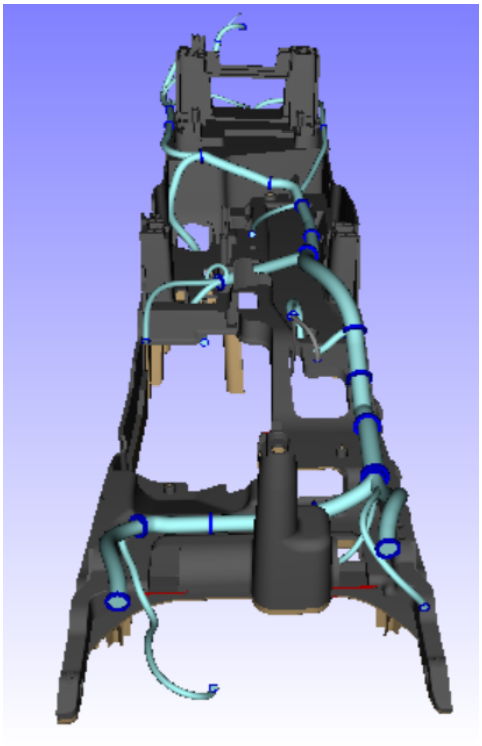
CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG



Optimization of Cable Harness Routing

Mathematical Modelling, Lagrangian Relaxation, and Subgradient Optimization

Master's thesis in Engineering Mathematics and Computational Science

Tobias Karlsson

Department of Mathematical Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Master's thesis 2020

Optimization of Cable Harness Routing

Mathematical Modelling, Lagrangian Relaxation, and Subgradient Optimization

Tobias Karlsson

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2020

Optimization of Cable Harness Routing
Mathematical Modelling, Lagrangian Relaxation, and Subgradient Optimization
Tobias Karlsson

Supervisors: Edvin Åblad and Tomas Hermansson, Fraunhofer-Chalmers Centre
Examiner: Ann-Brith Strömberg, Chalmers University of Technology

Master's Thesis 2020
Department of Mathematical Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Harness routing of an industrial instance.

Typeset in LaTeX
Gothenburg, Sweden 2020

Optimization of Cable Harness Routing
Mathematical Modelling, Lagrangian Relaxation, and Subgradient Optimization
Tobias Karlsson
Department of Mathematical Sciences Chalmers University of Technology and University of Gothenburg

Abstract

The problem of routing several cables, which should be grouped into a compound structure, can be a time consuming process when done manually. In this thesis, this problem is modelled as a mixed integer linear programming (MILP) problem. There are several factors to consider when designing a harness routing, and the MILP model contains two conflicting objectives which minimize two specific factors: the length of each distinct cable and the usage of space. A collection of Pareto optimal solutions is computed by assigning different weights to the objectives. Two other factors that are considered in the model formulation are minimum clearance to obstacles, modelled as hard constraints, and preferable zones for the routes as soft constraints. The problem is a large-scale optimization problem, and Lagrangian relaxation is utilized in the solution process. A deflected subgradient method is used to solve the Lagrangian dual problem, and to provide upper and lower bounds on the optimal objective value. Ergodic sequences of the Lagrangian subproblem solutions are utilized for branching decisions during the subgradient iterations, and are also utilized for constructing so-called core problems. Our approach is applied to an industrial test case and it results in a good harness design with respect to the factors mentioned above. For the test cases in this thesis, the relative duality gaps vary between 0.59% and 21.7% for varying objective weights. Our results also indicate that we can get good solutions within an acceptable time frame, that is in a few minutes. We suggest a number of possible improvements of our approach to reduce the computing times.

Keywords: cable routing, harness design, multi-objective optimization, mixed-integer linear programming, Lagrangian relaxation, subgradient optimization.

Acknowledgements

First of all I would like to thank my supervisors Edvin Åblad and Tomas Hermansson from Fraunhofer-Chalmers Centre (FCC). Two inspiring and gifted engineers, who gave a great deal of support and advises during the thesis work. Edvin has been a special contributor to the development of the algorithm and giving insight of theoretical parts.

Jonas Kressin from FCC has supported me on how to use the IPS software. Two skilled developers, Christian Larsen from FCC and Joakim Thorén (previously) from Industrial Path Solutions, also gave insights of the IPS functionality and supported me in my developing of my C++ skills.

I also want to thank Johan S. Carlson from FCC, and my examiner Ann-Brith Strömberg from Chalmers University of Technology, for support and guidance during the thesis work.

Tobias Karlsson, Gothenburg, September 2020

Contents

1
Introduction

1.1 Background and aim

Fraunhofer Chalmers Centre (FCC) is a research centre that offers research, software and services for a broad range of industrial applications. This includes modelling, simulation, and optimization of products and processes, which can boost technical development, improve efficiency, and cut costs. One key industrial application for FCC is quasi-static simulation of flexible cables. In order to set up such simulations, the cables first need to be routed in a collision-free way. Existing solutions can automatically route a single cable one at a time with respect to a triangulated geometry. Design constraints such as minimum bending radius and/or minimum clearance can also be imposed on the routing. However no solution for routing several cables simultaneously currently exists.

A cable harness is an assembly of cables or wires. A large number of cables are used in many different scenarios, such as in aircrafts, automobiles, computers, etc. There are advantages of having cable harnesses; the amount of space needed for the cables can be reduced, the increased security, and the time of installation can be reduced. It is common that routing designs are done manually, and when many requirements of the routing design have to be considered, this task can become complicated and very time consuming. By automating the cable harness routing process, several benefits can be achieved, e.g., the process can become less time consuming, and by optimizing the routing layout, costs can be reduced.

Figure 1.1 shows an example of how a cable harness layout can look like in an industrial application.

Figure 1.1: Installation of cable harnesses.

1.2 Problem description

The overall goal in this thesis is to investigate how to model and solve a harness routing problem as an optimization problem. We consider a bounded space in $\mathbb{R}^3$, where there are free space to route in and obstacles that the routes must avoid. In this space there are start and end points that will be connected with cables. A route in this scenario means the path of a cable through the space. The problem will be called the *cable harness routing problem* (CHRP), and the formulation of the problem in this thesis is one of several possible formulations. It is specifically of interest to investigate if one can obtain satisfactory solutions by formulating the CHRP as a multi-objective *mixed-integer linear program* (MILP) problem. It is also of interest to get an understanding of the problem complexity with this approach and to find good solution methods. The main research questions are formulated as follows:

- How can a multi-objective MILP model be formulated, resulting in a desired outcome for a given industrial instance?
- What methods and algorithms can be used to solve the model when regarding, among other, time complexity and how to handle more than one objective?

Considering the time complexity, an objective is that we should be able to solve industrial-size problem instances within, at most, a few minutes. The desired outcome is a bit open-ended. There are many factors that can be considered when designing harnesses; the ones considered in this thesis are:

- the length of the cables,
- the amount of space used by the cables,
- collision free routes,
- a minimum clearance, and
- preferable zones to route in.

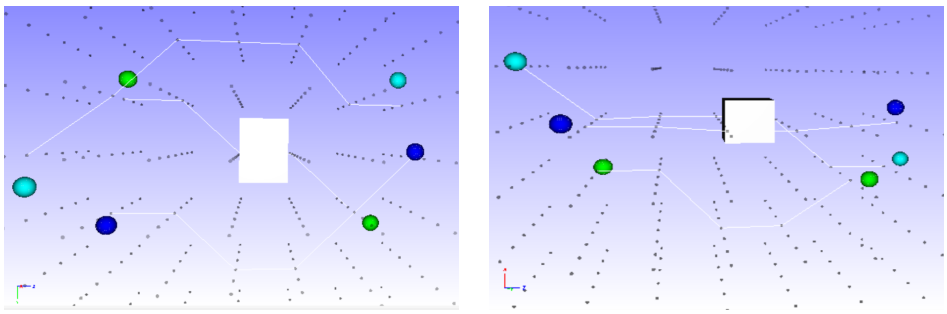
1.3 Methodology overview

An overview of the solution methodology is given here to, early in the thesis, clarify how the problem is formulated, and motivate why the specific mathematical tools are presented in Section 2.

The problem is formulated on a graph with nodes and directed edges. The approach is to discretize a 3D space with grid points as illustrated in Figure 1.2. The nodes represent the grid points, and the edges represent the physical paths that a route can take. We want to find paths, or routes, between node pairs with consideration to the factors mentioned in Section 1.2. A node pair consists of a start node and an end node. The grid points are uniformly distributed in a bounded cubic domain in $\mathbb{R}^3$ and can thus be identified as points in $\mathbb{N}^3$. The start and end nodes are defined anywhere in $\mathbb{R}^3$, and are mapped to their respective closest grid points. The resulting paths between the node pairs represent the cables. The grid is always cubic and the grid resolution is homogeneous.

There are two conflicting objectives in the MILP model that is formulated in this

thesis. One objective is to minimize the total route lengths of the cables, and the second objective is to minimize the number of nodes, or grid points, used by these routes altogether.



(a) (b)

Figure 1.2: Example of a discretized grid space with three start/end node pairs and a rectangular block obstacle. The paths (white lines) are routed between each node pair.

The problem is a large-scale optimization problem (for industrial-size instances). To get satisfactory solutions the grid often has to have a high resolution. Therefore, a method for handling large-scale problems is utilized, namely Lagrangian relaxation.

1.4 Limitations

The factors and requirements considered in this thesis are in general not enough for a cable (harness) routing. An important requirement that is not considered is to satisfy a minimum bending radius of the cable(s). The routing we are doing are nominal, meaning that it is a first step in a process where the cables later on will be simulated and where, for example, the effect of deformation due to gravity will be considered. Other factors and requirements that can be considered are mentioned in Section 1.5. It can be desirable to be able to specify waypoints of routings, but this is not considered in this thesis. These limitations are planned to be considered in future work by modifying solutions given from the model and algorithm developed in this thesis.

The model in this thesis is just one way to formulate the CHRP; it would be of interest to try other formulations, built on other decision variables or objective functions, but this is considered as future work. The grid discretization of the 3D space is always cubic, which is not always suitable, but which has simplified our implementation. In the future, the algorithm should be able to handle rectangular grids, and possibly also a non-homogeneous grid resolution. In zones where there are no obstacles, the requirement of the grid resolution will probably not be as high as in zones with a lot of obstacles.

1.5 Related work

This section gives a brief overview of the challenges of cable harness routing, which factors a harness designer might consider, which methods have been used previously to automate the process of harness routing, and the mathematical tools that has been used for the algorithm in this thesis.

There are a lot of factors that can be considered for a cable harness routing. One factor that first comes to mind is the length of the routing, which is for example mentioned in [1, 2, 3]. In electrical cables, the cable length is proportional to the voltage drop, which is another important concern for harness designers [4]. Other factors that can be considered are acceptable bend radii, position and distribution of fasteners used to constrain the harness [4], cable hazardous zones, e.g., zones with high temperature, moist, or vibrations [2].

In 2000, Ng et al. [4] mentioned that efforts have been made to automate the choice of a cable harness path through the use of Artificial Intelligence (AI), but with little success. They mention that a problem is that the harness routing problem is too open-ended and it is difficult to capture the design intent of the activity, and they imply that human input is required to guide the computer to reach an "optimal" solution. More recently, in 2016, Pemarathne and Fernando in [5] review proposed cable and wiring layout systems designed with AI. They state that their analysis have shown that the evolution of designing from manual trial and error to intelligent simulation software has proved vast improvement.

Conru in [6] defined the CHRP as a search problem in a graph. The author used a genetic algorithm to optimize a cable harness configuration by minimizing an objective function based on the number of cables passing through each edge in the graph. Conru mentions that the CHRP appears similar to the *Steiner tree problem* (STP) [7], although the STP is not really applicable for the problem formulation in the paper, since the cost corresponding to each edge in the graph is different for different harness configurations. Fernando and Kalganova in [3] use Ant Systems for multi-hose routing to find paths that minimizes the total length of the paths and maximizes the shared lengths of the paths.

Some solutions to the CHRP that is formulated in this thesis resemble Steiner trees. These solutions occur when the objective to minimize the amount of nodes that are used, is highly weighted. Steiner trees have been used in routing problems before, and even in relation to harnessing, Lin et al. in [8] proposes an algorithm for weight minimization of wires. They formulate the wire routing problem as an STP where the location of a Steiner vertex is selected for adding a splice connecting more than two wires. Klein and Haugland in [9] formulate a model for cost minimization of cable layouts (in the context of offshore wind farms) which has some resemblance with the STP. Lagrangian relaxation has been used for solving the STP, for example in [7]. More recently, Leitner et al. in [10] present a dual ascent algorithm for solving the STP. The STP might be useful to consider because some solutions to the problem in this thesis resemble a Steiner tree, although the STP is not considered in this thesis and is considered as future work.

The model developed in this thesis is Lagrangian relaxed, and a dual problem is formulated, which others have done before for resembling problems (see the discus-

sion above on the STP). The algorithm developed in this thesis utilizes, among other things, a deflected subgradient algorithm, and ergodic sequences for fixing variable values. Similar algorithms has been proposed before (but not specifically for the CHRP), for example in [11], [12]. The deflected subgradient algorithm is presented in [13]. Ergodic sequences are also used to construct so called core problems, as in [12].

2
Mathematical tools

This section presents background theory and mathematical tools that are used in this thesis to build the algorithms presented in Section 4, or that is related to the model presented in Section 3.

2.1 Shortest path problem

The *shortest path problem* (SPP) is relatable to the CHRP model in this thesis. SPP is a network optimization problem, i.e., it is defined on nodes and arcs, and the linear program (LP) formulation of this problem has *integrality property*, i.e., there exists an optimal integer solution [14, p. 216].

To show how the SPP can be formulated as an LP, we define the variables

$$x_{ij} = \begin{cases} 1 & \text{if arc } (i,j) \in A \text{ is part of the shortest path,} \\ 0 & \text{otherwise.} \end{cases}$$

The node set and arc set are denoted by V and A, respectively, and the arc costs are denoted as a_{ij}, $(i,j) \in A$. The LP model of the SPP can then be formulated as

$$\min \quad \sum_{(i,j)\in A} a_{ij}x_{ij}, \tag{2.1a}$$

$$\text{s.t.} \quad \sum_{i\in V:(i,k)\in A} x_{ik} - \sum_{j\in V:(k,j)\in A} x_{kj} = 0, \quad k \in V \setminus \{s,e\}, \tag{2.1b}$$

$$\sum_{i\in V:(s,i)\in A} x_{si} = 1, \tag{2.1c}$$

$$\sum_{i\in V:(i,e)\in A} x_{ie} = 1, \tag{2.1d}$$

$$x_{ij} \geq 0, \quad (i,j) \in A. \tag{2.1e}$$

The start node is denoted by s and the end node by e. The constraints (2.1b) are flow conservation constraints, stating that the flow into every node (that is not the start or end node) must flow out from it. The constraint (2.1c) makes sure that the path flows from the start node, and (2.1d) ensures that it reaches the end node. There is no need for binary constraints on the decision variables because of the integrality property. The SPP is not generally solved as an LP problem, but instead by using more efficient algorithms designed for finding shortest paths in a network, such as Dijkstra's algorithm, which is a special case of so-called Dynamic Programming algorithms, see [14, p. 192], [15]. However, these results are useful later in this thesis.

2.2 Lagrangian relaxation for mixed-integer linear programming

For problems with integer variables, it is common to apply a relaxation to recieve lower, or optimistic, bounds on the optimal objective function value. The solution to a relaxed problem can for example be used to prove infeasibility or optimality. Relaxations can be used in methods that search for an optimal solution, e.g., branch and bound, or find a good solution fast in cases when the original problem formulation is too complex. One popular relaxation method is Lagrangian relaxation [16], which approximates a difficult problem with a simpler one.

We have a MILP model when some variables are continuous and others are restricted to integer values. By denoting the continuous and integer variables as $\boldsymbol{x}_C$ and $\boldsymbol{x}_I$, respectively, we can formulate the model in matrix form as:

$$\begin{aligned} \min \quad & \boldsymbol{c}^T\boldsymbol{x}, && \text{(2.2a)}\\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, && \text{(2.2b)}\\ & \boldsymbol{x} \geq \boldsymbol{0}, && \text{(2.2c)}\\ & \boldsymbol{x}_I \in \mathbb{Z}^n, && \text{(2.2d)} \end{aligned}$$

where $\boldsymbol{x} = [\boldsymbol{x}_C^T \ \boldsymbol{x}_I^T]^T$ and n is the number of integer variables.

The procedure of Lagrangian relaxation will now be explained; it also can be read about in [16], [14, p. 455–463]. In Lagrangian relaxation we relax "complicating" constraints such that the relaxed problem is relatively easy to solve. Let us denote the feasible set to the problem (2.2) as $X := \{\boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b},\ \boldsymbol{x} \geq 0,\ \boldsymbol{x}_I \in \mathbb{Z}\}$, where the constraints constructing the set X are considered to be "simple" in the sense that the resulting optimization problem can be solved efficiently. Suppose now that in addition to these constraints, we have some "complicating" constraints denoted as $g_i(\boldsymbol{x}) \leq 0,\ i = 1, \ldots, m$. The resulting model would then be

$$\begin{aligned} \min \quad & \boldsymbol{c}^T\boldsymbol{x}, && \text{(2.3a)}\\ \text{s.t.} \quad & g_i(\boldsymbol{x}) \leq 0, \quad i = 1, \ldots, m, && \text{(2.3b)}\\ & \boldsymbol{x} \in X. && \text{(2.3c)} \end{aligned}$$

By relaxing the constraints (2.3b) the Lagrangian function for problem (2.3) is obtained as

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) := \boldsymbol{c}^T\boldsymbol{x} + \sum_{i=1}^{m} \lambda_i g_i(\boldsymbol{x}),$$

where $\lambda_i \geq 0,\ \ i = 1, \ldots, m$ are Lagrangian multipliers, or dual variables, and the relaxed problem takes the following form:

$$h(\boldsymbol{\lambda}) := \min_{\boldsymbol{x} \in X} L(\boldsymbol{x}, \boldsymbol{\lambda}). \tag{2.4}$$

This means that if the constraints (2.3b) are not satisfied, the Lagrangian function is penalized in the relaxed problem. If $\boldsymbol{x}$ is feasible in the original problem (2.3), we have that $\boldsymbol{c}^T\boldsymbol{x} \geq L(\boldsymbol{x}, \boldsymbol{\lambda}) \; \forall \, \boldsymbol{\lambda} \geq \mathbf{0}$, which implies that the Lagrangian function provides a lower bound on the optimal value of our original problem. The minimization problem (2.4) is called the Lagrangian subproblem. A solution to (2.4), for given dual variable values, will be denoted as $\boldsymbol{x}(\boldsymbol{\lambda})$.

The problem of finding the best relaxation bound is called the Lagrangian dual problem. The Lagrangian dual function is denoted by h, and the Lagrangian dual problem is formulated as

$$\max_{\boldsymbol{\lambda} \geq \mathbf{0}^m} \; h(\boldsymbol{\lambda}). \tag{2.5}$$

The Lagrangian dual problem is always a convex problem, which implies that it can be solved efficiently [17].

2.3 Subgradient optimization

Subgradient optimization is a popular method for solving the Lagrangian dual problem. It is an iterative procedure which can solve the problem of maximizing a non-smooth concave function, such as h. We say that $\boldsymbol{s} \in \mathbb{R}^m$ is a subgradient of h at $\boldsymbol{\lambda} \in \mathbb{R}^m_+$ if it is an element of the subdifferential:

$$\partial h(\boldsymbol{\lambda}) = \{\boldsymbol{s} \in \mathbb{R}^m \mid h(\boldsymbol{\gamma}) \leq h(\boldsymbol{\lambda}) + \boldsymbol{s}^T(\boldsymbol{\gamma} - \boldsymbol{\lambda}), \;\; \boldsymbol{\gamma} \in \mathbb{R}^m_+\}.$$

In the subgradient method the dual variables are updated as

$$\boldsymbol{\lambda}^{k+1} = \left(\max\left\{0, \; (\boldsymbol{\lambda}^k)_i + \alpha_k (\boldsymbol{d}^k)_i \right\} \right)_{i=1}^m, \tag{2.6}$$

where the search direction $d^k := s(\lambda^k)$ and $\alpha_k > 0$ is a step length at iteration k. One choice of subgradient is $s(\boldsymbol{\lambda}) = g(\boldsymbol{x})$ if g_i, $i = 1, \ldots, m$ in (2.3b) is linear.

To theoretically guarantee convergence, the step lengths have to fulfill the requirements $\alpha_k \to 0$ and $\sum_{i=1}^k \alpha_i \to \infty$, as $k \to \infty$. Step lengths that satisfy these requirements are given by $\alpha_k = \frac{1}{k}$, $k = 1, 2, \ldots$ [14, p. 470]. Example of other step lengths that guarantees convergence are divergent series step lengths [12, p. 512] and the Polyak step lengths [12, p. 515].

A common choice of step length is

$$\alpha_k = \delta_k \frac{u - h(\boldsymbol{\lambda}^k)}{\|\boldsymbol{d}^k\|^2},$$

where u is the best known upper bound on (2.5), and $\delta_k \in]0, 2[$. This does not satisfy the theoretical requirement for convergence, but it works well in practice [14, p. 470]. If $\delta_k = 1$, we have theoretical convergence [12, Prop. 15.8].

When the search direction is chosen to simply be a subgradient, the steps towards the optimal solution can be zigzagging. This zigzagging phenomenon can be reduced, so that the iterates $\boldsymbol{\lambda}^k$ approach an optimal solution faster, by using a so called

deflected subgradient search direction [13]. The deflected subgradient is computed as

$$\boldsymbol{d}^k = \boldsymbol{s}(\boldsymbol{\lambda}^k) + \Psi_k \boldsymbol{d}^{k-1}, \tag{2.7}$$

where $\boldsymbol{d}^0 = \mathbf{0}$ and $\Psi_k \geq 0$ is a deflection parameter. Belgacem and Amir [13] presents a deflection parameter that is computed as

$$\Psi_k = \begin{cases} \frac{-\eta(1-\beta)\boldsymbol{s}(\boldsymbol{\lambda}^k)\cdot\boldsymbol{d}^{k-1} + \beta\|\boldsymbol{s}(\boldsymbol{\lambda}^k)\|\|\boldsymbol{d}^{k-1}\|}{\|\boldsymbol{d}^{k-1}\|^2}, & \text{if } \boldsymbol{s}(\boldsymbol{\lambda}^k)\cdot\boldsymbol{d}^{k-1} < 0 \\ 0, & \text{otherwise,} \end{cases} \tag{2.8}$$

where $0 < \eta \leq 2$ and $\beta \in [0, 1]$. This deflection parameter makes $\boldsymbol{d}^k$ a convex combination of two deflected directions; one direction, $\boldsymbol{d}_{\text{MGT}}$, used in an algorithm called Modified Gradient Technique (MGT), and $\boldsymbol{d}_{\text{ADS}}$ used in the Average Direction Strategy (ADS). This means we have that $\boldsymbol{d}^k = (1-\beta)\boldsymbol{d}^k_{\text{MGT}} + \beta\boldsymbol{d}^k_{\text{ADS}}$.

Algorithm 1 shows the subgradient optimization procedure, which follows the algorithms in [14, p. 461] and [13]. The algorithm terminates when the difference between the upper and lower bound is satisfactory small, or when a maximum number of iterations has been reached.

Algorithm 1: Deflected subgradient algorithm

1. Choose initial values $\boldsymbol{\lambda}^0$. Set $k = 0$, lower bound $l = -\infty$, upper bound $u = \infty$, $\delta_0 = 2$, $p \in \mathbb{N}$, $\kappa \in]0, 1[$
2. Solve the Lagrangian subproblem for given $\boldsymbol{\lambda}^k$. This gives a solution $\boldsymbol{x}(\boldsymbol{\lambda}^k)$ and an optimistic bound $h(\boldsymbol{\lambda}^k)$. Find a feasible solution $\boldsymbol{x}_{\text{feas}}$ using $\boldsymbol{x}(\boldsymbol{\lambda}^k)$, and compute the pessimistic bound $\boldsymbol{c}^T\boldsymbol{x}_{\text{feas}}$. Update $l = \max\{l,\ h(\boldsymbol{\lambda}^k)\}$ and $u = \min\{u,\ \boldsymbol{c}^T\boldsymbol{x}_{\text{feas}}\}$.
3. Determine subgradient $\boldsymbol{s}(\boldsymbol{\lambda}^k) = g(\boldsymbol{x})$ and search direction $\boldsymbol{d}^k$ according to (2.7) and (2.8).
4. Update the step length $\alpha_k = \delta_k \frac{u - h(\boldsymbol{\lambda}^k)}{\|\boldsymbol{d}^k\|^2}$.
5. Compute $\boldsymbol{\lambda}^{k+1}$ according to (2.6)
6. Decrease δ_k as $\delta_k := \kappa\delta_k$ if l has not been improved the last p iterations.
7. Check convergence criterion and terminate if fulfilled.
8. Set $k := k + 1$ and go to 2

The parameter δ_k is decreased in step 6 if the lower bound is not improved in some number of iterations. The reason for this is to decrease the step length and take smaller steps in the subgradient direction. How big p should be and how δ_k is decreased depends on the problem and the instance. Belgacem and Amir in [13] uses $\kappa = 0.5$ and $p = 20$, in their example. In this thesis, those settings decreases δ_k too fast, and κ will instead be closer to 1.

2.4 Primal convergence in subgradient optimization with ergodic sequences

When the original problem is nonconvex, such as discrete optimization problems, there is usually a non-zero gap between the optimal primal and optimal dual objective values. This makes it more complicated to define the termination criteria, since

we can not assume that the gap will tend to zero. It is also desirable to generate good primal solutions. Utilizing so-called ergodic sequences might help regarding this. For a convex program, an ergodic sequence of subproblem solutions converges to the solution set of that problem [18, p. 293], provided that the step lengths in the dual subgradient algorithm and the weights defining the ergodic sequence fulfill certain (natural) requirements. For a mixed binary linear program, it can then be shown that the ergodic sequence converges to a solution set of a convexified version of the problem [12, p. 517]. The feasible set in the convexified version is defined as the intersection of the set defined by the inequality constraints in (2.3b) and the convex hull of the set X in (2.3c). The difference in optimal objective value of a MILP model and its LP-relaxation is called integrality gap. The smaller this gap is, the stronger is the model. When the problem formulation is strong, the convexified version lies close to the solution set of the original problem. So utilizing ergodic sequences for constructing primal feasible solutions hopefully results in good solutions.

The ergodic sequence $\{\tilde{\boldsymbol{x}}^t\}$ is defined as a convex combination of subproblem solutions, and can be computed as

$$\tilde{\boldsymbol{x}}^t := \frac{1}{\sum_{r=0}^{t-1} \alpha_r} \sum_{s=0}^{t-1} \alpha_s \boldsymbol{x}\left(\boldsymbol{\lambda}^s\right), \quad t = 1, 2, \ldots,$$

where the weights α_s are the same as the step lengths used in Algorithm 1. The updates can be done incrementally by the following approach [12, p. 517]:

$$\bar{\alpha}_1 := \alpha_0, \quad \bar{\alpha}_t := \bar{\alpha}_{t-1} + \alpha_{t-1}, \tag{2.9a}$$

$$\tilde{\boldsymbol{x}}^1 := \boldsymbol{x}(\boldsymbol{\lambda}^0), \tag{2.9b}$$

$$\tilde{\boldsymbol{x}}^t := \frac{\bar{\alpha}_t - \alpha_{t-1}}{\bar{\alpha}_t} \tilde{\boldsymbol{x}}^{t-1} + \frac{\alpha_{t-1}}{\bar{\alpha}_t} \boldsymbol{x}(\boldsymbol{\lambda}^{t-1}). \tag{2.9c}$$

Here, $\bar{\alpha}$ denotes the cumulative sum of the weights α_t. The convergence of the ergodic sequence of subproblem solutions with updates according to (2.9) is often slow [12, p. 517], partly because the weights α_t are lower for later iterates. An ergodic sequence, where later iterates are assigned higher weights than the earlier ones, can be obtained by using the so called s^k-rule [12, p. 519]. The ergodic sequence using the s^k-rule is defined by

$$\tilde{\boldsymbol{x}}^t := \sum_{s=0}^{t-1} \mu_s^t \boldsymbol{x}\left(\boldsymbol{\lambda}^s\right), \quad s = 0, \ldots, t-1,$$

with weights defined by

$$\mu_s^t := \frac{(s+1)^k}{\sum_{r=0}^{t-1} (r+1)^k}, \quad s = 0, \ldots, t-1, \quad t = 1, 2, \ldots, k > 0.$$

Larger values of k results in weights shifted towards later iterates.

2.5 Multi-objective linear programming

Multi-objective linear programming (MOLP) is a specific case of general multi-objective programming (also known as multi-objective optimization), where all functions defining the problem are linear. MOLP is an area in mathematical optimization that is concerned about problems with more than one objective to be optimized simultaneously. Mathematically a MOLP problem can be formulated as

$$\min \quad f_1(\boldsymbol{x}),\ f_2(\boldsymbol{x}), \ldots,\ f_n(\boldsymbol{x}) \tag{2.10a}$$
$$\text{s.t.} \quad \boldsymbol{x} \in X \tag{2.10b}$$

where n is the number of linear objective functions and $X = \{\boldsymbol{x} \geq \boldsymbol{0} \mid A\boldsymbol{x} \leq \boldsymbol{b}\}$ is the feasible set.

In many cases of MOLP, the objective functions are in conflict, which means that there is no solution that simultaneously optimizes each objective [19]. The goal is to find a solution that is optimal in some sense, and with conflicting objectives there exists so called *Pareto optimal* solutions. A solution is Pareto optimal, also called efficient, if none of the objective function values can be improved without worsening at least one of the other objective function values. Lootsma in [20, p. 230] formulates the definition as: a feasible solution $\tilde{\boldsymbol{x}} \in X$ is an efficient solution if there is no feasible solution $\boldsymbol{x} \in X$ such that

$$f_i(\boldsymbol{x}) \leq f_i(\tilde{\boldsymbol{x}}),\ i = 1, \ldots, n,$$
$$f_k(\boldsymbol{x}) < f_k(\tilde{\boldsymbol{x}}),\ \text{for some } k \in \{1, \ldots, n\}.$$

Moreover, a feasible solution $\tilde{\boldsymbol{x}}$ is weakly efficient, or weakly Pareto optimal, if there is no feasible solution $\boldsymbol{x}$ such that

$$f_i(\boldsymbol{x}) < f_i(\tilde{\boldsymbol{x}}),\ \ i = 1, \ldots, n.$$

This means that from a weakly efficient solution, it is impossible to improve all objective functions, but it may be possible to improve at least one other objective function without worsening the others. Figure 2.1 illustrates weakly efficient and efficient solutions on the *Pareto front* in the objective function space, in the case with two objective functions. The Pareto front is the set of all Pareto optimal solutions.

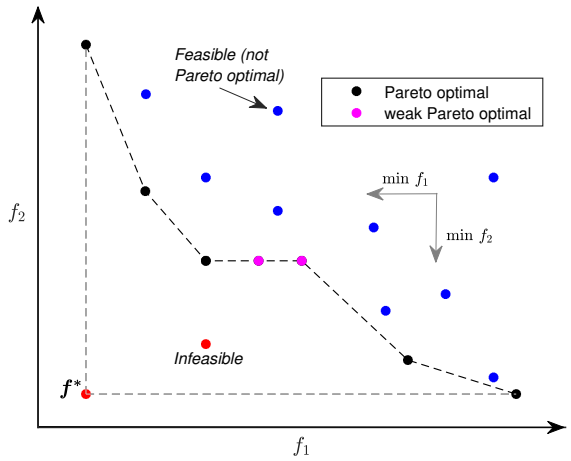


Figure 2.1: Illustration of a Pareto front with two objective functions to minimize, i.e., going in the direction of the grey arrows. The black line corresponds to the Pareto front and black points are Pareto optimal, while points on the dashed line (the pink ones) are weakly Pareto optimal. $\boldsymbol{f}^*$ corresponds to the ideal objective vector.

There exist different optimal solutions for each of the n conflicting objectives, which are found by solving

$$\min_{\boldsymbol{x} \in X} \quad f_i(\boldsymbol{x}),$$

for $i = 1, 2, \ldots, n$, and obtain the corresponding optimal objective values f_i^*, we get a so called ideal objective vector

$$\boldsymbol{f}^* = [f_1^*,\ f_2^*, \ldots, f_n^*].$$

In general, the ideal objective vector $\boldsymbol{f}^*$ does not correspond to a feasible solution, but Pareto solutions that are closer (in distance) to this point might be better [21, p. 410].

MOLP can be divided into two subfields: (1) find a representative set of solutions on the Pareto front, and (2) find a single solution that satisfies a subjective preference of a designer [20, p. 230], [19, p. 1106]. Generation of several solutions on the Pareto front is good for analyzing the problem, but in the end, there will be need for only one solution. Two different approaches are suggested for this purpose in [21]. One approach is divided into the following two steps:

Step 1: Find multiple trade-off (efficient) solutions with a wide range of the objective function values.

Step 2: Choose one of the obtained solutions using higher-level information.

In this approach, several solutions on the Pareto front are found before a final choice is made. The final choice by using "higher-level information" can often be non-technical and experience driven, [21, p. 407–408]. When we have two objective functions, i.e., $n = 2$, the Pareto front can be visualized in a 2D plot by plotting

the corresponding solution points in the objective function space, such as in Figure 2.1. This can help to choose a satisfactory solution. The second approach for finding one efficient solution can be done using a method sometimes referred to as weighted sum method, which will be discussed in more detail in Section 2.5.1. In this approach, higher-level information is first used to decide how the objectives should be prioritized, then a composite function is created into one scalar function. This composite function is then optimized over the feasible set to find a single trade-off optimal solution.

In the following subsections, two methods are presented, which are used in this report to handle the MOLP problem for different purposes. The method presented in Section 2.5.2 is used to compute all of the Pareto optimal and weakly Pareto optimal solutions. The weighted sum method in Section 2.5.1 is used to find a single point, or a few points on the Pareto front, and is also the method used together with the Lagrangian relaxation of the model described in Section 3.2.

2.5.1 Weighted sum method

For the weighted sum method a vector with positive weights $\boldsymbol{w} = [w_1, w_2 \ldots, w_n]$ is used to create the single objective function

$$F(\boldsymbol{x}) = \sum_{i=1}^{n} w_i f_i(\boldsymbol{x}). \tag{2.11}$$

A solution that minimizes (2.11) over the set X (defined in (2.10b)) is proven to be weakly efficient if $w_i \geq 0$, $i = 1, \ldots n$, and efficient if $w_i > 0$, $i = 1, \ldots n$ [22, p. 10].

With prior knowledge of a good preference vector, this is a suitable method to obtain an efficient solution. This method can be used to find multiple efficient solutions by using different preference vectors $\boldsymbol{w}$ and resolving the MOLP problem. A problem is that a uniform choice of weight vectors does not necessarily find a uniform set of solutions on the Pareto front. Another issue is that solutions lying on non-convex parts of the Pareto front can not be found with this approach, [21, p. 420]. When we have binary variables as in the model in this thesis, the Pareto front can be non-convex, implying there may exist (weakly) efficient solutions that can not be found with this method.

2.5.2 ε-constraint method

An approach that works better than the weighted sum method is the ε-constraint method, with respect to the issues mentioned in the end of Section 2.5.1. Here the MOLP problem is reformulated such that only one objective is optimized, while the rest are restricted using constraints. The reformulation of the problem looks as following:

$$\begin{aligned}
\min \quad & f_k(\boldsymbol{x}), && && \text{(2.12a)}\\
\text{s.t.} \quad & f_i(\boldsymbol{x}) \leq \varepsilon_i, && i \in \{1, \ldots, n\} \setminus k, && \text{(2.12b)}\\
& \boldsymbol{x} \in X. && && \text{(2.12c)}
\end{aligned}$$

An optimal solution to (2.12) is always weakly efficient [22, p. 12].

3

Model description

So far the main mathematical tools that are used in this thesis have been presented. In this section, we begin with definitions and notations that are used to formulate the model. How the nodes and arcs, and their corresponding costs, are defined, will also be described in Section 3.1. Thereafter, the model is presented, and the Lagrangian relaxation of the model is described and so is the reformulation for the ε-constraints method.

3.1 Definitions and notations

Let's define the following notations:

- V - set of nodes,
- K - set of cables,
- A_k - set of directed arcs $(\boldsymbol{i}, \boldsymbol{j})$ for cable $k \in K$,
- s_k - start node for cable $k \in K$,
- e_k - end node for cable $k \in K$,
- n_i - cost for using node $\boldsymbol{i} \in V$,
- a_{ij} - cost for traversing arc $(\boldsymbol{i}, \boldsymbol{j}) \in A_k$, $k \in K$.

The problem is defined as a graph in $\mathbb{R}^3$ with free space and obstacles. In this 3D space a grid is defined, where the grid-points are nodes in $V \subseteq \mathbb{N}^3$. Lets denote the number of grid-points in x-, y- and z-direction as d_x, d_y, $d_z \in \mathbb{N}$, respectively. A node $\boldsymbol{i} \in V$ corresponds to a grid-point in $\mathbb{N}^3$. In this thesis, a cubic grid has been used, i.e., $d_x = d_y = d_z$. Lets denote the number of grid points in a direction as $d = d_x$. An example of such a grid can be seen in Figure 3.1. The start and end positions of a cable k are located in $\mathbb{R}^3$, and these positions are mapped to their respective closest grid points such that $s_k, e_k \in V$, $k \in K$. From this setting, two specific questions arises:

1. How should the arcs be defined?
2. How should the node costs and arc costs be defined?

For each cable, there exists arcs from every node $\boldsymbol{i} \in V$ to all nodes in a neighbourhood of $\boldsymbol{i}$. This neighbourhood is defined as

$$N(\boldsymbol{i}) := \Big\{ \boldsymbol{j} \in V \setminus \boldsymbol{i} \mid \|\boldsymbol{i} - \boldsymbol{j}\|_\infty \leq 1 \Big\}.$$

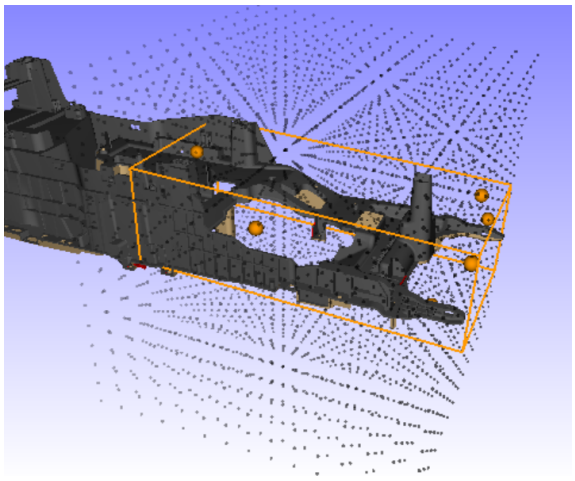


Figure 3.1: Grid points (nodes) as dark dots, in a 3D instance with orange start and end nodes. The start and end nodes are inside the rectangular box created by the orange lines.

Figure 3.2 illustrates the neighbourhood of a grid point. The set of all possible directed arcs for cable $k \in K$ is defined as

$$A_k := \Big\{ (\boldsymbol{i}, \boldsymbol{j}) \mid \boldsymbol{i} \in V \setminus e_k,\ \boldsymbol{j} \in V(\boldsymbol{i}) \setminus s_k \Big\}.$$

In other words, each arc is are going from one node to a node in its neighbourhood, but for each cable, no arcs start in its end node and no arcs end in its start node.

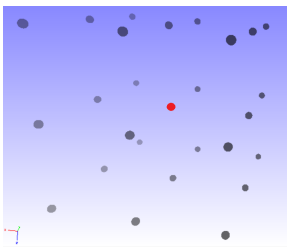


Figure 3.2: Dark grid points are in the neighbourhood of the red grid point.

This leads to question 2 in this section. The arc cost for arc $(\boldsymbol{i}, \boldsymbol{j}) \in A$, are defined as

$$a_{ij} = \frac{1}{d-1} \|\boldsymbol{i} - \boldsymbol{j}\|_2. \tag{3.1}$$

The factor $\frac{1}{d-1}$ normalizes the arc-costs between instances and makes the objective values end up in similar ranges. We also get that the total length, or cost, for a straight path from one side of the grid to the opposite side is equal to one. The arc costs are proportional to the Euclidian distances, which implies that $a_{ij} = a_{ji}$.

The node costs can be defined in different ways depending on the instance and what the designer prefers. For the tests in this thesis, the lowest cost of a node is

$$n_{\min} = \frac{1}{d-1}, \tag{3.2}$$

which is also the lowest arc cost. The highest cost for a node is equal to infinity. Those nodes are inside (in collision with) or too close to an obstacle. By not using infeasible nodes with cost equal to infinity, we satisfy collision free routes and minimum clearances. For every node, we have a distance, or a clearance, $c \in \mathbb{R}$ from the corresponding grid point in $\mathbb{R}^3$ to the closest point in $\mathbb{R}^3$ on an obstacle, see Figure 3.3.

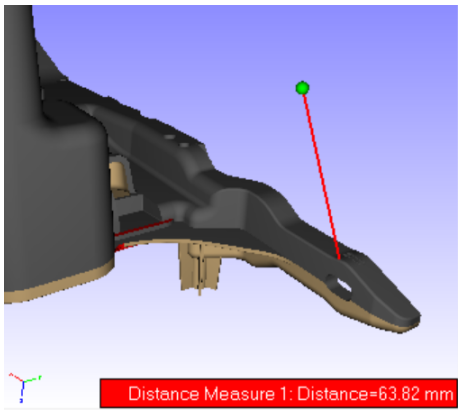
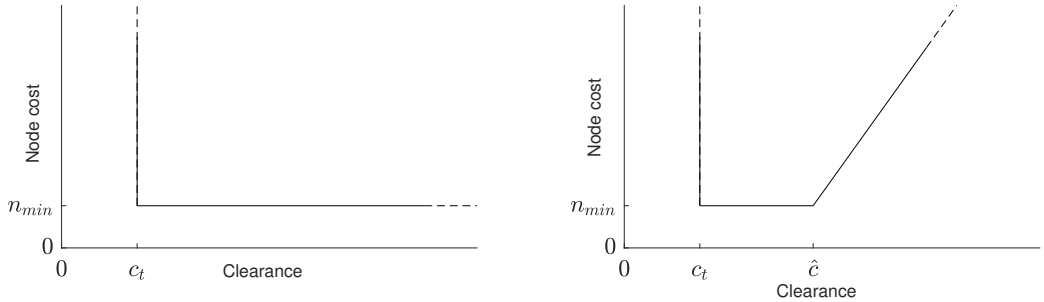


Figure 3.3: The red line shows the clearance for the green point.

In Figure 3.4 we can see examples of how the node costs can depend on the clearance. In the case in Figure 3.4(a), all nodes are equally expensive, except for the ones with a clearance below some threshold. If one prefer using nodes that are close to obstacles, the case in Figure 3.4(b) can be used, which penalizes nodes that are far away from any obstacle.



(a) (b)

Figure 3.4: Two different ways to define node costs, n_i, $i \in V$, as a function of clearance. $n_{\min} \in \mathbb{R}_+$ is the minimum node cost. $c_t, \hat{c} \in \mathbb{R}_+$ are thresholds.

3.2 Model

In this section the model that is formulated to solve the CHRP is presented. We define the decision variables as

$$x_i = \begin{cases} 1, & \text{if node } \boldsymbol{i} \in V \text{ is used,} \\ 0, & \text{otherwise,} \end{cases}$$

$$y_{ijk} = \begin{cases} 1, & \text{if arc } (\boldsymbol{i}, \boldsymbol{j}) \in A_k \text{ is traversed by cable } k \in K, \\ 0, & \text{otherwise.} \end{cases}$$

The model consists of two objective functions f_h and f_{SP}, defined as

$$f_h(\boldsymbol{x}) = \sum_{\boldsymbol{i} \in V} n_i x_i, \qquad \text{and} \qquad f_{SP}(\boldsymbol{y}) = \sum_{k \in K} \sum_{(\boldsymbol{i},\boldsymbol{j}) \in A_k} a_{ij} y_{ijk},$$

where f_h and f_{SP} relates to the harnessing aspect and the shortest paths for the cables, respectively.

To optimize for one objective function, a composite function is created by having a weighted sum of the objectives. The weights for f_h and f_{SP} are denoted by $w_h \geq 0$ and $w_{SP} \geq 0$, respectively. The model is formulated as following:

$$\min \quad \Big\{ w_h f_h(\boldsymbol{x}) + w_{SP} f_{SP}(\boldsymbol{y}) \Big\}, \tag{3.3a}$$

$$\text{s.t.} \quad \sum_{i:(i,l) \in A_k} y_{ilk} - \sum_{j:(l,j) \in A_k} y_{ljk} = 0, \qquad \boldsymbol{l} \in V \setminus \{s_k, e_k\},\ k \in K, \tag{3.3b}$$

$$\sum_{i:(s_k,i) \in A} y_{s_k ik} = 1, \qquad k \in K, \tag{3.3c}$$

$$\sum_{i:(i,e_k) \in A} y_{i e_k k} = 1, \qquad k \in K, \tag{3.3d}$$

$$\sum_{j:(l,j) \in A} y_{ljk} \leq x_l, \qquad k \in K,\ \boldsymbol{l} \in V, \tag{3.3e}$$

$$y_{ijk} \geq 0, \qquad k \in K,\ \ (\boldsymbol{i}, \boldsymbol{j}) \in A_k, \tag{3.3f}$$

$$x_i \in \{0, 1\}, \quad \boldsymbol{i} \in V. \tag{3.3g}$$

For future reference, define a feasible set S satisfying all constraints (3.3b)−(3.3g). Also, we will use $\boldsymbol{z} := (\boldsymbol{x}, \boldsymbol{y})$, and $f(\boldsymbol{z}) := w_h f_h(\boldsymbol{x}) + w_{SP} f_{SP}(\boldsymbol{y})$. The feasible set of paths for cable k is denoted $Y_k := \{\boldsymbol{y}_k \mid (3.3b) - (3.3d), (3.3f)\}$, i.e., the set of paths from s_k to e_k for cable $k \in K$. Lastly, we define $Y := \{\boldsymbol{y} \mid \boldsymbol{y}_k \in Y_k,\ k \in K\}$.

By setting $w_h = 0$ and $w_{SP} = 1$, the problem (3.3) can be viewed as finding the shortest paths for $|K|$ routes. When also considering the first objective function, i.e., when $w_h > 0$, the problem is also to consider the amount of space (in terms of number of nodes used) that is used. The difference between the constraints in this model and the SPP model (2.1), is the variables x_i and the added constraints (3.3e) and (3.3g). These constraints ensures that x_i is equal to one if any route uses node $\boldsymbol{i} \in V$. With the constraints (3.3e) we still have the integrality property of the set Y_k when $\boldsymbol{x}$ is binary; therefore the variables $\boldsymbol{y}$ can be considered as continuous. The model (3.3) is a *mixed binary linear program* (MBLP), a specific case of MILP.

An LP-relaxation of the model (3.3) means that the binary constraints (3.3g) are relaxed to $x_i \in [0, 1]$.

3.2.1 Lagrangian relaxation

The procedure of the Lagrangian relaxation of problem (3.3) will be shown here, resulting in a formulation of the Lagrangian subproblem and the Lagrangian dual problem.

By relaxing the constraints (3.3e) we get the following Lagrangian subproblem:

$$\min \quad \Big\{ f(\boldsymbol{x}, \boldsymbol{y}) + \sum_{k \in K} \sum_{i \in V} \lambda_{ik} \Big(\sum_{\boldsymbol{j} \in V : (\boldsymbol{i}, \boldsymbol{j}) \in A} y_{ijk} - x_i \Big) \Big\}, \tag{3.4a}$$

$$\text{s.t.} \quad \boldsymbol{x} \in \{0,1\}^{|V|}, \ \boldsymbol{y} \in Y, \tag{3.4b}$$

with Lagrangian dual variables $\lambda_{ik} \geq 0$, $k \in K$, $\boldsymbol{i} \in V$. The objective (3.4a) is penalized when the constraints (3.3e) are not fulfilled. In other words, we get a penalization whenever a node variable is equal to zero, but the node is used by some route.

Expand the relaxed objective function:

$$\begin{aligned} & f(\boldsymbol{x}, \boldsymbol{y}) + \sum_{k \in K} \sum_{i \in V} \lambda_{ik} \Big(\sum_{\boldsymbol{j} \in V : (\boldsymbol{i}, \boldsymbol{j}) \in A_k} y_{ijk} - x_i \Big) = \\ w_h \sum_{\boldsymbol{i} \in V} n_i x_i + w_{SP} \sum_{k \in K} \sum_{(\boldsymbol{i}, \boldsymbol{j}) \in A_k} a_{ij} y_{ijk} & + \sum_{k \in K} \sum_{\boldsymbol{i} \in V} \lambda_{ik} \Big(\sum_{\boldsymbol{j} \in V : (\boldsymbol{i}, \boldsymbol{j}) \in A_k} y_{ijk} - x_i \Big) = \\ & \sum_{\boldsymbol{i} \in V} \Big(w_h n_i - \sum_{k \in K} \lambda_{ik} \Big) x_i + \sum_{k \in K} \sum_{(\boldsymbol{i}, \boldsymbol{j}) \in A_k} \Big(w_{SP} a_{ij} + \lambda_{ik} \Big) y_{ijk} \end{aligned}$$

For given dual variable values, we can see that (3.4) separates into $|V|+|K|$ optimization problems. The Lagrangian subproblem thus reduces to solving the following problems:

$$\min \quad \sum_{\boldsymbol{i} \in V} \Big(w_h n_i - \sum_{k \in K} \lambda_{ik} \Big) x_i, \tag{3.5a}$$

$$\text{s.t.} \quad x_i \in \{0,1\}, \ \ \boldsymbol{i} \in V, \tag{3.5b}$$

$$\sum_{k \in K} \min \sum_{(\boldsymbol{i}, \boldsymbol{j}) \in A_k} \Big(w_{SP} a_{ij} + \lambda_{ik} \Big) y_{ijk}, \tag{3.6a}$$

$$\text{s.t.} \quad \boldsymbol{y}_k \in Y_k. \tag{3.6b}$$

This means that subproblem (3.6a) is to find $|K|$ separate shortest paths, which can be done efficiently with Dijkstra's algorithm. The problems (3.5a) are solved by setting $x_i = 0$ if $(w_h n_i - \sum_{k \in K} \lambda_{ik}) > 0$ and $x_i = 1$ otherwise. Since the problems (3.5a) and (3.6a) has integrality property the optimal objective value for the Lagrangian dual problem and the LP-relaxation of (3.3) are equal to each other.

The Lagrangian dual problem takes the following form:

$$\max_{\boldsymbol{\lambda} \geq \boldsymbol{0}} \quad \text{(3.5a)} \quad \text{and} \quad \text{(3.6a)}. \tag{3.7}$$

An equivalent formulation of the problem (3.7) is

$$\max_{\boldsymbol{\lambda}} \quad \sum_{k \in K} \min_{\boldsymbol{y}_k \in Y_k} \quad \sum_{\boldsymbol{i},\boldsymbol{j} \in A_k} \Big(w_{SP} a_{ij} + \lambda_{ik} \Big) y_{ijk}, \tag{3.8a}$$

$$\text{s.t.} \qquad \sum_{k \in K} \lambda_{ik} = w_h n_i, \quad \forall \boldsymbol{i} \in V, \tag{3.8b}$$

$$\boldsymbol{\lambda} \geq \mathbf{0}. \tag{3.8c}$$

An idea on how to solve problem (3.8) is to use subgradient optimization and project onto the constraints (3.8b)-(3.8c), and not only the non-negative orthant as for problem formulation (3.7). However, only the Lagrangian dual problem formulation (3.7) is consider in this thesis and the alternative formulation (3.8) is left for future work.

3.2.2 ε-constraint method model

When the ε-constraint method is used, the model (3.3) is reformulated. The objective weights are $w_h = w_{SP} = 1$, and the objective function f_1 is converted into a constraint. With $\varepsilon > 0$, the reformulation looks like following:

$$\min \quad f_{SP}(\boldsymbol{y}), \tag{3.9a}$$

$$\text{s.t.} \quad f_h(\boldsymbol{x}) \leq \varepsilon, \tag{3.9b}$$

$$\boldsymbol{x} \in \{0,1\}^{|V|}, \tag{3.9c}$$

$$\boldsymbol{y} \in Y. \tag{3.9d}$$

The problem (3.9) will be used to compute all Pareto and weakly Pareto optimal solutions with algorithm 6. The maximal value of ε is obtained by computing the shortest paths for all cables and evaluating the value of $f_h(\boldsymbol{x})$. ε is then decreased with the lowest node cost and problem (3.9) is solved in order to find a new solution, and this is repeated until the problem is infeasible.

4
Algorithm description

This section presents the algorithms that have been used and developed in this thesis. Connections to the mathematical tools in Section 2 are shown. The main algorithm that has been used for solving the CHRP problem (3.3) is presented in Section 4.1. The heuristics that have been developed and is a part of the main algorithm, are described in Section 4.2. Lastly, the algorithms to compute different Pareto optimal solutions are presented in Section 4.3.

4.1 Algorithm overview

The main algorithm is a deflected subgradient optimization procedure with one-sided branching in the sense of a diving heuristic; see [23, p. 17]. Heuristics are utilized to improve upper bounds by searching for feasible solutions with good quality. Ergodic sequences are used to determine which variables to fix. Similar approaches has been used before, for example in [11] and [12, p. 533].

In Figure 4.1 and in Algorithm 2, we can see an overview of the main algorithm, used for solving the CHRP (3.3). The heuristics that are a part of Algorithm (2), are presented in Section 4.2.

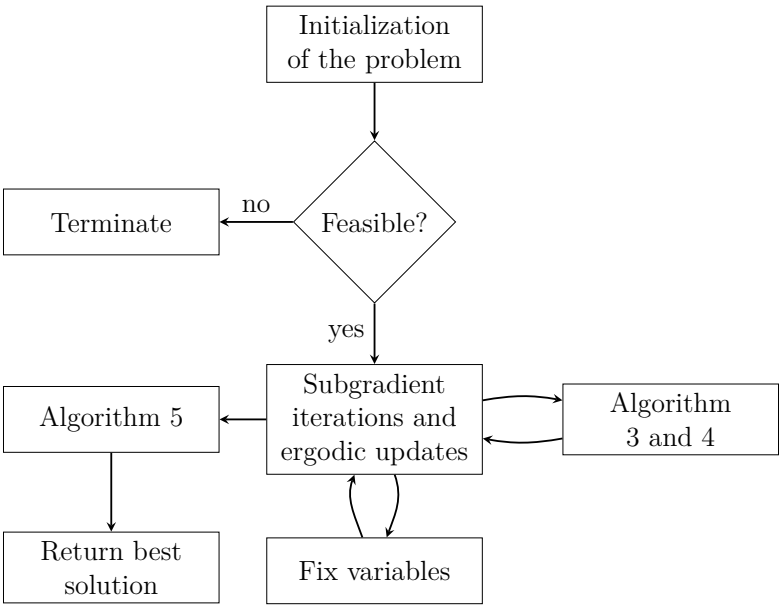


Figure 4.1: Flowchart of Algorithm 2.

Algorithm 2: Main algorithm

1. $t := 0$, $\boldsymbol{\lambda}^0 \in \mathbb{R}_+^{|K||V|}$, $(t_{\max}, t_{\text{fix}}, t_{\text{heur}}, b, p) \in \mathbb{N}$, $\delta_0 \in]0, 2]$, $l = 0$, $u = \infty$, $\epsilon \geq 0$, $\delta_{min} > 0$.
2. Find $\boldsymbol{\lambda}^{t+1}$, $\boldsymbol{z}(\boldsymbol{\lambda}^t)$ and $\boldsymbol{z}_{\text{feas}}(\boldsymbol{\lambda}^t)$ according to steps $2-7$ in algorithm (1), and update lower bound l and upper bound u.
3. Compute the ergodic sequence $\tilde{\boldsymbol{x}}^t$ (of the node-variables) according to (2.9)
4. If $t \geq t_{\text{fix}}$, fix b x-variables to zero. Choose the variables corresponding to the b smallest values of $\tilde{\boldsymbol{x}}^t$.
5. If t_{heur} is a divisor with t, apply Algorithm 3 and 4, to search for a better upper bound.
6. Check termination criteria, i.e., if $t \geq t_{\max}$, $\delta_t < \delta_{\min}$, or $100 \cdot \frac{u-l}{u} < \epsilon$. If any criterion is fulfilled, go to next step, otherwise set $t := t + 1$ and go to 2.
7. Apply Algorithm (5).

In the initialization of the problem, seen in Figure 4.1, feasibility are checked by making sure that none of the start and end nodes for the cables are in collision, and that there exists a feasible shortest path for every cable.

When solving the Lagrangian subproblems in step 2 of Algorithm 2, Dijkstra's algorithm is used for solving (3.6a). The problem (3.5a) is simply solved by setting the binary variables to 0 when the cost is positive and 1 when the cost is negative or zero. Dijkstra's algorithm and implementation of it can be read about in [15].

For the instances in this thesis, we have seen that that there is a fast improvement in early iterations. The value of t_{fix} has been chosen such that we start to fix variables when the rate of improvement has started to decrease.

The complication, mentioned in Section 2.4, about later iterations of the subproblem solutions being less weighted with the ergodic updates according to (2.9), is considered. The method with the s^k-rule has not been implemented and tested, but another simple idea has been tested: every i_{erg}:th iteration, start a "new" ergodic sequence vector $\tilde{\boldsymbol{x}}^t_{\text{new}} = \xi\tilde{\boldsymbol{x}}^t$, $\xi \in]0, 1[$. The vector $\tilde{\boldsymbol{x}}^t_{\text{new}}$ is then updated according to (2.9). The aim of this approach is not to obtain a primal feasible convergence directly with $\tilde{\boldsymbol{x}}^t_{\text{new}}$, but instead to decrease the weighting of earlier iterates and only use this vector as information to choose which variables to fix in the "branching" process in step 4 of Algorithm 2 and in Algorithm 4.

4.2 Heuristics

The heuristics are used to search for better upper bounds on the objective value. Algorithms 3 and 4 are used in step 5 in Algorithm 2. The last heuristic in section 4.2.2 is used once, in the last step of Algorithm 2.

4.2.1 Local search heuristic

The heuristic presented in Algorithm 3, is a simple but effective and fast heuristic. The idea is to reroute one cable at a time where it is cheap to use arcs that are used by other cables. An example of how the heuristic works is illustrated in Figure 4.2.

Algorithm 3: Local search heuristic

1. Given a feasible solution $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{y})$ to (3.3), define the best objective value $f_{\text{best}} := f(\boldsymbol{z})$.
2. Set $k := 1$, corresponding to cable 1.
3. If an arc is used by the cables $\tilde{k} \in K \setminus k$ in the solution $\boldsymbol{z}$, set the corresponding arc cost equal to a_{ij}, otherwise set the corresponding arc cost equal to $a_{ij} + n_i$.
4. Find the shortest path for cable k and denote the solution $\bar{\boldsymbol{z}}$. If $f(\bar{\boldsymbol{z}}) < f_{\text{best}}$, save this as a new solution $\boldsymbol{z}_{\text{new}} := \bar{\boldsymbol{z}}$ and set $f_{\text{best}} := f(\bar{\boldsymbol{z}})$.
5. Set $k := k + 1$ and go to 3, until $k = |K|$.
6. Set $\boldsymbol{z} := \boldsymbol{z}_{\text{new}}$ and go to 2 until the objective value has not been improved.

4.2.2 Core problem heuristics

The heuristics presented in this section resembles construction of so called core problems, as in [12, p. 531]. In the core problem, some of the variables in the original problem are assigned fixed values, which results in a problem with fewer variables, and hopefully yields a feasible and near optimal solution to the original problem.

The idea of Algorithm 4 is to fix variables (the binary variables), such that we get a simplified LP problem. With good choices of variables to bound we hopefully obtain a feasible solution with good quality. At each "branching" in Algorithm 4, variables are fixed to the value zero. An ergodic sequence is used to determine which variables to fix. It is called "*Cheap* core problem heuristic" because it is computationally cheap to execute compared to Algorithm 5.

Algorithm 4: Cheap core problem heuristic

Data: Ergodic sequence $\tilde{\boldsymbol{x}}^t$, feasible solution $\boldsymbol{z}$ to (3.3)
Result: Heuristic solution
Initialize set of bounded node variables $V_0 := \emptyset$;
Set positive number $\tilde{\sigma} < 1$;
Set a number $\sigma \in]0, 1[$;
$l = 0$;
while *LP-relaxation of* (3.3) *with nodes* $V \setminus V_0$ *is feasible* ***and*** $l < f(\boldsymbol{z})$ **do**
  if $\tilde{x}_i^t < \sigma$, $i = 1, \ldots, |V|$ **then**
    $V_0 := V_0 \cup \boldsymbol{i}$; `/*Fixing variable to zero*/`
  end
  Solve a LP-relaxation of (3.3) using only nodes in $V \setminus V_0$ and obtain solution $\boldsymbol{z}_{\text{LP}}$;
  $l := f(\boldsymbol{z}_{\text{LP}})$;
  Find feasible solution $\boldsymbol{z}_{\text{feas}}(\boldsymbol{z}_{\text{LP}})$ using some heuristic;
  if $f(\boldsymbol{z}_{feas}) < f(\boldsymbol{z})$ **then**
    $\boldsymbol{z} := \boldsymbol{z}_{\text{feas}}$;
  end
  $\sigma := \sigma + \tilde{\sigma}$;
end

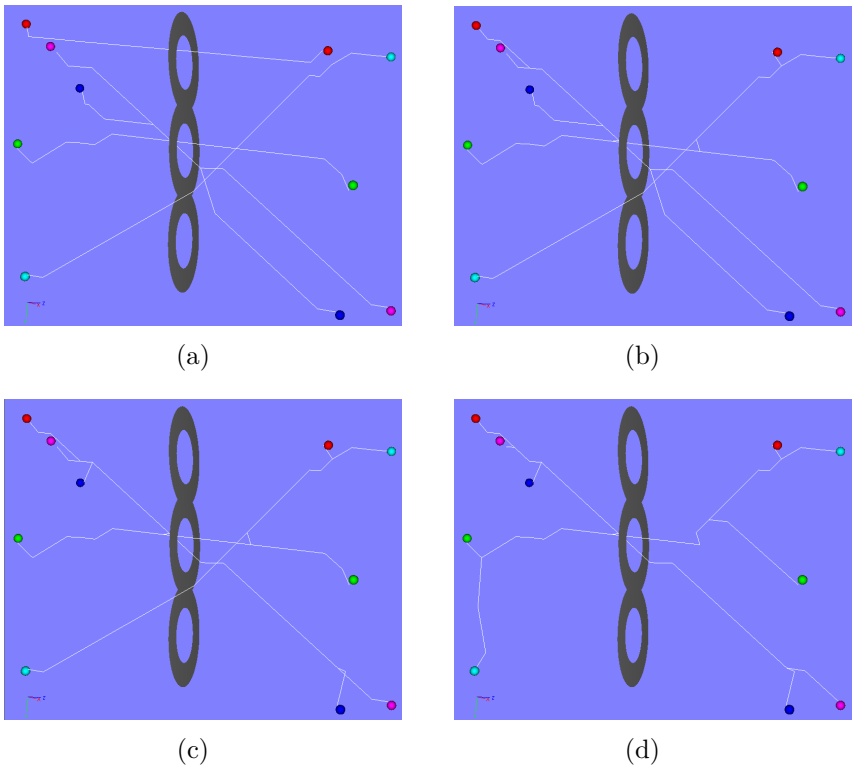


Figure 4.2: Example of applying Algorithm 3. Each node pair to connect has a distinct color. (a): Solution after some subgradient steps. (b): First iteration of Algorithm 3: the route connecting the red node pair has been rerouted. (c): Second iteration of Algorithm 3: the route connecting the blue node pair has been rerouted. (d): Fifth and final iteration: the routes connecting the green, turquoise and pink node pair have been rerouted.

The LP-relaxed problems are solved with Gurobi, and as will be seen in section 6, it will take too long computing time to solve an LP-relaxation of (3.3) for large instances. Therefore, the initialization of σ in Algorithm 4, should be chosen such that solving the LP-relaxation problem, using only nodes in $V \setminus V_0$, is computationally fast. Finding a feasible solution can be done, for example, by a rounding heuristic. The rounding of variables should then be done such that we have routes going from every start/end node pair. Suppose we have a solution $\boldsymbol{z}_{\text{LP}} = (\boldsymbol{x}_{\text{LP}}, \boldsymbol{y}_{\text{LP}})$ from solving a LP-relaxed problem, then finding a feasible solution can be done in the following steps:

- Round up variables that are not zero in $\boldsymbol{y}_{\text{LP}}$, such that there are routes going from s_k to e_k, $k \in K$. Denote the resulting solution as $\boldsymbol{y}_{\text{feas}}$.
- If a node is used in $\boldsymbol{y}_{\text{feas}}$, set the corresponding node variable to 1, otherwise set to 0.

The next heuristic finds a local optimum by using information throughout the execution of Algorithm 1. The idea is that arcs that appear often in the Lagrangian subproblems are more likely to be a part of the optimal solution. Let us define a variable that represents the frequency of appearance of arcs from $\hat{s}$ Lagrangian

subproblem solutions:

$$\boldsymbol{y}_{\text{freq}} := \sum_{s=1}^{\hat{s}} \boldsymbol{y}^s(\boldsymbol{\lambda}^s). \tag{4.1}$$

Algorithm 5: Core problem heuristic

Data: $\boldsymbol{y}_{\text{freq}}$
Result: Local optimal solution $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}})$ to (3.3)
Set number of nodes $\tilde{v} < |V|$;
Define set of nodes to be part of problem $\tilde{V} := \emptyset$;
Add nodes, used in the best solution found, to the set $\tilde{V}$;
while $|\tilde{V}| \leq \tilde{v}$ **do**
  $y_{ijk} := \text{argmax}_{ijk}\{\boldsymbol{y}_{\text{freq},ijk}\}$;
  $\tilde{V} := \tilde{V} \cup \{\boldsymbol{i}, \boldsymbol{j}\}$;
end
Solve (3.3) using nodes $\tilde{V}$ resulting in solution $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}})$;

The reduced dimension problem with node set $\tilde{V}$ in Algorithm 5, is solved with Gurobi. The problem (3.3) might be complicated to solve not only because there are many binary node-variables. If there are many cables, we get a large number of arc-variables which can result in a big time complexity. The number of these arc variables is reduced when the number of node variables is reduced. To decrease the dimension even more, one can choose to use at most n_{arc} arcs for every cable. A good choice could then be to use the variables corresponding to the n_{arc} biggest values in $\boldsymbol{y}_{\text{freq}}$, for every cable.

It would be possible to use an ergodic sequence for Algorithm 5 instead of (4.1). The subproblem solutions are equally weighted in $\boldsymbol{y}_{\text{freq}}$, and does therefore not suffer from lower weights on later iterates as the ergodic updates according to (2.9) does. Although, it would be interesting to use an ergodic sequence also for comparison.

4.3 Computing points on the Pareto front

The Pareto front is investigated in two different ways. I present an approach in Algorithm 6, which utilizes the ε-constraint method for an MBLP and finds all Pareto optimal, and weakly Pareto optimal solutions. The second algorithm, for computing Pareto optimal solutions, optimizes with different values of the objective weights. If the problem is computationally heavy, Algorithm 7 has an advantage since we can choose how many Pareto optimal solutions to compute; it is the only one that can be applied when Algorithm 2 is used as solver.

Algorithm 6: ε-constraint method with binary variables

Result: Set P containing all efficient and weakly efficient solutions for problem (3.3), with $w_h = w_{SP} = 1$.

$P := \emptyset$ $\boldsymbol{y}^* := \text{argmin}_{\boldsymbol{y} \in Y}\ f_{SP}(\boldsymbol{y})$;

Construct a feasible solution $(\boldsymbol{x}^*, \boldsymbol{y}^*) \in S$, such that $f_h(\boldsymbol{x})$ is minimized given $\boldsymbol{y}^*$;

$P := P \cup (\boldsymbol{x}^*, \boldsymbol{y}^*)$;

$n_{\min} := \min_i \{\boldsymbol{n}_i\}$;

$\varepsilon := f(\boldsymbol{x}^*, \boldsymbol{y}^*) - n_{\min}$;

while $\min_{(\boldsymbol{x},\boldsymbol{y}) \in S} \{f_{SP}(\boldsymbol{y}) \mid f_h(\boldsymbol{x}) \leq \varepsilon\}$ *is feasible* **do**

  $(\boldsymbol{x}, \boldsymbol{y}^*) := \text{argmin}_{(\boldsymbol{x},\boldsymbol{y}) \in S} \{f_{SP}(\boldsymbol{y}) \mid f_h(\boldsymbol{x}) \leq \varepsilon\}$;

  Construct a feasible solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$, such that $f_h(\boldsymbol{x})$ is minimized given $\boldsymbol{y}^*$;

  $P := P \cup (\boldsymbol{x}^*, \boldsymbol{y}^*)$;

  $\varepsilon := f_h(\boldsymbol{x}^*) - n_{\min}$

end

Given a routing solution $\boldsymbol{y}^*$, we obtain a feasible solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ which minimizes f_h by setting all node variables equal to zero, except for those who are used in the routes corresponding to $\boldsymbol{y}^*$.

Algorithm 7: Weighted sum

1. Choose some set of weights $s_w = \{(w_h^1, w_{SP}^1), (w_h^2, w_{SP}^2), \ldots, (w_h^m, w_{SP}^m)\}$, $w_h^i \geq 0$, $w_{SP}^i \geq 0$, $i = 1, 2, \ldots, m$.
2. Solve problem (3.3) using weights in s_w, with Gurobi or Algorithm 2, and obtain corresponding solutions $\boldsymbol{z}_1^*, \boldsymbol{z}_2^*, \ldots, \boldsymbol{z}_m^*$.

There are several ways to choose different weights to compute (weak) Pareto optimal solutions. One approach is to have that $w_h, w_{SP} \in [0, 1]$, $w_h + w_{SP} = 1$, and solve (3.3) for a uniformly distribution of the weights, e.g., $w_h = 0.1, 0.2, \ldots, 0.9$. This has shown to be a good strategy in this thesis, which can be seen in Section 6.1.3.

5
Implementation

5.1 Software

Our algorithms have been implemented in C++, mainly because the software IPS has an interface to C++. Industrial Path Solutions, or IPS, is a software tool developed by FCC. The instances are created in IPS.

Gurobi has been used to solve the MILP models (3.3) and (3.9), for small instances. Gurobi is a commercial optimization solver that handles, for example, MILP problems [24]. The Gurobi solver is also used in our C++ implementation.

5.2 Parameter settings

A not to extensive parameter settings investigation has been made for Algorithm 2. Motivation of some parameter settings are made in Section 6, and some settings have been directly taken from the literature. If nothing else is mentioned, the tests were made with the following settings for Algorithm 2:

- $w_h + w_{SP} = 1$, and generally $w_h = w_{SP} = 0.5$,
- $\lambda_{ik}^0 = n_{\min}, \ \boldsymbol{i} \in V, \ k \in K$,
- $\delta_0 = 2$,
- $\delta_{\min} = 10^{-5}$,
- $\epsilon = 3$,
- $p = 50$,
- $\kappa = 0.995$,
- $\beta = 0.75$, seen in (2.8),
- $\eta = 1.5$, seen in (2.8),
- $t_{\max} = 5000$,
- $t_{\text{heur}} = 800$,
- $t_{\text{fix}} = 1000$,
- $b = \lceil |V|/(3t_{\max}) \rceil$,
- $\tilde{v} = 8^3$, seen in Algorithm 5.

Also, if nothing else is mentioned, the node costs are set according to Figure 3.4(a), with $n_{\min}$ according to (3.2).

6
Tests and results

In this section different instances, or test cases, are presented. In Section 6.2 the results from solving an industrial test case is shown, where the essential point is to investigate if a satisfactory harness routing is obtained, but also to observe the computation time. We start with two simple test cases in Section 6.1 to investigate properties of the algorithm and the model, such as time complexity, how close we get to optimal solutions, and the appearance of the Pareto front.

When the gap is measured in percentage, it is calculated as in Algorithm 2, i.e., $100 \cdot \frac{\text{(upper bound)}-\text{(lower bound)}}{\text{lower bound}}$.

The tests were run using a i7-6700K processor at 4.00GHz with 4 cores.

6.1 Test cases

In Figure 6.1, we can see the two test cases used in this section. The case in Figure 6.1(a) is referred to as test case 1, and the case in 6.1(c) is referred to as test case 2. They have different numbers of cables, the start and end nodes are located differently, and they are not used to represent any realistic routing scenario. These test cases have feasible solutions for relative small dimensions, or grid sizes, compared to the industrial case studied in Section 6.2. This means that these instances can be solved to optimality with Gurobi, which is good for investigations.

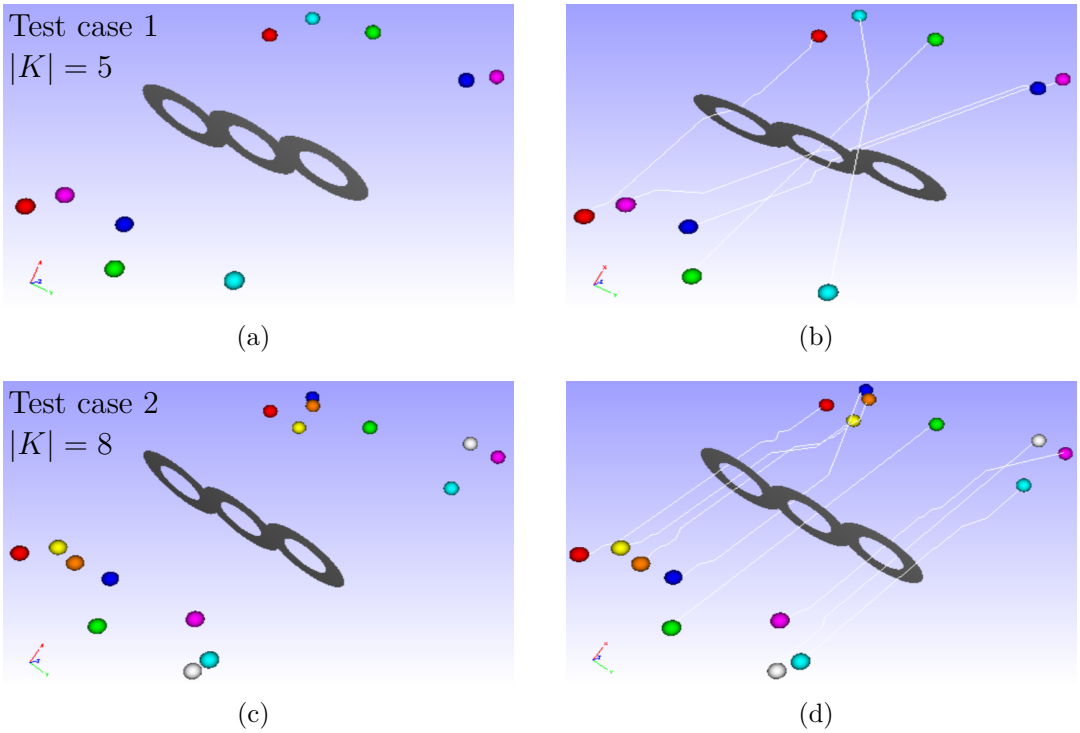
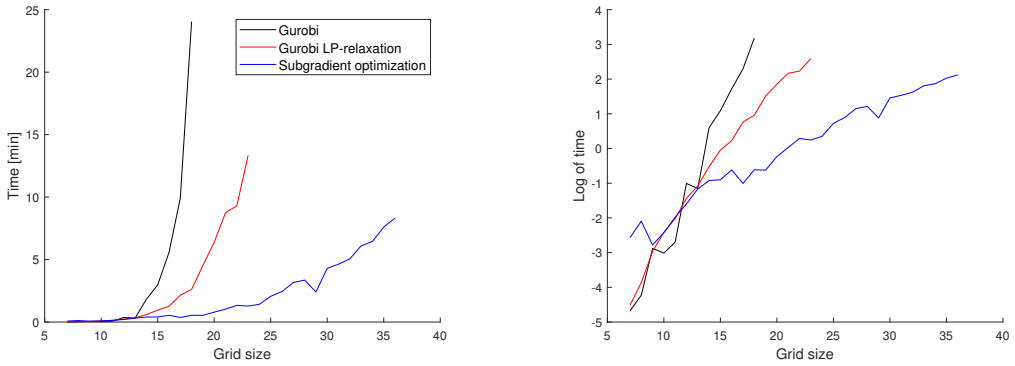


Figure 6.1: Two different test cases in (a) and (c), and their corresponding shortest path solutions (i.e., for $w_h = 0$) in (b) and (d), respectively. There is an obstacle in these instances in the form of three circles. The start/end node pairs to be connected have distinct colors.

6.1.1 Time complexity

Three factors that significantly affect the time complexity is the grid size (number of nodes), the number of cables, and the values of the objective weights. We can see the computation time plotted against increasing grid size in Figure 6.2, where the problem (3.3) is solved with Gurobi and Algorithm 2, and the LP-relaxed version with Gurobi. Solving the problem (3.3) with Gurobi seems to have an exponential trend, and solving it with Algorithm 2 looks more like a polynomial trend. The same thing can be said about the time complexity for an increasing number of cables; see Figure 6.3.

The more objective f_h is weighted, or preferred, the longer it takes to solve problem (3.3), and this applies for both solving the problem with Gurobi and with Algorithm 2. In figure 6.4, we can see the execution time for Algorithm 2 for test case 1 and 2, with increasing value of w_h.



(a) (b)

Figure 6.2: Test case 2. The plots show the time for solving (3.3), and its LP-relaxed version, with Gurobi, compared with the computation time when using Algorithm 2, for increasing grid size. The horizontal axis shows the number of grid points in one direction (the actual grid size is the cube of that value).

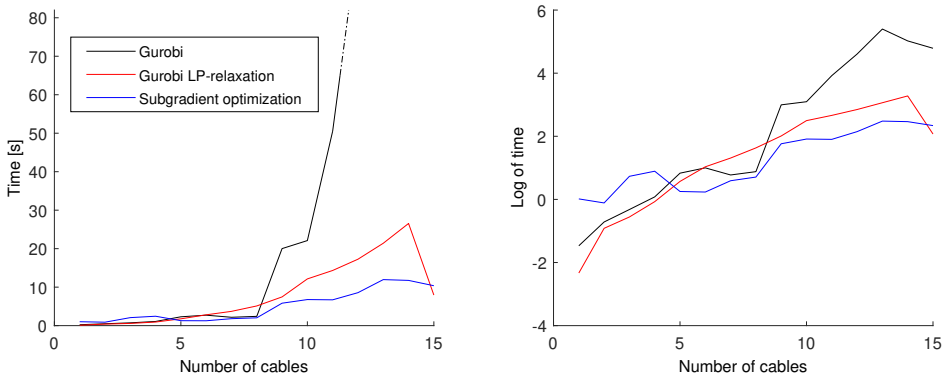


Figure 6.3: Modified version of test case 2 with more cables, $|V| = 10^3$. The figure shows the computation time and the log of the time for solving problem (3.3), plotted against the number of cables.

6.1.2 Convergence

In this section we investigate the convergence of the subgradient algorithm, the resulting gap for different test cases with Algorithm 2, and how close we get to optimal solutions with Algorithm 2.

Figure 6.5(a) shows the relative gap when applying the subgradient method (Algorithm 1) for test case 1 and 2. Figure 6.5(b) shows the time versus the number of iterations. This plot motivates the choice of having the maximum number of iterations set to 5000, since at that point the subgradient method improves in a slower rate, and it is not desirable that the algorithm runs more than around $200 - 300$ seconds. We can also observe in Figure 6.5 that the gap is bigger, and that the solution takes a bit longer time for larger values of w_h.

In Table 6.1, we can see the values of the resulting gap when applying Algorithm 2 on test case 1 and 2. The resulting gap depends on the instance and settings, e.g.,

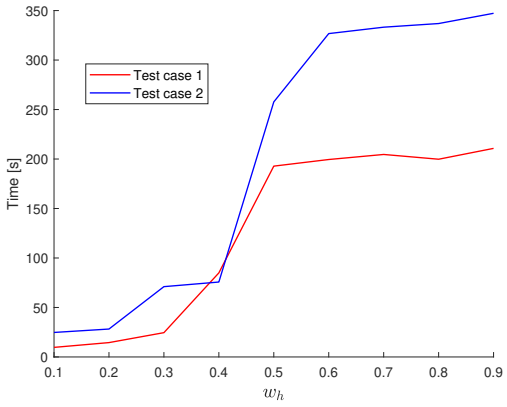
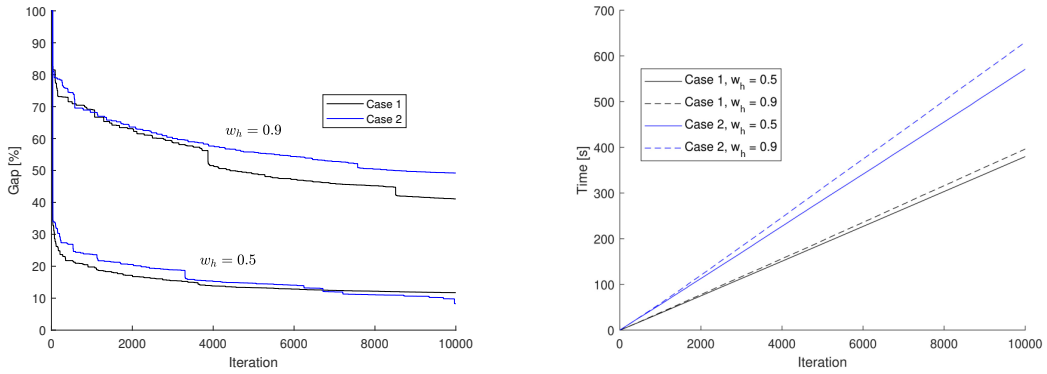


Figure 6.4: Algorithm 2 applied on test case 1 and 2 for $w_h = 0.1, 0.2, \ldots, 0.9$, $w_{SP} = 1 - w_h$. $|V| = 30^3$.



(a) (b)

Figure 6.5: The plot shows four tests; two different values of w_h for solving test case 1 and 2 with the subgradient method, i.e., Algorithm 1. (a) shows the relative gap, plotted against the number of iterations. (b) shows the computing time plotted against the number of iterations. $|V| = 30^3$, $\kappa = 0.999$, $t_{\max} = 10000$.

the value of the objective weights. By comparing Figure 6.5(a) and Table 6.1, we can see the impact of adding heuristics to the subgradient algorithm, for example; when we are only performing subgradient iterations, test case 1 and 2 with $w_h = 0.9$, reaches $50 - 60\%$ after 5000 iterations, but with Algorithm 2, we reach 21.7% and 14.5% for test case 1 and 2, respectively.

By observing the gap, it is not possible to see how close the upper bound is to the optimal objective value. A comparison of the resulting upper and lower bounds, the optimal value, and the optimal LP-relaxed value, can be seen in Figure 6.6. Here we can see that Algorithm 2 actually reaches, or is very close, to the optimal objective value, for different grid sizes. For the observed test cases, the main contributing factor to the size of the gap is the lower bound, which can in theory reach the LP-relaxed optimal objective value. In some cases there is a small integrality gap, as in Figure 6.6, and in such cases it is possible to reach small gaps, which occurs in Table 6.1.

w_h :	0.2	0.5	0.9
Gap [%] for test case 1	0.065	6.15	20.8
Gap [%] for test case 2	$3.5 \cdot 10^{-5}$	0.56	24.1

Table 6.1: The table shows the resulting relative gap, for different tests. $|V| = 30^3$, $w_{SP} = 1 - w_h$, $\epsilon = 0$.

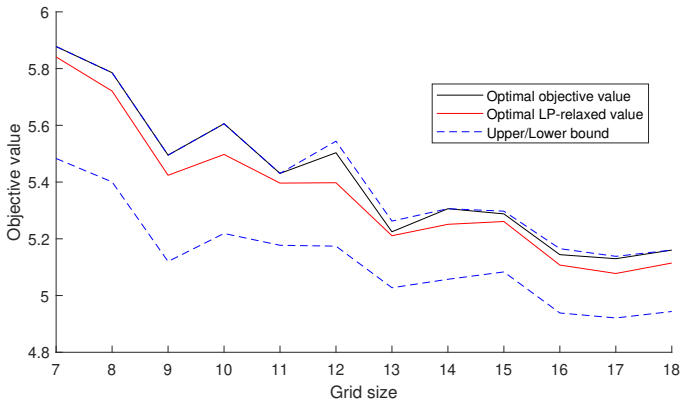


Figure 6.6: Test case 1. The optimal objective value, the optimal LP-relaxed objective value and the upper/lower bound from Algorithm 2, plotted for increasing grid size. The upper bound coincides with the optimal objective value for several grid sizes.

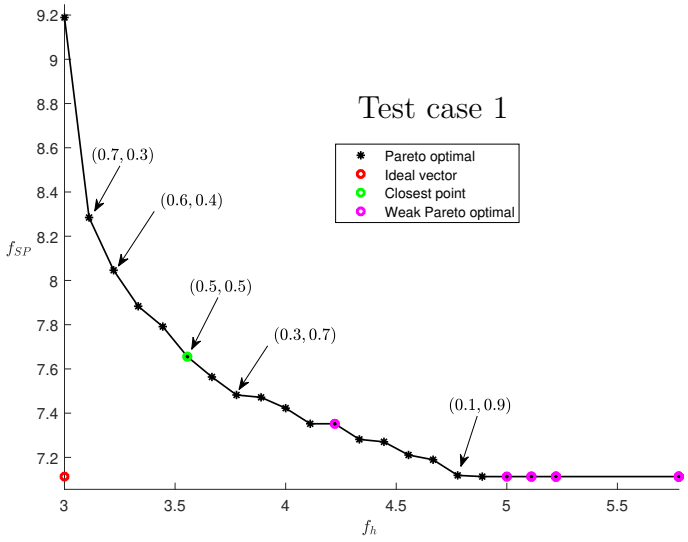
6.1.3 Pareto front

In Figure 6.7, the Pareto front was computed for test case 1 and 2 with Algorithm 6, using Gurobi. Problem (3.3) was also solved with Gurobi, with weights $w_h = 0.1, 0.2, \ldots, 0.9$, $w_{SP} = 1 - w_h$, and the corresponding resulting Pareto optimal solutions with these weights are also shown in the figure. Sometimes we end up in the same solution with different weights, the plot shows the lowest value of w_h, if several weights correspond to the same point.

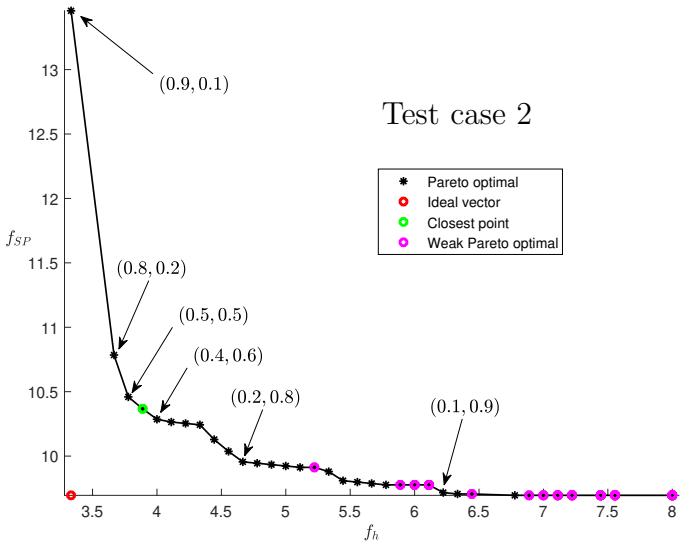
The ideal objective point corresponds to a non-existent feasible solution, where both objectives has their respective minimum values. It might be interesting to see which solutions that are close to the ideal objective point in Euclidean distance, to see if these solutions are "better", or more preferred from a subjective point of view.

Figures 6.8 and 6.9 show some solutions corresponding to the Pareto optimal solutions seen in Figure 6.7. These test cases do not represent any "real" example, and there are no concrete preference of how the harness routing should look like, but it can still be interesting to see how the Pareto optimal solutions differ in these cases.

We can also see that we get points on non-convex parts on the Pareto front, and these points can not be captured with the weighted sum method.



(a)



(b)

Figure 6.7: Pareto front computed with Algorithm 6, for test cases 1 and 2. The plots show all efficient and weakly efficient solutions. The green point is the one closest to the ideal point. The arrows show which point we end up in with weights (w_h, w_{SP}).

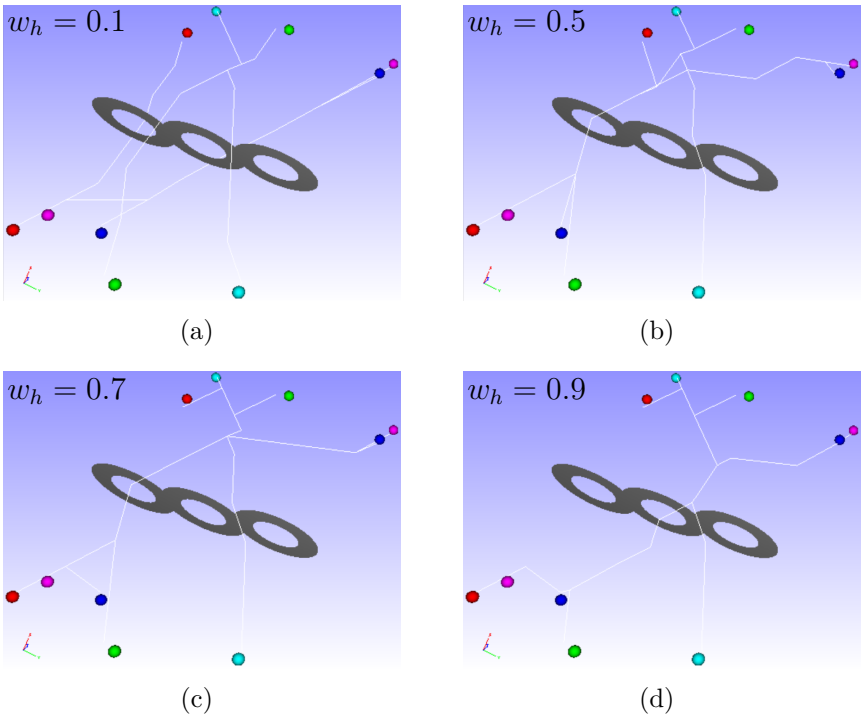


Figure 6.8: Pareto optimal solutions, corresponding to points on the Pareto front in figure 6.7(a). (d) corresponds to the Pareto point furthest to the left.

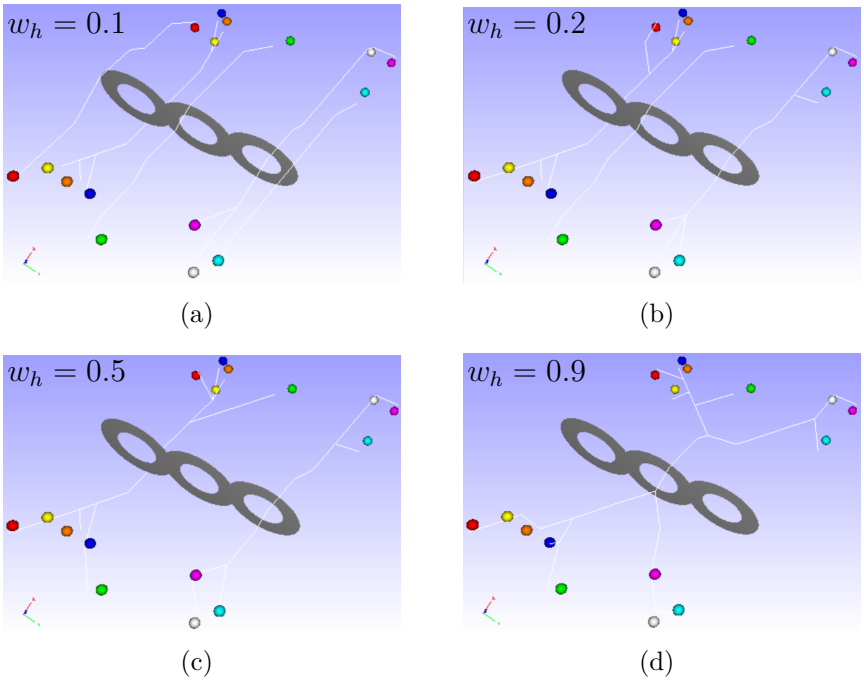


Figure 6.9: Pareto optimal solutions, corresponding to points on the Pareto front in Figure 6.7(b).

6.2 Solving an industrial test case

The industrial test case in this section has been manually routed before. We can see how the manually routed solution looks like in Figure 6.10.

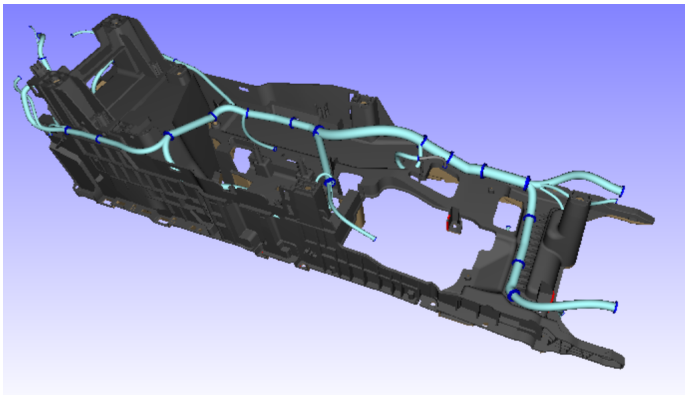
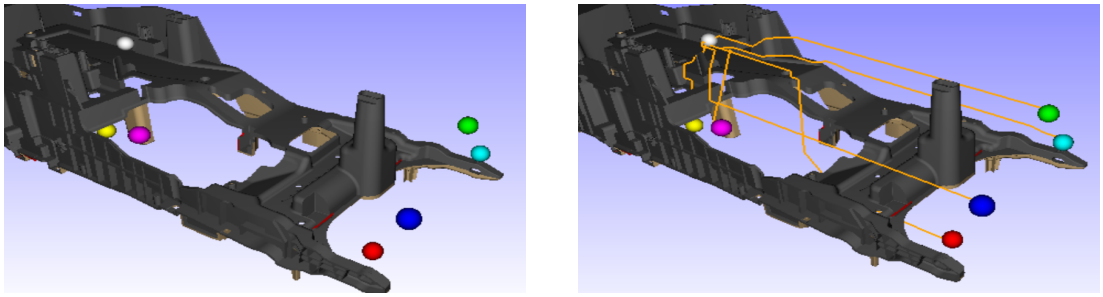


Figure 6.10: A manually routed industrial scene.

First, half of the scene is considered, where six start nodes has been chosen and joins the same end node, according to Figure 6.11.



(a) (b)

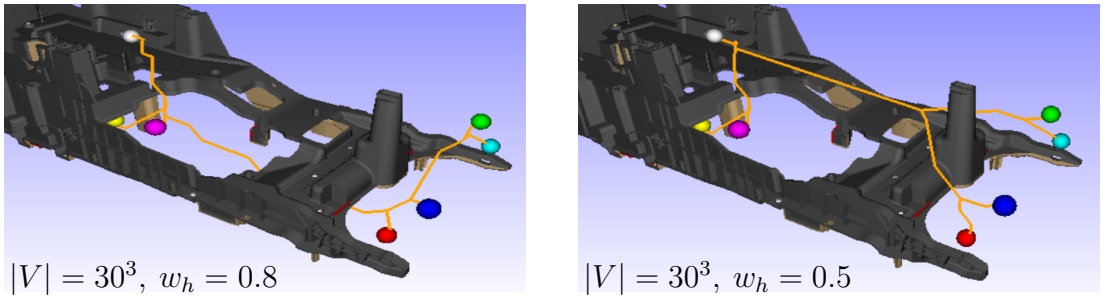
Figure 6.11: The first part of the industrial case to be routed. Six start nodes are chosen, illustrated with distinct colors, with a common white end node. (b) shows the shortest paths for the cables. Here, $|V| = 40^3$.

6.2.1 Choosing weights for the objective functions

Figure 6.12 shows harness routings for two different objective weight settings. When comparing with the manually routing in Figure 6.10, the resulting solution with weights $w_h = w_{SP} = 0.5$ in Figure 6.12(b) is more similar. It looks like f_h is weighted too much in Figure 6.12(a).

The solutions also depend on the grid size, i.e., the number of nodes. We do the following observations: If $|V|$ is too small, the problem is infeasible, and for larger grid sizes, it is possible to find paths through spaces with lower clearances. The latter is exemplified when we compare Figure 6.12(b) and Figure 6.13(a). When the number of nodes is increased from 30^3 to 40^3, the significant change is that the

cable corresponding to the yellow start node chooses a shorter path, going through a tighter space.



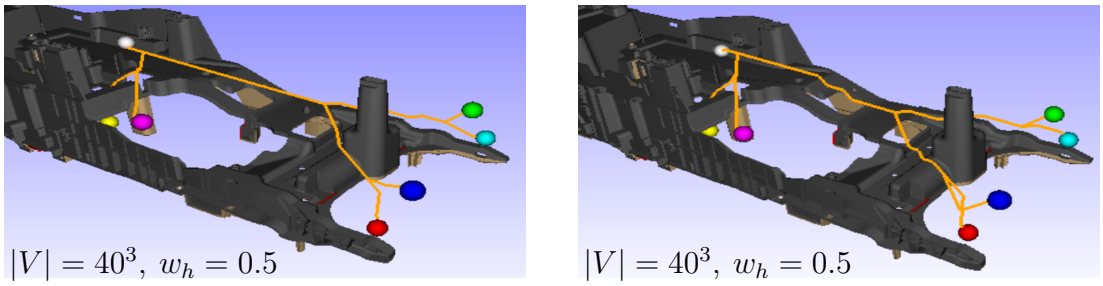
(a) (b)

Figure 6.12: Harness routing solutions for different objective weights.

6.2.2 Choosing node costs

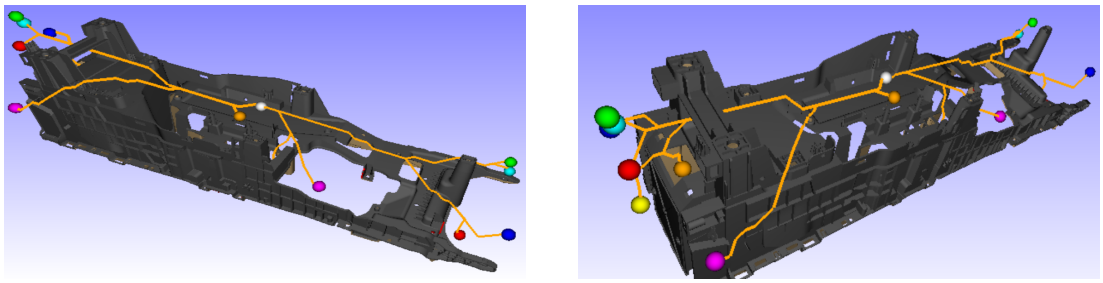
Figure 6.13 shows an example on how the node costs affects the harness routing. The node costs depends on the clearances. In Figure 6.13(a), the node costs are defined as in Figure 3.4(a), i.e., nodes very close to obstacles are equal to infinity and the rest are set to $n_{\min}$. Figure 6.13(b) has node costs according to Figure 3.4(b), where nodes close to obstacles (and satisfies minimum clearance) are cheap, and nodes with larger clearances increase in cost. As expected, we can observe that the routing follows paths closer to the obstacles in the latter case.

The harness routing in Figure 6.13(b) is the most similar routing as compared to the manually routed harness. It took 158.5 seconds and 1144 subgradient iterations for the test in Figure 6.13(b), and resulted in a gap of 1.52%. The rest of the instance is routed in Figure 6.14, with the same settings as in Figure 6.13(b).



(a) (b)

Figure 6.13: Results using different node costs.



(a) (b)

Figure 6.14: Continued routing from the scene in figure 6.13(b), for $|V| = 40^3$, $w_h = 0.5$.

7
Conclusion

A model of the CHRP has been formulated and a solution methodology has been proposed and implemented. The methodology consists of a Lagrangian relaxation of the CHRP model (3.3), and a subgradient optimization procedure for solving the Lagrangian dual problem and iterating towards a solution of the original problem. Ergodic primal sequences are utilized in this iterative process for fixing binary node variables. Two conflicting objectives have been composed into a weighted sum in the model. The objectives have been to minimize the length of the routes for each cable, and to minimize the usage of space. The harness routings are collision free, satisfies minimum clearances, and preferable zones to route in can be adjusted through settings of the node costs. The Pareto front has been computed for smaller instances with the ε-constraint method using Gurobi as a solver where the shape of the Pareto front has been investigated and some representative solutions from this set have been visualized. We can see that the Pareto front has non-convex parts, and solutions in these parts can not be found with the weighted sum method.

The results looks promising if the goal is to find a desired harness routing with respect to the mentioned factors and the approach in this thesis for solving the CHRP should be further investigated. The resulting solution, or harness layout, for the industrial test case looks good. However, to examine how satisfactory it is, it is necessary to have an examination of the routing from someone who has better insight of what is good or bad with the solution.x This result does not guarantee that the routing fulfills all necessary requirements. The procedure in this thesis is a first step towards automating the harness routing. Then modifications and simulations will continue such that bend radii meet the requirements, and the effect of deformation due to gravity is implemented, etc.

The resulting harness layout is not the only thing that is important with this approach, the computation time must also be acceptable. Our results show that it took 158 seconds for the industrial test case in Figure 6.13(b) with six cables and $|V| = 40^3$, and it took around eright minutes in Figure 6.2(b) with eight cables and $|V| = 35$ for test case 2. The algorithm should be able to handle these grid sizes and number of cables, but as it is for now, the computation time may be too long to be acceptable. There are several actions that can be made in regard to the time complexity:

- the code itself needs to be more thoroughly optimized;
- the grid size is unnecessarily large in some of the directions, and has an unnecessary high resolution in some zones, so the number of variables could be reduced significantly;
- further parameter tuning and algorithm improvements might be done such

that the convergence is faster and not so many iterations be needed;
- investigate alternative large-scale optimization methods that might be more efficient.

7.1 Future work

It has already been mentioned above what can be further investigated regarding the reduction of the time complexity. One thing that was mentioned was to test other large-scale methods and see how they perform compared to the approach in this thesis. Alternative methods could for example be decomposition techniques, such as Benders decomposition or column generation. Regarding faster convergence to an optimal solution, the s^t-rule mentioned in Section 2.4 could be implemented. We could also use the fact that we can get several optimal solutions to the Lagrangian subproblem (3.6a), which can be chosen between when computing the subgradients. One approach to find an ascent direction from the given subgradients is to follow the procedure described in [25, p. 288], where an ascent direction is found by solving an LP problem.

The implemented algorithm should be adjusted such that it can handle rectangular grids. This is expected to decrease the computation time since we would reduce the number of variables. It might also be of interest to have a dynamic grid where the grid points are more dense in zones where there are more obstacles.

It needs to be further investigated how satisfactory the resulting harness layouts are. This requires expertise knowledge about harness routings. Other questions that should be further investigated concern how to use the resulting solutions to satisfy more requirements on the routings, and what needs to be done to get a completely finished harness design. This includes, for example, requirements on the bend radii of the cables, and where to locate cable fasteners.

It can be desirable for the user to get more than one routing solution, so that a designer can make a subjective choice between different Pareto optimal solutions. A more effective solution methodology to generate several solutions with different objective weights should be investigated. A suggestion is to first compute a solution with some weights, and then use information from that solution to compute a new one with different weights. This could decrease the computation time for getting the second solution.

Bibliography

[1] G. I. Thantulage, "Ant colony optimization based simulation of 3D automatic hose/pipe routing," Ph.D. dissertation, Brunel University School of Engineering and Design PhD Theses, 2009.

[2] J. Weise, S. Benkhardt, and S. Mostaghim, "Graph-based multi-objective generation of customised wiring harnesses," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2019, pp. 407–408.

[3] T. Fernando and T. Kalganova, "Multi-colony ant systems for multi-hose routing," *International Journal of Computer Applications*, vol. 59, no. 2, 2012.

[4] F. Ng, J. M. Ritchie, J. Simmons, and R. Dewar, "Designing cable harness assemblies in virtual environments," *Journal of Materials Processing Technology*, vol. 107, no. 1–3, pp. 37–43, 2000.

[5] W. Pemarathne and T. Fernando, "Wire and cable routings and harness designing systems with AI, a review," in *2016 IEEE International Conference on Information and Automation for Sustainability (ICIAfS)*. IEEE, 2016, pp. 1–6.

[6] A. B. Conru, "A genetic approach to the cable harness routing problem," in *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, vol. 1, June 1994, pp. 200–205.

[7] L. Bahiense, F. Barahona, and O. Porto, "Solving Steiner tree problems in graphs with Lagrangian relaxation," *Journal of Combinatorial Optimization*, vol. 7, no. 3, pp. 259–282, 2003.

[8] C.-W. Lin, L. Rao, P. Giusto, J. D'Ambrosio, and A. Sangiovanni-Vincentelli, "An efficient wire routing and wire sizing algorithm for weight minimization of automotive systems," in *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2014, pp. 1–6.

[9] A. Klein and D. Haugland, "Obstacle-aware optimization of offshore wind farm cable layouts," *Annals of Operations Research*, vol. 272, no. 1–2, pp. 373–388, 2019.

[10] M. Leitner, I. Ljubić, M. Luipersbeck, and M. Sinnl, "A dual ascent-based branch-and-bound framework for the prize-collecting Steiner tree and related problems," *INFORMS Journal on Computing*, vol. 30, no. 2, pp. 402–420, 2018.

[11] P. Aldenvik and M. Schierscher, "Recovery of primal solutions from dual subgradient methods for mixed binary linear programming; a branch-and-bound approach," Master's thesis, University of Gothenburg, 2015.

[12] A.-B. Strömberg, T. Larsson, and M. Patriksson, "Mixed-Integer Linear Optimization: Primal–Dual Relations and Dual Subgradient and Cutting-Plane Methods," in *Numerical Nonsmooth Optimization*. Springer, 2020, pp. 499–547.

[13] R. Belgacem and A. Amir, "A new modified deflected subgradient method," *Journal of King Saud University-Science*, vol. 30, no. 4, pp. 561–567, 2018.

[14] J. Lundgren, M. Rönnqvist, and P. Värbrand, *Optimization*. Studentlitteratur AB, 2010.

[15] M. Chen, R. A. Chowdhury, V. Ramachandran, D. L. Roche, and L. Tong, *Priority queues and Dijkstra's algorithm*. Computer Science Department, University of Texas at Austin, 2007.

[16] C. A. Floudas and P. M. Pardalos, *Encyclopedia of Optimization*. Springer Science & Business Media, 2008.

[17] N. Andréasson, A. Evgrafov, and M. Patriksson, *An Introduction to Continuous Optimization: Foundations and Fundamental Algorithms*. Dover Publications, 2020, ch. 6.2.

[18] T. Larsson, M. Patriksson, and A.-B. Strömberg, "Ergodic, primal convergence in dual subgradient schemes for convex programming," *Mathematical Programming*, vol. 86, no. 2, pp. 283–312, 1999.

[19] K.-H. Chang, "Multiobjective Optimization and Advanced Topics," in *e-Design*, K.-H. Chang, Ed. Boston: Academic Press, 2015, ch. 19, pp. 1105–1173. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780123820389000193

[20] F. A. Lootsma, Ed., *Multi-Objective Linear Programming*. Boston, MA: Springer US, 1999, pp. 229–257. [Online]. Available: https://doi.org/10.1007/978-0-585-28008-0_10

[21] K. Deb, *Multi-objective Optimization*. Boston, MA: Springer US, 2014, pp. 403–449. [Online]. Available: https://doi.org/10.1007/978-1-4614-6940-7_15

[22] K. Miettinen, "Introduction to Multiobjective Optimization: Noninteractive Approaches," in *Multiobjective Optimization. Lecture Notes in Computer Science*, J. Branke, K. Deb, K. Miettinen, and R. Słowiński, Eds. Berlin, Heidelberg: Springer, 2008, vol. 5252. [Online]. Available: https://doi.org/10.1007/978-3-540-88908-3_1

[23] T. Berthold, "Primal heuristics for mixed integer programs," Master's thesis, Technische Universität Berlin, 2006.

[24] "Gurobi optimizer reference manual," 2020. [Online]. Available: http://www.gurobi.com

[25] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear programming: theory and algorithms*. John Wiley & Sons, 2006, ch. 6.