the TSP with disk neighborhoods is known as the Close-Enough Traveling Salesman Problem (CETSP). The CETSP originated as a problem of a company reading data from customers using the Radio-Frequency Identification (RFID) scanners, which can read data from a distance. It is proposed in [2] along with multiple heuristic approaches for solving the CETSP. In [3], the problem is approached using the Branch-and-Bound (BnB) for the TSP sequence optimization, and the Second-Order Cone Program (SOCP) mathematical model for solving the fixed-sequence subproblems. This second approach provides optimal solutions to the problem in an environment without obstacles.

The problem is more complicated in an environment with obstacles, as the paths become infeasible due to collisions with obstacles. Some use-case scenarios that can be modeled as the CETSP with obstacles are for example:

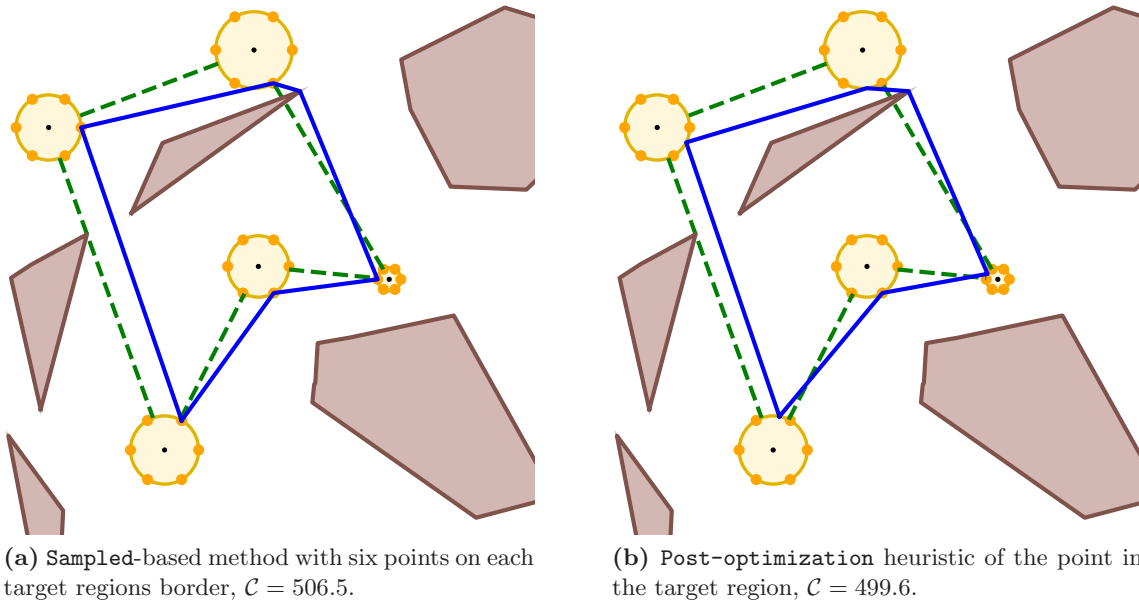
- Video mapping or data collection [4], such as in a farm field with a drone and a camera. The locations for scanning can be marked and since a camera has a wide view, it is sufficient to have the marker anywhere within the view, which adds a visiting neighborhood. The field can also have some scattered trees or farming equipment obstructing

the flight path.

- In a warehouse, all areas should periodically be visited and the visit to the positions confirmed with **RFID** sensors, as in [2]. However, the warehouse is full of obstacles and has different sections separated by walls, and the sensors have a sensing radius.
- In a supply run where a supply truck cannot remain in the target area for a long time (e.g. a battlefield), as noted in [3], the supplies are dropped in any position within the target area. Any obstacles can again occur, such as mountains, enemy camps, buildings, etc.

In [5] the **TSP** in the polygonal space is approached using the Self-Organizing Map (**SOM**). The **SOM** for the Traveling Salesman Problem with Neighborhoods (**TSPN**) is proposed in [6], where the target neighborhoods of the **TSPN** are approximated as polygons. In [7], the problem is approached using partitioning of the environment and modification of a solver for the Generalized Traveling Salesman Problem (**GTSP**), called the Large Neighborhood Search Heuristic for the **GTSP** (**GLNS**) [8]. It is initialized with a solution provided by the **SOM** in the environment without obstacles and then the solutions are modified to be feasible. However, these approaches are heuristic and there is no optimality guarantee.

To our best knowledge, there is no work approaching the **CETSP** with obstacles with the aim of finding the globally optimal solution. In this thesis, we aim to approach the **CETSP** with obstacles with the lower bound estimates to obtain the solution with the highest quality. We denote the problem studied in the thesis as the **CETSP<sub>obs</sub>**, and formally define the problem. The obstacles are approximated as polygons for this thesis with an arbitrary number of sides. An example of the problem with the proposed solution can be seen in Figure 1.1.



**Figure 1.1:** Example of the solutions produced by the **Sampled**-based method (Figure 1.1a) and mathematical **Post-optimization** heuristic (Figure 1.1b) method. The tour cost is denoted as  $\mathcal{C}$ , yellow circles are the target regions and orange points are their point samples, brown polygons are the obstacles, the green dashed line is the lower bound estimate using the Euclidean distance, and the blue line is the proposed Close-Enough Traveling Salesman Problem with Obstacles (**CETSP<sub>obs</sub>**) solution.

Since the environment contains polygonal obstacles that need to be avoided, we utilize the Visibility Graph (VG). The VG is an algorithm from computational geometry [9] that maps visible vertices of the polygonal environment to create a graph, where the visibility is denoted as edges between vertices. Graph-search algorithms such as Dijkstra can be then used to find the shortest path between two points in the polygonal space.

In the thesis, we propose the BnB method [3] to address the CETSP<sub>obs</sub> and propose a mathematical model formulation of the CETSP<sub>obs</sub> with polygonal obstacles. The model is formulated for a fixed sequence, inspired by the SOCP mathematical model for the CETSP from [3]. The addition of obstacles makes it non-linear and using binary variables transforms it from SOCP to a Mixed Integer Non-Linear Program (MINLP) problem. The BnB with the MINLP model takes a long time to compute a solution, therefore the Sampled-based method has also been utilized in the thesis. The disk neighborhoods borders are sampled and one of the samples has to be visited to consider the disk as visited, effectively transforming the problem to GTSP. With a large enough number of samples, this method approaches the optimal solution within the continuous domain. The solution of the Sampled-based method can be also improved upon with the mathematical model by providing the sampled solution as initialization, which we call the Post-optimization heuristic and is submitted in [10].

The proposed methods are tested on randomly generated instances. The computational results are discussed and compared with a SOM-based approach for solving the TSPN within the polygonal environment with obstacles [6] and the method for the CETSP<sub>obs</sub> called the GLNS for the CETSP (GLNS-CETSP) [7]. The Sampled-based method is the fastest to compute out of the proposed methods, and both the computational time and cost of the solution are highly dependent on the number of samples. The Post-optimization heuristic can improve the costs of the solutions provided by the proposed Sampled-based method, and it can also improve the reference methods [6, 7]. The proposed method obtains better solutions than the SOM approach, and solutions with better or comparable quality to the GLNS-CETSP solutions when using the mathematical model.

The structure of this thesis is as follows. The related work is described in Chapter 2. The problem formulation is introduced in Chapter 3, and the necessary background knowledge is described in Chapter 4. The BnB algorithm and the mathematical model used in the proposed method are presented in Chapter 5. The method is empirically evaluated in Chapter 6 along with the discussion of results. Finally, the work is concluded in Chapter 7.



## Related Work

The herein studied Close-Enough Traveling Salesman Problem with Obstacles ( $\text{CETSP}_{\text{obs}}$ ) is a variant of the well-known Traveling Salesman Problem ( $\text{TSP}$ ). There are only a few approaches to the  $\text{CETSP}_{\text{obs}}$  or the closely related problem of Traveling Salesman Problem with Neighborhoods ( $\text{TSPN}$ ) with obstacles to the best of the author's knowledge. Therefore, we also present an overview of solutions to the  $\text{TSPN}$ , and Multi-Goal Path Planning / Multi-Target Planning ( $\text{MTP}$ ) with obstacles. Moreover, we present a brief overview of the optimization solvers, since we solve the  $\text{CETSP}_{\text{obs}}$  with a sequence-based mathematical model.

First, the possible solution approaches to the  $\text{TSP}$  are introduced in Section 2.1. Its extended variant is the  $\text{TSPN}$ , where instead of visiting goals, the aim is to visit any point within a region, and methods addressing the  $\text{TSPN}$  are described in Section 2.2. Existing methods for planning with obstacles are introduced in Section 2.3, and finally, mathematical optimization solvers are described in Section 2.4.

### 2.1 Traveling Salesman Problem

The  $\text{TSP}$  is a well-known problem from  $\text{MTP}$  [1]. The problem is to find a tour with a minimum length that visits all locations, and the initial and final locations are the same. The  $\text{TSP}$  uses the Euclidean distances between locations to measure the path length, and it can be referred to as the Euclidean Traveling Salesman Problem ( $\text{ETSP}$ ) in the literature. Many variants of the  $\text{TSP}$  exist, using different distance metrics, asymmetric distances (distance from target a to target B is different than from B to A), which is called the Asymmetric Traveling Salesman Problem ( $\text{ATSP}$ ), adding neighborhoods of the target location, adding obstacles etc.

Although the  $\text{TSP}$  assignment might seem simple, the problem is NP-hard [1] unless  $P=NP$ , therefore, it cannot be solved in polynomial time. The  $\text{TSP}$  is a special case of the Hamiltonian cycle, and it has been proven that computing the Hamiltonian cycle is NP-hard, therefore the  $\text{TSP}$  is also NP-hard [1]. For the  $\text{TSP}$  with  $n$  goals, there are  $(n - 1)!$  solutions and the factorial grows fast with  $n$ .

Many approaches had been proposed to solve the  $\text{TSP}$ , which can be divided into several categories, such as heuristics, and exact methods. The heuristic approaches aim to find any solutions efficiently. To name a few [11]: the Nearest Neighbor ( $\text{NN}$ ) algorithm, which starts

with an arbitrary goal and adds the next nearest goal to the end of the sequence, and repeats this step until all goals are added. Similar to the **NN** are the insertion methods, which can insert the next goal not only at the end but to any position in the sequence.

The unsupervised learning approaches to the **TSP** are also studied, such as the Self-Organizing Map (**SOM**) [12]. **SOM** is a two-layer neural network, where the output layer is a ring of neurons. It is trained by adjusting itself to the input data, in this case, the coordinates of the goals, by iteratively drawing the ring of neurons closer to the goals while minimizing the ring's length. The advantage of the **SOM** is its low computational complexity, but the number of iterations to train the output layer to obtain the solution is not known in advance.

The exact approaches aim to find the optimum solutions in less than exponential time. The solution can be found using the brute force approach, which becomes intractable with just a few dozen of goals. Therefore, approaches that can tighten the solution space have been used. One of the most studied exact algorithms is the Branch-and-Bound (**BnB**) method, which branches on sub-problems by prioritizing the promising solutions first and using the bounding function to eliminate sequences with high tour costs [13]. The subproblems are relaxations of the original problem, for example, a tour on a subset of the given goals of the **TSP**. The bounding in **BnB** helps to limit which solutions are branched further, leading to faster computation of the optimal solution.

Another exact approach is the Mixed Integer Linear Program (**MILP**)-based approach [14], where the problem is formulated using an objective function with constraints and solved using optimization solvers. The formulation is Mixed-Integer due to the use of binary variables to indicate which edges between goals are within the solution. The solvers are usually optimized for the **TSP** and use specific solution space cuts to find the solution faster. In **MILP**, the **BnB** or its variations (such as Branch-and-Cut) can also be used to branch on the subproblems, however here the problem relaxation is usually a linear subspace.

## 2.2 Traveling Salesman Problem with Neighborhoods

In the **TSP**, the exact locations are visited and the optimization concerns only the sequence order. The herein studied problem of the **TSPN** is a variant of the **TSP**, where the aim is to visit a neighborhood (area) instead. Because of this, in addition to the combinatorial optimization of the **TSP**, the continuous optimization is addressed to determine the points of visit within the neighborhood. The region is considered to be visited, if the solution touches any point of the neighborhood including the border.

The neighborhood of the goal location can vary in both shape and size. In [15], polyhedra or ellipsoid neighborhoods are considered. To solve the **TSPN**, the authors propose a Mixed Integer Non-Linear Program (**MINLP**) for both symmetric and asymmetric distances. Since the **MINLP** formulation is difficult to solve, the authors propose modifications of the spatial **BnB** [16], such as subtour elimination and global search space cuts. The authors also discuss a heuristic solution to the problem using different **MINLP** relaxations to obtain a tight upper bound for the **BnB**.

In [2], the problem called the Close-Enough Traveling Salesman Problem (**CETSP**) is introduced, which uses disk-shaped neighborhoods. The problem is introduced in relation to **MTP** with the Radio-Frequency Identification (**RFID**) scanners, which can read the **RFID** tags from a distance. Visiting a point in the disk-shaped (circular) neighborhood can result in a shorter tour than visiting the disk centers. The approach proposed in [2] is to simplify the

initial **CETSP** by grouping the input target regions by their mutual distance and aggregating these into as few nodes as possible such that by visiting the node, all the target regions within the group are visited. The **TSP** tour on the nodes is then a feasible **CETSP** solution at the same time guaranteed by the node grouping. Multiple heuristic approaches are proposed to find the initial clusters and reduce their number, along with iterative improvement strategies. A similar approach to [2] is described in [17], which also uses clustering of the nodes and then a convex hull algorithm. In [18], various heuristic approaches to solving the **CETSP** are discussed along with the computational results.

The mentioned approaches for solving the **CETSP** propose heuristic solutions. In [3], the authors propose a method for solving the **CETSP** yielding the optimal solution with the use of the **BnB** algorithm. They employ a mathematical model for solving the subproblems (fixed sub-sequences) that be solved to optimality since it is formulated as the Second-Order Cone Program (**SOCP**) from the convex optimization class [19]. The proposed algorithm works as follows: the initialization consists of three targets with maximum tour length, and the **SOCP** problem is solved on the three-target sequence. Afterward, the solution is branched by adding the next target into the sequence at different positions, and the solution value is always obtained by the **SOCP**. The authors solve the problem both in 2D and 3D space and can find optimal solutions in many of the used instances from in [3] and [18].

## 2.3 Multi Goal Path Planning in Environment with Obstacles

There are different approaches to tackling path planning in an environment with obstacles. In [20], the authors describe a non-convex optimization method named Covariant Hamiltonian Optimization for Motion Planning (**CHOMP**) between two points when an obstacle is on the straight line in between. It uses a covariant gradient update rule that provides fast convergence to a local optimum. The authors apply this method within a multi-dimensional motion planning, which is for example motion planning for a robotic hand with a high number of Degrees of Freedom (**DoF**).

In [21], the authors propose an optimization method for planning with obstacles and evaluate it for robot manipulators with multiple **DoF** with results up to 18 **DoF**. The proposed optimization method is similar to the **CHOMP**, which starts with a straight-line trajectory that interferes with obstacles and optimizes it to obtain a feasible solution. The method is formulated using a sequential convex optimization, which solves a non-convex optimization problem by repeatedly solving convex subproblems. A part of the strategy is to turn infeasible constraints into penalties, leading to zero constraints violation. This paper focuses on the mathematical formulations, constraints adding and modifying constraints during the solving process, which is not the aim of this thesis.

The **CHOMP** is a method for motion planning, where smooth trajectories traversable under physical constraints and without sharp turns are desired. In **MTP**, obstacle avoidance is usually considered on a larger scale with the intermediate path instead. In a polygonal domain where the obstacles are represented by polygons, graph approach can be employed to avoid obstacles, namely the Visibility Graph (**VG**) [9]. The **VG** can be efficiently constructed within the 2D polygonal space. Then, the path found using the **VG** is the shortest path as shown in [9]. In the **TSP** problem with obstacles in the polygonal domain, this can be used to obtain the distances between the targets and solved as the Non-Euclidean **TSP** [22].

In [5], the authors propose the use of the **SOM** for solving the **TSP** in an environment with obstacles. The **SOM** can be represented as a two-layered competitive learning network,

where the nodes start in a ring and are iteratively drawn towards the vertices representing the targets. It can be used for solving the **TSP** as well, as it requires only a cost function of the resulting path, which in the Euclidean distance. The approach proposed in the paper uses an approximation of the shortest path based on the convex partitioning of a polygonal space, which is faster than the computation of the shortest path using the **VG**.

The **SOM** is also used for solving the **TSPN** with polygonal neighborhoods, in the polygonal domain with obstacles [6]. The approach is similar to [5], where mainly the distance function differs and different **SOM** variations are proposed. In [7], the **CETSP<sub>obs</sub>** is tackled by modification of an algorithm for solving the Generalized Traveling Salesman Problem (**GTSP**) called the Large Neighborhood Search Heuristic for the **GTSP** (**GLNS**) [8]. The **GTSP** can be considered as a discretized **TSPN**, where instead of a neighborhood, one point from a set of samples is to be visited for each target. The proposed heuristic approach is denoted as **GLNS** for the **CETSP** (**GLNS-CETSP**), and it can be used with different initialization methods and configurations of computational precision. It works by combining the **GLNS** algorithm with the proposed distance computation technique for finding the shortest distance between a point-circle-point sequence.

In [23] the authors tackle the problem of finding the shortest curvature-constrained path in an environment with obstacles, named Curvature-Constrained **TSP** with Obstacles (**CTSPO**). The authors describe various heuristic algorithms for solving the **CTSPO** feasibly. Because of the presence of obstacles and curvature constraints, some tours can become infeasible, which is also addressed by the authors. The main idea is to discretize the goals with heading angles, which are part of the Dubins vehicle state representation [24]. Then, solve the problem of finding the curvature-constrained path with obstacles separately. By solving this subproblem, the distance between every two samples can be obtained to create a distance matrix. Since the goal visits are sampled, the final tour is then found on the auxiliary graph constructed from the goal samples in sequence order. The graph contains the distance between samples, from which the shortest path can be obtained, and this technique can be also used for the **GTSP**.

## 2.4 Mathematical Modeling and Optimization Solvers

A mathematical optimization problem is defined by an objective function and constraints. The objective function is the optimization goal, which is either minimized or maximized. The constraints define the bounds on the variables which are part of the optimization problem. They can be either equality constraints or bounds for a (strictly) less or greater value.

One of the problems of mathematical optimization is the Linear Program (**LP**), and it is a widely used formulation for optimization problems in business, logistics, etc. [25]. The variables within the objective function and constraints are linear, meaning that there is no variable multiplication and only linear operations (addition, subtraction and multiplication by constant numbers). The **LP** is a problem of continuous optimization. In some applications, it might be necessary to restrict some constraints to integer values or even to binary values. This model can be understood as an extension of the **LP**, and is called Mixed Integer Linear Program (**MILP**) or **MIP**. To solve the **MILP** problem, different approaches can be used including the **BnB** algorithm [26]. The **BnB** for **MILP** works by first solving the problem as the continuous **LP**, and branching if the integer variable has a non-integer value in the solution. The **BnB** finds the exact solution to the **MILP** problem.

The convex optimization [19] is a category of mathematical optimization, where the objec-



tive function and constraints satisfy the convexity property. For a function, convexity means that the line segment connecting any two points on the function lies above the graph or on it [27]. Similarly, for a convex polygon, a line segment connecting any two points on the polygon lies completely within the polygon. Convexity in the optimization problem guarantees that the found local optimum is also the global optimum [27].

Another category of mathematical optimization is the Non-Linear Program (NLP). The objective function and/or constraints are non-linear, therefore the NLP solution space is non-convex. The NLP is usually solved with numerical methods such as gradient descent, but the found solution might be only locally optimal. Many approaches for solving NLP exist, but finding the global optimum might be computationally expensive, therefore most of the approaches focus on finding at least a local optimum. A version of the NLP with integer and/or binary variables is called Mixed Integer Non-Linear Program (MINLP).

There are many optimization solvers for mathematical optimization, and in Table 2.1, the overview of selected solvers and their supported optimization categories are depicted.

**Table 2.1:** Supported mathematical optimization categories of selected solvers.

Solver name	LP	MILP	SOCP	MISOCP	NLP	MINLP
CPLEX [28]	✓	✓	✓	✓	-	-
Ipopt [29]	✓	-	✓	-	✓	-
Juniper [30]	-	-	✓	✓	✓	✓
Bonmin [31]	-	-	-	-	✓	✓
Alpine [32, 33]	-	-	-	-	✓	✓

A mathematical model proposed in the thesis is in the MINLP category, therefore we need a solver able to solve such a problem. Solving the MINLP is computationally demanding, and different solvers provide various performances, and a few are mentioned next. A solver which can solve the LP, SOCP and NLP is the Ipopt [29], however, it cannot address their Mixed-Integer variants. The Juniper solver [30] is an open-source solver for MINLP developed for Julia [34] to allow modifications of the BnB parameters for the user. Another considered solver is (MI)NLP Bonmin solver [31], which is a part of the AMPL initiative for uniting mathematical solvers under one framework. The (MI)NLP COUENNE solver [35], which is used in [15] is also within the AMPL library. The two mentioned solvers for MINLP can find a local optimum. To get the global optimum, another solver called Alpine [32, 33] can be used. The Alpine solver requires sub-solvers for the MILP, NLP, and MINLP. In the thesis, the IBM-developed solver CPLEX is used among others [28]. The CPLEX can solve both problems from convex optimization (LP, SOCP), and their mixed integer variants (MILP, Mixed Integer Second-Order Cone Program (MISOCP)).

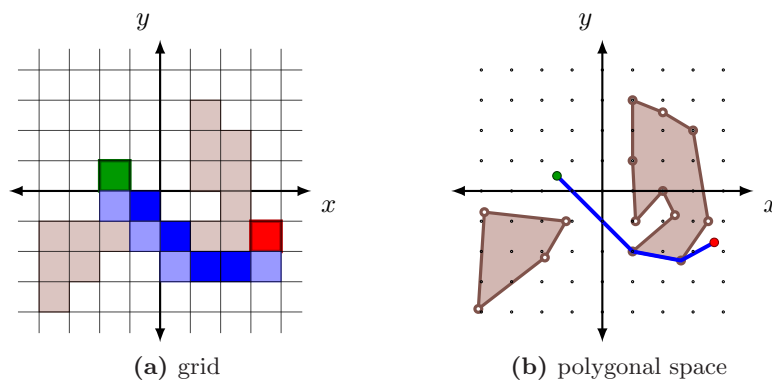


# Problem Statement

In the thesis, we study the Close-Enough Traveling Salesman Problem (**CETSP**) in the polygonal environment with obstacles. The **CETSP** is to find the most cost-efficient closed tour visiting all target disks while avoiding obstacles. The most cost-efficient path is the one minimizing the tour length. Hence, in this chapter, we first introduce the polygonal domain environment in Section 3.1. Then the problem formulation of the Traveling Salesman Problem (**TSP**) is provided in Section 3.2, and the formulation of the Traveling Salesman Problem with Neighborhoods (**TSPN**) is in Section 3.3 along with the disk-neighborhood variant called the **CETSP**. Finally, the obstacles are introduced into the formulation of the problem solved in this thesis, and it is provided in Section 3.4.

## 3.1 Polygonal domain

The problem studied in this thesis is within the 2D polygonal domain. The points in the polygonal environment are defined by coordinates  $(x, y) \in \mathbb{R}^2$ . This is in contrast to e.g. the 2D grid, where each of the coordinates is equal to a position in a grid. Both mentioned environments can be seen in Figure 3.1 with a path between two coordinates. The path in the



**Figure 3.1:** Example of 2D configuration spaces with obstacles (brown) and a path (blue) from start (green) to finish (red).

grid space has to be specified as all the squares in the path, whereas the path in the polygonal space is specified only by the intermediate space. The polygonal domain is separated into free space and space occupied by polygonal obstacles.

### 3.1.1 Obstacle Representation

There are different options for how to represent obstacles. Real-environment obstacles are for example a tree, a pit, or a cluster of people, and can have complex borders that are hard to represent using geometric shapes. The obstacles can be approximated by shapes that are easier to describe with mathematical prescription.

A polygonal obstacle  $O$  is a polygon with  $l$  sides and can be defined by a sequence of vertices  $O = (\mathbf{o}_1, \dots, \mathbf{o}_l)$ ,  $\mathbf{o}_i = (x_i, y_i) \in \mathbb{R}^2$ . The number of vertices can be different for each obstacle, and the whole set  $\Omega$  of  $m$  obstacles is

$$\Omega = \{O_1, \dots, O_m\} = \{(\mathbf{o}_1^1, \mathbf{o}_2^1, \dots, \mathbf{o}_{l_1}^1), \dots, (\mathbf{o}_1^m, \mathbf{o}_2^m, \dots, \mathbf{o}_{l_m}^m)\},$$

$$m \in \mathbb{N}, \quad l_i \in \mathbb{N}^+ \quad \forall i \in \{1, \dots, m\}, \quad (3.1)$$

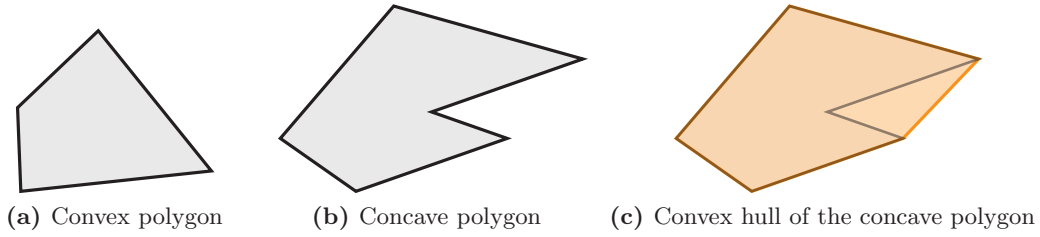


Figure 3.2: Example of convex and concave polygons.

In Figure 3.2, examples of convex (Figure 3.2a) and concave (Figure 3.2b) polygons along with a convex hull (Figure 3.2c) are shown. A convex hull is the encasement of a concave polygon  $P_c$  within a convex polygon  $P_x$  such that all points of  $P_c$  are contained in  $P_x$ .

## 3.2 Traveling Salesman Problem

The **TSP** is a well-known problem to find a path with minimum length, that visits a given set of  $n$  target locations  $C = \{\mathbf{c}_1, \dots, \mathbf{c}_n\}$  starting and finishing in the same position. Hence, the aim is to find the optimal sequence of visits to  $C$ .

The **TSP** formal definition is shown in Problem 1, where  $\mathbf{c}_i \in \mathbb{R}^2, i \in \{1, \dots, n\}$  is the location in 2D Euclidean space, and  $\Sigma$  is the visiting sequence being optimized. The distance between the goals is Euclidean and is denoted as  $\|\mathbf{c}_i - \mathbf{c}_j\|$  between locations  $\mathbf{c}_i, \mathbf{c}_j$ .

### Problem 1 (Traveling Salesman Problem (TSP))

$$\min_{\Sigma} \quad \|\mathbf{c}_{\sigma_n} - \mathbf{c}_{\sigma_1}\| + \sum_{i=1}^{n-1} \|\mathbf{c}_{\sigma_i} - \mathbf{c}_{\sigma_{i+1}}\|, \quad (3.2)$$

$$\text{s.t.} \quad \Sigma = (\sigma_1, \dots, \sigma_n), \quad (3.3)$$

$$\sigma_i \in \{1, \dots, n\}, \quad (3.4)$$

$$\sigma_i \neq \sigma_j \text{ for } i \neq j, \quad (3.5)$$

### 3.3 Traveling Salesman Problem with Neighborhoods

The **TSPN** is a variant of the **TSP**, where instead of visiting locations exactly, at least one point within their neighborhoods is visited. The path with minimum length is visiting a given set of  $n$  regions  $\mathcal{R} = \{R_1, \dots, R_n\}$ ,  $R_i \subset \mathbb{R}^2$ , determining the optimal visiting sequence  $\Sigma$ , and at the same time optimizing the points  $P$  within the regions.

The problem consists of two parts: finding the optimal  $\Sigma$  is a problem of combinatorial optimization, and finding the best points within the neighborhoods is a problem of continuous optimization. The **TSPN** formal definition is shown in Problem 2, where the optimization goal is to minimize the variable Euclidean distance between the points  $P$  within regions  $\mathcal{R}$ , visiting the target regions in the sequence  $\Sigma$ .

#### Problem 2 (Traveling Salesman Problem with Neighborhoods (TSPN))

$$\min_{\Sigma, P} \quad \|\mathbf{p}_{\sigma_n} - \mathbf{p}_{\sigma_1}\| + \sum_{i=1}^{n-1} \|\mathbf{p}_{\sigma_i} - \mathbf{p}_{\sigma_{i+1}}\| \quad (3.6)$$

$$\text{s.t.} \quad P = \{\mathbf{p}_1, \dots, \mathbf{p}_n\} \in \mathbb{R}^{n \times 2}, \quad (3.7)$$

$$\mathbf{p}_i \in R_i \quad \forall R_i \in \mathcal{R}, \quad (3.8)$$

$$\Sigma = (\sigma_1, \dots, \sigma_n), \quad (3.9)$$

$$\sigma_i \in \{1, \dots, n\}, \quad (3.10)$$

$$\sigma_i \neq \sigma_j \text{ for } i \neq j, \quad (3.11)$$

$$(3.12)$$

A variant of the **TSP** with disk-shaped neighborhoods (regions) is called the **CETSP**. The neighborhoods are given as a set of  $n$  target regions  $\mathcal{S} = \{S_1, \dots, S_n\}$ ,  $S_i = (\mathbf{c}_i, \delta_i)$  where  $\mathbf{c} \in \mathbb{R}^2$  is the center point and  $\delta \geq 0$  is the disk radius. The **CETSP** is to determine the shortest cyclic tour visiting all disk regions Problem 3 is the formulation of the **CETSP**. The optimization goal is the same as in the **TSPN**, but the cost is computed over locations in the goal's circular area given by the radius  $\delta$ .

#### Problem 3 (Close-Enough Traveling Salesman Problem)

$$\min_{\Sigma, P} \quad \|\mathbf{p}_{\sigma_n} - \mathbf{p}_{\sigma_1}\| + \sum_{i=1}^{n-1} \|\mathbf{p}_{\sigma_i} - \mathbf{p}_{\sigma_{i+1}}\| \quad (3.13)$$

$$\text{s.t.} \quad \|\mathbf{p}_i - \mathbf{c}_i\| \leq \delta_i \quad \forall i \in \{1, \dots, n\} \quad (3.14)$$

$$P = \{\mathbf{p}_1, \dots, \mathbf{p}_n\} \in \mathbb{R}^{n \times 2}, \quad (3.15)$$

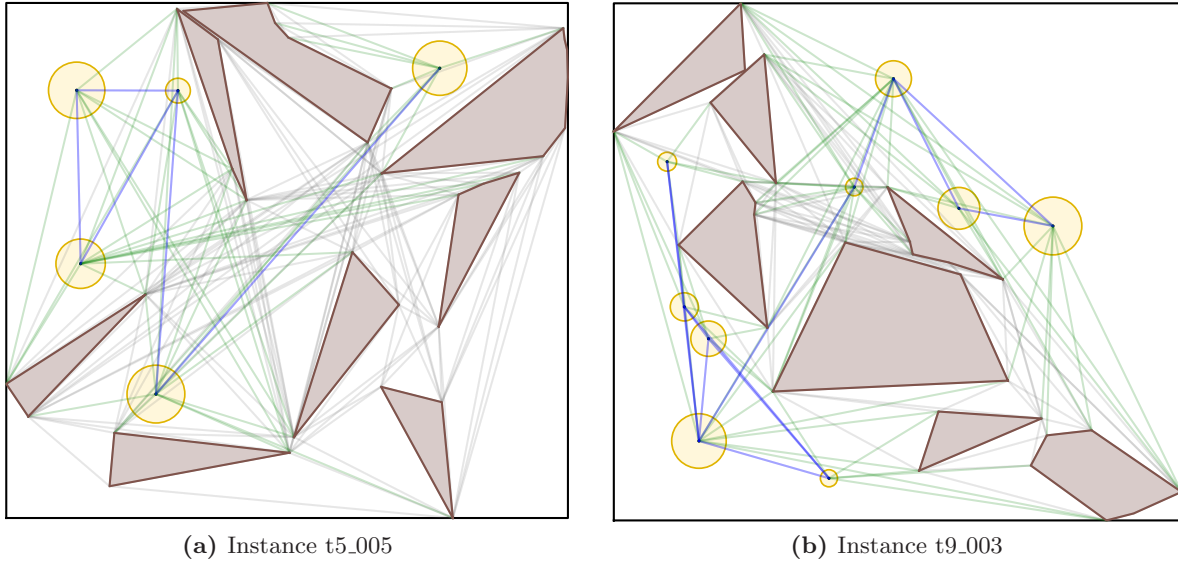
$$\Sigma = (\sigma_1, \dots, \sigma_n) \in \mathbb{Z}^n, \quad (3.16)$$

$$1 \leq \sigma_i \leq n, \quad (3.17)$$

$$\sigma_i \neq \sigma_j \text{ for } i \neq j. \quad (3.18)$$

### 3.4 Close-Enough Traveling Salesman Problem with Obstacles

The problem studied in this thesis is the Close-Enough Traveling Salesman Problem with Obstacles (**CETSP<sub>obs</sub>**), where the obstacles are defined as polygons. The aim of the **CETSP<sub>obs</sub>** is the same as the **CETSP**, but the found tour avoids the obstacles in the environment.



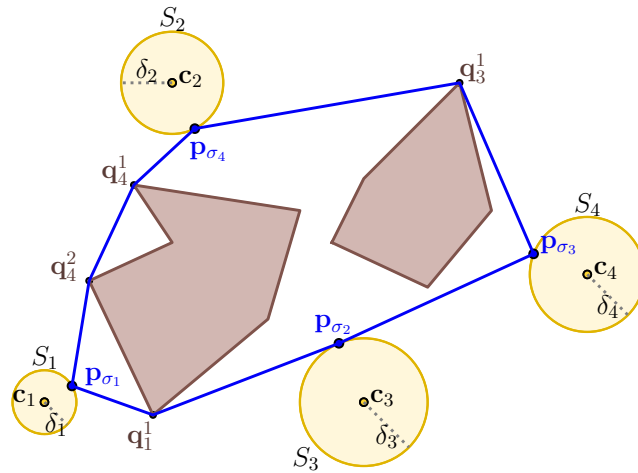
**Figure 3.3:** Example of the instances for the  $\text{CETSP}_{\text{obs}}$ . The target regions are shown as yellow disks, and the obstacles are shown as brown polygons. The green lines are the edges in a Visibility Graph (VG) and the blue lines show direct visibility between target region centers.

The instances used for such a problem are shown in Figure 3.3. In this case, a straight line path between two subsequent target regions might not be possible, and in such cases, the path consists of multiple line segments with intermediate points. We denote the set of intermediate points as

$$\begin{aligned}
 Q_i &= (\mathbf{q}_0^i, \mathbf{q}_1^i, \dots, \mathbf{q}_{k_i}^i), \\
 k_i &\in \mathbb{N}_0 \quad \forall i \in \{1, \dots, n\}.
 \end{aligned}
 \tag{3.19}$$

This denotes  $k_i$  consecutive intermediate points between  $\mathbf{p}_i$  and  $\mathbf{p}_j$ .

If  $k_i = 0$ , there are no intermediate points in between  $\mathbf{p}_i$  and  $\mathbf{p}_j$ . The notation used in this problem is shown in Figure 3.4.



**Figure 3.4:** Notation used for the  $\text{CETSP}_{\text{obs}}$ .

The addition of the intermediate points means that the distance is now computed as the sum of the length of the segments of the multi-segment line. The length with intermediate points denoted  $\mathcal{L}^*$  is defined as

$$\mathcal{L}^*(\mathbf{p}_i, \mathbf{p}_j) = \begin{cases} \|\mathbf{p}_i - \mathbf{p}_j\| & \text{if } k_i = 0, \\ \|\mathbf{p}_i - \mathbf{q}_i^1\| + \|\mathbf{q}_i^1 - \mathbf{p}_j\| & \text{if } k_i = 1, \\ \|\mathbf{p}_i - \mathbf{q}_i^1\| + \left( \sum_{l=1}^{k_i-1} \|\mathbf{q}_i^l - \mathbf{q}_i^{l+1}\| \right) + \|\mathbf{q}_i^{k_i} - \mathbf{p}_j\| & \text{if } k_i \geq 2, \\ \infty & \text{if obstacle interference.} \end{cases} \quad (3.20)$$

The formulation of the  $\text{CETSP}_{\text{obs}}$  is in Problem 4. The optimization goal is to minimize the distance function  $\mathcal{L}^*$  in between all target regions by optimization of the sequence  $\Sigma$ , the points position within target region  $P$  and the intermediate points on obstacles  $\mathcal{Q}$ .

#### Problem 4 (Close-Enough Traveling Salesman Problem with Obstacles)

$$\min_{\Sigma, P, \mathcal{Q}} \mathcal{L}^*(\mathbf{p}_{\sigma_n}, \mathbf{p}_{\sigma_1}) + \sum_{i=1}^{n-1} \mathcal{L}^*(\mathbf{p}_{\sigma_i}, \mathbf{p}_{\sigma_{i+1}}), \quad (3.21)$$

$$\text{s.t. } \|\mathbf{p}_i - \mathbf{s}_i\| \leq \delta_i \quad \forall i \in \{1, \dots, n\}, \quad (3.22)$$

$$P = \{\mathbf{p}_1, \dots, \mathbf{p}_n\} \in \mathbb{R}^{n \times 2}, \quad (3.23)$$

$$\mathcal{Q} = \{Q_1, \dots, Q_n\}, Q_i = \{q_1^i, \dots, q_{k_i}^i\}, \quad (3.24)$$

$$\Sigma = (\sigma_1, \dots, \sigma_n) \in \mathbb{Z}^n, \quad (3.25)$$

$$1 \leq \sigma_i \leq n, \quad (3.26)$$

$$\sigma_i \neq \sigma_j \text{ for } i \neq j. \quad (3.27)$$



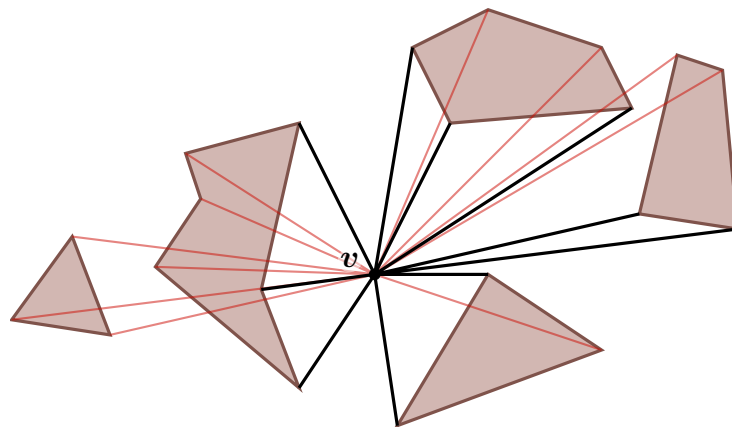


# Background

In this chapter, we summarize two concepts from computer science utilized in the proposed method for solving the Close-Enough Traveling Salesman Problem with Obstacles (CETSP<sub>obs</sub>). The concepts are described here for the thesis to be self-contained, and they are the Visibility Graph (VG) in Section 4.1, and Priority Queue (PQ) in Section 4.2. In the proposed method, the direct visibility area between two disks named **cone** is used, and to obtain it, the goniometric (trigonometric) functions are used. Since the concept is important but not necessary to understand the proposed method, it is described in this chapter in Section 4.3.

## 4.1 Visibility Graph

The VG is a well-known method from computational geometry [9], which can be used to find the shortest paths within a polygonal environment. The VG is a graph  $\mathcal{G}_{\text{vis}} = (V, E)$ , where the set of vertices  $V$  is obtained as all polygon vertices from the input map. The set of edges  $E$  contains all edges between vertices  $\mathbf{v} \in \mathbb{R}^2$ , which have direct visibility and are computed during the initialization of the VG. Direct visibility of an edge  $e = (\mathbf{v}_i, \mathbf{v}_j), \mathbf{v}_i, \mathbf{v}_j \in V$  means



**Figure 4.1:** VG from a single vertex  $\mathbf{v}$ . Brown polygons are the obstacles, black lines are edges  $E$  in the  $\mathcal{G}_{\text{vis}}$  from  $\mathbf{v}$ , and red lines are edges to the remaining vertices that are not visible from  $\mathbf{v}$ . Note that only a subset of  $E$  is displayed and the edges between obstacles are not shown for clarity.

that there is no obstacle intersecting  $e$ . The VG on a set of polygonal obstacles with a total of  $n$  vertices can be computed in  $\mathcal{O}(n^2 \log n)$  time [9]. An example of a simple visibility graph from one point is shown in Figure 4.1.

#### 4.1.1 Shortest Path in the Visibility Graph

The Dijkstra algorithm [9] is a graph algorithm for finding the shortest paths between the graph vertices. The graph can be for example the VG as described in Section 4.1. The shortest path between vertices of VG can be found using the Algorithm 1.

---

**Algorithm 1:** Shortest paths using the VG.

---

**Input:**  $\Omega$  – set of polygonal obstacles

$\mathbf{v}_{\text{start}}, \mathbf{v}_{\text{goal}}$  – starting and goal vertices

**Output:**  $P = (\mathbf{v}_{\sigma_1}, \dots, \mathbf{v}_{\sigma_k})$  – the shortest visiting sequence of  $k$  vertices

---

- 1  $\mathcal{G}_{\text{vis}} \leftarrow$  Visibility Graph from  $\Omega$  vertices  $\cup (\mathbf{v}_{\text{start}}, \mathbf{v}_{\text{goal}})$
  - 2 to every edge  $(\mathbf{v}_i, \mathbf{v}_j) \in \mathcal{G}_{\text{vis}}$  assign weight, which is the Euclidean distance  $|\mathbf{v}_i - \mathbf{v}_j|$
  - 3 use Dijkstra algorithm to compute a shortest path  $P$  between  $\mathbf{v}_{\text{start}}, \mathbf{v}_{\text{goal}}$  in  $\mathcal{G}_{\text{vis}}$
- 

## 4.2 Priority Queue

The PQ [36] is an abstract data structure for accessing the element with the extremal sorting value, which is usually the smallest or largest number. The PQ stores the elements and provides the following functionality.

**Retrieval of an element with an extremal value** The elements in the PQ have an assigned value. This functionality allows to retrieve the element with the extremal value.

**Peek to the element with an extremal value** The element stays in the queue, but its assigned sorting value can be obtained.

**Insertion of an element with assigned sorting value** The PQ must be able to insert more data into the storage and use the value for deciding the extremal element.

The defined operations put a need on the data structure for which it can be optimized. A possible solution would be to use a sorting function on a simple array, but the best computational complexity for sorting an array is  $\mathcal{O}(n \log n)$ . An example of an efficient implementation of the PQ is a Heap [36], which is represented as a tree with the extremal value in its root. The computational complexity for element insertion and retrieval is  $\mathcal{O}(\log n)$ , and for peek it is  $\mathcal{O}(1)$ .

## 4.3 Computation of Area Between Two Disks

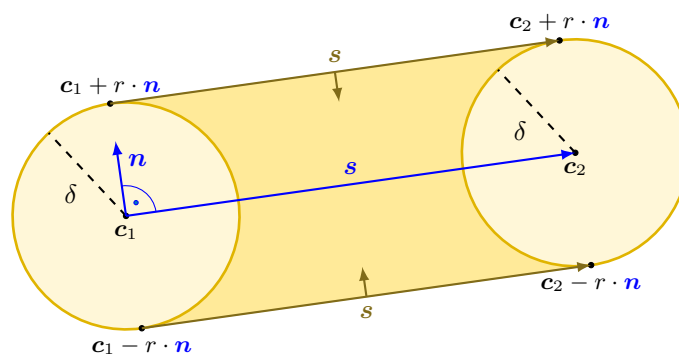
The area between two disks is a part of the proposed method as a constraint for the solution space and detecting possible path interference with obstacles. However, the computation is simply done by a combination of the goniometric functions, therefore it is shown here in advance for completeness. The computation of the area between two disks has computational complexity  $\mathcal{O}(1)$ .

The area between two disks, or two circles, encapsulates the area where the line connecting any two points from the first and second disk goes through. We call this area, which creates the solution space of the possible connecting lines, a **cone**. The cone is bordered by two line segments, which are tangent to both disks.

The cone can be calculated using goniometric functions and the computation differs based on the disks' radii because of the different angles of the tangents. In Section 4.3.1, we describe the computation of the cone when the disks have the same radii. In Section 4.3.2 the computation is described for different radii.

### 4.3.1 Equal Radii

In the case where the two disks have equal radii, the computation of the cone relies mostly on the line segment connecting their centers. The cone is encapsulated by two tangents, which are the same as the vector connecting the centers, shifted by its normal vector of the size of the radius, as illustrated in Figure 4.2. In this case, the tangents are parallel.



**Figure 4.2:** Area of direct visibility (cone) between two targets with same radii. The targets are the black circles with their centers  $\mathbf{c}_1, \mathbf{c}_2$  and radius  $\delta$ . The radii are denoted by the dashed line, the connecting line is denoted  $\mathbf{s}$ , the normal vector is denoted  $\mathbf{n}$ ,  $|\mathbf{n}| = 1 < r$ , and the cone is in yellow color, including the area of the disks.

The disks are identified with their centers  $\mathbf{c}_1, \mathbf{c}_2$  and radius  $\delta$ . The vector connecting  $\mathbf{c}_1$  and  $\mathbf{c}_2$  is calculated as

$$\mathbf{s} = \mathbf{c}_2 - \mathbf{c}_1, \quad (4.1)$$

and the normal vector  $\mathbf{n}$  with length 1, perpendicular to  $\mathbf{s}$  is calculated as

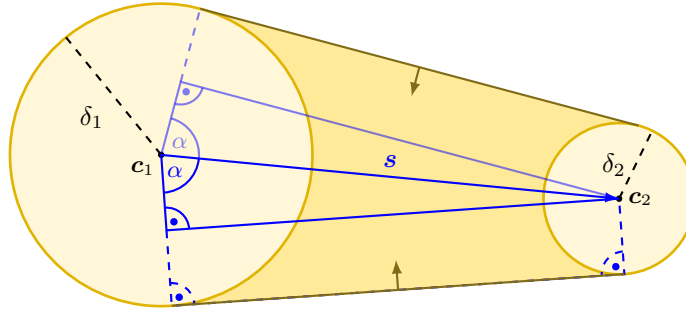
$$\mathbf{n} = \left( \frac{-s_2}{\|\mathbf{s}\|}, \frac{s_1}{\|\mathbf{s}\|} \right) \quad (4.2)$$

The border of the cone is then defined by the target regions, and the borders are defined as one of the two equal options:

- line segment between  $\mathbf{c}_1 \pm \delta \mathbf{n}$  and  $\mathbf{c}_2 \pm \delta \mathbf{n}$
- vector  $\mathbf{s}$  starting in  $\mathbf{c}_1 \pm \delta \mathbf{n}$

### 4.3.2 Non-equal Radii

In the case where the two disks have different radii, the computation of the cone is more difficult as additional angles need to be obtained. The cone and the angles necessary for its



**Figure 4.3:** Area of possible solution space (cone) between two targets with different radii. The targets are the black circles with their centers  $c_1, c_2$ , the radii are denoted by the dashed line, the connecting line is denoted  $s$ , the angles used for cone calculation are denoted  $\alpha, \beta$ , and the cone is in green color with dark green border.

computation are illustrated in Figure 4.3.

The disks are defined with their centers  $c_1, c_2$  and radii  $\delta_1, \delta_2$ . The vector  $s$  connecting  $c_1$  and  $c_2$  is calculated as in (4.1), and the angle of  $s$  with respect to the coordinate system denoted as  $\gamma$  is calculated as

$$\gamma = \arctan\left(\frac{s_2}{s_1}\right). \quad (4.3)$$

The angle of the border, tangent to both disks is denoted  $\alpha$ , and it is placed as shown in Figure 4.3. It can be seen that there is a right-angle triangle and that the vector  $s$  is its hypotenuse, and the difference  $\delta_1 - \delta_2$  is the length of one of its sides. With this information, the angle  $\alpha$  can be obtained as

$$\alpha = \arccos\left(\frac{|\delta_1 - \delta_2|}{\|s\|}\right) \quad (4.4)$$

The borders are then defined as the line segment between  $c_1 + \delta_1 n_{\pm}$  and  $c_2 + \delta_2 n_{\pm}$ , where  $n_+$  and  $n_-$  are two vectors defined as

$$n_{\pm} = (\cos(\gamma \pm \alpha), \sin(\gamma \pm \alpha)). \quad (4.5)$$

# Proposed Method

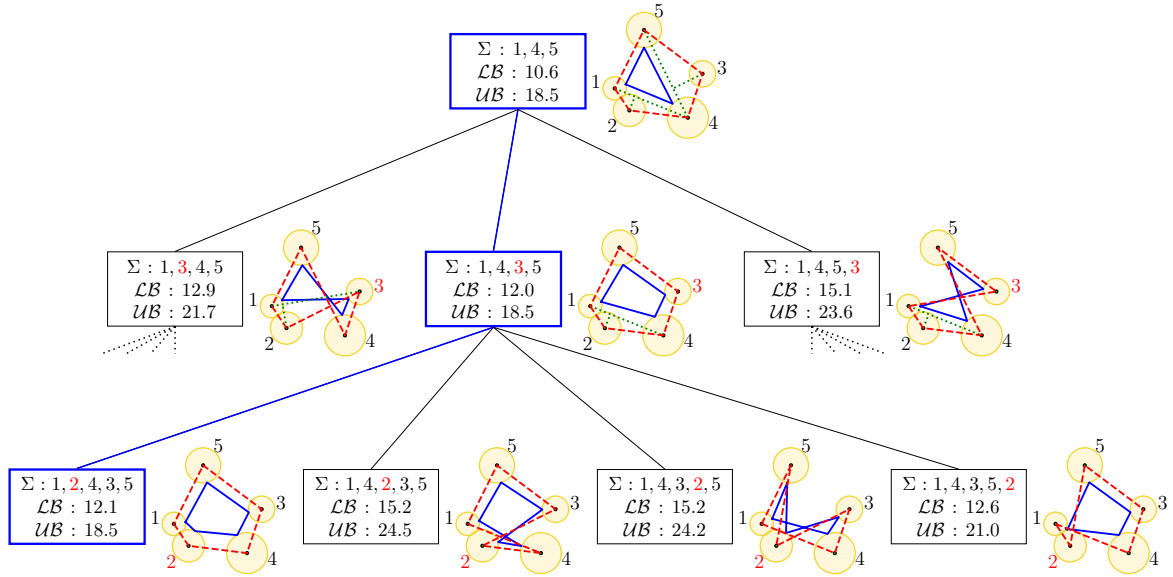
In this chapter, the proposed method for solving the Close-Enough Traveling Salesman Problem with Obstacles (**CETSP<sub>obs</sub>**) is described. We approach the problem with the lower bound estimates and the exact algorithm to obtain the best solution quality. The proposed method is based on the Branch-and-Bound (**BnB**) algorithm, and it is presented in Section 5.1. The **CETSP<sub>obs</sub>** consists of the problem of finding the optimal sequence which is a problem from combinatorial optimization, and of finding the best point within the neighborhood, which is a problem from continuous optimization. The **BnB** algorithm branches on subsequences of the given target regions, and two methods for finding solutions on fixed target regions sequence have been proposed. The first method samples the target regions, therefore reducing the problem from continuous optimization to combinatorial optimization, and it is described in Section 5.2. Second, the mathematical model for representing the problem along with the explanation of the used obstacle constraints is presented in Section 5.3

## 5.1 Branch-and-Bound for the Sequence Optimization

There are different approaches to sequence optimization for the Traveling Salesman Problem (**TSP**) (Section 3.2). Since the problem is NP-hard and the number of possible solutions is  $(n - 1)!$  where  $n$  is the number of targets, testing all solutions to find the optimal using the brute force search becomes intractable with the increasing number of target regions. Branch-and-Bound (**BnB**) is an algorithm for solving problems from mixed-integer and combinatorial optimization. The **BnB** approach in this thesis for solving the **CETSP<sub>obs</sub>** is based on [3], which is used for solving the Close-Enough Traveling Salesman Problem (**CETSP**) to optimality. An example of the **BnB** algorithm for a small **CETSP** instance is illustrated in Figure 5.1.

The **BnB** algorithm creates and expands a solution tree consisting of nodes with partial problem solutions. The solution tree is created by branching the already existing nodes, where the next node for branching is selected as the one with the smallest lower bound. Since the resulting solution tree could expand until it would contain all of the many possible sequences, the **BnB** uses the bounding of the nodes to reduce the size of the search tree.

Some terminology is used for describing the proposed **BnB** approach for solving the **CETSP<sub>obs</sub>** in this thesis. The terms are as follows, marked by the bold text.



**Figure 5.1:** BnB example on a CETSP instance. The BnB is first initialized with three targets and then branched by adding the next target into the sequence  $\Sigma$ . Then, by the priority queue, the next node with the smallest lower bound  $\mathcal{LB}$  is examined and branched using the **branching rule**. The next popped node contains all targets in  $\Sigma$ , therefore the BnB algorithm ends and returns the solution with  $\Sigma = (1, 2, 4, 3, 5)$ . Nodes are shown in rectangles, and the popped nodes have blue outlines. For each node, the corresponding solution of the instance is shown next to it on the right with targets as yellow disks, the  $\mathcal{LB}$  as the blue line, and the upper bound  $\mathcal{UB}$  as the red line. The green dotted lines show the heuristic insert of targets for the  $\mathcal{UB}$  (Algorithm 3).

**Node** Denoted as  $\mu$ , the node of the solution tree is a **partial problem**. The partial problem contains an ordered subset of the given target regions, whereas the full problem of the  $\text{CETSP}_{\text{obs}}$  contains all given target regions. Each node is unique by the sequence of target regions, and it contains the lower and upper bound solutions for this sequence. If the node is considered as promising based on its lower bound, the node is branched with the **branching rule**, creating further nodes in the search tree.

**Branching rule** The rule is used for creating children nodes from a parent node. In our case in branching, the sequence is extended with an additional target region, which is selected to maximize the lower bound. Since it can be at different positions in the sequence, the target region is inserted into each position to create multiple node children. The node children are inserted into the Priority Queue (PQ) and can be examined as part of the branching, which is the expansion of the subproblems to eventually contain all the target regions.

**Solution** In the implementation, the **Solution** is a structure containing the following information: the solution cost  $\mathcal{C}$ , the visited target regions  $\Sigma$ , the sequence  $\Psi$  including both the target regions and intermediate obstacle points, and lastly the exact points  $P$  on which the cost was computed. The lower bound contains a solution with the goals which are then branched upon, and the upper bound contains a feasible solution containing all the target regions.

**Lower bound** Denoted  $\mathcal{LB}$ , the  $\mathcal{LB}$  is a possibly infeasible solution to the  $\text{CETSP}_{\text{obs}}$  which

can contain only a subset of the target regions. It has a lower cost than or at most equal cost to the optimal solution to the full problem. Its exact form in this thesis is described in Section 5.1.2.

**Upper bound** Denoted  $\mathcal{UB}$ , and it is a solution to the full problem with higher than or equal cost to the optimal solution to the full problem. The  $\mathcal{UB}$  solution is feasible and can be accepted as the solution to the problem if the time limit is exceeded, therefore making the **BnB** algorithm an any-time algorithm. Its exact form in this thesis is described in Section 5.1.3.

**Global bounds** The  $\mathcal{LB}$  and  $\mathcal{UB}$  are obtained for each node. The bounds of nodes in the search tree update the global bounds of the **BnB** algorithm, and the global bounds are denoted as  $\mathcal{LB}$  and  $\mathcal{UB}$ . First, the global bounds can be initialized as:  $\mathcal{LB} = 0, \mathcal{UB} = \infty$ . Whenever a node  $\mu$  is created and if  $\mu.\mathcal{LB} \leq \mathcal{UB}$ , the global bounds are updated as

$$\mathcal{LB} = \max(\mathcal{LB}, \mu.\mathcal{LB}), \quad (5.1)$$

$$\mathcal{UB} = \min(\mathcal{UB}, \mu.\mathcal{UB}). \quad (5.2)$$

**Bounding** The bounding is a crucial part of the **BnB** algorithm, as it eliminates unpromising partial solutions from the solution tree. It uses the global  $\mathcal{UB}$  for node elimination if for the node  $\mu$ ,  $\mu.\mathcal{LB} > \mathcal{UB}$ . This bounding can be applied because for any node  $\mu$ , the inequality

$$\mu.\mathcal{LB} \leq \mu.\mathcal{C} \leq \mu.\mathcal{UB} \quad (5.3)$$

is valid due to the  $\mathcal{LB}$  and  $\mathcal{UB}$  definition, where  $\mu.\mathcal{C}$  is the found solution to the node target region sequence  $\mu.\Sigma$ .

The overview of the method is described in Algorithm 2. First, the initial node is created with three target regions that maximize the  $\mathcal{LB}$  as described in Section 5.1.1. Whenever a new **BnB** node  $\mu$  is created, it is added to the **PQ** as described in Section 4.2 with the sorting value being  $\mu.\mathcal{LB}$ . From the initial node  $\mu_{\text{init}}$ , new nodes are branched using the **branching rule**. The computation of the  $\mathcal{LB}$  estimates is described in detail in Section 5.1.2, and the  $\mathcal{UB}$  computation is described in Section 5.1.3. Each new node not eliminated by the **bounding rule**, is added to the **PQ** (children of the last examined node), and global  $\mathcal{LB}$  and  $\mathcal{UB}$  are updated as in (5.1) and (5.2). The next examined node is the one with the smallest  $\mathcal{LB}$  retrieved from the **PQ**.

If the bounds are close to the solution, they are called tight bounds. The tight bounds are desired because, with tight bounds, the **BnB** algorithm can find the optimal solution faster. This is because the nodes with the  $\mathcal{LB}$  higher than the global  $\mathcal{UB}$  are cut off earlier, resulting in fewer branches and thus a smaller search tree.

The **BnB** algorithm finishes successfully when all nodes are either opened or bounded resulting in an empty **PQ**, or unsuccessfully if the maximum computation time limit is reached. The leaf node is the node  $\mu_{\text{leaf}}$  that has all the target regions added in the  $\Sigma$ , therefore the solution to the full problem using the proposed method can be found. The solution is then used to update the global  $\mathcal{UB}$  which can be retrieved as a feasible solution at any time during the tree search.

**Algorithm 2:** Proposed Branch-and-Bound algorithm for the  $\text{CETSP}_{\text{obs}}$ .

---

**Input:**  $\mathcal{S} = \{S_1, \dots, S_n\}$  – set of target regions  
 $\mathcal{VG}$  – Visibility Graph between obstacle points and target regions center  
 $t_{\max}$  – maximum computation time

**Output:**  $\mathcal{LB}, \mathcal{UB}$  – global lower and upper bound solutions.

---

```

1  $\mu_{\text{root}} \leftarrow$  empty BnB node
2  $\mu_{\text{root}}.\Sigma \leftarrow$  select root sequence from  $\mathcal{S}$  using  $\mathcal{VG}$  // (5.4)
3  $\mu_{\text{root}}.\mathcal{LB}, \mu_{\text{root}}.\mathcal{UB} \leftarrow$  compute bounds from  $\mu_{\text{root}}.\Sigma$  // Sections 5.1.2 and 5.1.3
4  $\mathcal{LB}, \mathcal{UB} \leftarrow \mu_{\text{root}}.\mathcal{LB}, \mu_{\text{root}}.\mathcal{UB}$  // initialize global bounds
5  $\mathcal{PQ} \leftarrow [\mu_{\text{root}}]$  // PQ of BnB nodes initialized with  $\mu_{\text{root}}$ 
6 while  $\mathcal{PQ}$  is not empty and  $t < t_{\max}$  do
7    $\mu \leftarrow$  get node with smallest  $\mathcal{LB.C}$  from  $\mathcal{PQ}$ 
8   if  $\mu.\mathcal{LB}.\Sigma$  contains all goals then
9      $\mathcal{P} \leftarrow$  proposed solution on  $\mu.\mathcal{LB}.\Sigma$  // Section 5.1.4
10    if  $\mathcal{P.C} < \mathcal{UB.C}$  then
11       $\mathcal{UB} \leftarrow \mathcal{P}$  // update global upper bound
12    end
13  end
14  // branching rule - insert next goal into  $\mu.\Sigma$  that maximizes the distance
15   $\nu \leftarrow$  target region furthest from  $\mu.\mathcal{LB}.\Sigma$  // next goal for insertion
16  children  $\leftarrow \{\}$ 
17  foreach  $i \in \{1, \dots, |\mu.\Sigma|\}$  do // insert goal candidate into each index
18     $\mu' \leftarrow$  empty BnB node
19    // insert  $\nu$  to sequence  $\mu'.\Sigma$  at index  $i$ 
20     $\mu'.\Sigma \leftarrow (\mu'.\sigma_1, \dots, \mu'.\sigma_i, \nu, \mu'.\sigma_{i+1}, \dots, \mu'.\sigma_{|\mu.\Sigma|})$ 
21     $\mu'.\mathcal{LB}, \mu'.\mathcal{UB} \leftarrow$  compute bounds from  $\mu'.\Sigma$  // Sections 5.1.2 and 5.1.3
22    add  $(\mu')$  to children
23    // update global bounds
24     $\mathcal{LB} \leftarrow \arg \max_{\mu'.\mathcal{LB}, \mathcal{LB}} (\mu'.\mathcal{LB.C}, \mathcal{LB.C})$ 
25     $\mathcal{UB} \leftarrow \arg \min_{\mu'.\mathcal{UB}, \mathcal{UB}} (\mu'.\mathcal{UB.C}, \mathcal{UB.C})$ 
26  end
27  // bounding
28  children  $\leftarrow$  filter children using  $(\lambda x \rightarrow x.\mathcal{LB.C} \leq \mathcal{UB.C})$ 
29  foreach child in children do
30    insert child into  $\mathcal{PQ}$ 
31  end
32 end
33 return  $\mathcal{LB}, \mathcal{UB}$  // solution after computational time limit reached

```

---

### 5.1.1 Initialization

The initialization of the BnB for the  $\text{CETSP}_{\text{obs}}$  is shown in Algorithm 2 on lines 1–5. The initial solution is selected to maximize the global  $\mathcal{LB}$ . The initial three targets  $(S_i, S_j, S_k)$  in



Algorithm 2 on Line 2 are selected as

$$(S_i, S_j, S_k) = \arg \max_{(S_i, S_j, S_k) \in \mathcal{S}} (\mathcal{L}^*(S_i, S_j) + \mathcal{L}^*(S_j, S_k) + \mathcal{L}^*(S_k, S_i)). \quad (5.4)$$

For every combination of three target regions, get the tour length, and the order of the targets does not matter since the distance is symmetric and the tour is a cycle. The number of possible combinations is calculated with the binomial coefficient  $\binom{n}{3}$  [37], and selecting the root sequence can be implemented with a three-for-loop algorithm, thus the computational complexity is  $\mathcal{O}(n^3)$ .

### 5.1.2 Lower Bound Estimates

In this thesis, we propose several methods to address the lower bound estimation. The lower bound, denoted  $\mathcal{LB}$ , is a possibly infeasible solution to the  $\text{CETSP}_{\text{obs}}$  with the cost lower or equal to the final solution.

Because of this condition, the used lower bound must be smaller than the optimal solution. We study the three following options to estimate the  $\mathcal{LB}$  for subsequences of partial problems.

**$\mathcal{LB}_{\text{SOCP}}$**  The Second-Order Cone Program (SOCP) [3] (Model 5.3.1) model-based lower bound, and the solution to the  $\text{CETSP}$  [3]. It disregards the obstacles and solves the relaxed problem, therefore it is possibly infeasible in the  $\text{CETSP}_{\text{obs}}$  if an obstacle interferes with the solution.

**$\mathcal{LB}_{\text{Euclid}}$**  For every two subsequent target regions  $S_i = (\mathbf{c}_i, \delta_i), S_j = (\mathbf{c}_j, \delta_j)$  in  $\Sigma$ , we determine the shortest path disregarding the obstacles, calculated as

$$\mathcal{LB}_{\text{Euclid}}(S_i, S_j) = \|\mathbf{c}_i - \mathbf{c}_j\| - \delta_i - \delta_j. \quad (5.5)$$

The  $\mathcal{LB}_{\text{Euclid}}$  is an infeasible solution for the original problem because the result is not a continuous path. It also has a property that the  $\mathcal{LB}$  of a child node can be lower than its parent because of the radii subtraction, which breaks the triangle inequality.

**$\mathcal{LB}_{\text{VG}}$**  For every two subsequent target regions  $S_i, S_j$  in  $\Sigma$ , we determine the shortest path found with the Visibility Graph (VG), calculated using the  $\mathcal{L}^*$  (3.20) as

$$\mathcal{LB}_{\text{VG}}(S_i, S_j) = \mathcal{L}^*(\mathbf{c}_i, \mathbf{c}_j) - \delta_i - \delta_j. \quad (5.6)$$

This bound is tighter than the  $\mathcal{LB}_{\text{Euclid}}$ , and it is also infeasible and has the same property as  $\mathcal{LB}_{\text{Euclid}}$ . The  $\mathcal{LB}$  is used for sorting the PQ and if when the children nodes have smaller  $\mathcal{LB}$  than the parent nodes, it can result in a more depth-first search.

### 5.1.3 Upper Bound Computation

The upper bound, denoted  $\mathcal{UB}$ , is a feasible solution of each node to the  $\text{CETSP}_{\text{obs}}$  problem containing all the target regions, with the cost higher or equal to the optimal solution. It should be fast to compute since it is used as a bound for each node, but should also be close to the final solution.

We propose two methods to compute a tight upper bound but first, we need to add the remaining target regions to the sequence  $\mu.\Sigma$  for which we use a heuristic insert procedure described Algorithm 3.

---

**Algorithm 3:** Heuristic sequence insert of target regions.
 

---

**Input:**  $\mathcal{S} = \{S_1, \dots, S_n\}$  – set of target regions  
 $\Sigma = \{\sigma_1, \dots, \sigma_k\}, k \leq n$  – the sequence from node  $\mu$  of goals already inserted  
 $\mathcal{VG}$  – Visibility Graph between obstacle points and target regions center

**Output:** updated  $\Sigma'$  containing all  $S \in \mathcal{S}$

---

```

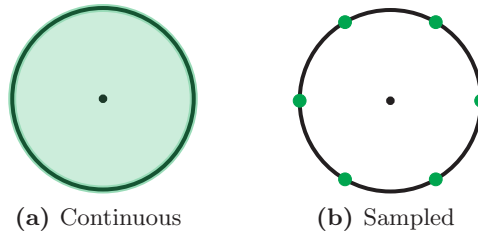
1  $A \leftarrow \{\sigma \mid \sigma \in \{1, \dots, n\}, \sigma \notin \Sigma\}$  // select indexes that are not already in  $\Sigma$ 
2  $\Sigma' = \{\sigma'_1, \dots, \sigma'_k\} \leftarrow$  copy of  $\Sigma$ 
3 foreach  $a \in A$  do
4    $d_{\min} \leftarrow \infty$  // minimum difference by adding the goal
5    $i_{\min} \leftarrow 0$  // insertion index leading to  $d_{\min}$ 
6   // compute difference created by inserting  $a$  at index  $i$  in  $\Sigma'$ 
7   foreach  $i \in \{1, \dots, |\Sigma'|\}$  do
8     // the path between  $(S_{\sigma'_i}, S_{\sigma'_{i+1}})$  can contain intermediate points  $q$  on
9     // the obstacle borders, creating a multi-line segment.
10    foreach line segment between  $(S_{\sigma'_i}, S_{\sigma'_{i+1}})$  computed by  $\mathcal{VG}$  do
11       $d \leftarrow$  distance of target region  $S_a$  from the line segment
12      if  $d < d_{\min}$  then // update minimum difference
13         $d_{\min} \leftarrow d$ 
14         $i_{\min} \leftarrow i$ 
15      end
16    end
17  end
18  // update  $\Sigma'$  with  $a$  inserted at  $i_{\min}$ th index
19   $\Sigma' \leftarrow \{\sigma'_1, \dots, \sigma'_{i_{\min}}, a, \sigma'_{i_{\min}+1}, \dots, \sigma'_{|\Sigma'|}\}$ 
20 end
21 return  $\Sigma$ 

```

---

### 5.1.4 Proposed Solution

We propose to use two approaches to solve the  $\text{CETSP}_{\text{obs}}$ , which differ in the possible points on target regions, see Figure 5.2 – and  $\text{MINLP}$  model approach in Figure 5.2a, and the  $\text{Sampled}$ -based method in Figure 5.2b. Both approaches for finding a solution are decoupled - they are computed on the fixed target regions sequence.



**Figure 5.2:** Solution points on target disk. In the continuous case, the points produced by the mathematical model can be anywhere within the green area including the border as shown in Figure 5.2a. In the sampled case, the point visiting the disks is one of the six uniformly sampled points on the disk border, denoted by the green dots in Figure 5.2b.

In the **Sampled**-based method, target region borders are sampled, and the problem is then to select one sample for each region to minimize the tour cost. The approach for finding the best samples is described in Section 5.2. In the **MINLP** model, the mathematical model optimizes the points within the target regions, as described in Section 5.3. The target regions can be initialized either by the result of the **Sampled**-based method, or with the sequence of target regions itself, and adding intermediate points when necessary with the visibility graph.

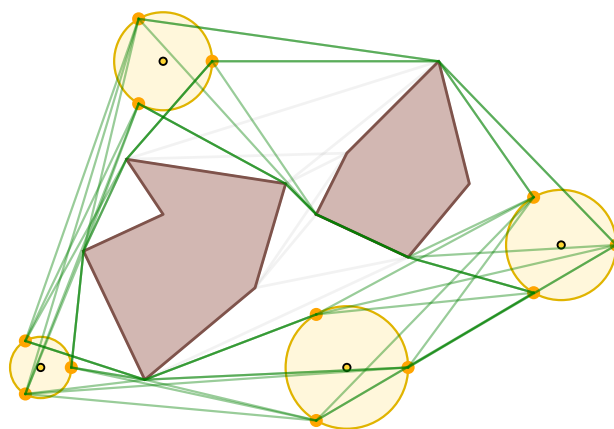
## 5.2 Sampled-Based Method

In the **Sampled**-based method, we transform the continuous target regions into sets of samples, as shown in Figure 5.2b. This transforms the continuous optimization problem of point selection into a combinatorial one, namely from **CETSP<sub>obs</sub>** to Generalized Traveling Salesman Problem (**GTSP**) with non-Euclidean distances provided by the **VG**. The tour still has to visit all the target regions, but now it should visit one of the sampled points.

The samples  $K_i$  for target region  $S_i$  are selected uniformly on the target region border as

$$K_i = \{\mathbf{p}_1^i, \dots, \mathbf{p}_k^i\}, \mathbf{p}_j^i = \mathbf{c}_i + \delta_i(\cos(j\theta), \sin(j\theta)), \theta = \frac{2\pi j}{k} \quad \forall j \in \{1, \dots, k\}, \quad (5.7)$$

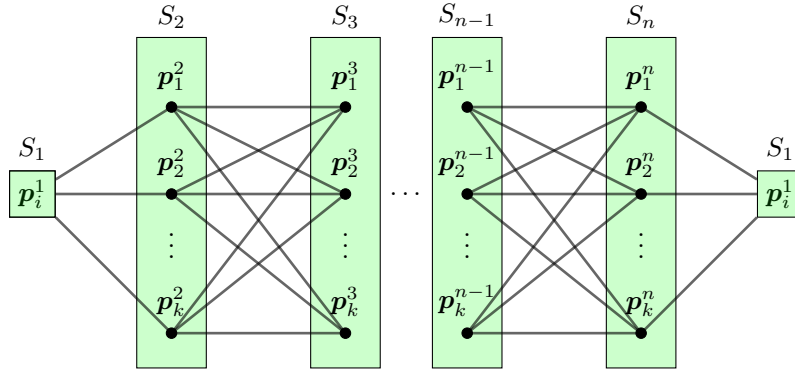
The sampling of the region border is sufficient since it is visited even if the solution goes through the target region. With the number of samples approaching infinity, the solution is equal to the optimized version. An illustration of target sampling in an example instance is shown in Figure 5.3.



**Figure 5.3:** Instance with sampled target regions, where the number of samples on the border is three, shown by the orange dots. The green lines show all the combinations of paths between two subsequent target regions.

When the samples are created and points are fixed, the **VG** is used to find the shortest path between every two samples. Then, only the best samples for each position need to be determined. This is done by the auxiliary graph as shown in Figure 5.4, with  $\mathcal{O}(nk)$  computational complexity to find the optimal solution, where  $n$  is the number of targets and  $k$  is the number of samples in each target region. By using this graph approach, the solution found on the samples is the best possible within the sampled space.

The computational time and solution cost of this approach are dependent on the number of samples. With a smaller number of samples, it is relatively fast and straightforward to



**Figure 5.4:** Auxiliary graph with target samples. The distance is calculated using the VG from the first target region  $S_1$  point  $p_i^1$ , where  $i$  is the selected sample, in a cycle again up to  $p_i^1$ . This computation is repeated for all points from  $S_1$ , and the best-found length from all of the points  $p_i^1, i \in \{1, \dots, k\}$  is the optimal length within the sampled space.

compute. With a higher number of samples, the computation takes longer time since also the VG is computed with a higher number of vertices. By sampling the target region the unlimited number of visiting point placements in the continuous domain is limited to only the sample number, therefore some possibly better cost solutions in the continuous neighborhood are lost.

### 5.3 Mixed Integer Non-Linear Program for the $\text{CETSP}_{\text{obs}}$

In this section, we define the mathematical model as the Mixed Integer Non-Linear Program (MINLP) for the  $\text{CETSP}_{\text{obs}}$  when the sequence of visits is fixed. The model for finding the optimal solution to the  $\text{CETSP}$  [3] is described in Section 5.3.1. This model is formulated as a problem from convex optimization, therefore the found optimum is the global optimum of the problem.

The model for the  $\text{CETSP}$  does not consider obstacles, and therefore we propose an extension of this model with added constraints for polygonal obstacles. The obstacle constraints, along with the full model are in Section 5.3.2. The obstacle constraints however are added only when the obstacle interferes with the solution between each two subsequent regions, as described in Section 5.3.3. Lastly, the model only optimizes the points in the input target regions. In some instances, the solution quality might increase if further obstacle points are included in the solution, and the process of adding the points is described in Section 5.3.4.

#### 5.3.1 Second-Order Cone Program for the $\text{CETSP}$

The problem of the  $\text{CETSP}$  is defined as the minimization of the tour length. Note that the sequence  $\Sigma$  would determine the numbering of the target regions  $S$ . In this section, the numbering of  $S$  by  $\Sigma$  is omitted for cleaner definitions. The sequence of the target regions  $S'$  is fixed and is denoted as

$$\begin{aligned} S' &= (S_1, \dots, S_k), \\ S_i &= (\mathbf{c}_i, \delta_i), \mathbf{c}_i \in \mathbb{R}^2, \delta_i \geq 0 \quad \forall i \in \{1, \dots, k\}, k \leq n. \end{aligned} \quad (5.8)$$

The optimization goal is the sum of distances  $\mathbf{f} = \{f_1, \dots, f_k\}$  between the points  $\mathbf{x}$  found in the circular area from center  $\mathbf{c}$  with radius  $\delta$ . Due to the tour cyclicity, we define that the target region with index  $k + 1$  to be equivalent to the target with index 1 as

$$\mathbf{x}_{k+1} \triangleq \mathbf{x}_1. \quad (5.9)$$

The objective goal and constraints with helper variables are defined in Model 5.3.1.

### Model 5.3.1 (SOCP for CETSP)

$$\min_X \sum_{i=1}^n f_i, \quad (5.10)$$

$$\text{s.t. } f_i^2 \geq \mathbf{w}_i^T \mathbf{w}_i \quad \forall i \in \{1, \dots, n\}, \quad (5.11)$$

$$\mathbf{w}_i = \mathbf{x}_{i+1} - \mathbf{x}_i \quad \forall i \in \{1, \dots, n\}, \quad (5.12)$$

$$\mathbf{v}_i^T \mathbf{v}_i \leq \delta_i^2 \quad \forall i \in \{1, \dots, n\}, \quad (5.13)$$

$$\mathbf{v}_i = \mathbf{c}_i - \mathbf{x}_i \quad \forall i \in \{1, \dots, n\}. \quad (5.14)$$

$$f_i \in \mathbb{R}, \quad \mathbf{w}_i \in \mathbb{R}^2, \quad \mathbf{x}_i \in \mathbb{R}^2, \quad \mathbf{v}_i \in \mathbb{R}^2 \quad \forall i \in \{1, \dots, k\}.$$

The decision variables are the points in the target regions  $X = \{\mathbf{x}_1, \dots, \mathbf{x}_k\}$ . The auxiliary variable  $\mathbf{w}_i$  (5.12) contains the vector difference of the two subsequent points  $\mathbf{x}_i, \mathbf{x}_{i+1}$ , and  $f_i$  (5.11) represents the minimized length of vector  $\mathbf{w}_i$ . They are used to calculate the tour cost in the objective function (5.10) as a sum over all variables  $\mathbf{f}$ . The variable  $\mathbf{v}_i$  (5.13) contains the vector difference of the point  $\mathbf{x}_i$  and its corresponding target region center  $\mathbf{c}_i$ . The length of vector  $\mathbf{v}_i$  must be smaller than the target region radius  $\delta_i$  (5.14). The model expects an obstacle-free environment, and it can be extended using obstacle constraints to find feasible solutions.

### 5.3.2 Polygonal Obstacles Constrained in a Half-Plane

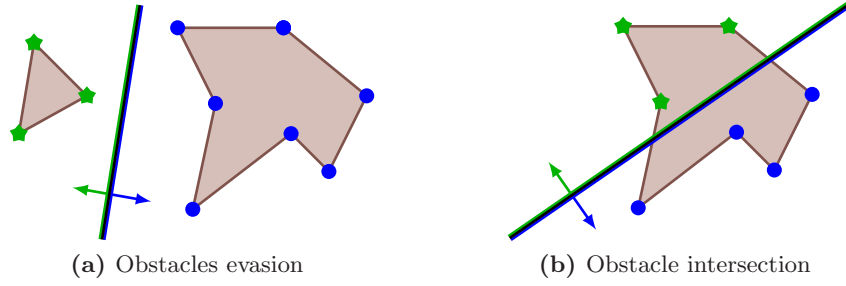
The obstacle constraints are used in the mathematical model, therefore they need to be in such a form to be solvable by the current solvers. The well-studied obstacle representation is the polygon. The polygon is usually represented as a sequence of points that are connected by line segments, and it can have an arbitrary number of sides. In the proposed approach, we use only the polygon points to determine an intersection with the solution.

Figure 5.5 shows an example of the solution line segment generating two half-planes and possible obstacle positions. The line segment is created by the two sequential points  $\mathbf{x}_i, \mathbf{x}_{i+1}$ , and the half-plane is separated by the corresponding line. If all the points of the obstacle border are in one half-plane, the solution does not interfere with the obstacle, as illustrated in Figure 5.5a, otherwise the obstacle intersects the solution as in Figure 5.5b.

The line can be prescribed by a parametric equation  $ax_1 + bx_2 + c = 0$ , where  $a, b, c \in \mathbb{R}$  are the defining constants. The line consists of all points  $\mathbf{p} \in \mathbb{R}^2$  for which the equation is valid. Prescription of the line generated by segment of points  $\mathbf{x}_{i+1}, \mathbf{x}_i \in \mathbb{R}^2$  are

$$-d_2 p_1 + d_1 p_2 + q = 0, \quad (5.15)$$

$$\mathbf{d} = \mathbf{x}_{i+1} - \mathbf{x}_i, \quad q = d_2 x_{i,1} - d_1 x_{i,2}. \quad (5.16)$$



**Figure 5.5:** Points in half-planes generated by a solution line segment. The line with blue-green outlines represents the border between half-planes. Brown polygons are the obstacles with their vertices shown with a dot or a star symbol, and the colored symbol represents half-plane affiliation.

The meaning of the left side of (5.15) for point  $\mathbf{p} \in \mathbb{R}^2$  anywhere in the plane is

$$-d_2 p_1 + d_1 p_2 + q \quad \begin{cases} > 0 & \mathbf{p} \text{ is in the first half-plane,} \\ = 0 & \mathbf{p} \text{ is on half-plane border,} \\ < 0 & \mathbf{p} \text{ is in the other half-plane.} \end{cases} \quad (5.17)$$

We want the obstacle to be on one side of the half-plane or the other. For this reason, we model the constraints using the Big-M method, which by using a large constant  $M$  and binary variable  $y \in \{0, 1\}$  activates one of the two half-plane constraints. The constraints for a single obstacle  $O = \{\mathbf{o}_1, \dots, \mathbf{o}_l\}$  being in one of the half-planes generated by solution parametric line between every two subsequent points  $\mathbf{x}_i, \mathbf{x}_{i+1}$  are

$$-d_{i,2} o_{j,1} + d_{i,1} o_{j,2} + q_i \leq 0 + M y_i \quad \forall i \in \{1, \dots, n\}, j \in \{1, \dots, l\}, \quad (5.18)$$

$$-d_{i,2} o_{j,1} + d_{i,1} o_{j,2} + q_i \geq 0 - M(1 - y_i) \quad \forall i \in \{1, \dots, n\}, j \in \{1, \dots, l\}, \quad (5.19)$$

$$\mathbf{d}_i = \mathbf{x}_{i+1} - \mathbf{x}_i \quad \forall i \in \{1, \dots, n\}, \quad (5.20)$$

$$q_i = d_{i,2} x_{i,1} - d_{i,1} x_{i,2} \quad \forall i \in \{1, \dots, n\}, \quad (5.21)$$

$$\mathbf{o}_j \in O \quad \forall j \in \{1, \dots, l\}, \quad (5.22)$$

The mathematical model is defined as the minimization of the length tour with a given set of target regions  $\mathcal{S}$  as in (5.8), and a set of obstacles  $\Omega$  (3.1). The set of obstacles is defined as a set of polygons defined by points in order, where each polygon can have a different number of sides.

The optimization goal is the sum of distances  $\mathbf{f}$  between the points  $\mathbf{x}$  found in the circular area from center  $\mathbf{c}$  with radius  $\delta$ . The constraints are added for each of two consecutive target regions  $S_i, S_{i+1}$  if the obstacle is within their cone (further described in Section 5.3.3). The tour is cyclic, therefore (5.9) holds. The objective goal and constraints with auxiliary variables are defined in Model 5.3.2.

**Model 5.3.2 (MINLP for CETSP<sub>obs</sub>)**

$$\min_X \sum_{i=1}^n f_i, \quad (5.23)$$

$$\text{s.t. } f_i^2 \geq \mathbf{w}_i^T \mathbf{w}_i \quad \forall i \in \{1, \dots, n\} \quad (5.24)$$

$$\mathbf{w}_i = \mathbf{x}_{i+1} - \mathbf{x}_i \quad \forall i \in \{1, \dots, n\} \quad (5.25)$$

$$\mathbf{v}_i^T \mathbf{v}_i \leq \delta_i^2 \quad \forall i \in \{1, \dots, n\} \quad (5.26)$$

$$\mathbf{v}_i = \mathbf{c}_i - \mathbf{x}_i \quad \forall i \in \{1, \dots, n\} \quad (5.27)$$

$$-d_{i,2}o_{l,1}^k + d_{i,1}o_{l,2}^k + q_i \leq M y_{i,k} \quad (5.28)$$

$$\forall i \in \{1, \dots, n\}, k \in \{1, \dots, K\}, l \in \{1, \dots, l_k\} \text{ if } O_k \text{ in cone of } \mathbf{x}_i, \mathbf{x}_{i+1}$$

$$-d_{i,2}o_{l,1}^k + d_{i,1}o_{l,2}^k + q_i \geq -M(1 - y_{i,k}) \quad (5.29)$$

$$\forall i \in \{1, \dots, n\}, k \in \{1, \dots, K\}, l \in \{1, \dots, l_k\} \text{ if } O_k \text{ in cone of } \mathbf{x}_i, \mathbf{x}_{i+1}$$

$$\mathbf{d}_i = \mathbf{x}_{i+1} - \mathbf{x}_i \quad \forall i \in \{1, \dots, n\} \quad (5.30)$$

$$q_i = d_{i,2}x_{i,1} - d_{i,1}x_{i,2} \quad \forall i \in \{1, \dots, n\} \quad (5.31)$$

$$f_i \in \mathbb{R}, \mathbf{w}_i \in \mathbb{R}^2, \mathbf{x}_i \in \mathbb{R}^2, \mathbf{v}_i \in \mathbb{R}^2, \mathbf{y}_i \in \{0, 1\}^m, \mathbf{d}_i \in \mathbb{R}^2, q_i \in \mathbb{R} \quad \forall i \in \{1, \dots, n\}.$$

The first constraints are the same as in Model 5.3.1, equations (5.11)–(5.14), and the remaining constraints are added for the obstacles. The constraints (5.28), (5.29) are the half-plane constraints for each side of the line segment. The binary variable  $y_{i,k}$  switching the constraints for the Big-M method is indexed for each two consecutive target regions index  $i$  and each obstacle index  $k$ . The in cone selection of indexes is described in Section 5.3.3. The remaining variables are auxiliary, and they represent the difference vector  $\mathbf{d}_i$  (5.30) of the line between two consecutive solution points  $\mathbf{x}_i, \mathbf{x}_{i+1}$ , and the constant  $q_i$  (5.31) representing the constant of the line.

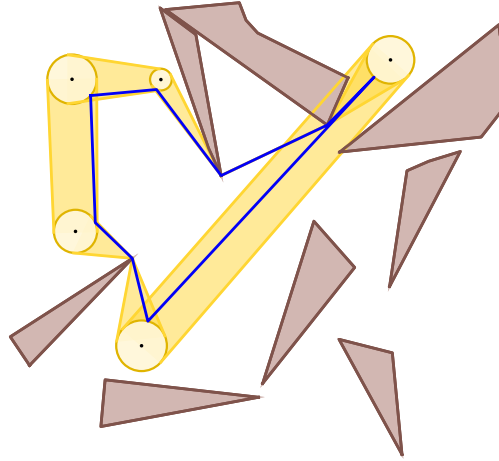
**5.3.3 Conditions for Adding Obstacle Constraints to Model**

The position of the obstacle in half-planes generated by the line segment solution is only relevant for those obstacles that can interfere with the solution. Also, adding constraints for every obstacle and for each of the two subsequent target regions could result in a large number of constraints in the model that are always valid. The area where the solution line occurs is known in advance and is visualized in Figure 5.6. The area between two points is denoted in this thesis as a **cone**, and its computation is described in Section 4.3.

To decide if the constraint is needed for the specific two target regions, a cone between them is calculated and then the intersection of the cone area with an obstacle is checked. If the obstacle blocks the cone completely, any straight line between the two target regions would be infeasible. In such a case, the intermediate points on the obstacle borders are added using the VG, as is described in the statement of Section 3.1. The creation of the model with the cone constraints and the intermediate points is shown in Algorithm 4.

The position of the obstacle in half-plane works, because we made some assumptions on the instances. These assumptions are

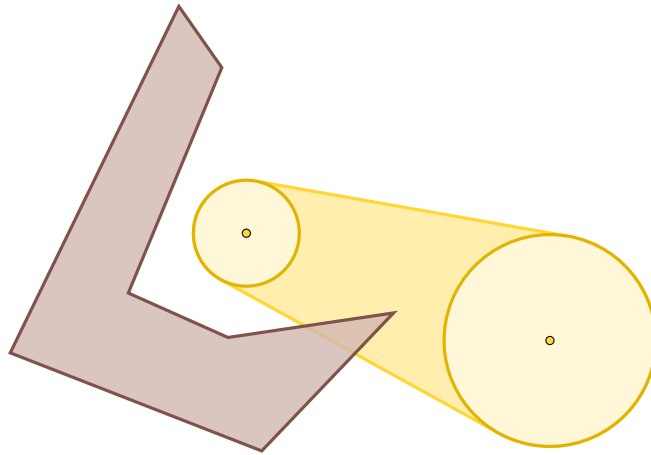
1. The obstacles do not interfere with each other.



**Figure 5.6:** Area of possible solutions between targets on a  $\text{CETSP}_{\text{obs}}$  instance. The brown polygons are the obstacles, the yellow disks are the target regions, and the orange areas between the target regions are the cones. The blue line is a possible solution to the instance.

2. No target region interferes with any obstacle.
3. The target regions are allowed to interfere.

The first assumption is important for the computation of  $\text{VG}$  due to the implementation constraints [38]. The second assumption is important for the half-plane constraint. The case scenario where the constraint is required, otherwise the solution could not be found, is shown in Figure 5.7.



**Figure 5.7:** Example of a target region and obstacle scenario where the half-plane constraint would fail due to obstacle presence inside of the disk.

Another limitation of the used constraints is that the obstacle should occur entirely in one half-plane. If a complex concave polygon obstacle partly occurs inside the cone and then creates a corner behind the target region as shown in Figure 5.8, it could also be blocking a valid solution using the half-plane constraint.

This issue can be solved by convexification of the obstacle polygon. The obstacle is separated into multiple smaller obstacles that are convex, and the original obstacle is their



---

**Algorithm 4:** Create optimization model with cone.
 

---

**Input:**  $\mathcal{S} = \{S_1, \dots, S_n\}$  – set of target regions  
 $\Sigma = \{\sigma_1, \dots, \sigma_n\}$  – order of visit to the target regions  
 $\mathcal{Q} = \{Q_1, \dots, Q_n\}$  – set of intermediate points on obstacles, where  
 $Q_i = \{\mathbf{q}_1, \dots, \mathbf{q}_{k_i}\} \in \mathbb{R}^{k_i \times 2}, k_i \in \mathbb{N}_0 \forall i \in \{1, \dots, n\}$   
 $\Omega = \{O_1, \dots, O_m\}$  – set of obstacles, where  
 $O_i = \{\mathbf{o}_1^i, \dots, \mathbf{o}_{l_i}^i\} \in \mathbb{R}^{l_i \times 2}, l_i \in \mathbb{N} \forall i \in \{1, \dots, m\}$

**Output:** optimization model  $\mathcal{M}$

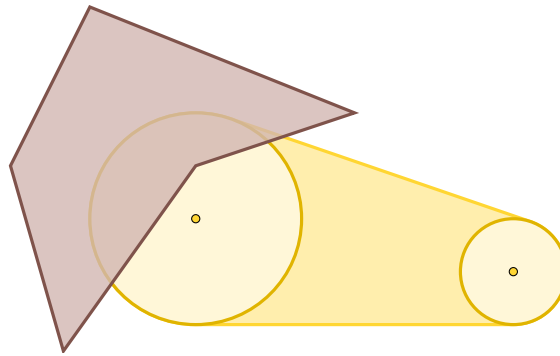
---

```

1  $\Psi \leftarrow ()$  // list of target regions in order
2 for  $\sigma_i \in \Sigma$  do // append all targets and intermediate points in a sequence
3    $\Psi \leftarrow \Psi \sqcup S_{\sigma_i}$ 
4    $\Psi \leftarrow \Psi \sqcup_{j=0}^{k_{\sigma_i}} S(\mathbf{c} = \mathbf{q}_{\sigma_i}, \delta = 0)$  // add intermediate points on obstacles
// as target regions with zero radius
5 end
6  $\mathcal{M} \leftarrow$  optimization Model 5.3.1 with variables  $\{\mathbf{x}_1, \dots, \mathbf{x}_{|\Psi|}\} \in \mathbb{R}^{|\Psi| \times 2}$ 
7 forall  $i \in \{1, \dots, |\Psi|\}$  do // iterate all cones
8   to model  $\mathcal{M}$  add constraint  $\mathbf{d}_i = \mathbf{x}_{i+1} - \mathbf{x}_i$ 
9   to model  $\mathcal{M}$  add constraint  $q_i = \mathbf{d}_{i,2}\mathbf{x}_{i,1} - \mathbf{d}_{i,1}\mathbf{x}_{i,2}$ 
10  forall  $j \in \{1, \dots, m\}$  do // iterate all obstacles
11    if obstacle  $O_j$  is in cone between target regions  $\Psi_i, \Psi_{i+1}$  then
12      to model  $\mathcal{M}$  add binary variable  $y_{i,j}$ 
13      forall  $l \in \{1, \dots, l_j\}$  do // iterate all points in obstacle
14        to model  $\mathcal{M}$  add constraint  $-\mathbf{d}_{i,2}\mathbf{o}_{l,1}^j + \mathbf{d}_{i,1}\mathbf{o}_{l,2}^j + q_i \leq My_{i,j}$ 
15        to model  $\mathcal{M}$  add constraint  $-\mathbf{d}_{i,2}\mathbf{o}_{l,1}^j + \mathbf{d}_{i,1}\mathbf{o}_{l,2}^j + q_i \geq -M(1 - y_{i,j})$ 
16      end
17    end
18  end
19 end
20 return  $\mathcal{M}$ ;

```

---



**Figure 5.8:** Example of a target region and obstacle scenario with a complex concave obstacle around the target region.

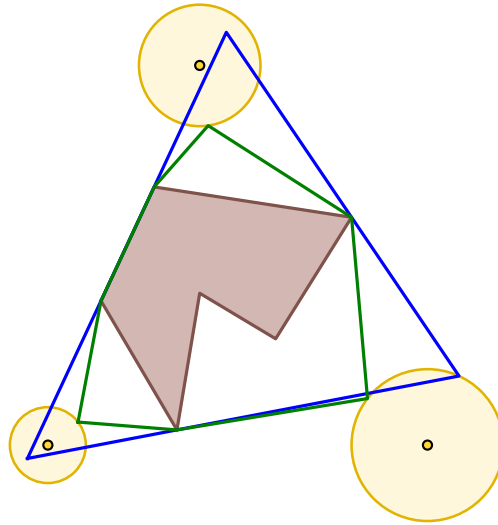
unification. The separation can be done for example with triangulation by ear clipping [39]. The triangulation of a single obstacle with  $l$  sides takes utmost  $\mathcal{O}(n^3)$  based on the imple-

mentation. Due to floating point sensitivity in the calculations, the **VG** is computed first, and the obstacle triangulation is computed afterward, and the separated obstacle is used only for the **MINLP** constraints.

### 5.3.4 Discussion of the Model Solution Quality

The model finds a solution over a set of target regions, along with the dummy target regions added by the **VG**. Given the set  $\Psi$ , as in Algorithm 4, the solution found by Model 5.3.2 is optimized. However, the question is if the whole sequence given in  $\Psi$  is optimal, and that is not proved in this thesis.

During the examination, however, it is found that the optimized solution touches the obstacle border points, as in the example in Figure 5.9. In this case, one can see that the



**Figure 5.9:** Example solution where the addition of intermediate solution points can improve its cost. The blue line is the **MINLP** model solution with  $\mathcal{C} = 174$  only on yellow target regions in the sequence input to Model 5.3.2, and the green line is the **MINLP** model solution with  $\mathcal{C} = 128$ , after addition of the points that touch the solution into the sequence.

solution could be improved if the solution line had multiple segments instead of a single straight segment. For this purpose, an algorithm to include these touching points is proposed in Algorithm 5. The algorithm first finds a solution  $\xi$  for the input list of target regions and obstacles with Model 5.3.2. Then it enters a loop and the algorithm examines the solution lines between every two consecutive target regions, and if any obstacle point is on the solution line, it is added to the list of target regions with zero radius. If there are no added points, the loop ends. A new solution  $\xi'$  is computed from the updated list of target regions, and if the new solution has a higher cost  $\xi'.\mathcal{C}$  than the previous solution  $\xi.\mathcal{C}$ , the loop ends. Otherwise,  $\xi$  is updated with  $\xi'$  and the loop continues. When the loop ends, the latest solution  $\xi$  is returned.

### 5.3.5 The **MINLP** as Post-optimization Heuristic

The proposed **MINLP** mathematical Model 5.3.2 is from the Non-Linear optimization category, and finding the optimum can be difficult with high computational complexity. On the other hand, the **Sampled**-based method from Section 5.2 solves a discretized problem

---

**Algorithm 5:** Pointify model solution.
 

---

**Input:**  $\Psi = \{S_1, \dots, S_k\}$  – sequence of target regions  
 $\mathcal{VG}$  – visibility graph between obstacle points and target regions center  
 $\Omega = \{O_1, \dots, O_m\}$  – set of obstacles  
**Output:**  $\xi = (\Psi, \mathcal{C})$  – visiting sequence of the locations  $\mathcal{P}$ .

---

```

1  $\xi \leftarrow$  find solution points and tour cost using  $\mathcal{M}$ 
2 while True do // iterative solution improvement
3    $\Psi' \leftarrow ()$  // updated sequence of target regions
4    $\mathcal{Q} \leftarrow \{Q_i, \dots, Q_{|\Psi|}, Q_i = \emptyset$  // intermediate points for model creation
5   foreach consecutive line segment  $X_i = (S_i, S_{i+1})$  from  $\mathcal{S}$  do
6      $\Psi' \leftarrow \Psi' \sqcup S_i$ 
7     // for obstacle points that are in cone, check if they lay on the line
8      $Q_i \leftarrow$  found obstacle points touching solution between  $X$ 
9     order  $Q_i$  by the direction of  $X$ 
10    end
11    if  $\Psi' = \Psi$  then // no more obstacle points are added
12      return  $\xi$ 
13    end
14     $\mathcal{M} \leftarrow$  mathematical model created from  $\Psi', \mathcal{O}, \mathcal{Q}$  with Algorithm 4
15     $\xi' \leftarrow$  find solution points and tour cost using  $\mathcal{M}$ 
16    if  $\xi'.\mathcal{C} > \xi.\mathcal{C}$  then // added point prolongs the tour cost
17      return  $\xi$ 
18    else
19       $\xi \leftarrow \xi'$ 
20    end
21     $\Psi \leftarrow \Psi'$ 
22 end

```

---

with deterministic computation. We can use the `Sampled`-based method to find the initial sequence including the intermediate obstacle points, and then use Algorithm 4 to obtain the `MINLP` model. The obtained model is then optimized to obtain the solution. We denote this approach as the `Post-optimization` heuristic, and it can be also used as the `UB` solution.



# Empirical Evaluation

In this chapter, we empirically evaluate the performance of the proposed Branch-and-Bound (BnB) method. We also generate random instances for the method evaluation with up to ten target regions, and they are described in Section 6.1. In the proposed method, different options for the lower bound estimates are specified, and their influence on the BnB algorithm is described in Section 6.2. Different methods for finding the upper bound solutions on fixed sequence were proposed, which are also examined in terms of solution quality and computational time in Section 6.3 and compared to the reference methods.

All the proposed solutions have been implemented in Julia language [34], version 1.8, and were executed on a personal computer with Intel CPU i7-10700 CPU @ up to 2.9 GHz. The used optimization solver for the MINLP model is the Juniper solver [30] version 0.7 with Ipopt [29] version 3.13, which are part of the Julia package system. For the Second-Order Cone Program (SOCP) model for computing the  $\mathcal{LB}_{\text{SOCP}}$ , the CPLEX solver [28] version 0.6 is used.

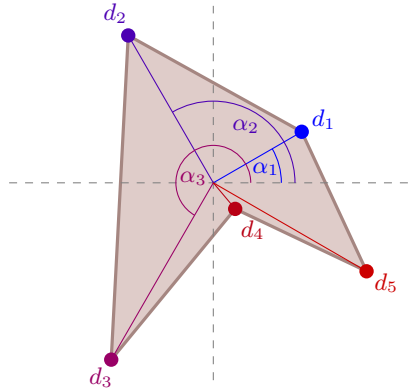
## 6.1 Instances

In the previous chapter, the constraints on the feasible instances have been described. The constraints are that obstacles cannot interfere with the target region or another obstacle, and this is therefore prohibited during the instance generation.

The instances are generated as follows. First, a set of target regions is generated within the specified limits. The targets can interfere with each other, so there is no additional check of their locations. Then the set of obstacles is generated, and for each added obstacle, both the interferences with any obstacle and any target region are checked. If the obstacle interferes, it is generated again at random.

Global bounds on the coordinates in the 2D plane are set, and they are  $[x_{\min}, x_{\max}]$  on the  $x$ -axis, and  $[y_{\min}, y_{\max}]$  on the  $y$ -axis. The radius of the target region must be within  $[\delta_{\min}, \delta_{\max}]$ . The number of target regions can be fixed or generated in the specified range. The number of targets should be greater than three because sequences shorter than or equal to three do not need to be optimized.

The number of obstacles can also be fixed or random, and the values for generating a single polygon are shown in Figure 6.1. The obstacle is specified by its border points and has to be

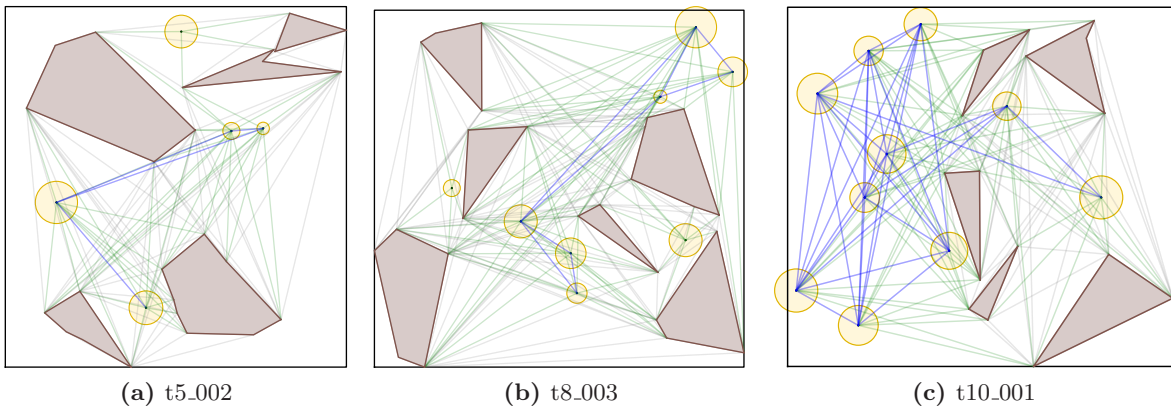


**Figure 6.1:** Process for random obstacle generation with  $s = 5$  sides,  $\alpha_1 < \alpha_2 < \dots < \alpha_5$ , absolute value  $|\alpha_i - \alpha_{i+1}| \leq \pi$ , center point  $(x, y)$  is at the intersection of the dashed lines.

a polygon with no crossing sides. First, a center point  $(x, y)$  is randomly generated, where  $x \in [x_{\min}, x_{\max}]$  and  $y \in [y_{\min}, y_{\max}]$ . The number of obstacle border points can be fixed or random, and they are generated by angles  $A = \{\alpha_1, \dots, \alpha_s\}$  and lengths  $D = \{d_1, \dots, d_s\}$ , where  $s$  is a randomly generated number of polygon sides. The set of angles is generated randomly from the range  $[0, 2\pi)$  and sorted in ascending order. The distance between every two succeeding angles must be  $\leq \pi$  because there is a risk that the obstacle borders would cross otherwise. The distances  $D$  are also generated randomly within the specified range, and the final coordinates for obstacle  $O$  are calculated as

$$O = ((x + d_i \cos \alpha_i, y + d_i \sin \alpha_i) \mid i \in \{1, \dots, s\}). \quad (6.1)$$

The generated obstacles cannot interfere with each other because of the Visibility Graph (VG) computation. The VG is computed for all the obstacles and target region centers after they are generated for the instance, and altogether the data are saved to an instance file. The instances are named as  $t_n.00i$ , where  $n$  is the number of target regions in an obstacle and  $i$  is the identifier of the specific instance,  $i \leq 6$ . An example of the generated data is shown in Figure 6.2.



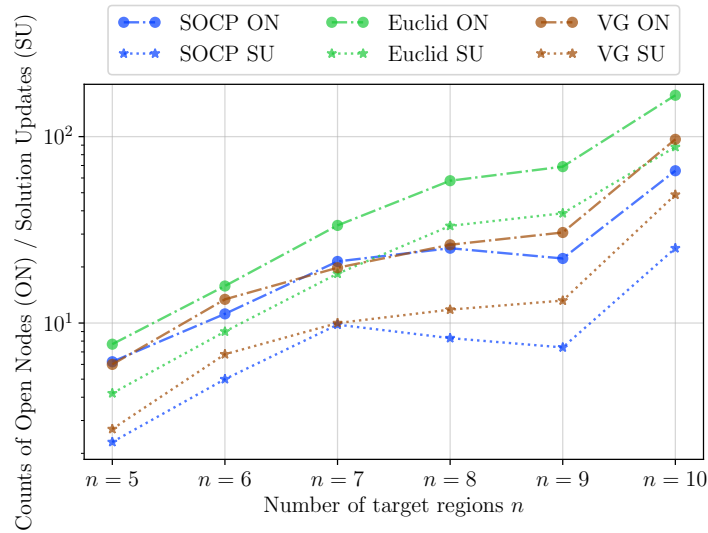
**Figure 6.2:** Example of randomly generated instances. Brown polygons are the obstacles, yellow discs are the target regions, and the lines are the edges of VG.

## 6.2 Evaluation of the Lower Bound Estimates

The lower and upper bounds are used in the **BnB** algorithm for bounding of the solution tree size. In Section 5.1.2, three different lower bounds (denoted  $\mathcal{LB}$ ) were described -  $\mathcal{LB}_{\text{SOCP}}$  computed with the **SOCP** model used for solving the Close-Enough Traveling Salesman Problem (**CETSP**),  $\mathcal{LB}_{\text{Euclid}}$  computed as the Euclidean distance minus the radii of the target regions, and  $\mathcal{LB}_{\text{VG}}$  computed as the distance found using **VG** minus the radii.

The upper bounds (denoted  $\mathcal{UB}$ ) are found using the heuristic insertion of all target regions into the sequence as described in Section 5.1.3. Once the **BnB** node contains all the target regions, the global **BnB**  $\mathcal{UB}$  is updated by the proposed  $\mathcal{UB}$  solution. In this section, we focus on the influence of the different  $\mathcal{LB}$  estimates.

The number of examined nodes for the sampled solution method with  $k = 6$  point samples on the target region border is shown in Figure 6.3, and the corresponding data are in Table 6.1. From Figure 6.3 it can be seen that the  $\mathcal{LB}_{\text{SOCP}}$  opens the lowest number of nodes from all the proposed  $\mathcal{LB}$  types, and  $\mathcal{LB}_{\text{Euclid}}$  the most nodes. Similarly for the solution updates, which



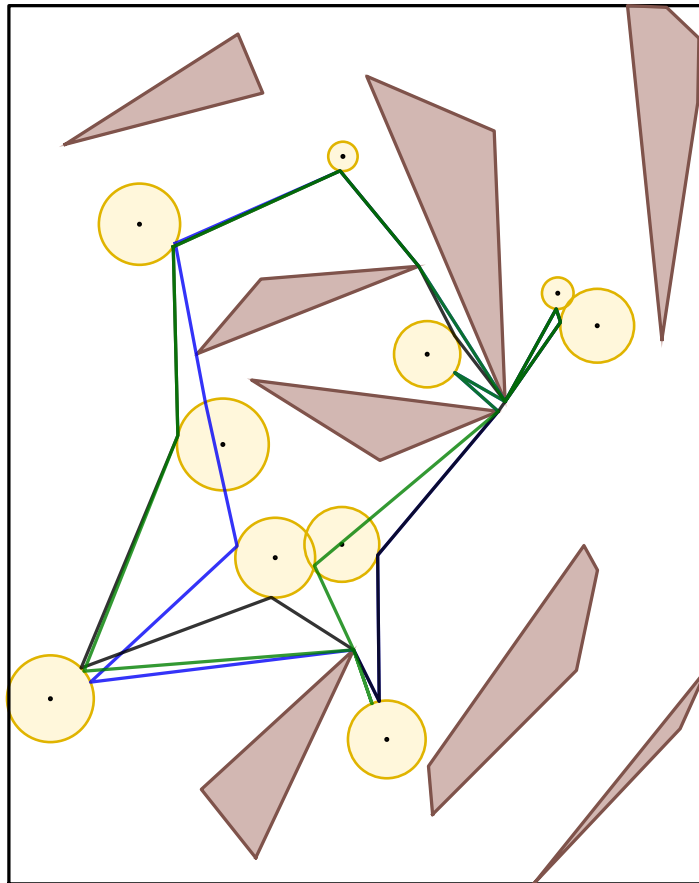
**Figure 6.3:** Different number of examined lower bound values for the **Sampled**-based method with the number of samples  $k = 64$ .

are always lower than the number of open nodes, since only the open leaf nodes are considered for the solution. The full data in Table 6.1 shows also the time needed to find the solution, and it can be seen that the computational time for the different  $\mathcal{LB}$  estimates is similar. This is because even though the  $\mathcal{LB}_{\text{SOCP}}$  expanded fewer nodes, the model optimization takes longer than accessing precomputed distances as in  $\mathcal{LB}_{\text{Euclid}}$  and  $\mathcal{LB}_{\text{VG}}$ . The sampling-based method is deterministic, therefore two runs for each instance are considered sufficient. Furthermore, all results for instances where the computational time has not been reached have the same solution value  $\mathcal{C}$ . The largest discrepancy of  $\mathcal{C}$  is found for the instance  $\tau 10.004$ , where the cost found by the visibility graph is the lowest. This is because all three  $\mathcal{LB}$  types timed out, and the  $\mathcal{C}$  is the latest result since **BnB** is an any-time algorithm. The found solutions for this specific instance are shown in Figure 6.4.

**Table 6.1:** Comparison of the  $\mathcal{LB}$  types for **Sampled**-based method with  $k = 64$ . The best values for each row are marked by bold font. The red values in columns  $t$  are when the 3600 s time limit of the **BnB** algorithm is reached in at least one run, and blue values in columns  $\mathcal{C}$  are when the value differs from the minimum.

Instance	$\mathcal{LB}_{\text{SOCP}}$			$\mathcal{LB}_{\text{Euclid}}$			$\mathcal{LB}_{\text{VG}}$		
	$\mathcal{C}$	$t$ [s]	ON	$\mathcal{C}$	$t$ [s]	ON	$\mathcal{C}$	$t$ [s]	ON
t5_001	<b>499.7</b>	<b>307.7</b>	<b>7.0</b>	<b>499.7</b>	308.2	8.0	<b>499.7</b>	332.9	<b>7.0</b>
t5_002	<b>616.9</b>	<b>292.6</b>	<b>6.0</b>	<b>616.9</b>	293.0	9.0	<b>616.9</b>	296.6	<b>6.0</b>
t5_003	<b>687.8</b>	<b>413.2</b>	<b>7.0</b>	<b>687.8</b>	414.7	<b>7.0</b>	<b>687.8</b>	420.3	<b>7.0</b>
t5_004	<b>647.2</b>	<b>344.0</b>	<b>5.0</b>	<b>647.2</b>	348.3	6.0	<b>647.2</b>	349.7	<b>5.0</b>
t5_005	<b>691.8</b>	<b>339.5</b>	6.0	<b>691.8</b>	343.5	9.0	<b>691.8</b>	341.4	<b>4.0</b>
t5_006	<b>451.7</b>	96.6	<b>6.0</b>	<b>451.7</b>	<b>96.0</b>	7.0	<b>451.7</b>	97.1	7.0
t6_001	<b>693.3</b>	<b>767.8</b>	<b>18.0</b>	<b>693.3</b>	792.3	19.0	<b>693.3</b>	782.5	<b>18.0</b>
t6_002	<b>719.1</b>	686.9	<b>8.0</b>	<b>719.1</b>	705.5	<b>8.0</b>	<b>719.1</b>	<b>683.3</b>	<b>8.0</b>
t6_003	<b>451.2</b>	<b>775.3</b>	<b>11.0</b>	<b>451.2</b>	800.0	20.0	<b>451.2</b>	780.5	15.0
t6_004	<b>566.5</b>	<b>551.2</b>	<b>10.0</b>	<b>566.5</b>	570.4	20.0	<b>566.5</b>	556.1	20.0
t6_005	<b>690.6</b>	723.7	9.0	<b>690.6</b>	750.8	12.0	<b>690.6</b>	<b>722.8</b>	<b>6.0</b>
t7_001	<b>487.9</b>	<b>490.4</b>	<b>12.0</b>	<b>487.9</b>	495.8	34.0	<b>487.9</b>	497.7	34.0
t7_002	<b>896.2</b>	<b>989.7</b>	23.0	<b>896.2</b>	1021.6	24.0	<b>896.2</b>	999.2	<b>18.0</b>
t7_003	<b>782.8</b>	1047.3	55.0	<b>782.8</b>	1051.2	81.0	<b>782.8</b>	<b>1018.8</b>	<b>22.0</b>
t7_004	<b>693.9</b>	<b>1142.9</b>	<b>6.0</b>	<b>693.9</b>	1151.1	13.0	<b>693.9</b>	1160.9	13.0
t7_005	<b>764.0</b>	<b>977.8</b>	<b>11.0</b>	<b>764.0</b>	988.2	15.0	<b>764.0</b>	979.9	12.0
t8_001	<b>779.0</b>	<b>1671.9</b>	45.0	<b>779.0</b>	1726.4	130.0	<b>779.0</b>	1685.4	<b>37.0</b>
t8_002	<b>737.6</b>	<b>1545.0</b>	15.0	<b>737.6</b>	1546.8	<b>14.0</b>	<b>737.6</b>	1545.5	<b>14.0</b>
t8_003	<b>780.1</b>	<b>1921.2</b>	<b>42.0</b>	<b>780.1</b>	1974.8	119.0	<b>780.1</b>	1951.9	49.0
t8_004	<b>912.4</b>	<b>1615.1</b>	<b>17.0</b>	<b>912.4</b>	1671.1	28.0	<b>912.4</b>	1625.5	21.0
t8_005	<b>653.3</b>	<b>1648.6</b>	<b>22.0</b>	<b>653.3</b>	1658.9	32.0	<b>653.3</b>	1656.5	25.0
t8_006	<b>609.0</b>	1367.8	<b>10.0</b>	<b>609.0</b>	<b>1364.8</b>	25.0	<b>609.0</b>	1369.4	12.0
t9_001	<b>745.3</b>	<b>3010.5</b>	<b>27.0</b>	<b>745.3</b>	3026.9	76.0	<b>745.3</b>	3043.5	30.0
t9_002	<b>1007.0</b>	<b>1978.9</b>	<b>29.0</b>	<b>1007.0</b>	2110.1	109.0	<b>1007.0</b>	2014.0	49.0
t9_003	<b>823.0</b>	2629.5	19.0	<b>823.0</b>	2665.5	39.0	<b>823.0</b>	<b>2617.7</b>	<b>18.0</b>
t9_004	<b>797.4</b>	<b>3728.5</b>	<b>7.0</b>	<b>797.4</b>	<b>3766.8</b>	<b>7.0</b>	<b>797.4</b>	<b>3843.1</b>	<b>7.0</b>
t9_005	<b>707.2</b>	<b>2000.1</b>	<b>29.0</b>	<b>707.2</b>	2088.0	114.0	<b>707.2</b>	2064.1	49.0
t10_001	<b>780.9</b>	<b>2081.8</b>	<b>14.0</b>	<b>780.9</b>	2149.6	116.0	<b>780.9</b>	2240.9	71.0
t10_002	<b>884.6</b>	<b>3232.6</b>	<b>185.0</b>	<b>884.6</b>	<b>3600.4</b>	365.5	<b>884.6</b>	<b>3499.1</b>	205.0
t10_003	<b>653.1</b>	<b>3263.6</b>	38.0	<b>653.1</b>	3350.0	60.0	<b>653.5</b>	<b>3490.2</b>	<b>33.5</b>
t10_004	<b>851.2</b>	<b>3887.0</b>	16.0	<b>852.4</b>	<b>3875.1</b>	<b>8.0</b>	<b>794.4</b>	<b>4079.7</b>	<b>8.0</b>
t10_005	<b>638.0</b>	<b>3124.1</b>	<b>75.0</b>	<b>638.0</b>	3359.9	284.0	<b>638.0</b>	<b>3387.5</b>	166.0





**Figure 6.4:** Solutions for instance `t10_004` with `Sampled`-based method and  $k = 64$  found before timeout  $t_{\max} = 3600$  s was reached with different  $\mathcal{LB}$  estimates. The black line is the solution found when  $\mathcal{LB}_{VG}$  is used, the blue line is the  $\mathcal{LB}_{Euclid}$  and the green line is the  $\mathcal{LB}_{SOCP}$ .

### 6.3 Evaluation of the Upper Bound Solutions

In this section, the cost of the solutions and the corresponding computational time needed are compared. The cost and time are dependent on the different number of samples for the `Sampled`-based method, described in Section 5.2 and `Post-optimization` heuristic, described in Section 5.3.5. For the latter and the Mixed Integer Non-Linear Program (`MINLP`) formulation described in Section 5.3, the cost and time are dependent on the optimization solver.

The cost and time for the `Sampled`-based method with `SOCP` lower bound  $\mathcal{LB}_{SOCP}$  is shown in Figure 6.5, and the same data are also shown in Table 6.2. The table shows that with a larger number of target region samples  $k$  the solution improves since the proposed `Sampled`-based method is exact. At the same time, the computational time increases logarithmically. The peak for  $n = 10, k = 64$  at the end in Figure 6.5a is caused by the timeout. A feasible solution is still obtained as the value of the global upper bound since the `BnB` is an any-time algorithm.

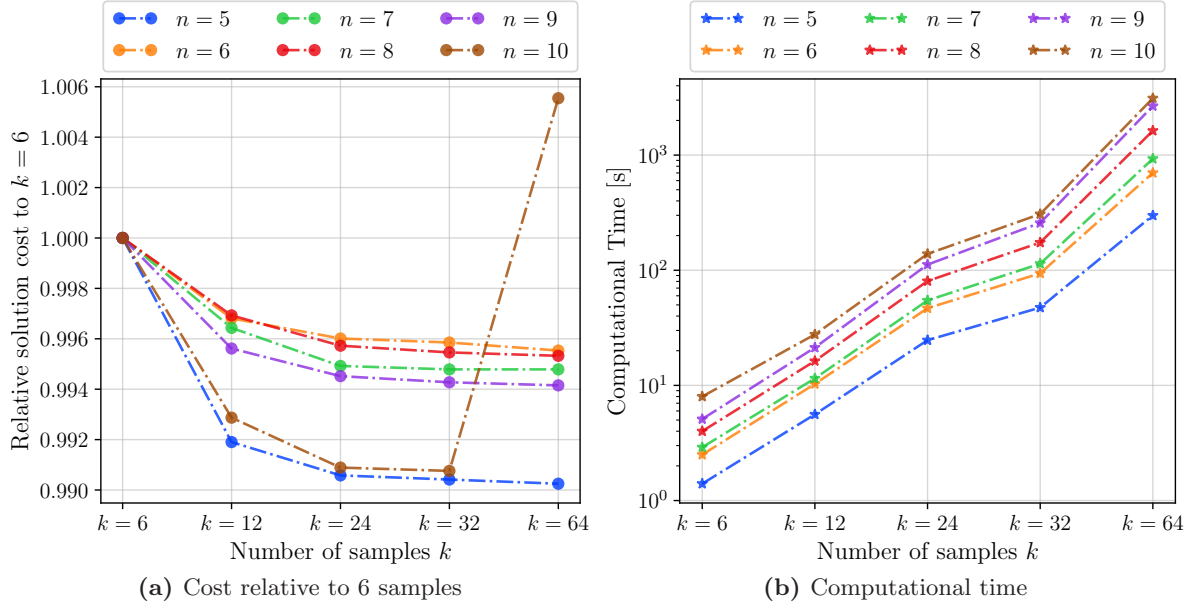
The table Table 6.2 the `MINLP` model and `Post-optimization` heuristic methods along with the computational times are shown. The cost  $\mathcal{C}$  obtained by the `Post-optimization` heuristic of  $k = 6$  samples is better than `Sampled`-based method for  $k = 64$ , and the

**Table 6.2:** Sampled-based method with different number of samples  $k$  and  $\mathcal{LB}_{\text{SOCP}}$ . The best values for each row are marked by bold font and the times  $t$  where the time limit  $t_{\text{max}}$  of 3600s is reached are in red. The dash symbolizes missing results.

Instance	Sampled $k = 6$		Sampled $k = 12$		Sampled $k = 32$		Sampled $k = 64$		Post-opt $k = 6$		MINLP	
	$\mathcal{C}$	$t$ [s]	$\mathcal{C}$	$t$ [s]	$\mathcal{C}$	$t$ [s]	$\mathcal{C}$	$t$ [s]	$\mathcal{C}$	$t$ [s]	$\mathcal{C}$	$t$ [s]
t5_001	506.5	<b>1.1</b>	500.3	4.5	499.8	42.1	499.7	307.7	<b>499.6</b>	57.2	<b>499.6</b>	70.2
t5_002	626.6	<b>1.1</b>	617.7	4.6	<b>616.9</b>	41.6	<b>616.9</b>	292.6	<b>616.9</b>	80.9	<b>616.9</b>	149.9
t5_003	690.0	<b>2.1</b>	689.0	8.3	687.9	67.9	687.8	413.2	<b>687.7</b>	234.2	705.6	1225.8
t5_004	652.0	<b>1.6</b>	647.7	6.4	647.4	55.4	<b>647.2</b>	344.0	<b>647.2</b>	1786.6	649.2	1085.4
t5_005	697.6	<b>1.6</b>	694.4	6.5	691.9	55.3	<b>691.8</b>	339.5	692.1	37.2	695.0	71.6
t5_006	457.8	<b>0.8</b>	452.0	3.1	451.7	22.3	451.7	96.6	<b>451.6</b>	15.7	<b>451.6</b>	31.7
t6_001	697.5	<b>2.7</b>	694.5	11.1	693.4	103.3	693.3	767.8	<b>693.2</b>	<b>5263.2</b>	698.0	<b>4362.6</b>
t6_002	721.6	<b>3.1</b>	719.5	12.6	719.2	107.6	719.1	686.9	<b>719.0</b>	1070.7	<b>719.0</b>	1024.5
t6_003	453.0	<b>2.3</b>	451.6	9.3	<b>451.2</b>	90.2	<b>451.2</b>	775.3	<b>451.2</b>	337.9	<b>451.2</b>	325.6
t6_004	568.0	<b>1.6</b>	567.9	6.7	566.7	66.1	<b>566.5</b>	551.2	566.8	1595.6	566.8	2257.4
t6_005	694.4	<b>2.9</b>	691.0	11.7	690.8	102.8	<b>690.6</b>	723.7	<b>690.6</b>	55.5	691.5	38.7
t7_001	489.1	<b>2.5</b>	488.5	9.7	488.0	82.2	<b>487.9</b>	490.4	<b>487.9</b>	697.5	488.7	376.6
t7_002	903.2	<b>2.5</b>	896.9	10.2	896.3	109.7	<b>896.2</b>	989.7	–	–	959.9	<b>16504.2</b>
t7_003	784.2	<b>3.5</b>	783.9	13.5	782.9	127.9	<b>782.8</b>	1047.3	<b>782.8</b>	<b>3913.7</b>	876.7	<b>4029.3</b>
t7_004	699.1	<b>2.9</b>	696.0	12.1	694.0	128.7	<b>693.9</b>	1142.9	698.2	<b>15016.1</b>	698.0	<b>13362.1</b>
t7_005	768.3	<b>2.9</b>	765.8	12.1	764.0	120.9	764.0	977.8	<b>763.9</b>	1321.5	766.5	505.5
t8_001	782.3	<b>6.7</b>	780.1	25.2	<b>779.0</b>	228.8	<b>779.0</b>	1671.9	782.3	<b>3779.1</b>	786.5	<b>4045.9</b>
t8_002	740.5	<b>2.9</b>	738.4	12.4	737.7	147.5	<b>737.6</b>	1545.0	740.5	3095.5	758.9	1089.7
t8_003	786.5	<b>3.8</b>	780.8	15.2	780.2	176.4	<b>780.1</b>	1921.2	786.5	<b>3658.2</b>	799.5	<b>4550.0</b>
t8_004	914.7	<b>3.8</b>	912.7	15.9	<b>912.4</b>	171.8	<b>912.4</b>	1615.1	914.7	3399.4	953.3	<b>3767.6</b>
t8_005	655.6	<b>3.4</b>	655.4	14.3	653.5	164.3	<b>653.3</b>	1648.6	<b>653.3</b>	2999.7	672.4	<b>4100.8</b>
t8_006	612.5	<b>3.5</b>	610.9	14.8	609.2	155.1	<b>609.0</b>	1367.8	<b>609.0</b>	1321.2	<b>609.0</b>	1287.9
t9_001	749.8	<b>7.2</b>	746.7	29.7	745.4	324.0	<b>745.3</b>	3010.5	749.8	<b>4276.5</b>	755.1	<b>4264.3</b>
t9_002	1010.4	<b>4.0</b>	1007.8	16.1	1007.2	197.1	1007.0	1978.9	<b>1006.9</b>	<b>4040.5</b>	1011.7	<b>4387.1</b>
t9_003	829.1	<b>5.2</b>	824.0	22.2	823.2	265.6	<b>823.0</b>	2629.5	829.1	824.8	823.5	1363.4
t9_004	801.0	<b>5.1</b>	797.9	22.3	797.5	303.7	<b>797.4</b>	<b>3728.5</b>	<b>797.4</b>	<b>3759.7</b>	<b>797.4</b>	<b>3813.8</b>
t9_005	713.9	<b>4.0</b>	709.4	16.1	707.3	197.6	707.2	2000.1	<b>707.1</b>	<b>3631.4</b>	835.2	<b>4430.3</b>
t10_001	788.3	<b>4.1</b>	782.7	16.5	781.0	196.2	780.9	2081.8	<b>780.8</b>	<b>14437.8</b>	–	–
t10_002	892.6	<b>11.7</b>	886.7	35.4	884.7	351.3	<b>884.6</b>	3232.6	897.7	<b>7868.3</b>	907.3	<b>4073.0</b>
t10_003	656.3	<b>3.4</b>	654.4	15.8	653.4	252.0	<b>653.1</b>	3263.6	<b>653.1</b>	<b>17503.8</b>	–	–
t10_004	805.1	<b>11.7</b>	797.7	38.8	794.5	400.8	851.2	<b>3887.0</b>	<b>794.3</b>	<b>5346.5</b>	–	–
t10_005	644.9	<b>9.3</b>	638.7	32.1	638.2	337.4	<b>638.0</b>	3124.1	<b>638.0</b>	<b>58214.1</b>	–	–

Post-optimization heuristic could perform better in all cases if more than 6 samples were used. The solutions found directly by MINLP model usually take longer  $t$ , and  $\mathcal{C}$  is equal to or worse than the Post-optimization heuristic. This is because the MINLP solver does not find the global optimum, and in many instances, not all the sequences were examined because of the time limit. The computational time  $t$  fluctuates for the two methods without any resemblance of the tendency of  $t$  for the Sampled-based method. The time limit  $t_{\text{max}}$  was set to

3600s, and the optimization solver itself had the time limit set to 1000s. In some instances,  $t$  was many times higher than  $t_{\max}$ . This is caused by the MINLP optimization solver, which, even though the time limit was reached, did not stop the improvement loop.



**Figure 6.5:** The comparison of the solution cost and time for the Sampled-based method with SOCP lower bound and different number of samples. With a higher number of samples  $k$  the cost consistently decreases, and the computational time increases logarithmically.

In table Table 6.3 the proposed Sampled-based method is compared with the reference solutions found using the Self-Organizing Map (SOM) [6] and the Large Neighborhood Search Heuristic for the Generalized Traveling Salesman Problem (GTSP) (GLNS) for the CETSP (GLNS-CETSP) [7]. The reference methods are heuristic with the solution depending on the initialization, and their cost is averaged over multiple runs. For the proposed Sampled-based method,  $\mathcal{C}$  decreases with the number of samples  $k$ , since the method is deterministic. Even though the SOM approach also uses samples for the target regions to transform the circle to a  $k$ -sided polygon, the cost  $\mathcal{C}$  fluctuates due to the heuristic nature of the algorithm.

In Table 6.4 the comparison of the proposed and reference methods [6, 7] with and without the Post-optimization heuristic is shown. The proposed Post-optimization heuristic improves the solutions found with Sampled-based method. For the solutions from the reference methods, the Post-optimization heuristic does not always improve the solutions. This can happen because the initializing sequence uses the target regions, and the used mathematical solver found only a local optimum. The better performance of the Post-optimization heuristic on the proposed solution can be explained by multiple solution updates, whereas in the reference methods only the final solution was provided to the Post-optimization heuristic. Also, the  $\mathcal{UB}$  solution in the proposed method is only updated when improvement is reached, whereas in Table 6.4 the cost found by the Post-optimization heuristic for the reference methods is shown nevertheless if the improvement is reached or not.

All of the used data instances and solutions from the proposed Sampled-based method, Post-optimization heuristic and the reference SOM and GLNS-CETSP methods are shown in Appendix in Chapter A.

**Table 6.3:** Comparison of solution cost  $\mathcal{C}$  for proposed and reference methods. The best values for each row are marked in bold font. The dash symbolizes missing results.

Instance	Sampled-based method				SOM [6]				GLNS-CETSP [7]
	$k = 6$	$k = 12$	$k = 24$	$k = 64$	$k = 6$	$k = 12$	$k = 24$	$k = 64$	
t5_001	506.5	500.3	499.8	499.7	513.8	512.6	515.3	510.2	<b>499.6</b>
t5_002	626.6	617.7	617.0	<b>616.9</b>	638.8	629.7	630.7	630.1	<b>616.9</b>
t5_003	690.0	689.0	688.1	687.8	693.0	692.5	691.0	692.1	<b>687.7</b>
t5_004	652.0	647.7	647.4	<b>647.2</b>	653.3	652.1	650.0	650.1	649.2
t5_005	697.6	694.4	692.0	<b>691.8</b>	709.7	705.3	702.7	705.5	692.1
t5_006	457.8	452.0	451.9	451.7	460.4	456.5	456.3	455.5	<b>451.6</b>
t6_001	697.5	694.5	693.6	693.3	708.8	705.8	703.3	704.1	<b>693.2</b>
t6_002	721.6	719.5	719.1	719.1	729.2	728.1	724.7	727.5	<b>719.0</b>
t6_003	453.0	451.6	451.5	<b>451.2</b>	469.9	469.9	474.9	473.3	<b>451.2</b>
t6_004	568.0	567.9	566.8	<b>566.5</b>	–	–	–	–	566.9
t6_005	694.4	691.0	690.8	<b>690.6</b>	700.7	699.8	700.0	698.9	<b>690.6</b>
t7_001	489.1	488.5	488.2	<b>487.9</b>	498.1	497.8	497.4	496.1	488.1
t7_002	903.2	896.9	896.3	<b>896.2</b>	924.0	913.8	913.7	914.8	898.2
t7_003	784.2	783.9	782.9	<b>782.8</b>	794.8	801.9	793.8	796.2	788.8
t7_004	699.1	696.0	694.1	<b>693.9</b>	710.0	706.9	704.4	703.7	<b>693.9</b>
t7_005	768.3	765.8	764.1	<b>764.0</b>	782.1	780.7	784.8	779.0	<b>764.0</b>
t8_001	782.3	780.1	779.4	<b>779.0</b>	792.5	794.4	789.0	788.6	795.3
t8_002	740.5	738.4	738.0	<b>737.6</b>	752.8	745.6	746.1	746.0	<b>737.6</b>
t8_003	786.5	780.8	780.3	<b>780.1</b>	800.4	792.9	803.8	799.0	781.8
t8_004	914.7	912.7	912.5	<b>912.4</b>	937.3	944.0	933.2	938.5	1070.1
t8_005	655.6	655.4	653.7	<b>653.3</b>	665.1	667.6	664.0	662.8	<b>653.3</b>
t8_006	612.5	610.9	609.4	<b>609.0</b>	616.2	616.2	614.9	615.3	<b>609.0</b>
t9_001	749.8	746.7	745.4	<b>745.3</b>	758.8	757.1	754.7	754.6	<b>745.3</b>
t9_002	1010.4	1007.8	1007.4	<b>1007.0</b>	1018.5	1021.8	1020.1	1019.8	1019.3
t9_003	829.1	824.0	823.3	<b>823.0</b>	857.5	853.6	847.9	848.3	<b>823.0</b>
t9_004	801.0	797.9	797.6	<b>797.4</b>	806.1	804.8	809.5	803.0	<b>797.4</b>
t9_005	713.9	709.4	707.6	707.2	728.1	723.7	727.2	728.4	<b>707.1</b>
t10_001	788.3	782.7	781.2	780.9	797.7	805.5	801.2	800.5	<b>780.8</b>
t10_002	892.6	886.7	884.9	<b>884.6</b>	900.4	897.8	894.5	895.9	896.0
t10_003	656.3	654.4	653.5	<b>653.1</b>	664.5	661.8	661.2	661.4	653.2
t10_004	805.1	797.7	<b>794.8</b>	851.2	809.7	803.7	802.3	799.7	860.6
t10_005	644.9	638.7	638.3	<b>638.0</b>	647.7	644.2	644.3	643.2	<b>638.0</b>

**Table 6.4:** Comparison of solution cost  $\mathcal{C}$  with Post-optimization heuristic for proposed and reference methods. The best values for each row are marked by bold font. The dash symbolizes missing results.

Instance	Sampled $k = 6$	Post-opt $k = 6$	MINLP	SOM [6]	SOM Post-opt	GLNSC [7]	GLNSC Post-opt
t5_001	506.5	<b>499.6</b>	<b>499.6</b>	515.3	514.5	<b>499.6</b>	<b>499.6</b>
t5_002	626.6	<b>616.9</b>	<b>616.9</b>	630.7	639.3	<b>616.9</b>	<b>616.9</b>
t5_003	690.0	<b>687.7</b>	705.6	691.0	694.0	<b>687.7</b>	<b>687.7</b>
t5_004	652.0	<b>647.2</b>	649.2	650.0	653.5	649.2	649.2
t5_005	697.6	<b>692.1</b>	695.0	702.7	710.4	<b>692.1</b>	<b>692.1</b>
t5_006	457.8	<b>451.6</b>	<b>451.6</b>	456.3	460.0	<b>451.6</b>	<b>451.6</b>
t6_001	697.5	<b>693.2</b>	698.0	703.3	709.5	<b>693.2</b>	<b>693.2</b>
t6_002	721.6	<b>719.0</b>	<b>719.0</b>	724.7	728.7	<b>719.0</b>	<b>719.0</b>
t6_003	453.0	<b>451.2</b>	<b>451.2</b>	474.9	469.8	<b>451.2</b>	<b>451.2</b>
t6_004	568.0	<b>566.8</b>	<b>566.8</b>	–	–	566.9	566.9
t6_005	694.4	<b>690.6</b>	691.5	700.0	700.0	<b>690.6</b>	<b>690.6</b>
t7_001	489.1	<b>487.9</b>	488.7	497.4	499.1	488.1	488.1
t7_002	903.2	–	959.9	913.7	921.8	<b>898.2</b>	898.5
t7_003	784.2	<b>782.8</b>	876.7	793.8	792.7	788.8	788.8
t7_004	699.1	698.2	698.0	704.4	712.7	<b>693.9</b>	<b>693.9</b>
t7_005	768.3	<b>763.9</b>	766.5	784.8	779.2	764.0	764.0
t8_001	<b>782.3</b>	<b>782.3</b>	786.5	789.0	793.8	795.3	795.3
t8_002	740.5	740.5	758.9	746.1	752.8	<b>737.6</b>	<b>737.6</b>
t8_003	786.5	786.5	799.5	803.8	815.3	<b>781.8</b>	<b>781.8</b>
t8_004	<b>914.7</b>	<b>914.7</b>	953.3	933.2	941.5	1070.1	1070.1
t8_005	655.6	<b>653.3</b>	672.4	664.0	665.1	<b>653.3</b>	<b>653.3</b>
t8_006	612.5	<b>609.0</b>	<b>609.0</b>	614.9	616.1	<b>609.0</b>	<b>609.0</b>
t9_001	749.8	749.8	755.1	754.7	759.5	<b>745.3</b>	<b>745.3</b>
t9_002	1010.4	<b>1006.9</b>	1011.7	1020.1	1019.6	1019.3	1019.3
t9_003	829.1	829.1	823.5	847.9	850.5	<b>823.0</b>	<b>823.0</b>
t9_004	801.0	<b>797.4</b>	<b>797.4</b>	809.5	807.1	<b>797.4</b>	<b>797.4</b>
t9_005	713.9	<b>707.1</b>	835.2	727.2	726.8	<b>707.1</b>	<b>707.1</b>
t10_001	788.3	<b>780.8</b>	–	801.2	798.1	<b>780.8</b>	<b>780.8</b>
t10_002	892.6	897.7	907.3	<b>894.5</b>	906.2	896.0	896.0
t10_003	656.3	<b>653.1</b>	–	661.2	664.7	653.2	653.2
t10_004	805.1	<b>794.3</b>	–	802.3	811.5	860.6	860.6
t10_005	644.9	<b>638.0</b>	–	644.3	647.5	<b>638.0</b>	<b>638.0</b>



# Conclusion

In this thesis, we study the Close-Enough Traveling Salesman Problem (**CETSP**) in an environment with obstacles. It is a variant of the Traveling Salesman Problem (**TSP**), which is an NP-hard problem, therefore the studied problem is challenging as well. The approaches for solving the Close-Enough Traveling Salesman Problem with Obstacles (**CETSP<sub>obs</sub>**) and related work are overviewed, and the descriptions of background methods are provided to make the thesis self-contained.

The studied problem is denoted as the **CETSP<sub>obs</sub>** and its formal definition has been provided. The proposed Branch-and-Bound (**BnB**) algorithm is presented, along with the proposed upper bound solutions (denoted **UB**). For the **BnB**, three different types of lower bound estimates (denoted **LB**) are used: the Second-Order Cone Program (**SOCP**) model solution of the **CETSP** disregarding the obstacles denoted **LB<sub>SOCP</sub>**, the Euclidean distance minus the radii of the target regions denoted **LB<sub>Euclid</sub>**, and the distance found using the Visibility Graph (**VG**) minus the radii denoted **LB<sub>VG</sub>**. The mathematical model formulation for **CETSP<sub>obs</sub>** is formulated as the Mixed Integer Non-Linear Program (**MINLP**) for a fixed target region sequence. The constraints use half-plane obstacle separation, and the cone area is used to minimize the number of constraints added to the **MINLP** model.

Two different **UB** solution methods for the fixed sequence of target regions found by the **BnB** algorithm are proposed. The first takes advantage of the **MINLP** model, which was provided with the description of obstacle constraints and the sequence optimization is discussed. The second proposed method is the **Sampled**-based method, which effectively transforms the **CETSP<sub>obs</sub>** to the Generalized Traveling Salesman Problem (**GTSP**), where the **VG** is used to compute the shortest path between every two samples. This approach finds the optimal solution within the discretized problem by selecting the best samples using the auxiliary graph. The two methods can be combined, and **MINLP** model is used to improve the solutions found by the **Sampled**-based method by optimizing the found solution sequence including the obstacle points, called the **Post-optimization** heuristic. The proposed methods for **UB** computation are employed for the **BnB** nodes containing all the given target regions in the sequence (lead nodes), and the solution is also used to update the global **UB** to tighten the bound to the global optimum.

The proposed method is empirically evaluated on randomly generated instances. The different types of **LB** estimates were compared by the number of opened **BnB** nodes and the

number of  $UB$  updates in leaf nodes. The  $UB$  solution approaches were evaluated as well, and the obtained solution costs  $\mathcal{C}$  were the same for all  $\mathcal{LB}$  types if the solution is found within the time limit. However, the computational time to find the solution is different due to a different number of examined nodes. The proposed  $BnB$  method has a longer computational time than the two heuristic approaches used for comparison [6, 7], which is expected since the  $BnB$  method is used for finding the exact solutions. The **Post-optimization** heuristic can also be used on the solutions given by the approaches [6, 7], and is in review in [10].

The proposed  $UB$  solutions that use the  $MINLP$  mathematical model are quite slow to computational and become intractable for larger instances. This is due to the proposed  $BnB$  method and the infeasible  $\mathcal{LB}$ , but also the speed of the optimization solvers for  $MINLP$ . A tighter  $\mathcal{LB}$  estimate could be used, e.g. the windowing approach with the distance found with  $MINLP$  model, where the  $\mathcal{LB}$  estimate is computed on a subset of the target region sequence.

The obstacle constraints in the  $MINLP$  model are selected to have the lowest polynomial complexity. Therefore, the half-plane representation is used in the thesis, because the constraint variables are at most to the power of two. During the design phase, different constraints for obstacle avoidance were considered, such as obstacle representation as a disk, where the intersection constraints had variables to the fourth power. The second option for the polygonal obstacle was the intersection of the line segments of the polygonal obstacle border to the solution line segment, which had floating point precision issues. Therefore, a possible future work would be to further examine the different obstacle constraints.



# Bibliography

- [1] David L Applegate, Robert E Bixby, Vašek Chvátal, and William J Cook. The Traveling Salesman Problem. In *The Traveling Salesman Problem*. Princeton university press, 2007.
- [2] Damon J Gulczynski, Jeffrey W Heath, and Carter C Price. The Close Enough Traveling Salesman Problem: A discussion of several heuristics. In *Perspectives in operations research*, pages 271–283. Springer, 2006.
- [3] Walton Pereira Coutinho, Roberto Quirino do Nascimento, Artur Alves Pessoa, and Anand Subramanian. A Branch-and-Bound algorithm for the Close Enough Traveling Salesman Problem. *INFORMS Journal on Computing*, 28(4):752–765, 2016.
- [4] Jindřiška Deckerová, Jan Faigl, and Vít Krátký. Traveling Salesman Problem with neighborhoods on a sphere in reflectance transformation imaging scenarios. *Expert Systems with Applications*, 198:116814, 2022.
- [5] Jan Faigl, Miroslav Kulich, Vojtěch Vonásek, and Libor Přeučil. An application of the Self-Organizing Map in the non-Euclidean Traveling Salesman Problem. *Neurocomputing*, 74(5):671–679, 2011.
- [6] Jan Faigl, Vojtěch Vonásek, and Libor Přeučil. Visiting convex regions in a polygonal map. *Robotics and Autonomous Systems*, 61(10):1070–1083, 2013.
- [7] Lukáš Fanta. *The Close Enough Traveling Salesman Problem in the polygonal domain*. PhD thesis, Master’s thesis, CTU in Prague, 2021.
- [8] Stephen L Smith and Frank Imeson. GLNS: An effective large neighborhood search heuristic for the generalized Traveling Salesman Problem. *Computers & Operations Research*, 87:1–19, 2017.
- [9] Mark De Berg, Marc Van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Introduction*. Springer, 1997.
- [10] Jindřiška Deckerová, Kristýna Kučerová, and Jan Faigl. Towards improvement of heuristic solutions of the Close Enough Traveling Salesman Problem in an environment with obstacles, 2023. (in review).

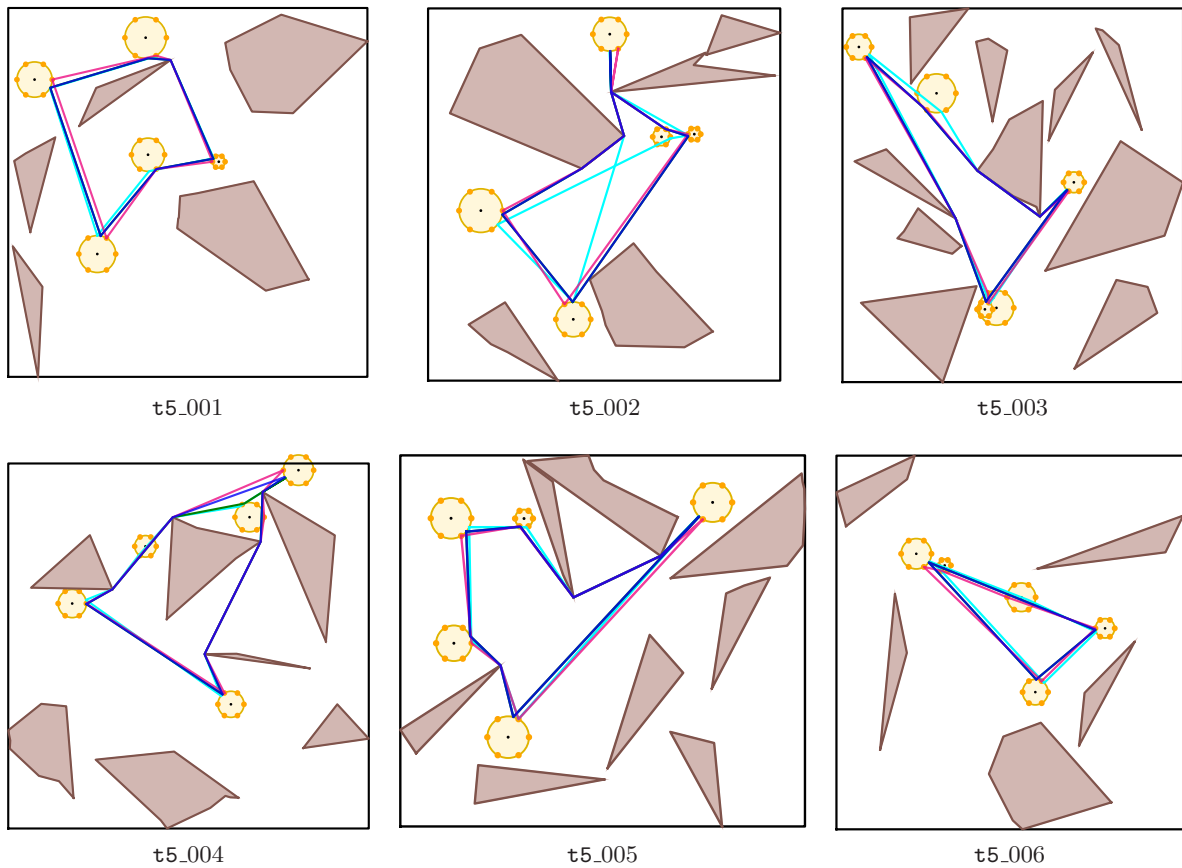
## Bibliography

- [11] Daniel J Rosenkrantz, Richard E Stearns, and Philip M Lewis, II. An analysis of several heuristics for the Traveling Salesman Problem. *SIAM journal on computing*, 6(3):563–581, 1977.
- [12] Hui-Dong Jin, Kwong-Sak Leung, Man-Leung Wong, and Z-B Xu. An efficient Self-Organizing Map designed by genetic algorithms for the Traveling Salesman Problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 33(6):877–888, 2003.
- [13] Egon Balas and Paolo Toth. Branch and Bound methods for the Traveling Salesman Problem. Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group, 1983.
- [14] George Bernard Dantzig, Delbert R Fulkerson, and Selmer Martin Johnson. On a Linear-Programming, combinatorial approach to the Traveling-Salesman Problem. *Operations research*, 7(1):58–66, 1959.
- [15] Iacopo Gentilini, François Margot, and Kenji Shimada. The Travelling Salesman Problem with neighbourhoods: MINLP solution. *Optimization Methods and Software*, 28(2):364–378, 2013.
- [16] Pietro Belotti, Jon Lee, Leo Liberti, François Margot, and Andreas Wächter. Branching and Bounds tightening techniques for non-convex MINLP. *Optimization Methods and Software*, 24(4-5):597–634, 2009.
- [17] Jing Dong, Ning Yang, and Ming Chen. Heuristic approaches for a TSP variant: The automatic meter reading shortest tour problem. *Extending the horizons: Advances in computing, optimization, and decision technologies*, pages 145–163, 2007.
- [18] William Kenneth Mennell. *Heuristics for solving three routing problems: Close Enough Traveling Salesman Problem, Close-Enough Vehicle Routing Problem, and Sequence-Dependent Team orienteering Problem*. University of Maryland, College Park, 2009.
- [19] Stephen Boyd, Stephen P Boyd, and Lieven Vandenbergh. *Convex optimization*. Cambridge university press, 2004.
- [20] Nathan Ratliff, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa. CHOMP: Gradient optimization techniques for efficient motion planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 489–494. IEEE, 2009.
- [21] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014.
- [22] John Saalweachter and Zygmunt Pizlo. Non-Euclidean Traveling Salesman Problem. *Decision modeling and behavior in complex and uncertain environments*, pages 339–358, 2008.
- [23] Luitpold Babel. Curvature-Constrained Traveling Salesman tours for aerial surveillance in scenarios with obstacles. *European Journal of Operational Research*, 262(1):335–346, 2017.

- [24] Lester E Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, 79(3):497–516, 1957.
- [25] George Bernard Dantzig. Linear Programming. *Operations research*, 50(1):42–47, 2002.
- [26] Gautam Mitra. Investigation of some Branch and Bound strategies for the solution of Mixed Integer Linear Programs. *Mathematical Programming*, 4:155–170, 1973.
- [27] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Fundamentals of convex analysis*. Springer Science & Business Media, 2004.
- [28] Christian Bliet, Pierre Bonami, and Andrea Lodi. Solving Mixed-Integer Quadratic Programming Problems with IBM-CPLEX: a progress report. In *Proceedings of the twenty-sixth RAMP symposium*, pages 16–17, 2014.
- [29] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale Nonlinear Programming. *Mathematical Programming*, 106:25–57, 2006.
- [30] Ole Kröger, Carleton Coffrin, Hassan Hijazi, and Harsha Nagarajan. Juniper: An Open-source Nonlinear Branch-and-Bound Solver in Julia. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 377–386. Springer International Publishing, 2018.
- [31] Pierre Bonami et al. Bonmin: Basic Open-source Nonlinear Mixed Integer Programming. <https://github.com/coin-or/Bonmin>, 2004. Accessed: 2023-04-13.
- [32] Harsha Nagarajan, Mowen Lu, Site Wang, Russell Bent, and Kaarthik Sundar. An adaptive, multivariate partitioning algorithm for global optimization of nonconvex programs. *Journal of Global Optimization*, 2019.
- [33] Harsha Nagarajan, Mowen Lu, Emre Yamangil, and Russell Bent. Tightening McCormick relaxations for nonlinear programs via dynamic multivariate partitioning. In *International Conference on Principles and Practice of Constraint Programming*, pages 369–387. Springer, 2016.
- [34] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.
- [35] Pierre Bonami et al. COUENNE: Convex Over and Under Envelopes for Nonlinear Estimation. <https://github.com/coin-or/Couenne>, 2006. Accessed: 2023-04-13.
- [36] Peter van Emde Boas, Robert Kaas, and Erik Zijlstra. Design and implementation of an efficient Priority Queue. *Mathematical systems theory*, 10(1):99–127, 1976.
- [37] Herbert John Ryser. *Combinatorial mathematics*, volume 14. American Mathematical Soc., 1963.
- [38] Christian Reksten-Monsen et al. Taipanrex/pyvisigraph: Given a list of simple obstacle polygons, build the visibility graph and find the shortest path between two points. <https://github.com/TaipanRex/pyvisigraph>, Last modified: 2018. Accessed: 2023-05-11.
- [39] David Eberly. Triangulation by ear clipping. *Geometric Tools*, pages 2002–2005, 2008.



# Instances



**Figure A.1:** The randomly generated instances with proposed and reference solutions. The cyan path is the solution found by **SOM** [6], the green path is the solution found by **GLNS-CETSP**[7], the red path is found using the **Sampled**-based method with 6 samples, the blue path is found with **Post-optimization heuristic**.

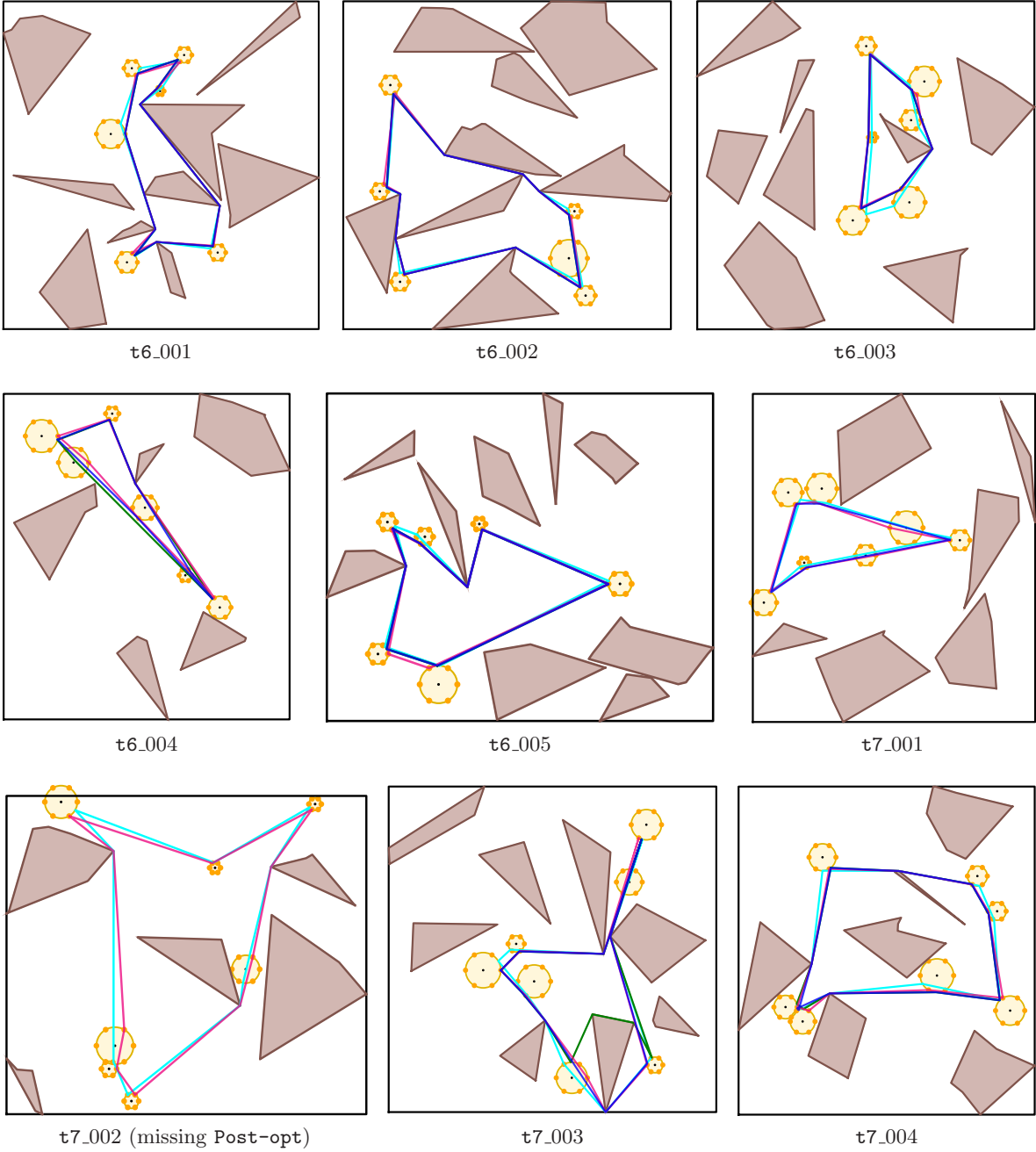


Figure A.2: The randomly generated instances with proposed and reference solutions contd.

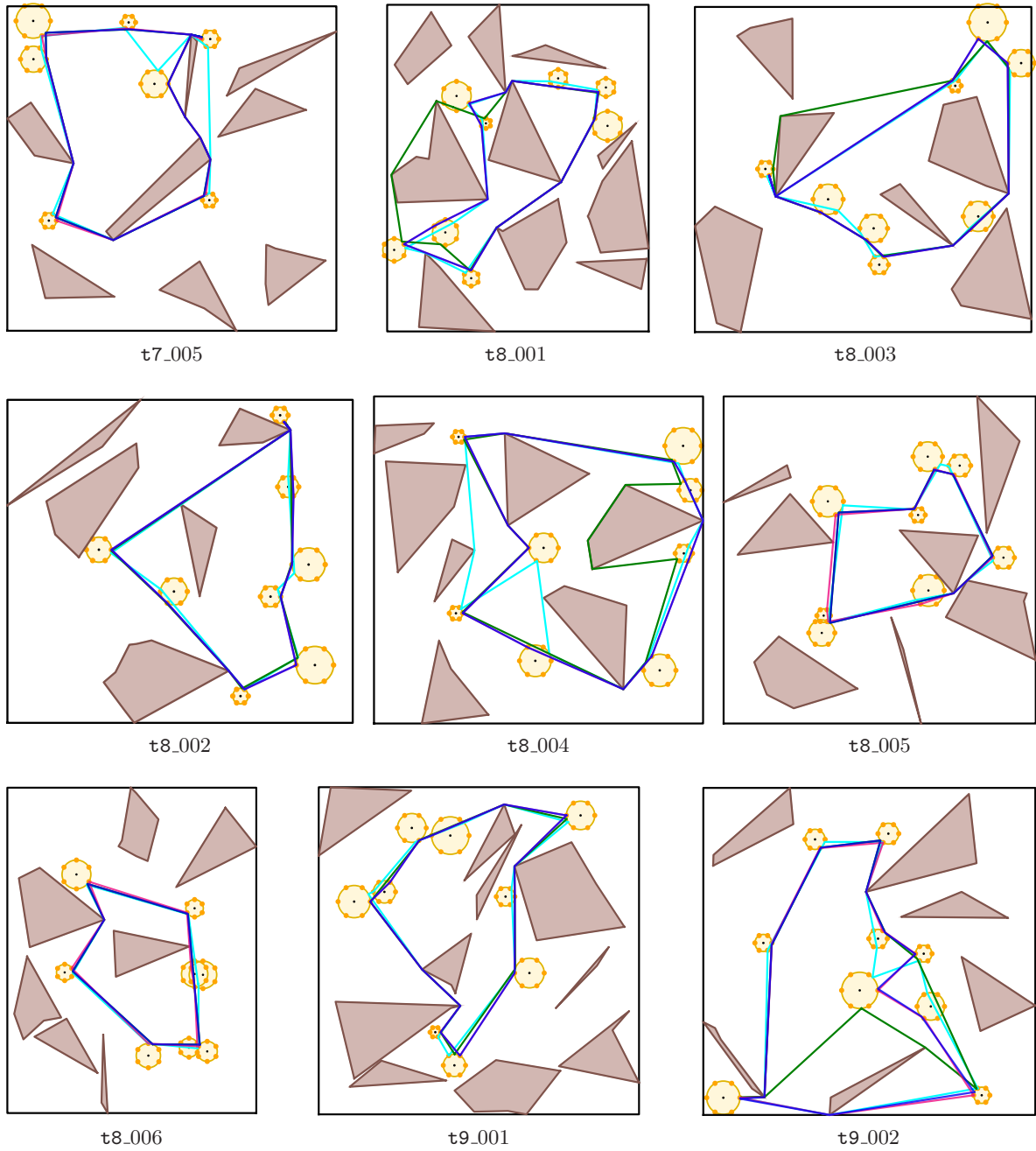


Figure A.3: The randomly generated instances with proposed and reference solutions contd.

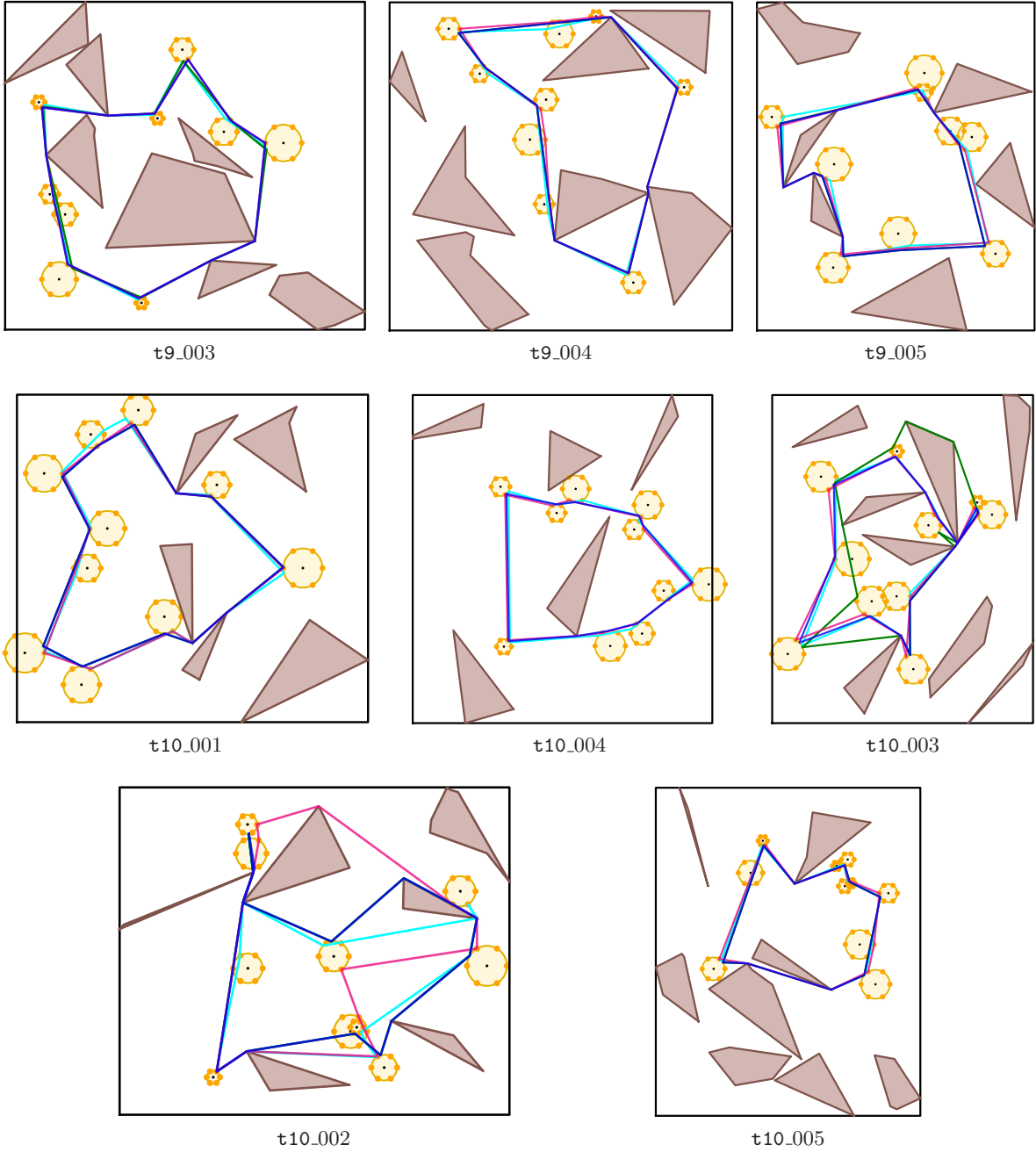


Figure A.4: The randomly generated instances with proposed and reference solutions contd.