Elevator packing with various objects using cyclic placement method and quadratic line search

Lauri Jokinen

School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 11.5.2022

Supervisor

Prof. Ahti Salo (Prof. Harri Ehtamo)

Advisors

Prof. Harri Ehtamo

M.Sc. (tech.) Mirko Ruokokoski

Aalto University
School of Science

Aalto University
School of Science

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Lauri Jokinen

Title Elevator packing with various objects using cyclic placement method and quadratic line search

Degree programme Engineering Physics and Mathematics

Major Systems and Operations Research **Code of major** SCI3055

Supervisor Prof. Ahti Salo (Prof. Harri Ehtamo)

Advisors Prof. Harri Ehtamo, M.Sc. (tech.) Mirko Ruokokoski

Date 11.5.2022 **Number of pages** 62+14 **Language** English

Abstract

In this Master's thesis, we will develop an algorithm for packing an elevator with various objects. For simplicity, our model is two-dimensional and we model the objects as capsules and rectangles and their combinations. The objects will arrive from an elevator door one by one until no more can be fitted. The aim here is to maximize the number of objects in the elevator, with no overlapping.

The optimization model is based on the concept of object overlapping. This means that initially, we let the objects first overlap each other. Then, we adjust their places to minimize their overlapping area. So, the objective function is the total overlapping area of all the objects. A feasible configuration in the elevator is one with the total overlapping area being equal to zero.

For overlapping, we use two concepts. One is the geometrical overlapping area which is found in the literature. The other is overlapping distance, a concept developed here. Even though the objective function is not differentiable, we use gradient-based optimization methods. We show a way of simplifying the gradient evaluation by canceling a significant amount of counterterms, which speeds up the algorithm drastically.

We first solve the basic capsule packing problem. We use the gradient method, the Broyden–Fletcher–Goldfarb–Shanno method (BFGS), and the cyclic placement method (CPM) which is a special case of a block coordinate method. From the practical point of view, the CPM is the best method. In fact, the CPM decomposes the problem so that only one capsule is adjusted at a time in a sequential fashion. Especially with a modified quadratic line search method, the CPM works very efficiently. These observations are a result of extensive numerical testing.

We finally use our basic capsule packing algorithm for certain application cases. In the first application, we construct a model for capsule and rectangle combinations, which model passengers with suitcases. In the second case, we fit shopping carts and passengers into the elevator.

In this thesis, we form a mathematical optimization model for packing various capsule-rectangle combinations, and study various optimization methods to solve them effectively. Our focus is on the models and algorithms, rather than solving particular cases. We also develop a considerable amount of theory to study the concepts and methods used.

Keywords Elevator packing, Capsule packing problem, Optimization, Global optimization, Gradient method, Broyden–Fletcher–Goldfarb–Shanno method, Cyclic placement method, Quadratic line search method

Aalto University
School of Science

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi
Diplomityön tiivistelmä

Tekijä Lauri Jokinen

Työn nimi Hissin täyttö erilaisilla kappaleilla, käyttäen syklistä sijoittelumenetelmää ja kvadraattista viivahakua

Koulutusohjelma Teknillinen fysiikka ja matematiikka

Pääaine Systeemi- ja operaatiotieteet **Pääaineen koodi** SCI3055

Työn valvoja Prof. Ahti Salo (Prof. Harri Ehtamo)

Työn ohjaajat Prof. Harri Ehtamo, DI Mirko Ruokokoski

Päivämäärä 11.5.2022 **Sivumäärä** 62+14 **Kieli** Englanti

Tiivistelmä

Tässä diplomityössä kehitetään algoritmi hissin täyttämiseen erilaisilla kappaleilla. Mallimme on kaksiulotteinen ja mallinnamme kappaleet kapseleina, suorakulmioina ja niiden yhdistelminä. Kappaleet saapuvat hissin ovesta yksi kerrallaan, kunnes hissi on täynnä. Tarkoitus on maksimoida kappaleiden määrä hississä siten, etteivät ne ole päällekkäin.

Optimointimallimme perustuu kappaleiden päällekkäisyyteen. Annamme kappaleiden olla ensin päällekkäin, ja sitten minimoimme tätä päällekkäisyyttä. Kohdefunktiomme on siis summa kaikista päällekkäisistä pinta-aloista. Käypä konfiguraatio hississä on silloin, kun kohdefunktion arvo on nolla.

Käytämme päällekkäisyyden mittaamiseen kahta käsitettä. Ensimmäinen on geometrinen leikkauspinta-ala, mikä löytyy kirjallisuudesta. Toinen on päällekkäisyyspituus, joka on kirjoittajan kehittämä konsepti. Vaikka kohdefunktio ei ole differentioituva, käytämme gradienttipohjaisia menetelmiä. Näytämme, miten gradientin numeerista laskentaa voidaan sieventää kumoamalla vastatermejä, mikä nopeuttaa algoritmia merkittävästi.

Ensin ratkaisemme kapselin pakkaustehtävän. Ratkaisuun käytämme gradienttimenetelmää, Broyden–Fletcher–Goldfarb–Shanno -menetelmää (BFGS), sekä syklistä sijoittelumenetelmää (CPM). CPM on paras menetelmä kaikkiin tässä työssä esitettyihin tehtäviin. CPM hajauttaa tehtävän siten, että vain yhtä kappaletta liikutetaan kerrallaan. Etenkin räätälöidyllä kvadraattisella viivahaulla CPM toimii erittäin tehokkaasti. Väitteet perustellaan kattavan simulaatiodatan avulla.

Lopuksi käytämme algoritmia kahdessa sovelluksessa. Ensimmäisessä sovelluksessa tutkimme kapselin ja suorakulmion yhdistelmää, jolla mallinamme ihmisiä matkalaukun kanssa. Toisessa sovelluksessa täytämme hissejä ostoskärryillä sekä ihmisillä.

Tässä diplomityössä muodostamme matemaattisen optimointimallin kapseli-suorakulmio kombinaatioiden pakkaamiseen, ja tutkimme minkälaiset optimointimenetelmät ratkaisevat sen tehokkaasti. Keskitymme mallien ja algoritmien yleisiin ominaisuuksiin, emmekä niinkään esimerkkitapausten ratkaisuun. Työssä on myös huomattava määrä teoriaa käytettyihin konsepteihin ja menetelmiin liittyen.

Avainsanat Hissin pakkaus, kapselin pakkausongelma, optimointi, globaali optimointi, gradienttimenetelemä, Broyden–Fletcher–Goldfarb–Shanno'n menetelmä, syklinen asettelumenetelmä, kvadraattinen viivahakumenetelmä

Preface

Honestly, writing this thesis was hard work. Writing about mathematical concepts and algorithms is difficult – especially about new concepts for which there is no existing notation. Putting these concepts into words is an artform, and without it, the theory may become useless. However, I'm pleased with the result, and I can honestly say that I'm proud of this work.

I want to thank Harri for spending hours and hours on this thesis. To him I tried to explain my wildest ideas, and he kept my feet on the ground. Also, my motivation was quite low at times but Harri couraged me to keep going. I want to thank Mirko for his support and supply of materials.

This thesis was written under the COVID-19 pandemic, thus home was my workplace most of the time. It was important to have support there too, and therefore, Loviisa receives the most thanks of all. Also, my family gave me support whenever I needed it, as did the members of the Polytech Choir.

For financial support during the work, I want to thank the Foundation for Aalto University Science and Technology, the KONE Corporation, and the Department of Mathematics and Systems Analysis.

Mathematics is not a careful march down a well-cleared highway, but a journey into a strange wilderness, where the explorers often get lost. Rigor should be a signal to the historians that the maps have been made, and the real explorers have gone elsewhere.

— W. S. Anglin, Mathematics author

Servinkuja, 21.6.2021 Lauri Jokinen

Symbols and abbreviations

Symbols

n	Number of capsules in the elevator.
n'	Number of capsule-rectange conbinations in the elevator.
N	A set defined as $\{1, \dots, n\}$.
N'	A set defined as $\{n+1, \dots, n+n'\}$.
m	Index for an object in an elevator, $1 \leq m \leq n+n'$.
k	Iteration index.
$\mathbf{b}$	Displacement parameters for the elevator box.
$\mathbf{x}$ or $\mathbf{x}'$	Displacement parameters for an object.
$\mathbf{x}^m$	Displacement parameters for an object indexed with m.
$\mathbf{x}_k^m$	Displacement parameters for an object indexed with m, and iteration index k.
$\mathbf{x}_k$	Optimization parameters in a column vector (containing the placement parameters for all the objects in the elevator)
X	Set containing the placement parameters for all the objects in the elevator.
$E(\mathbf{x}, \mathbf{x}')$	Overlapping distance of the capsules $\mathbf{x}$ and $\mathbf{x}'$.
$A(\mathbf{x}, \mathbf{x}')$	Geometrical intersection area of the objects $\mathbf{x}$ and $\mathbf{x}'$.
$A(\mathbf{x})$	Geometrical area of the object $\mathbf{x}$.

Abbreviations

BFGS	Broyden–Fletcher–Goldfarb–Shanno method.
CPM	Cyclic placement method.
QLS	Quadratic line search method.

Contents

Abstract 3

Abstract (in Finnish) 4

Preface 5

Symbols and abbreviations 6

Contents 7

1 Introduction 9
- 1.1 Background . . . 9
- 1.2 Thesis summary . . . 9
- 1.3 Thesis contents . . . 10
- 1.4 Literature review . . . 11

2 Capsule packing problem 12
- 2.1 Defining capsules and rectangles . . . 14
- 2.2 Overlapping . . . 15
  - Overlapping area . . . 15
  - Overlapping distance . . . 18
- 2.3 Objective function . . . 23

3 Optimization methods 26
- 3.1 Simplifying the gradient evaluation . . . 26
- 3.2 Capsule packing problem with gradient method and BFGS . . . 29
- 3.3 Cyclic placement method . . . 31
- 3.4 Global optimization and comparison of methods . . . 32
- 3.5 Numerical simulations, convergence analysis . . . 33

4 Gradient method with quadratic line search 35
- 4.1 Definition of the quadratic line search . . . 35
- 4.2 Numerical simulations . . . 37

5 Passengers with suitcases 46
- 5.1 The model . . . 47
- 5.2 Objective function . . . 49
- 5.3 Results . . . 49

6 Shopping cart problem 52
- 6.1 One fixed shopping cart . . . 52
- 6.2 One to three moving shopping carts . . . 54

7 Discussion and conclusions 58
- 7.1 Results and their reliability . . . 59

7.2 Future research 59

References **60**

A Intersection points of circles and line segments **63**

B Algorithm for a minimum distance between two line segments **65**

C Simulation data **67**
C.1 Global solving times 67
C.2 Example result for each scenario 72

D Lemma 3 **76**

1 Introduction

1.1 Background

Optimal packing of passenger elevators, that usually also carry goods, e.g., shopping carts, luggage, etc., is not studied very much in the literature. Mathematically, the problem belongs to so-called packing problems which deal with, e.g., maximizing the number of similar objects in a box.

A recent ISO standard (8100-32:2020) [1] describes naive methods for calculating person capacities in elevators. In the standard, we divide the elevator's rated load by the average mass of a person or divide the lift floor area by the average area of a person, projected to the floor. Recently, Mirko Ruokokoski studied elevator packing, where passengers were modeled as ellipses [2, 3]. Generally, ellipse packing problem is quite well understood [4], but most of these studies approach the problem heuristically. Ruokokoski's approach is analytical and uses analytical optimization methods, as we shall do here as well.

Nevertheless, packing of passengers and goods is a different problem compared to packing passengers only: instead of ellipses, we use certain type of capsules, rectangles, and their combinations. This thesis has been done in professor Harri Ehtamo's research group in Aalto University, where also other optimization problems related to elevators has been studied, e.g., optimal routing problem for elevator groups [5, 6].

1.2 Thesis summary

In the thesis, we describe a model of an elevator and try to find methods for packing the elevator reliably and efficiently. We model the elevator floor as a two-dimensional rectangle, onto which passengers enter one by one and adjust their places in a way that avoids "overlapping". The two-dimensional model is suitable, considering that objects, let alone passengers, are rarely packed on top of each other in an elevator. Also, reducing to two dimensions allows much faster problem-solving.

We model the packed passengers and objects as capsules and rectangles. We also present a model for a passenger with a suitcase, as a combination of a capsule and a rectangle. These shapes are convenient because their geometry allows us to evaluate their overlapping area in a closed form, rather than numerically (e.g., in the case of two ellipses [2, 7]). We further simplify the problem by using passengers of constant size and having only one wide elevator door. The simplifications make the simulation results easier to replicate, but the algorithm is applicable for more complex models.

The elevator packing algorithm adds passengers and objects one by one. In between, we need to confirm that all the objects can fit into the elevator. This is solved by minimizing the total overlapping area, or a measure for it. At a feasible configuration in the elevator, the total overlapping area (or objective function value) is zero. There may be infinitely many feasible arrangements, and also infinitely many local non-feasible configurations.

For overlapping, we use geometrical overlapping area, as well as overlapping distance, for which the concept is developed by the author. Overlapping distance uses

special geometry of capsules to find if, and how much, two capsules are overlapping. The method is quick to compute and is analytically quite simple. The contributions of the author are discussed in more detail in Chapter 7.

The objective function is continuous and non-linear, but not necessarily differentiable, so minimizing it will require a robust non-linear optimization method. Despite this fact, we use gradient-based methods here: gradient method and Broyden-Fletcher-Goldfarb-Shanno (BFGS) method. We show how the numerical gradient evaluation can be simplified by canceling counterterms, making the computation much faster [8]. The effect is significant, halving the evaluation time with just five objects, and quartering it with eleven objects.

A block coordinate method is a well-studied optimization method and perhaps one of the simplest ways of decomposing an optimization problem [9]. Because of the structure of our objective function, we can formulate an effective block coordinate method, which we call the cyclic placement method (CPM) [8]. In fact, the method improves the placement of one object at a time and leaves the rest of the objects fixed in their positions. We sequentially improve the placement of each object in a continuous sequence.

A gradient method is as good as its line search method. Line search methods balance between accuracy and speed, and choosing between these depends on the application. In this thesis, we use a quadratic line search (QLS) [10, 11], which attempts to combine the best of both. The method is implemented in such a way that only a single objective function evaluation is needed [10], which makes it a powerful line search method when used with the CPM.

We apply the methods to two elevator packing application cases. First, we fill the elevator with passengers some of which have suitcases. The other application case is filling the elevator with shopping carts and passengers. We implement the algorithms in *Matlab* and use its *Optimization Toolbox* and *Polygons* -packages in the simulations.

1.3 Thesis contents

In the next Chapter, we describe the capsule packing problem and show how the overlapping area of two objects can be evaluated. We also define the overlapping distance. A function for total overlapping is presented in Chapter 2.3, where we analyze the function as an objective function for non-linear optimization algorithms. Chapter 3 presents a method for comparing different optimization methods, using expected global solving time. We show how the numerical evaluation of the objective function's gradient can be simplified. In the Chapter, we also present the optimization methods: the gradient method, the BFGS, and the CPM. We show how the methods perform with the capsule packing problem. In Chapter 4, we define the QLS and show how the method, combined with CPM, performs with the capsule packing problem. We also apply the QLS to a few simple example problems. Chapters 5 and 6 present two different elevator packing applications. In Chapter 5, we pack elevators with passengers, some of which are coupled with a suitcase, and in Chapter 6, we pack the elevator car with shopping carts and passengers.

1.4 Literature review

There is a lot of documentation about packing problems, because of the number of variations: packing problems may have distinctively different objective functions and different restrictions depending on the application. E.g., when cutting pieces from striped fabric, the orientation of the pieces matter. Most papers on packing problems handle one or a few different kinds of shapes. A circle packing problem is a well-studied topic [12], as well as an ellipse packing problem [2, 4, 13] and rectangle packing [14]. However, literature on the capsule packing problem is rather limited [8]. Some methods for solving packing problems include tabu-search [14], and genetic algorithm [15]. A common approach to a packing problem is to gradually expand the objects [4, 13], or shrink the box until maximum packing density is reached. This approach is not applicable to elevator packing, since we cannot scale the packed objects nor the elevator.

In his paper, Ruokokoski [2] showed an approach to an elevator packing problem, with ellipses, but otherwise similar to ours with capsules and rectangles. However, sometimes the objects tend to stay on top of each other, and the simulation durations are rather long. Our algorithm manages to avoid these flaws.

An algorithm for a capsule packing was presented in [8], for which this thesis is essentially a refinement and an extension. Here, we use many of the methods used in [8]: the algorithm for the overlapping area of two capsules or rectangles; the objective function; the CPM; the simplification of the objective function's gradient. We use the same objective function as in [8], except ignoring penalty terms, which were used to block the capsules being on top of each other. We here find that the problem can be solved without the penalty terms when global solving techniques are used.

The CPM is a special case of a block coordinate method [8, 9]. Variations to the block coordinate method include acceleration step [16], Hooke and Jeeves method [11, 16], random indexing [17], and Gauss-Southwell [18] methods. We will compare these variations in more detail in Chapter 3.3. A stochastic gradient method is also a popular method for decomposing a large optimization problem. However, the block coordinate method is a stochastic gradient method, if we use random indexing [19].

A question of global search (e.g. random search) and local search (e.g. gradient method) is always relevant when we have non-convex optimization problems. This is often referred to as the balance of exploration and exploitation [20]. The objective function in this thesis has many local minima, so it is necessary to use some kind of random search as well. In this thesis, we use simple trial and error with different initial conditions. More sophisticated random search methods and heuristics are out of the scope of this thesis.

Line search is a very well-studied topic, as well as the quadratic interpolation line search method [11, 10]. The method shown in [10] utilizes also cubic interpolation, extending the method shown here.

2 Capsule packing problem

Our aim here is to pack an elevator floor with as many capsules as possible so that the capsules do not overlap. So, we consider the elevator as a rectangle box, of which the upper edge acts as an elevator door. We initially place the capsules in a random position (placement and orientation) so that the capsules' center points lie on the door threshold. We present the method and the algorithm in Figure 1; in Figure 2 we present a flow chart for the elevator packing algorithm. We define n to be the number of capsules in the elevator. We start by initializing $n \leftarrow 0$ and define solution S_0 as an "empty solution" with no capsules. Next, we shall add a capsule at the door and increment $n \leftarrow n + 1$. The capsules may overlap initially, and we aim to fit all the capsules in the box with no overlapping. We minimize the overlapping with, e.g., a gradient method. Let S_n be a generated solution with the least overlap. If solution S_n is feasible, i.e., no overlaps remain, we proceed with adding the next capsule, etc. If the solution is not feasible, we return the solution S_{n-1}, which is the previous feasible solution.

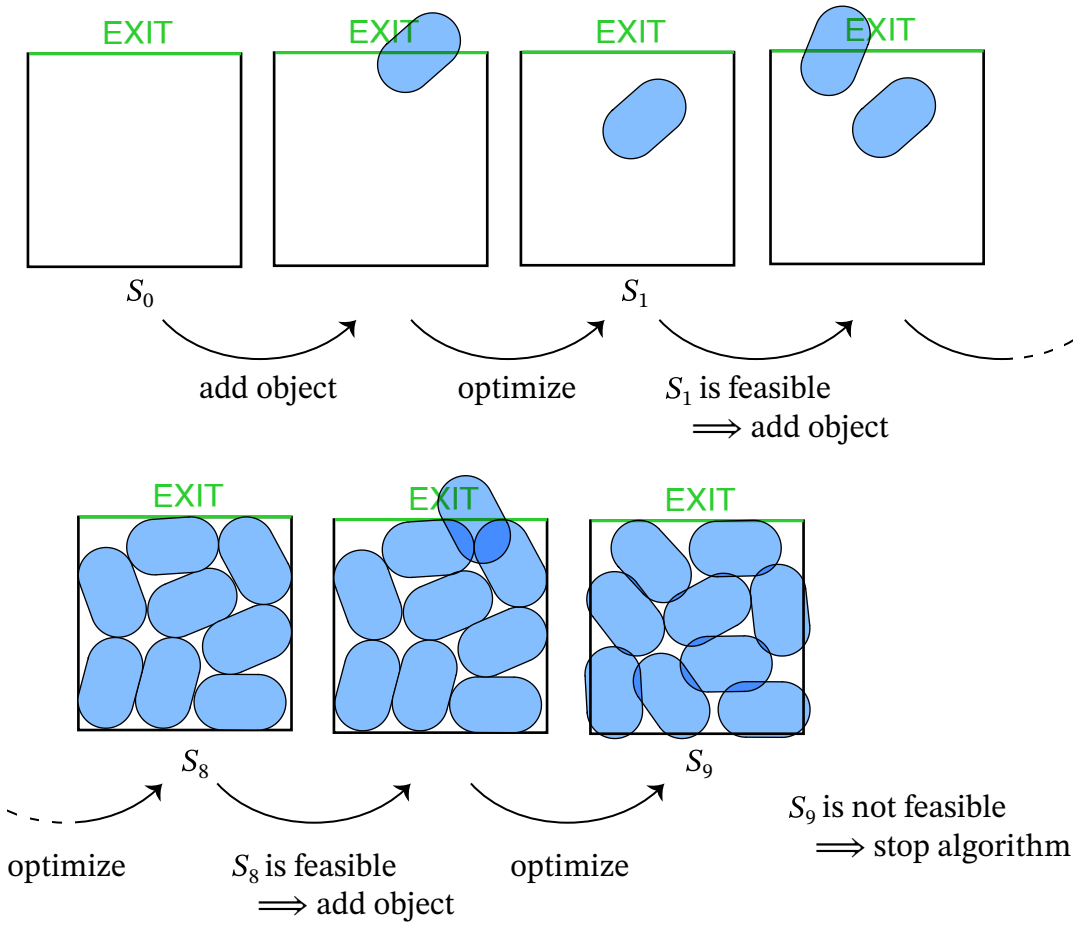


Figure 1: Algorithm for the capsule packing problem. Feasibility in this context means that the objects do not overlap.

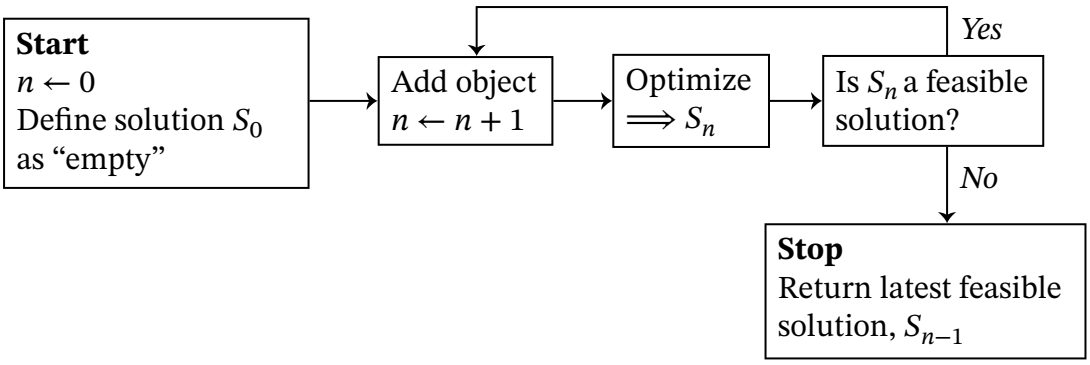


Figure 2: Flow chart of the capsule packing algorithm.

ISO standard 8100-30:2019 [1] specifies elevator dimensions along with their rated capacities, see Table 1. We choose three of these sizes to be used in the simulations, indicated in bold in the Table. American body sizes are shown in Table 2 [21]. To simplify the problem, we'll use constant-sized passengers based on these dimensions, 455×275 mm.

Table 1: Elevator floor sizes from ISO 8100-30:2019 standard [1]. The first column is from CEN 81-1:1998 standard [22]. Chosen sizes for the simulations are in bold.

Rated capacity	Rated load (kg)	Width (mm)	Depth (mm)	Area (m^2)
8	630	1100	1400	1.54
10	**800**	**1350**	**1400**	**1.89**
13	1000	1600	1400	2.24
17	**1275**	**2000**	**1400**	**2.8**
18	1350	2000	1500	3.0
21	1600	2100	1600	3.36
24	1800	2350	1600	3.76
26	**2000**	**2350**	**1700**	**3.995**

Table 2: Human dimensions, U.S. adults aged 19 to 65 years [21]. The rightmost column is used for the dimensions of passengers.

Measure	Men, median	Women, median	Average, with clothes
Shoulder breadth	470 mm	400 mm	**455 mm**
Chest depth	255 mm	255 mm	**275 mm**

2.1 Defining capsules and rectangles

We define capsules and rectangles (objects) in the same way as in [8]. Mathematically, both are defined analogously. A capsule is defined by a rectangle and two similar semicircles, see Figure 3. A rectangle is defined as a capsule but without circles. Define

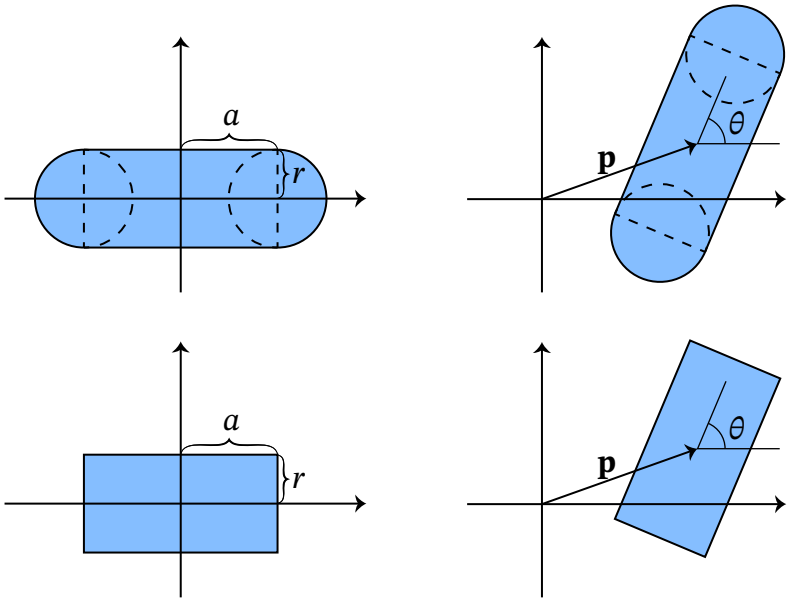


Figure 3: Definition of capsule and rectangle, in terms of their dimensions and displacement parameters.

$a \in \mathbb{R}_+$ and $r \in \mathbb{R}_+$, to be the dimensions of the capsule. Position vector $\mathbf{p} \in \mathbb{R}^2$ and angle $\theta \in \mathbb{R}$ define its location and orientation. We first define a capsule centered at the origin, and then displace it, as in Figure 3. We define a capsule rectangle by its four corner points,

$$\mathbf{R}_1 = \begin{bmatrix} a \\ r \end{bmatrix}, \ \mathbf{R}_2 = \begin{bmatrix} -a \\ r \end{bmatrix}, \ \mathbf{R}_3 = \begin{bmatrix} -a \\ -r \end{bmatrix}, \ \mathbf{R}_4 = \begin{bmatrix} a \\ -r \end{bmatrix}.$$

Then, define capsule circles, both with radius r, and center points,

$$\mathbf{C}_1 = \begin{bmatrix} a \\ 0 \end{bmatrix}, \ \mathbf{C}_2 = \begin{bmatrix} -a \\ 0 \end{bmatrix}.$$

The above construction defines a capsule centered at the origin. Define a rotation matrix,

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

We define rotated and displaced capsule rectangle corner points and capsule circle center points as,

$$\begin{aligned} \mathbf{r}_i &= R(\theta)\,\mathbf{R}_i + \mathbf{p}; \quad \text{for } i = 1,2,3,4, \\ \mathbf{c}_i &= R(\theta)\,\mathbf{C}_i + \mathbf{p}; \quad \text{for } i = 1,2. \end{aligned}$$

A point $\mathbf{q}$ belongs to the capsule when it belongs to the rectangle or either of the circles. I.e., the point $\mathbf{q}$ belongs to the capsule iff,

$$\begin{aligned} &\big[(-a \le Q_1 \le a) \wedge (-r \le Q_2 \le r)\big] \\ \vee&\big[(Q_1 + a)^2 + (Q_2)^2 \le r\big] \\ \vee&\big[(Q_1 - a)^2 + (Q_2)^2 \le r\big], \end{aligned} \tag{1}$$

where $\mathbf{Q} = [\, Q_1 \;\; Q_2 \,]^\top = R(-\theta)(\mathbf{q} - \mathbf{p})$. We show an equivalent and more practical definition in Chapter 2.2.

2.2 Overlapping

The idea of the elevator packing algorithm is to initially allow the capsules to intersect and then minimize their overlap. For this, we need a measure of how much two capsules are overlapping. We develop two approaches: overlapping area and overlapping distance. The latter approach takes advantage of the capsules' special geometry but is not applicable to pure rectangles. The overlapping distance is faster to evaluate than the overlapping area and is also applicable to three-dimensional capsules. We define a vector of a capsule's displacement parameters, $\mathbf{x} = [\, \mathbf{p}^\top \;\; \theta \,]^\top$, which we'll use in the definition of the overlapping functions.

Overlapping area

In this thesis, we'll use the same method as presented in [8]. The contours of capsules and rectangles are formed of line segments and semicircles; hence, they are convex sets and their intersections are convex. Thus, we can divide the intersection area into convex polygon and convex circle segments. We shall consider two intersecting capsules; the case for rectangles is analogous. For the sake of simplicity, we consider capsules of the same size, however, a generalization to different sizes is possible. Define a set C, containing the points of a capsule defined in terms of $\mathbf{x} = [\, \mathbf{p}^\top \;\; \theta \,]^\top \in \mathbb{R}^3$, and $a, r \in \mathbb{R}_+$. Define another set, C', in terms of a capsule defined by $\mathbf{x}' = [\, (\mathbf{p}')^\top \;\; \theta' \,]^\top \in \mathbb{R}^3$, a and r.

We define the set of *fixed capsule point(s)* (fcp(s)) for each capsule (see Figure 4a),

$$\mathrm{FCP} = \left\{ R(\theta) \begin{bmatrix} \pm a \\ \pm r \end{bmatrix} + \mathbf{p}, \text{ with every combination of } \pm \text{-signs} \right\},$$

$$\mathrm{FCP}' = \left\{ R(\theta') \begin{bmatrix} \pm a' \\ \pm r' \end{bmatrix} + \mathbf{p}', \text{ with every combination of } \pm \text{-signs} \right\}.$$

We also define *capsule intersection point(s)*, (cip(s)), of two capsules in terms of intersection points of the capsules' line segments and semicircles, see Figure 4b. Analytical formulas for cips are provided in Appendix A. The capsules may also share joint contours, thus producing infinitely many intersection points. These cases are also handled in Appendix A, and examples are seen in Figure 5. Define the cips of C and C' as a set, CIP.

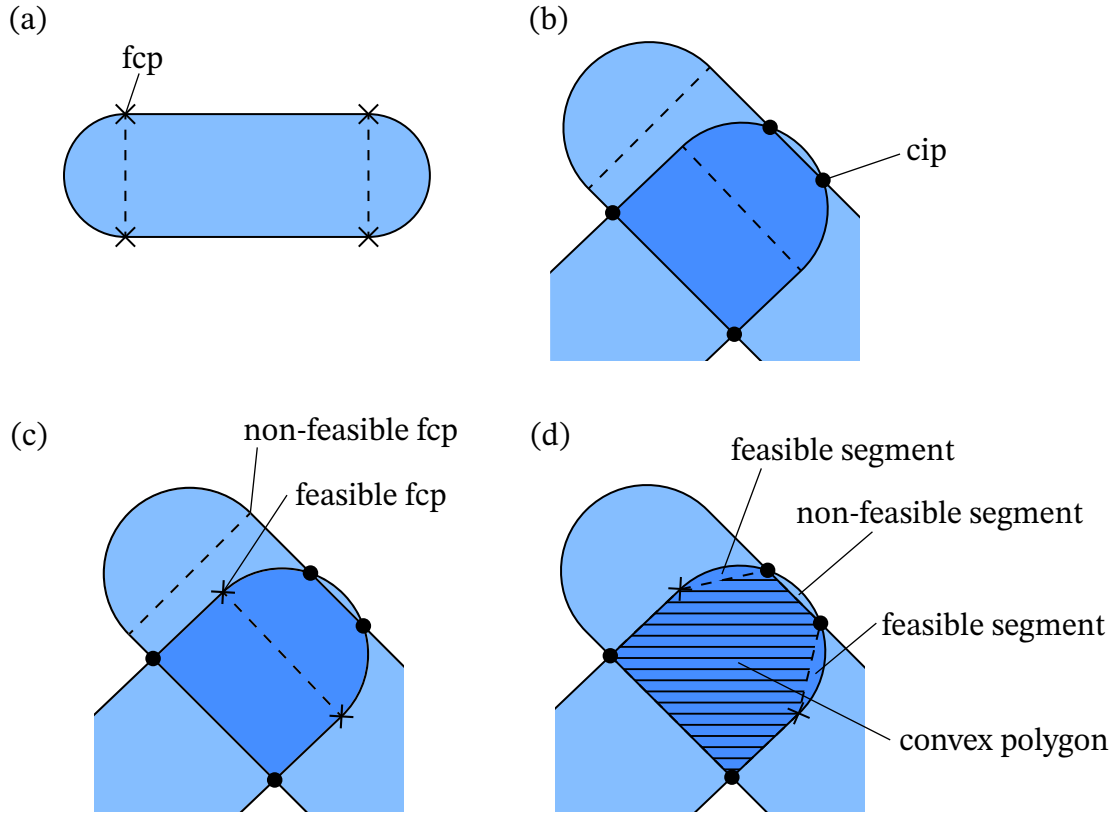


Figure 4: An example of area evaluation. We present fcps in (a), and cips in (b). In (c) we have cips and such fcps, which lie at the intersection; the points that define the convex polygon. In (d), we have the convex polygon with feasible and non-feasible circle segments.

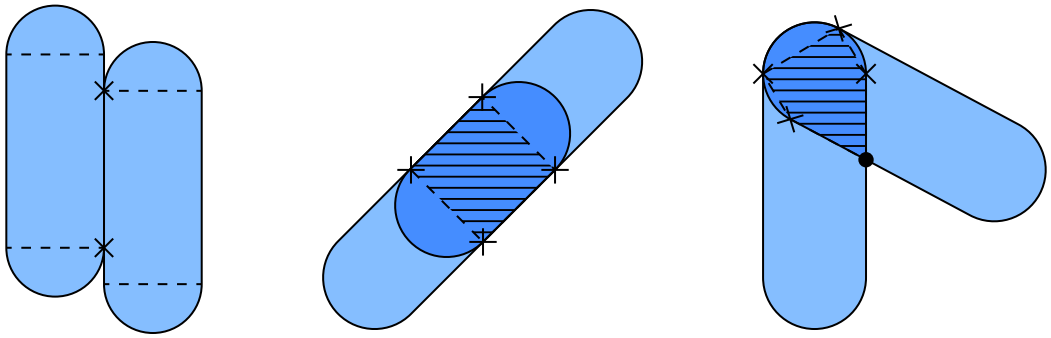


Figure 5: Examples where capsules share joint contours.

The convex polygon is defined as a convex hull that covers the following finite set of points (see Figure 4c),

$$\mathrm{CIP} \cup (\mathrm{FCP} \cap I) \cup (\mathrm{FCP}' \cap I),\ I = C \cap C';$$

the polygon is seen in Figure 4d. The convex polygon and its area can be numerically evaluated using *Matlab* commands `convhull` and `polyarea`. The command `convhull`

returns the hull's corner points, arranged counterclockwise.

Next, we shall find all feasible segments, evaluate their areas and add them to the intersection area. A feasible segment is defined to be any segment that is a subset of the intersection I, and whose intersection area with the convex polygon is zero. All feasible segments are on the boundary of the convex polygon and in its exterior. Thus, we shall look for a feasible segment at each edge of the convex polygon, and limit our search to the exterior of the convex polygon.

Next, we define the segment arc's center point and show how it can be used to determine a segment's feasibility. Consider a corner point $\mathbf{h}_1$, on the convex polygon (shown in Figure 6), and a subsequent corner point counterclockwise, $\mathbf{h}_2$. We shall

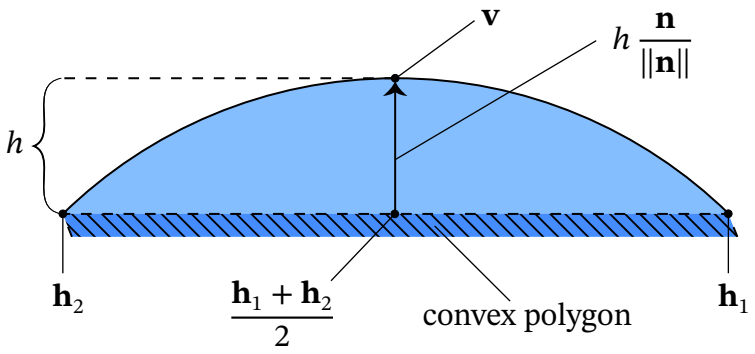


Figure 6: Arc center point. Because the convex polygon's corner points are in counterclockwise order, the vector $\mathbf{n}$ is oriented as pointing towards the exterior of the convex polygon.

define the arc's center point with the following formula (see Figure 6),

$$\mathbf{v} = \frac{\mathbf{h}_1 + \mathbf{h}_2}{2} + h\frac{\mathbf{n}}{\|\mathbf{n}\|},$$

where h is the height of the segment and $\mathbf{n}$ is a normal vector for the convex polygon's edge, oriented towards the exterior of the polygon. The formula for the segment's height is [23],

$$h = r - \frac{1}{2}\sqrt{4r^2 - \|\mathbf{h}_2 - \mathbf{h}_1\|^2}.$$

We shall construct $\mathbf{n}$ by turning vector $\mathbf{h}_2 - \mathbf{h}_1$ clockwise by $\pi/2$, thus making $\mathbf{n}$ normal to $\mathbf{h}_2 - \mathbf{h}_1$, and directed towards the exterior of the polygon (see Figure 6),

$$\begin{aligned}
\mathbf{n} &= R(-\pi/2)(\mathbf{h}_2 - \mathbf{h}_1) \\
&= \begin{bmatrix} \cos(-\pi/2) & -\sin(-\pi/2) \\ \sin(-\pi/2) & \cos(-\pi/2) \end{bmatrix} (\mathbf{h}_2 - \mathbf{h}_1) \\
&= \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} (\mathbf{h}_2 - \mathbf{h}_1).
\end{aligned}$$

Here, R is the rotation matrix defined in Chapter 2.1. The segment is either feasible, thus $\mathbf{v} \in I$, or the segment is not feasible, which implies $\mathbf{v} \notin I$. Hence, if $\mathbf{v} \in I$ (which can be found by using Equation (1)), we evaluate the area of the segment by [23],

$$A_{\text{seg}} = \frac{1}{2} r^2(\phi - \sin\phi), \ \phi = 2 \arcsin \frac{\|\mathbf{h}_1 - \mathbf{h}_2\|}{2r},$$

and add it to the total area. If $\mathbf{v} \notin I$, the segment is not feasible, and its area is not taken into account.

We define the intersection area between two capsules, $\mathbf{x}$, and $\mathbf{x}'$, as $A(\mathbf{x}, \mathbf{x}')$, and the area of a capsule as $A(\mathbf{x})$, which is a constant. It can be shown that A is continuous as a function of $(\mathbf{x}, \mathbf{x}')$.

Overlapping distance

With any two circles, it is trivial to find out how much they are overlapping: one finds the distance between their center points and notes if it's smaller or larger than the sum of their radii. We can use a similar approach with two capsules by realizing that a capsule's contour can be defined by the set of points at a constant distance from a line segment, see Figure 7. In this Chapter, we define overlapping distance E, see Figure 8, which measures how much two capsules overlap. Also, we shall refine E in a special case, where the line segments of the capsules intersect, see Figure 8c.

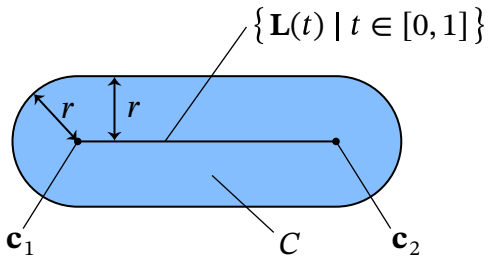


Figure 7: Illustration of the fact that a capsule's contour is at a constant distance from a line segment.

We use a new, equivalent definition, for the set containing the points of a capsule,

$$C = \left\{ \mathbf{c} \,\middle|\, \mathbf{c} = \mathbf{L}(t) + r\,\mathbf{d}, \text{ with some } t \in [0,1], \text{ and } \mathbf{d} \in D \right\}. \tag{2}$$

where $r > 0$ fixed, D is a closed unit disc and,

$$\mathbf{L}(t) = t\,\mathbf{c}_1 + (1-t)\,\mathbf{c}_2, \ \mathbf{c}_1, \mathbf{c}_2 \in \mathbb{R}^2.$$

For all $\mathbf{c} \in C$, following inequality holds,

$$r \geq \min_{t \in [0,1]} \|\mathbf{c} - \mathbf{L}(t)\|. \tag{3}$$

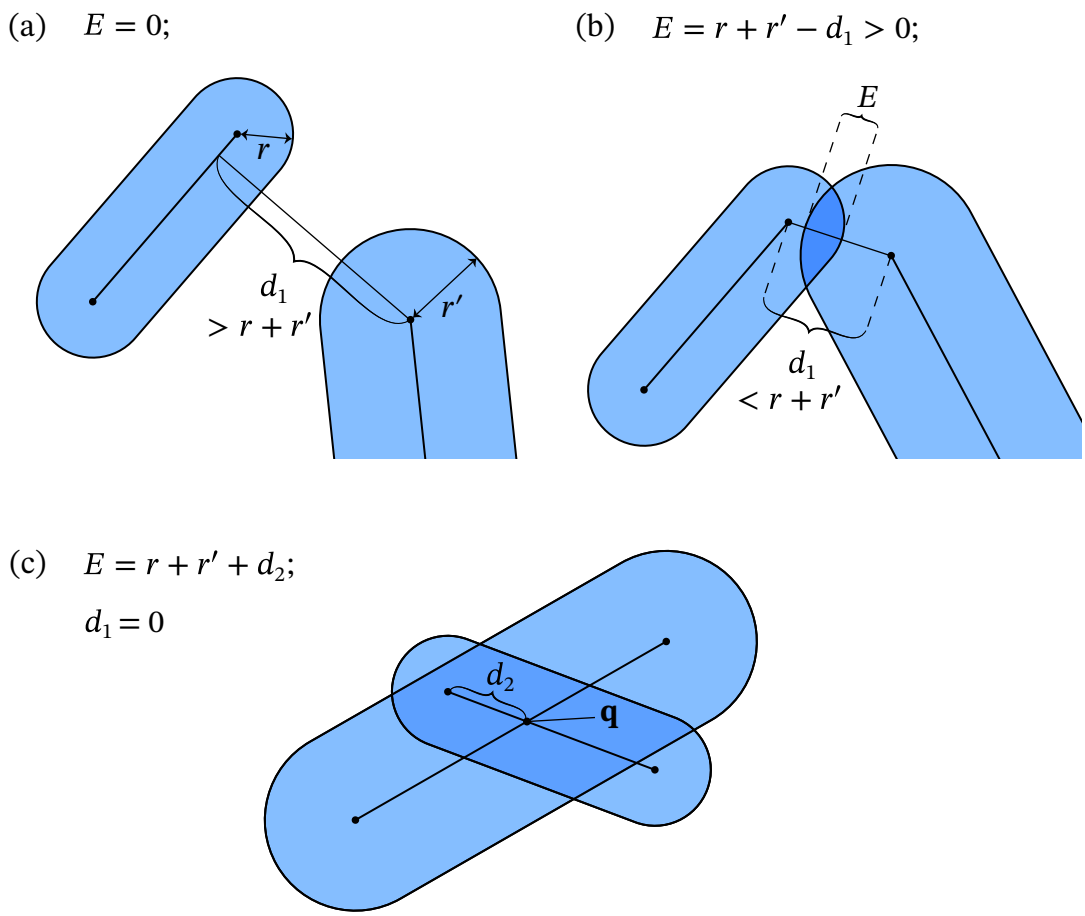


Figure 8: Value of function E in different cases. (a) When d_1 is large enough ($d_1 \geq r + r'$), the capsules are not overlapping, hence we define $E = 0$. In this case, the overlapping distance is equal to the overlapping area. In (b), d_1 is smaller, but non-zero ($0 < d_1 < r + r'$), implying that the capsules are overlapping. In this case, we define $E = r + r' - d_1 > 0$. In cases (a) and (b), a value for d_2 is not defined. In (c) the line segments intersect, i.e., $d_1 = 0$, in which case we define d_2 (d_2 is defined in Equation (6), and in sentence below it), and $E = r + r' + d_2$.

Define another capsule C', analogously to the above definition,

$$C' = \left\{ \mathbf{c}' \;\middle|\; \mathbf{c}' = \mathbf{L}'(t') + r'\,\mathbf{d}', \text{ with some } t' \in [0,1], \text{ and } \mathbf{d}' \in D \right\},$$
$$\mathbf{L}'(t') = t'\,\mathbf{c}'_1 + (1-t')\,\mathbf{c}'_2,$$
$$\mathbf{c}'_1, \mathbf{c}'_2 \in \mathbb{R}^2,$$
$$r' > 0, \text{ fixed.}$$

Note that we let the capsules be of different sizes. Next, we shall write the minimum distance between the line segments $\mathbf{L}$ and $\mathbf{L}'$,

$$d_1 = \min_{t,t' \in [0,1]} \|\mathbf{L}(t) - \mathbf{L}'(t')\|, \tag{4}$$

see Figures 8a, 8b. Analytical formula for d_1 is provided in Appendix B. The mathematical definition of d_2 in Figure 8c is given at the end of this section, in Equation (6), and the sentence below it. With Lemma 1 and Theorem 1 we show how we can use the formula in Equation (4) to find out if two capsules overlap.

Lemma 1. *Let D be a closed unit disc, $\boldsymbol{a} \in \mathbb{R}^2$ and $b \geq 0$. Then,*

$$\min_{\boldsymbol{x} \in D} \|\boldsymbol{a} + b\boldsymbol{x}\| = \max\{\|\boldsymbol{a}\| - b,\ 0\}, \tag{5}$$

holds.

Proof. Note, that the lemma holds for $b = 0$. Suppose now that $b > 0$. We shall consider the left-hand side of Equation (5) and divide the problem into two cases: (i) $\|\mathbf{a}\| \leq b$, (ii) $\|\mathbf{a}\| > b$. Figure 9 illustrates the cases and solutions visually. Consider (i). By choosing $\mathbf{x} = -\mathbf{a}/b$, the left hand side of Equation (5) is zero. A value of a norm is non-negative, thus our choice $\mathbf{x} = -\mathbf{a}/b$ is a solution to the minimization problem.

Case (ii) can be proven with two inequalities. First, we obtain a lower bound for the left hand side of Equation (5), using triangle inequality [24],

$$\min_{\mathbf{x} \in D} \|\mathbf{a} + b\mathbf{x}\| \geq \min_{\mathbf{x} \in D} \left| \|\mathbf{a}\| - \|b\mathbf{x}\| \right| = \|\mathbf{a}\| - b, \text{ for } \|\mathbf{a}\| > b.$$

An upper bound is found by a substitution of $\mathbf{x} = -\mathbf{a}/\|\mathbf{a}\|$,

$$\min_{\mathbf{x} \in D} \|\mathbf{a} + b\mathbf{x}\| \leq \left\| \mathbf{a} - b\frac{\mathbf{a}}{\|\mathbf{a}\|} \right\| = \|\mathbf{a}\| \left(1 - \frac{b}{\|\mathbf{a}\|}\right) = \|\mathbf{a}\| - b.$$

Combining the above results, we get a solution to the minimization problem,

$$\min_{\mathbf{x} \in D} \|\mathbf{a} + b\mathbf{x}\| = \begin{cases} 0, & \text{for } \|\mathbf{a}\| - b \leq 0 \\ \|\mathbf{a}\| - b, & \text{for } \|\mathbf{a}\| - b > 0 \end{cases}$$
$$= \max\{\|\mathbf{a}\| - b,\ 0\}.$$

□

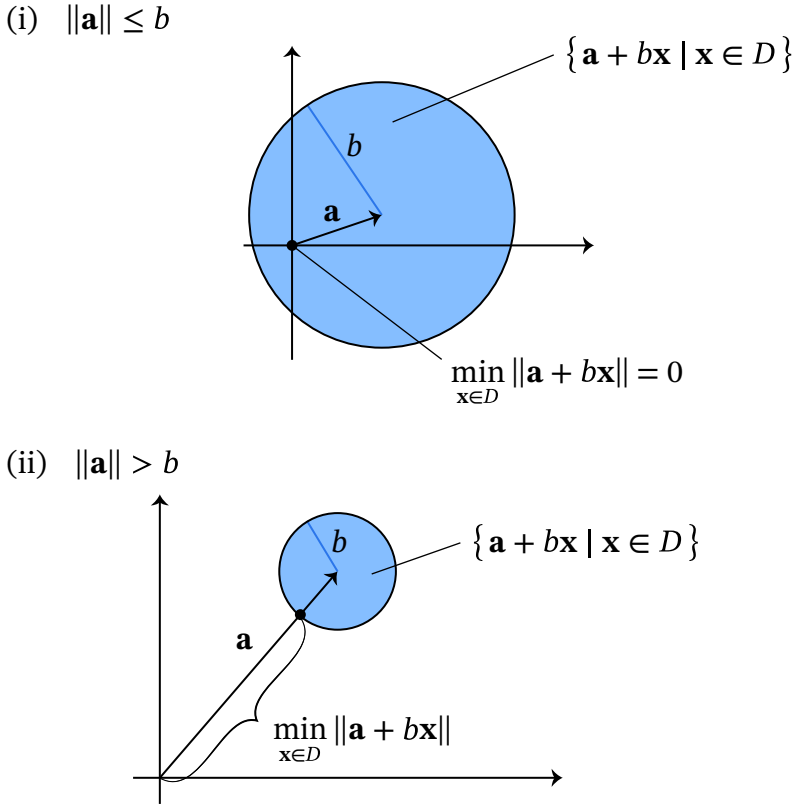


Figure 9: Illustration of Lemma 1.

Theorem 1. *$C \cap C' \neq \emptyset$ iff $d_1 \le r + r'$.*

Proof. For the equivalence, we show that: (i) $C \cap C' \neq \emptyset$ implies $d_1 \le r + r'$, (ii) $C \cap C' = \emptyset$ implies $d_1 > r + r'$. Consider (i). Let's take a point $\mathbf{a} \in C \cap C'$. By Equation (3), and the triangle inequality,

$$\begin{aligned} r + r' &\ge \min_{t\in[0,1]} \|\mathbf{a} - \mathbf{L}(t)\| + \min_{t'\in[0,1]} \|\mathbf{a} - \mathbf{L}'(t')\| \\ &\ge \min_{t,t'\in[0,1]} \|\mathbf{a} - \mathbf{L}(t) - \mathbf{a} + \mathbf{L}'(t')\| \\ &= \min_{t,t'\in[0,1]} \|\mathbf{L}(t) - \mathbf{L}'(t')\| = d_1. \end{aligned}$$

This proves (i). Equation $C \cap C' = \emptyset$ holds if we have $0 < \min_{\mathbf{c}\in C,\, \mathbf{c}'\in C'} \|\mathbf{c} - \mathbf{c}'\|$. Let D be a closed unit disc. By using Equation (2) and Lemma 1,

$$\begin{aligned} 0 &< \min_{\mathbf{c}\in C,\, \mathbf{c}'\in C'} \|\mathbf{c} - \mathbf{c}'\| \\ &= \min_{t,t'\in[0,1],\ \mathbf{d},\mathbf{d}'\in D} \|(\mathbf{L}(t) + r\mathbf{d}) - (\mathbf{L}'(t') + r'\mathbf{d}')\| \\ &= \max\left\{ \min_{t,t'\in[0,1],\ \mathbf{d}\in D} \|\mathbf{L}(t) + r\mathbf{d} - \mathbf{L}'(t')\| - r',\ 0 \right\}. \end{aligned}$$

Zero on the right-hand side is ruled out because it leads to a contradiction. Hence,

$$\begin{aligned} 0 &< \min_{t,t'\in[0,1],\ \mathbf{d}\in D} \|\mathbf{L}(t) - \mathbf{L}'(t') + r\mathbf{d}\| - r' \\ &= \max\left\{ \min_{t,t'\in[0,1]} \|\mathbf{L}(t) - \mathbf{L}'(t')\| - r,\ 0 \right\} - r' \\ &= \min_{t,t'\in[0,1]} \|\mathbf{L}(t) - \mathbf{L}'(t')\| - r - r' \\ &= d_1 - r - r', \end{aligned}$$

which proves (ii). □

Theorem 1 implies that overlapping of C and C' occurs iff $r + r' - d_1 \geq 0$. Thus, we aim to move capsules so that $r + r' - d_1 < 0$ holds. Let $\mathbf{x}$ and $\mathbf{x}'$ be the displacement vectors of C and C', respectively. We shall now define the overlapping distance,

$$E(\mathbf{x}, \mathbf{x}') = \begin{cases} 0, & \text{for } r + r' \leq d_1 \\ r + r' - d_1, & \text{for } 0 < d_1 < r + r', \end{cases}$$

which is zero when the capsules do not overlap and is greater than zero when they overlap, see Figure 8. However, an issue arises in a case when $d_1 = 0$, i.e., when the line segments $\mathbf{L}$ and $\mathbf{L}'$ intersect, see Figure 8c. In these cases, the overlapping distance is constant, $E = r + r'$, and the value is probably constant also at the neighborhood of these points, with respect to the placement parameters, $(\mathbf{x}, \mathbf{x}')$. A constant function value at a neighborhood implies a zero gradient vector. The purpose of function E is to be minimized with a numerical optimizer which often requires a non-zero gradient vector. For our purpose, we shall redefine E, when $d_1 = 0$, such that the gradient vector will not be zero. We define a measure for the overlapping of the two line segments $\mathbf{L}$ and $\mathbf{L}'$ (see Figure 8c),

$$d_2 = \begin{cases} \min_{\mathbf{w}\in W} \|\mathbf{q} - \mathbf{w}\|_2, & \text{for } \mathbf{L}(t) \nparallel \mathbf{L}'(t) \\ 0, & \text{for } \mathbf{L}(t) \parallel \mathbf{L}'(t), \end{cases} \tag{6}$$

$$\mathbf{q} = \left\{ \mathbf{L}(t) \mid \mathbf{L}(t) = \mathbf{L}'(t');\ t, t' \in [0,1] \right\},$$

$$W = \left\{ \mathbf{c}_1,\ \mathbf{c}_2,\ \mathbf{c}'_1,\ \mathbf{c}'_2 \right\}.$$

The above formula for d_2 measures the distance from the line segments' intersection point $\mathbf{q}$, to the nearest line segment endpoint. Note, that $\mathbf{q}$ is defined uniquely only when $d_1 = 0$, and $\mathbf{L}(t) \nparallel \mathbf{L}'(t)$. A special case when $d_1 = 0$, and $\mathbf{L}(t) \parallel \mathbf{L}'(t)$ (i.e., the line segments $\mathbf{L}$ and $\mathbf{L}'$ overlap, thus having infinitely many intersection points), is not a problem with numerical optimization algorithms, so in this case, we can simply define $d_2 = 0$. We can now include this case in the definition of $E(\mathbf{x}, \mathbf{x}')$,

$$E(\mathbf{x}, \mathbf{x}') = \begin{cases} 0, & \text{for } r + r' \leq d_1 \\ r + r' - d_1, & \text{for } 0 < d_1 < r + r' \\ r + r' + d_2, & \text{for } d_1 = 0. \end{cases}$$

In the last case the radii r, r' are added to ensure the continuity of the function E.

2.3 Objective function

In this section, we shall define a function, that corresponds to the total overlapping of objects in an elevator. Minimizing this function leads to configurations with the least overlapping. Define a set of objects $\mathbf{x} = \{\mathbf{x}^1, \ldots, \mathbf{x}^n\}$, where $\mathbf{x}^i = [\,(\mathbf{p}^i)^\top \;\; \theta^i\,]^\top$ are capsules of the same size. Define $N = \{1, \ldots, n\}$, and a rectangular box with chosen dimensions and whose center point is fixed to the origin. We shall refer to this box by $\mathbf{b} = [\,0 \;\; 0 \;\; 0\,]^\top$. Our overall objective function consists of two parts. Overlapping between all the objects is defined by summing the overlapping of every pair of objects,

$$F_A(X) = \sum_{i,j \in N, i<j} E(\mathbf{x}^i, \mathbf{x}^j),$$

and the formula,

$$F_B(X) = \sum_{i \in N} [A(\mathbf{x}^i) - A(\mathbf{x}^i, \mathbf{b})],$$

gives the sum of the object areas that are outside of box $\mathbf{b}$. Our overall objective function $F(X)$, is defined as,

$$F(X) = F_A(X) + F_B(X) = \sum_{i,j \in N, i<j} E(\mathbf{x}^i, \mathbf{x}^j) + \sum_{i \in N} [A(\mathbf{x}^i) - A(\mathbf{x}^i, \mathbf{b})].$$

All terms of the sum in F are non-negative, so F must also be non-negative. Feasible capsule configurations in the elevator are defined as points, where $F(X) = 0$. Some overlapping regions may be counted more than once with the formula for F, e.g., in the case of three capsules all overlapping each other at some point. However, the overlapping will be minimized by minimizing F, even if some overlapping areas are counted more than once.

Since functions A and E are continuous functions, $F(X)$ is also continuous. However, it's not necessarily differentiable everywhere, since it is easy to produce corners both in its graph and contours, see Figures 10 and 11. Convergence of non-linear optimization algorithms is usually obtained by assuming a continuous gradient of the objective function [11], thus the objective function here may be challenging.

The total number of terms in $F_A(X)$ is

$$\binom{n}{2} = \frac{n(n-1)}{2} = \mathcal{O}(n^2),$$

and in $F_B(X)$ there are n terms. Assuming strictly positive evaluation times for functions A and E, the computation time of $F(X)$ grows at a speed of $\mathcal{O}(n^2)$.

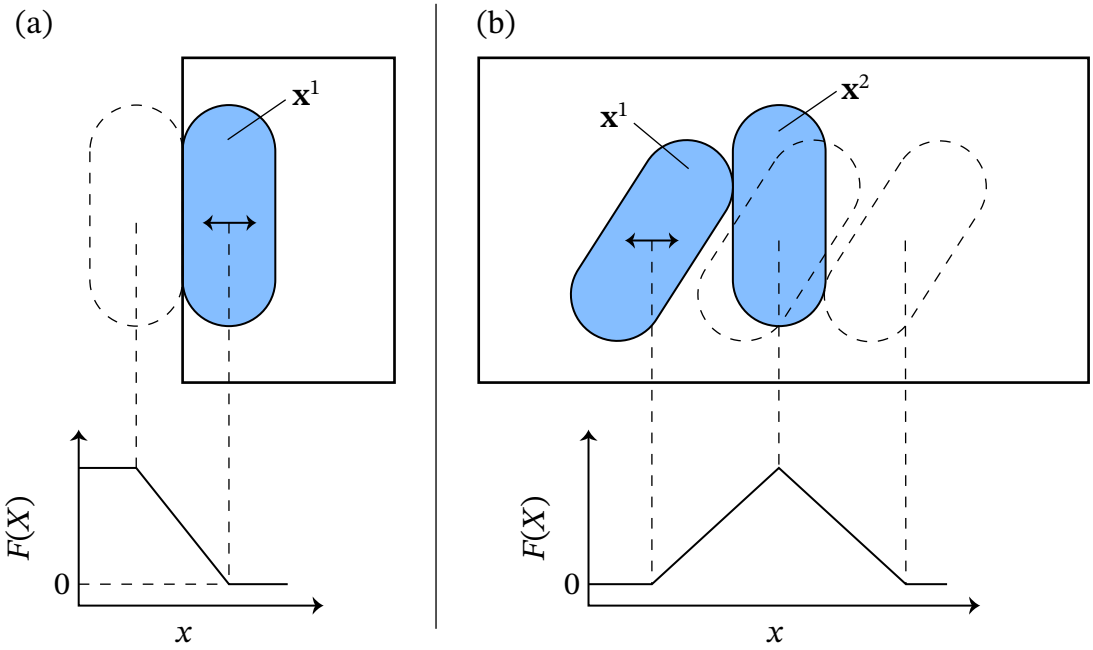


Figure 10: Examples of corners in the objective function graph. The objective function is shown with respect to x parameter of capsule $\mathbf{x}^1 = [\, x \;\; y \;\; \theta \,]^\mathsf{T}$, i.e., the capsule $\mathbf{x}^1$ is moved horizontally, capsule $\mathbf{x}^2$ is fixed in its position. In (a) the corners are due to function $A(\mathbf{x}^1, \mathbf{b})$, as the capsule crosses an edge of the box. In (b) the corners are due to function $E(\mathbf{x}^1, \mathbf{x}^2)$, as the capsule $\mathbf{x}^1$ crosses the capsule $\mathbf{x}^2$.

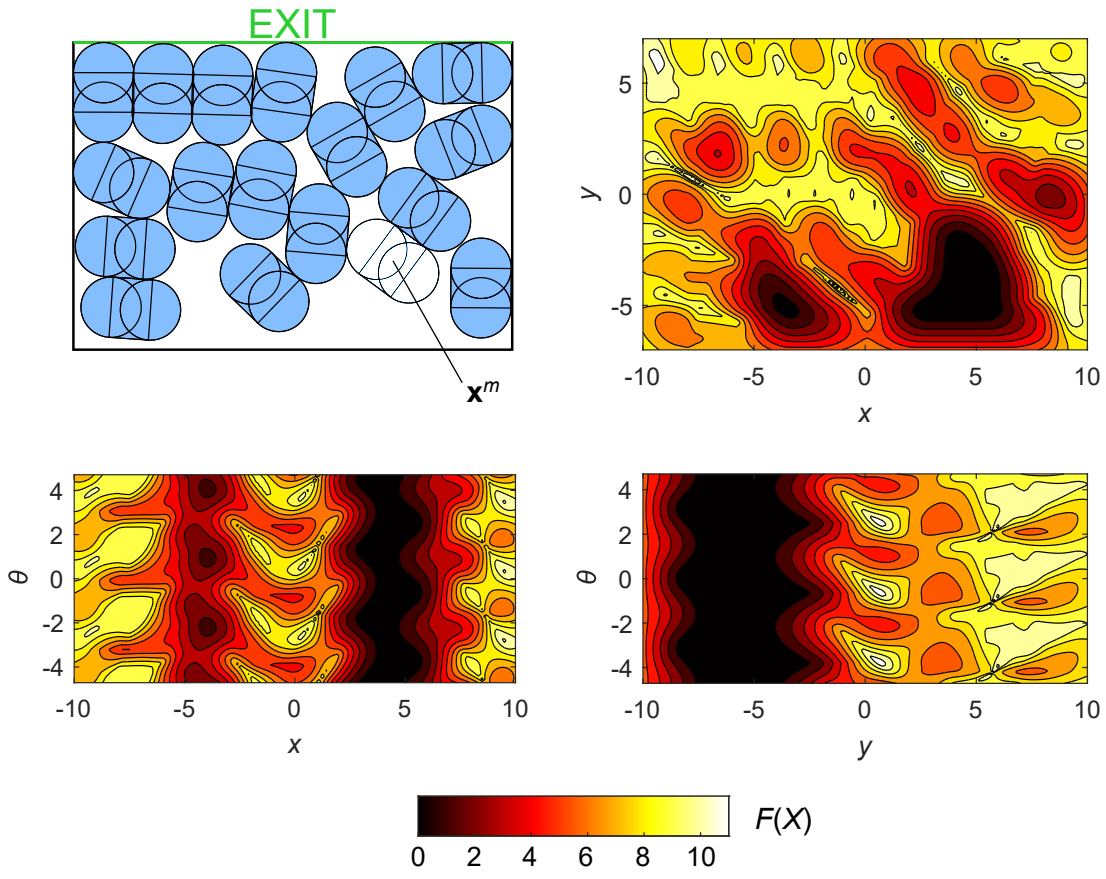


Figure 11: Objective function landscape, i.e., constant contours, with respect to the displacement of capsule $\mathbf{x}^m = [x\ y\ \theta]^\top$. E.g., in the top-left graph, parameter θ is kept fixed, as in the top right graph. Corners can be seen in all three graphs. The length unit here is 0.1 m.

3 Optimization methods

Before analyzing various optimization methods for the minimization of $F(X)$, we need to define a stopping criterion for the algorithms. A local optimum is indicated with respect to the max-norm of the objective function's gradient, $\|\nabla F(X)\|_\infty < 10^{-4}$. The max-norm is chosen because it works consistently with different numbers of objects and different elevator sizes.

Sometimes gradient-based algorithms converge slowly towards the optimum. Also, because of the challenging landscape of the objective function, an exact solution $F(X) = 0$, may be difficult to find. Thus, we relax the problem by allowing a tiny overlap at a solution. This relaxation can be tolerated because human bodies, here called passengers, are not ideal rigid objects. We define the criterion for a solution as,

$$\max\left\{ E(\mathbf{x}^i, \mathbf{s}^j),\ A(\mathbf{x}^i,\ \mathbf{b}) - A(\mathbf{x}^i) \mid i, j \in N,\ i < j \right\} < 10^{-3}.$$

Note, that the choice of the length scale also matters, because displacement parameters in vector $\mathbf{p}^i$, are of distance unit whereas the angle parameter θ^i is unitless. This means that if we scale the problem larger (e.g., change the unit from m to cm), the system becomes more sensitive with respect to the angle parameter θ^i when compared to the parameters $\mathbf{p}^i$. Scaling the optimization parameters is called diagonal scaling [11]. Diagonal scaling is optimal when the contours of the objective function near an optimum are circles with center points at the optimum. A counterexample would be a narrow valley, in which an optimum is obtained (an extreme case is shown in Chapter 4.2). Diagonal scaling often affects gradient descent's convergence [11], whereas Newton and Quasi-Newton methods do diagonal scaling by using second-order information of the objective function. Also, the conjugate gradient method corrects the diagonal scaling of the problem. Diagonal scaling would affect the contours shown in Figure 11, by scaling the graphs horizontally or vertically. In this thesis, we use 0.1 m as our unit of length, and from the Figure, we see that this choice for the scale is suitable, because there are not many narrow valleys.

Implementation quality can affect the computation times, even if the underlying mathematics remains the same. In this thesis, we use *Matlab*'s professionally implemented algorithms and our own implementations. If professionally implemented, our methods could potentially be significantly faster. Thus, we use the number of A and E function evaluations as our metric for time, which is independent of implementation qualities. Also, the results are more accurately replicable, because, with a high-level language such as *Matlab*, the computing times may even vary from run to run.

To evaluate the gradient numerically, we use finite differences. The gradient of the problem could be evaluated with a central difference or auto-derivative method for greater precision [25], but the computing times would be higher.

3.1 Simplifying the gradient evaluation

Here we simplify the objective function gradient evaluation, which is presented in [8]. The method is based on a decomposition, where gradients are calculated for each

capsule separately. The decomposition allows us to remove canceling counter terms, which are present when ∇F is evaluated directly with finite differences. The method is very effective even with a small number of objects, as we will see later in this Chapter.

We shall define a function that sums overlapping with respect to capsule $\mathbf{x}^m$. Overlapping between all objects and the capsule $\mathbf{x}^m$ is,

$$f_A^m(\mathbf{x}^m) = \sum_{i \in N \setminus \{m\}} E(\mathbf{x}^i, \mathbf{x}^m),$$

and its area outside the elevator is,

$$f_B^m(\mathbf{x}^m) = A(\mathbf{x}^m) - A(\mathbf{x}^m, \mathbf{b}).$$

Function f^m is defined as a sum of the above two formulas,

$$f^m(\mathbf{x}^m) = f_A^m(\mathbf{x}^m) + f_B^m(\mathbf{x}^m).$$

With Lemma 2 and Theorem 2, we will show how ∇F can be evaluated by using f^m. Later we shall compare the computation time between ∇F and the formula presented in Theorem 2.

Lemma 2. *The identity $F(X) = F(X \setminus \{\boldsymbol{x}^m\}) + f^m(\boldsymbol{x}^m)$ holds, for all $m \in N$.*

Proof. We prove the claim by separating terms from $F(X)$. For simplicity we'll separate the terms regarding $\mathbf{x}^n$, thus proving $F(X) = F(X \setminus \{\mathbf{x}^n\}) + f^n(\mathbf{x}^n)$. The proof then holds for any index by reordering the set X. Define $X_0 = X \setminus \{\mathbf{x}^n\}$, and $N_0 = \{1, \dots, n-1\}$. The formula for $F_A(X)$ can be rewritten as,

$$\begin{aligned} F_A(X) = \sum_{i,j \in N, i<j} E(\mathbf{x}^i, \mathbf{x}^j) &= \sum_{j=2}^{n} \sum_{i=1}^{j-1} E(\mathbf{x}^i, \mathbf{x}^j) \\ &= \sum_{j=2}^{n-1} \sum_{i=1}^{j-1} E(\mathbf{x}^i, \mathbf{x}^j) + \sum_{i=1}^{n-1} E(\mathbf{x}^i, \mathbf{x}^n) \\ &= \sum_{i,j \in N_0, i<j} E(\mathbf{x}^i, \mathbf{x}^j) + \sum_{i \in N \setminus \{n\}} E(\mathbf{x}^i, \mathbf{x}^n) \\ &= F_A(X_0) + f_A^n(\mathbf{x}^n). \end{aligned} \tag{7}$$

$F_B(X)$ can be written similarly,

$$\begin{aligned} F_B(X) &= \sum_{i \in N} [A(\mathbf{x}^i, \mathbf{b}) - A(\mathbf{x}^i)] \\ &= \sum_{i \in N_0} [A(\mathbf{x}^i) - A(\mathbf{x}^i, \mathbf{b})] + [A(\mathbf{x}^n) - A(\mathbf{x}^n, \mathbf{b})] \\ &= F_B(X_0) + f_B^n(\mathbf{x}^n). \end{aligned} \tag{8}$$

Using Equations (7) and (8), we get,

$$\begin{aligned} F(X) &= F_A(X) + F_B(X) \\ &= F_A(X_0) + f_A^n(\mathbf{x}^n) + F_B(X_0) + f_B^n(\mathbf{x}^n) \\ &= F(X_0) + f^n(\mathbf{x}^n), \end{aligned}$$

which proves the claim. □

Theorem 2. *The identity*

$$\nabla F(X) = \left[\nabla f^1(\boldsymbol{x}^1)^\top \quad \ldots \quad \nabla f^n(\boldsymbol{x}^n)^\top\right]^\top \tag{9}$$

holds.

Proof. Write $\mathbf{x}^m = [\, x_1^m \;\; x_2^m \;\; x_3^m \,]^\top$, and consider a partial derivative of F, with respect to the component x_i^m. We define a displaced object, $\mathbf{y}^m$, whose elements are defined as,

$$y_j^m = \begin{cases} x_j^m, & \text{for } j \neq i, \\ x_j^m + h, & \text{for } j = i, \end{cases}$$

where $h > 0$. Also, define $X_0 = X \setminus \{\mathbf{x}^m\}$ and $Y = X_0 \cup \{\mathbf{y}^m\}$. Then we prove the claim by writing the partial derivatives and using Lemma 2 as follows,

$$\begin{aligned} \frac{\partial F(X)}{\partial x_i^m} = \lim_{h\to 0} \frac{F(Y) - F(X)}{h} &= \lim_{h\to 0} \frac{F(X_0) + f^m(\mathbf{y}^m) - F(X_0) - f^m(\mathbf{x}^m)}{h} \\ &= \lim_{h\to 0} \frac{f^m(\mathbf{y}^m) - f^m(\mathbf{x}^m)}{h} \\ &= \frac{\partial f^m(\mathbf{x}^m)}{\partial x_i^m}. \end{aligned} \tag{10}$$

□

In Equation (10), terms $F(X_0)$ cancel in the computation of partial derivatives, when $F(X)$ is used. The unnecessary computations add up to a significant proportion when the number of objects is large, thus making the right-hand side of Equation (9) faster to evaluate. Next, we shall compare the evaluation speed of both sides of Equation (9).

On the left-hand side of (9), we're evaluating the gradient of $F(X)$ directly. We have n objects, and each object has three degrees of freedom. Thus, the gradient of $F(X)$ for all the parameters with forward differences requires $3n + 1$ function evaluations. Since $F(X)$ is computed in $\mathcal{O}(n^2)$ time (see Chapter 2.3), the gradient is evaluated in $\mathcal{O}(n^2)(3n + 1) = \mathcal{O}(n^3)$ time.

Consider now the expression on the right-hand side of (9). The computation time of function f^m grows at a linear rate, $\mathcal{O}(n)$. Computation of $\nabla f^m(\mathbf{x}^m)$ requires four function evaluations, thus ∇f^m is evaluated in $\mathcal{O}(n) \cdot 4 = \mathcal{O}(n)$ time. For ∇F, we need ∇f^m, for all $m \in N$. Thus, the evaluation time grows at a rate of $\mathcal{O}(n) \cdot n = \mathcal{O}(n^2)$.

The reduction in evaluation time, from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$, is significant. A more detailed study on the evaluation times is provided in [8], where we noted that at least 50 % reduction in computation time is reached with just five capsules, and at least 75 % reduction is reached with 11 capsules. We shall use the right-hand side of (9) with all algorithms presented below.

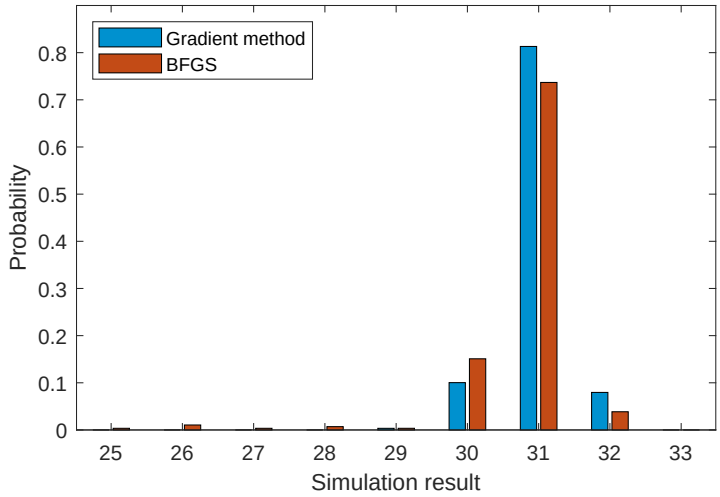


Figure 12: Obtained distribution (normalized frequency) of the result of a simulation, with a box size of 2350×1700 mm. The gradient method finds solutions with 32 passengers (global optimum) more often than BFGS.

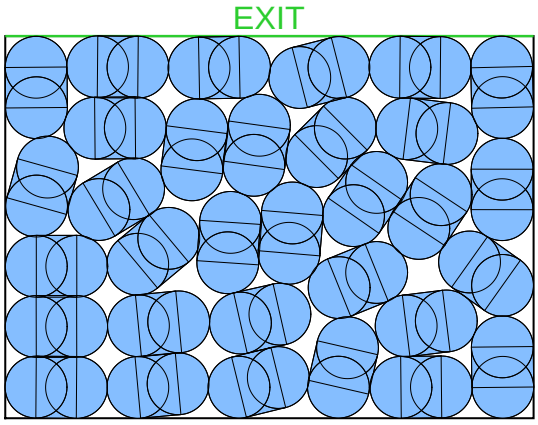


Figure 13: 32 passengers in a 2350×1700 mm elevator. The arrangement is computed by the BFGS.

3.2 Capsule packing problem with gradient method and BFGS

Familiar non-linear optimization methods include the gradient method and BFGS. Define all the optimization parameters as a single column vector, $\mathbf{x}_k = [\,(\mathbf{x}_k^1)^\top \;\; \ldots \;\; (\mathbf{x}_k^n)^\top\,]^\top$, where $k \in \mathbb{N}$ is an iteration index. In Algorithms 1 and 2 we have a gradient method and a lightly modified BFGS, where gradient method iteration is used when the Hessian inverse update becomes near singular. In both methods, we use *Matlab*'s optimization algorithm for the line search, which is an exact line search method.

We now apply these methods for the capsule packing problem described in Chapter 2: we add capsules to the box one by one until the optimizer cannot solve the packing problem implying that the box is full. Simulation results are presented in Figure

Algorithm 1 Gradient method

inputs: initial conditions $\mathbf{x}_1$
$k \leftarrow 0$.
while *neither stopping criteria is satisfied* **do**
  $k \leftarrow k + 1$
  $\alpha_k \leftarrow \arg\min_\alpha \left\{F(\mathbf{x}_k - \alpha \nabla F(\mathbf{x}_k))\right\}$
  $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k - \alpha_k \nabla F(\mathbf{x}_k)$
end while
outputs: $\mathbf{x}_{k+1}$

Algorithm 2 BFGS

inputs: initial conditions $\mathbf{x}_1$
$k \leftarrow 0$
$B_0^{-1} \leftarrow I$
$\mathbf{x}_0 \leftarrow \mathbf{x}_1$
while *neither stopping criteria is satisfied* **do**
  $k \leftarrow k + 1$
  $\mathbf{s}_k \leftarrow \mathbf{x}_k - \mathbf{x}_{k-1}$
  $\mathbf{y}_k \leftarrow \nabla F(\mathbf{x}_k) - \nabla F(\mathbf{x}_{k-1})$
  ▷ If-statement in case the update formula for B^{-1} is near singular
  if $(\mathbf{s}_k^\top \mathbf{y}_k)^2 < 10^{-4}$ **then**
  ▷ A gradient method step
    $B_k^{-1} \leftarrow B_{k-1}^{-1}$
    $\alpha_k \leftarrow \arg\min_\alpha \left\{F(\mathbf{x}_k - \alpha \nabla F(\mathbf{x}_k))\right\}$
    $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k - \alpha_k \nabla F(\mathbf{x}_k)$
  else
  ▷ Hessian inverse update and a Quasi-Newton step
    $$B_k^{-1} \leftarrow B_{k-1}^{-1} + \frac{\left(\mathbf{s}_k^\top \mathbf{y}_k + \mathbf{y}_k^\top B_{k-1}^{-1} \mathbf{y}_k\right)\left(\mathbf{s}_k \mathbf{s}_k^\top\right)}{\left(\mathbf{s}_k^\top \mathbf{y}_k\right)^2} - \frac{B_{k-1}^{-1} \mathbf{y}_k \mathbf{s}_k^\top + \mathbf{s}_k \mathbf{y}_k^\top B_{k-1}^{-1}}{\mathbf{s}_k^\top \mathbf{y}_k}$$
    $\alpha_k \leftarrow \arg\min_\alpha \left\{F(\mathbf{x}_k - \alpha B_k^{-1} \nabla F(\mathbf{x}_k))\right\}$
    $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k - \alpha_k B_k^{-1} \nabla F(\mathbf{x}_k)$
  end if
end while
outputs: $\mathbf{x}_{k+1}$

12, where we have simulated elevator packing with capsules with different initial conditions, as described in Chapter 2.1. From the Figure, we can see that the obtained capacity differs from run to run, thus, the optimization methods are not always able to find a global optimum. An optimized setting is shown in Figure 13. Numerical simulations are handled in more detail in Chapter 3.5.

3.3 Cyclic placement method

The cyclic placement method, CPM, was presented in [8], where preliminary results for the capsule packing problem were obtained. In the method, the problem is decomposed into subproblems, in which we optimize the placement of a single capsule at a time, and keep other capsules fixed. Once the placement of a capsule is optimized, we move on to the next object. We usually need to optimize the objects' placements many times to reach an optimum. Since we're optimizing the placement of a single capsule, we can use f^m as the objective function. The CPM is presented in Algorithm 3.

In [8], the individual optimization tasks (line 7 in Algorithm 3) were solved by using *Matlab*'s `active-set` algorithm. We use the same algorithm here, with a limit of three iterations, as was also done in [8].

Algorithm 3 CPM

1: **inputs:** $\mathbf{x}_1^i, \forall i \in N$
2: $k \leftarrow 0$
3: **while** *both stopping criteria are not satisfied* **do**
4: $\quad k \leftarrow k + 1$
$\quad \triangleright$ *Modulo operator:* $a \,\%\, n := a - n\lfloor a/n \rfloor$
5: $\quad m \leftarrow (k-1) \,\%\, n + 1$
6: $\quad \mathbf{x}_{k+1}^i = \mathbf{x}_k^i, \ \forall i \in N \setminus \{m\}$
7: $\quad \mathbf{x}_{k+1}^m = \arg\min_{\mathbf{x}} f^m(\mathbf{x})$
8: **end while**
9: **outputs:** $\mathbf{x}_{k+1}^i, \forall i \in N$

The CPM is a special case of the block coordinate method [9]. The block coordinate method has various modifications. These include random indexing [26], acceleration step [16], Hooke and Jeeves method [16], and Gauss-Southwell rule [27]. The performance of the first three methods are more or less based on a heuristic than on theory [11], and testing these in practice with the packing problem shows a negligible change in the convergence speed, compared to Algorithm 3.

With the Gauss-Southwell rule, we would choose and optimize the capsule with the largest gradient (e.g., compared by Euclidian norm). Thus, we would need to evaluate the gradient of all capsules at each iteration. The modification can accelerate the convergence if the structure of the problem allows the gradients to be quickly computable [27]. However, our objective function does not have the structure required,

hence the Gauss-Southwell rule should perform worse than a regular block coordinate method. This claim is verified by simulations.

3.4 Global optimization and comparison of methods

Comparing optimization methods is not a trivial task, especially when the objective function contains many local optima and multiple solutions. Some algorithms may find feasible solutions more often than others, but on the other hand, it might consume more time. In this thesis, we try to find the global optimum (most capsules with $F(X) = 0$) by trial and error starting from different initial conditions. As a measure of the speed of global optimization, we use the expected time for finding a global optimum.

Assume simulations with some algorithm and a problem (e.g., fit most capsules into a given box). Assume also, that we record the outcome of each simulation (global optimum obtained or not), and the time the algorithm takes to finish for each simulation. In this thesis the individual simulations are statistically independent, thus we can study the simulation data with simple probability.

We shall index the simulation trials, $S = \{1, \ldots, s\}$, and define computation time for each simulation, t_s, $\forall s \in S$. We split the set S to subsets P and Q: in P we have such simulation indices, whose outcome is a global optimum, and Q is the complement of P; $Q = S \setminus P$. We shall define a sample probability, $p = |P|/|S|$, and its complement, $q = 1 - p$. We shall define separate sets of computing times: $T_P = \{t_s \mid s \in P\}$, and $T_Q = \{t_s \mid s \in Q\}$. We denote $\overline{T}_P$ and $\overline{T}_Q$ to be sample means for the above two sets, respectively.

Table 3: Scenarios where we find a global optimum.

Number of simulations until a global optimum is obtained	Probability	Expected time	Probability × Expected time
1	p	$\overline{T}_P$	$p\,\overline{T}_P$
2	qp	$\overline{T}_Q + \overline{T}_P$	$qp\,(\overline{T}_Q + \overline{T}_P)$
3	q^2p	$2 \cdot \overline{T}_Q + \overline{T}_P$	$q^2p\,(2 \cdot \overline{T}_Q + \overline{T}_P)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$i+1$	q^ip	$i \cdot \overline{T}_Q + \overline{T}_P$	$q^ip\,(i \cdot \overline{T}_Q + \overline{T}_P)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
Sum	1		$\overline{T}_Q(1/p - 1) + \overline{T}_P$

When measuring the time for finding a global optimum, we stop the search once we find a global optimum. I.e., at the final trial, we find a global optimum, and in the preceding trials we do not find one. In Table 3 we have listed all possibilities for the number of simulation trials until a global optimum is found. The expected time for

finding a global optimum is thus,

$$\sum_{i=0}^{\infty} q^i p \left(i \overline{T}_Q + \overline{T}_P\right) = \overline{T}_Q \left(\frac{1}{p} - 1\right) + \overline{T}_P.$$

The sum is simplified with the following formula,

$$\sum_{i=0}^{\infty} i\, q^i = \frac{q}{(1-q)^2}, \text{ for } |q| < 1,$$

which can be obtained by differentiating the formula for geometric series [28].

3.5 Numerical simulations, convergence analysis

We ran simulations with the three optimization methods, with the capsule packing problem described in Chapter 2.1, which models an elevator car being filled with passengers. The results can be seen in Table 4, where we have expected global solving times (see Chapter 3.4) for each method. We see that the gradient method and BFGS produce similar expected global solving times with all three elevator sizes. The CPM is significantly worse compared to the other two: with the medium-sized elevator, the problem is solved in a significantly longer time, and we could not find even a single global solution with the largest elevator size.

The distributions of solutions are presented in Figure 14. With the two bigger elevator sizes, the CPM with *Matlab*'s solver finds solutions far more rarely compared to the other two. Simulation data can be seen in Appendix C.

Table 4: Expected global optimization time of a capsule packing problem. Computing times are shown in a unit of million area calculations. The fastest algorithm for each problem is shown in bold.

Elevator size (mm)	Capacity		Gradient method	BFGS	CPM (*Matlab*'s solver)
	ISO-std.	Simulations			
1350 × 1400	10	15	**57.6**	89.8	65.5
2000 × 1400	17	22	28.9	**19.5**	84.6
2350 × 1700	26	32	**411**	474	–

The CPM with *Matlab*'s optimizer does not perform well. Nevertheless, in the next Chapter, we will show that this state of affairs is not the final truth of the things. We will there define a CPM with a different optimizer that outperforms all the methods described in this Chapter. Capacity in our simulations is systematically greater than in the ISO standard. This is because any personal space is ignored in the simulations, and also our chosen dimensions for passengers may be smaller than in reality.

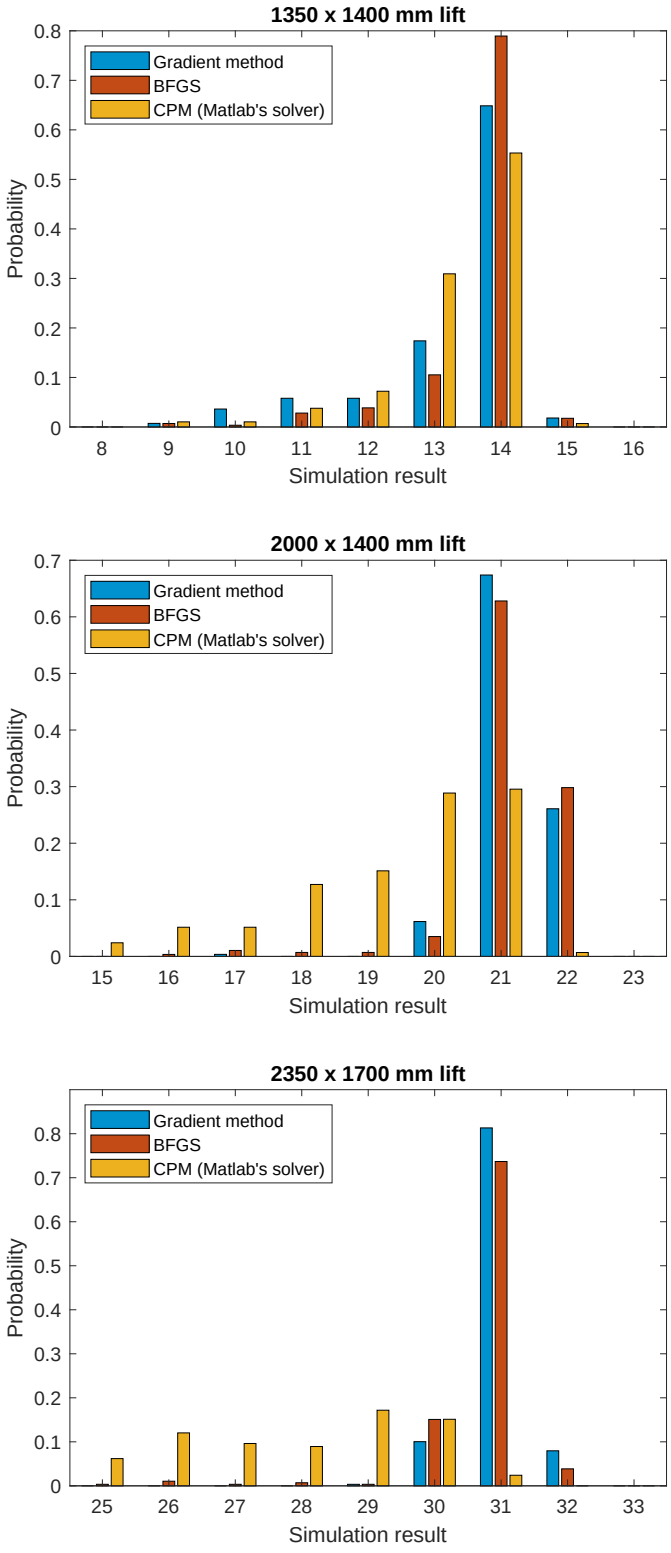


Figure 14: Distribution of the number of fitted objects, for each optimization method.

4 Gradient method with quadratic line search

4.1 Definition of the quadratic line search

In the following analysis, let $f : \mathbb{R}^n \mapsto \mathbb{R}$ be any function such that $f \in C^1$. The quadratic line search (QLS) implementation exploits the already evaluated gradient and function value of f at a point $\mathbf{x}_k$ [10]. With this information, only one function evaluation is needed for fitting a quadratic model, from which we obtain an approximation for the optimal step length. The method is computationally light and gives the exact optimal step with quadratic functions, which we shall also show. Later we also make use of the method associated with CPM to solve packing problems and see how it compares to the methods shown in Chapter 3.

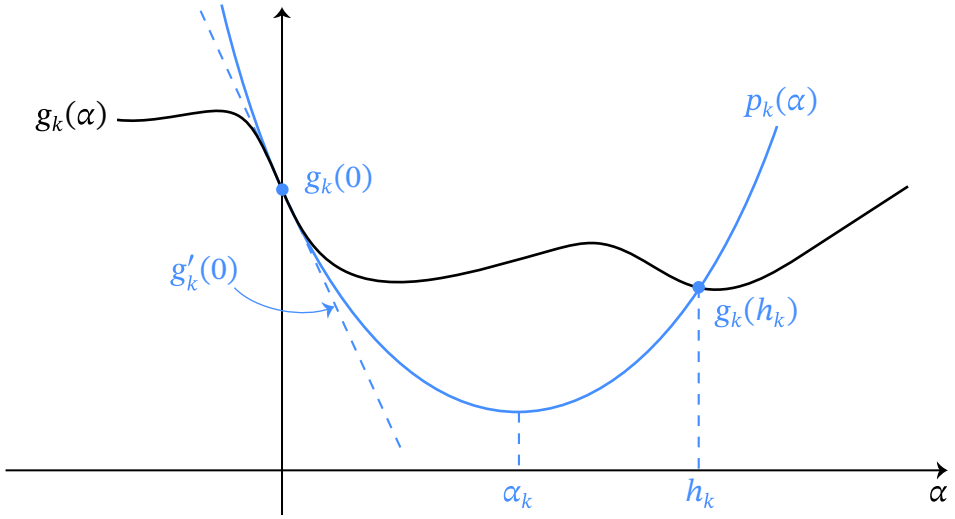


Figure 15: Example of a quadratic approximation of a function.

Define the line search function,

$$g_k(\alpha) = f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)). \tag{11}$$

We shall construct a parabola $p_k(\alpha)$, as in Figure 15, with some chosen $h_k > 0$. The parabola is defined by the following properties: $p_k(0) = g_k(0)$, $p_k'(0) = g_k'(0)$ and $p_k(h_k) = g_k(h_k)$. The parabola satisfying the first two conditions is,

$$p_k(\alpha) = g_k(0) + \alpha\, g_k'(0) + \frac{\alpha^2}{2} p_k''(0); \tag{12}$$

and since $p_k(h_k) = g_k(h_k)$, we have,

$$g_k(h_k) = g_k(0) + h_k\, g_k'(0) + \frac{h_k^2}{2} p_k''(0), \tag{13}$$

if and only if

$$\frac{1}{2} p_k''(0) = \frac{1}{h_k^2}\big(g_k(h_k) - g_k(0) - h_k\, g_k'(0)\big). \tag{14}$$

We next optimize $p_k(\alpha)$ in Equation (12), by solving the zero value of the derivative. The optimum α_k is defined by,

$$\alpha_k = -\frac{g_k'(0)}{p_k''(0)}.$$

Point α_k is the minimum point of the parabola when $p_k''(0) > 0$. Using (14),

$$\alpha_k = \frac{h_k^2}{2} \cdot \frac{-g_k'(0)}{g_k(h_k) - g_k(0) - h_k\, g_k'(0)}. \tag{15}$$

The formula above is presented in [11, 10], and the method is known as the quadratic interpolation line search method. Next, using Equation (11), we obtain $g_k(0) = f(\mathbf{x}_k)$ and,

$$g_k'(\alpha) = -\nabla f\big(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)\big)^\top \nabla f(\mathbf{x}_k) \implies g_k'(0) = -\|\nabla f(\mathbf{x}_k)\|^2. \tag{16}$$

Substituting these to Equation (15) gives [10],

$$\alpha_k = \frac{h_k^2}{2} \cdot \frac{\|\nabla f(\mathbf{x}_k)\|^2}{g_k(h_k) - f(\mathbf{x}_k) + h_k \|\nabla f(\mathbf{x}_k)\|^2}. \tag{17}$$

With the gradient method, numerical values for $\nabla f(\mathbf{x}_k)$ and $f(\mathbf{x}_k)$ are evaluated before the line search. Thus, only $g_k(h_k)$, is needed to obtain the step size α_k. In the simulations, a normalized parameter $h_k = u_k / \|\nabla f(\mathbf{x}_k)\|$, is used, because it is numerically more stable. With this parametrization, Equation (17) becomes,

$$\alpha_k = \frac{u_k^2}{2} \cdot \left[f\left(\mathbf{x}_k - u_k \frac{\nabla f(\mathbf{x}_k)}{\|\nabla f(\mathbf{x}_k)\|} \right) - f(\mathbf{x}_k) + u_k \|\nabla f(\mathbf{x}_k)\| \right]^{-1}.$$

Quadratic objective function

The quadratic line search method (QLS) has an interesting property: if $f(\mathbf{x})$ is quadratic and strictly convex, the method solves the line search problem exactly with any $h_k > 0$ (not necessarily with the above definition given by $u_k / \|\nabla f(\mathbf{x}_k)\|$). In this case, all line search problems are also quadratic, so we shall define $g_k(\alpha)$ as,

$$g_k(\alpha) = g_k(0) + \alpha\, g_k'(0) + \frac{\alpha^2}{2} g_k''(0),$$

with $g_k(0), g_k'(0), g_k''(0) \in \mathbb{R}$. Since f is strictly convex, we have $g_k''(0) > 0$. The minimum of g_k is at,

$$\arg\min_{\alpha \in \mathbb{R}} g_k(\alpha) = -\frac{g_k'(0)}{g_k''(0)}. \tag{18}$$

Using Equation (15), we have the following,

$$\begin{aligned}\alpha_k &= \frac{h_k^2}{2} \cdot \frac{-g_k'(0)}{g_k(h_k) - g_k(0) - h_k\, g_k'(0)} \\ &= \frac{h_k^2}{2} \cdot \frac{-g_k'(0)}{\left[g_k(0) + h_k\, g_k'(0) + \frac{h_k^2}{2} g_k''(0)\right] - g_k(0) - h_k\, g_k'(0)} \\ &= -\frac{g_k'(0)}{g_k''(0)}. \end{aligned} \tag{19}$$

Comparing Equations (18) and (19), we see that the QLS produces the minimum point exactly, with any $h_k > 0$. An example quadratic function is presented later in Chapter 4.2.

Convergence analysis

Assume that ∇f is Lipschitz-continuous, i.e., there is $L \geq 0$, for which,

$$\forall\, \mathbf{v}, \mathbf{w} : \ \|\nabla f(\mathbf{v}) - \nabla f(\mathbf{w})\| \leq L\|\mathbf{v} - \mathbf{w}\|.$$

In [29] it reads that the gradient method converges, if $0 < \alpha_k < 2/L$, $\forall k \geq 0$. Here, the left-hand side inequality holds (see Lemma 3, in Appendix D),

$$\alpha_k = \frac{\|\nabla f(\mathbf{x}_k)\|^2}{p_k''(0)} \geq \frac{1}{L} > 0.$$

The inequality $\alpha_k < 2/L$ holds at least in a neighborhood of a local optimum point, but not necessarily globally. More detailed convergence analysis is out of the scope of this thesis.

4.2 Numerical simulations

Simple example problems

In this Chapter, we shall implement a gradient method and compare three line search methods: exact line search, and the QLS with two choices for h_k. The used parameters are $h_{k,1} = 5 \cdot 10^{-2}/\|\nabla f(\mathbf{x}_k)\|$, and, $h_{k,2} = 5 \cdot 10^{-3}/\|\nabla f(\mathbf{x}_k)\|$. The exact line search method is implemented by using *Matlab*'s optimization algorithm. In this Chapter, we use autodifferentiation for gradient evaluation.

As a time unit, we use step count, because it is independent of the implementation performance between the methods. However, the QLS only requires one objective function evaluation, whereas we let multiple evaluations with the exact line search method. Thus, the exact line search method is expected to consume more time per step compared to the QLS.

In the examples, we use three test functions (see Table 5), all of which have a single minimum point, and no maxima. Functions f^A and f^B are convex and f^C is not convex.

The function f^A is a quadratic function, and we see that the QLS does not differ from the exact line search method. From Figure 16 it can be seen that the methods "zigg-zagg" their way closer to the optimum, and also, the linear convergence rate of all the methods is seen.

With functions f^B and f^C, the QLS performs similarly or better than the exact line search, see Figures 17 and 18. In both cases, the QLS takes shorter steps at the beginning and later converges faster. The function f^C is Rosenbrock's "banana function" [30], where both methods show tight "zigg-zagging". This behavior is typical for gradient methods at this problem.

With functions f^B and f^C, we can see that the choice of h_k matters with non-quadratic functions. With the function f^B, the smaller $h_{k,2}$ was faster, whereas with the function f^C, the larger $h_{k,1}$ was the fastest. The first few steps are similar with both h_k, and differences become more prominent at later steps.

Table 5: Test functions.

Function	Initial point, $\mathbf{x}_0$	Optimum, $\mathbf{x}_*$	$f^i(\mathbf{x}_*)$
$f^A(x_1, x_2) = x_1^2 + 8x_2^2 - 0.5x_1x_2$	$(10,\ -10)$	$(0, 0)$	0
$f^B(x_1, x_2) = \exp(x_1 + 2x_2 - 0.1) + \exp(-x_1 - 0.2) + \exp(x_1 - 2x_2 - 0.1)$	$(-2,\ -4)$	$(-0.3966,\ 0)$	2.434
$f^C(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$	$(-2,\ 0)$	$(0, 0)$	0

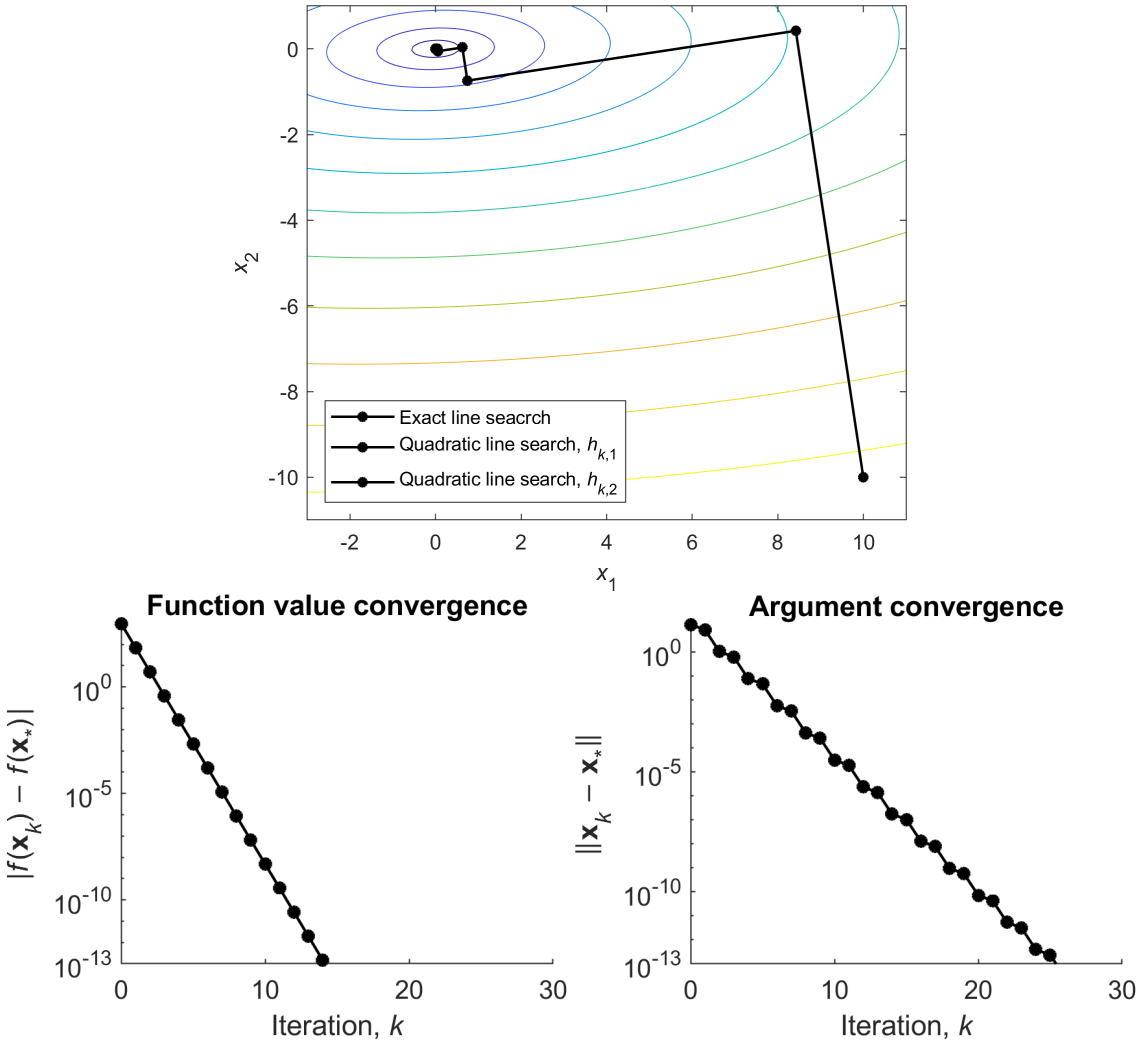


Figure 16: Function f^A. The line search methods work in the same way with any quadratic function.

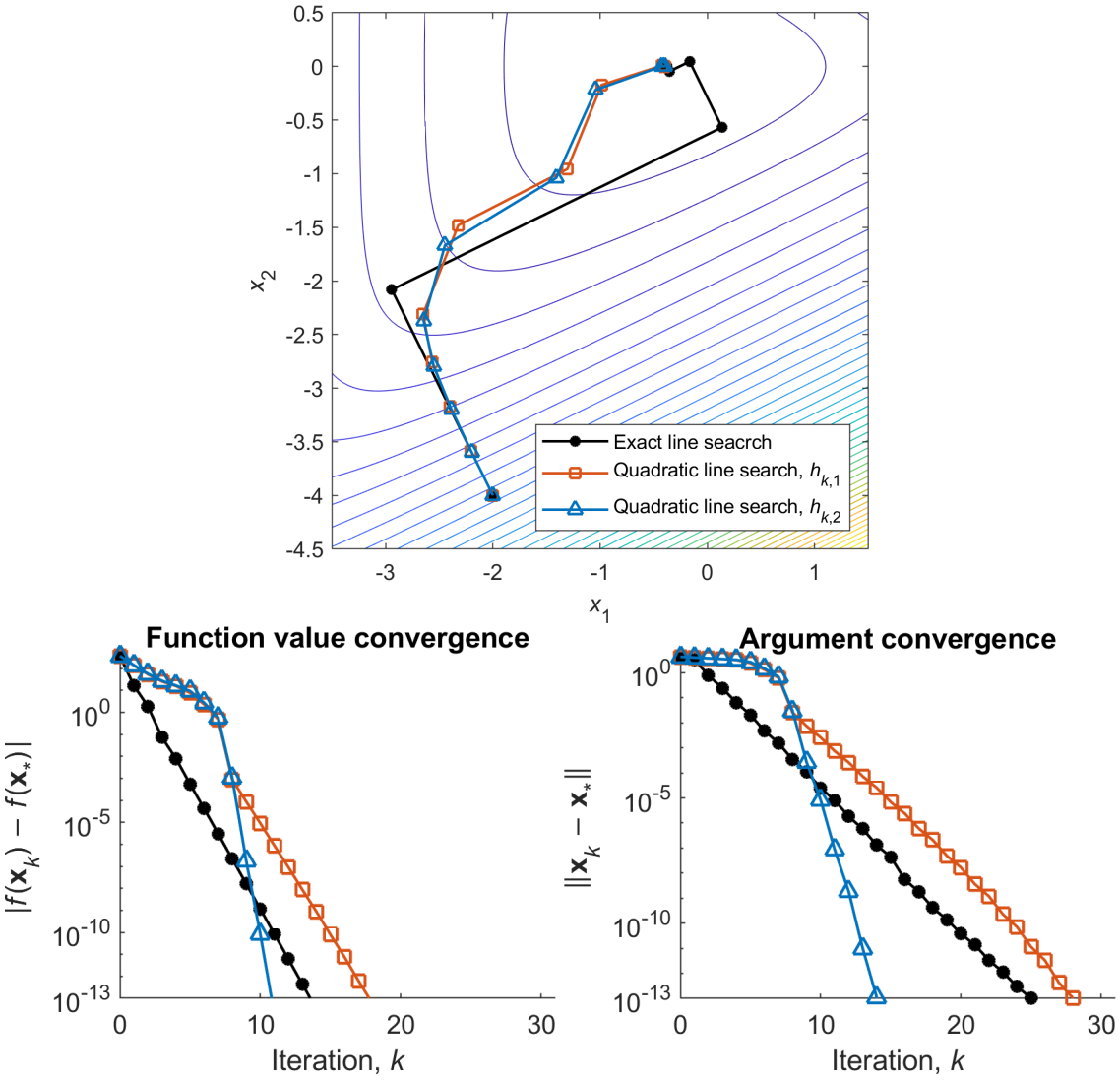


Figure 17: Function f^B. The QLS is coping well against the exact line search method.

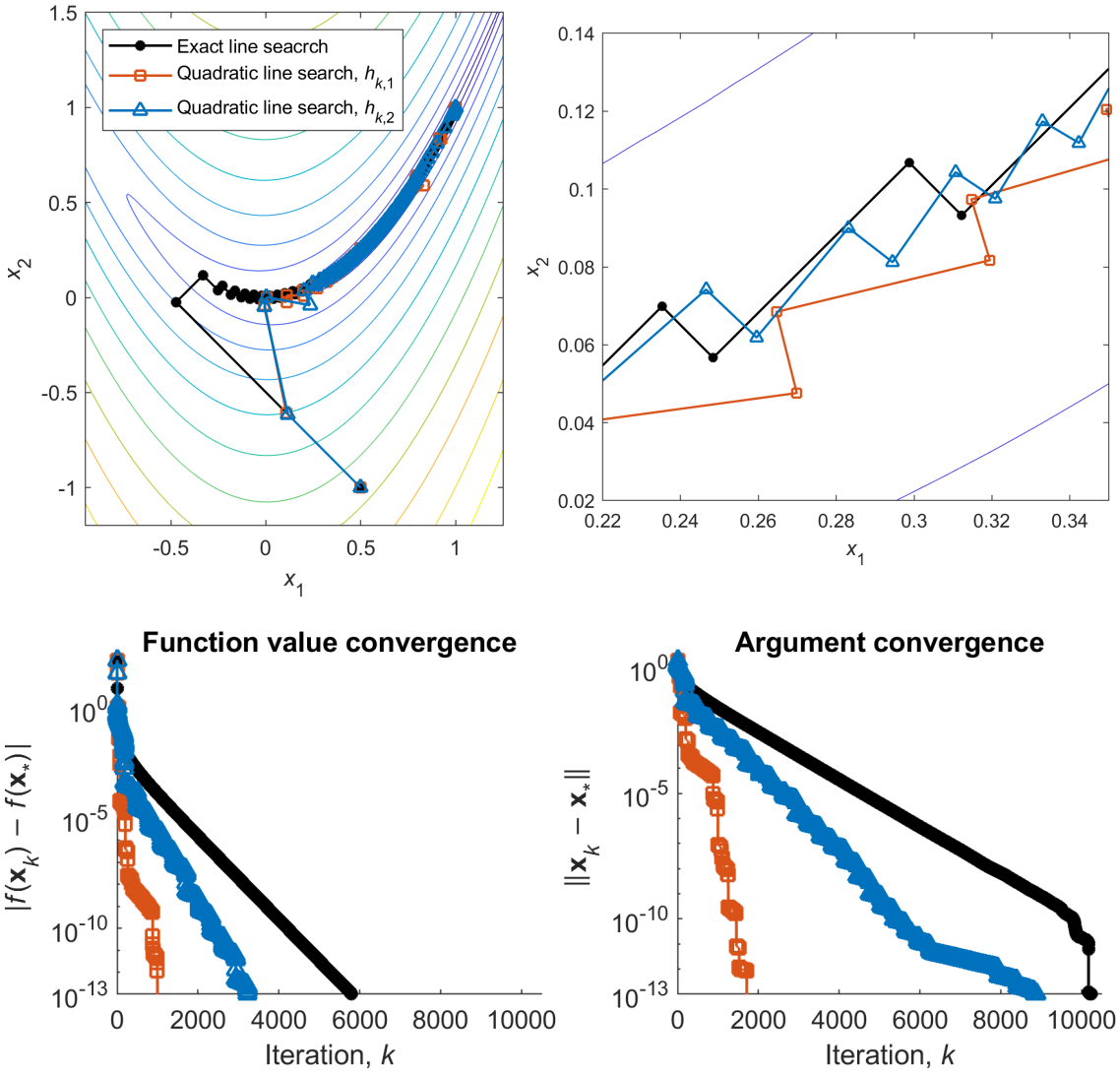


Figure 18: The function f^C, i.e., the Rosenbrock function.

Capsule packing problem

Now we shall construct a method for the capsule packing problem. The method is a gradient method with QLS method, with added random search and acceleration step. We use the cyclic placement method, CPM (see Chapter 3.3), in which we optimize placement of a single capsule at a time, with f^m as the objective function. We compare this method with the previously shown results, in Chapter 3.5.

With the CPM, the objective function is different at every iteration, so methods which use data from previous iterations are not applicable. Fortunately, the gradient method with QLS does not require data from previous iterations, so it can be very effective at this problem. Our objective function is rather heavy to evaluate, so the computationally light QLS should have an advantage.

See Algorithm 4 for the complete algorithm that is used with the CPM. The CPM is presented in Algorithm 3, in Chapter 3.3.

We redefine the parameter h_k to be

$$h_k = \frac{M}{\sqrt{k/n}\,\|\nabla f^m(\mathbf{x}_k^m)\|}.$$

We shall define a random search [31], and acceleration step [16], which in this case accelerates the convergence and also help to find global optima. We use a limit for the line search, to prevent $\alpha_k > L/2$, and also to prevent objects from, e.g., crossing each other in the elevator. We choose to limit the step to 5 cm, i.e., $\|\mathbf{x}_{k+1} - \mathbf{x}_k\| < 0.5 = M$. The parameter h_k is decreasing but maintains numerical stability.

With random search, we form a hypersphere around the current point $\mathbf{x}_k$, sample random points from inside the hypersphere, and choose the point where the objective function value is the smallest. Studying other random search methods and heuristics is out of the scope of this thesis.

The acceleration step is as follows. After a step of gradient method, we sample a number, $\rho_k \in \{1, 2\}$, which gets value 1 with 70 % probability, and 2 with 30 % probability. We modify the gradient method step to $\mathbf{x}_{k+1}^m = \mathbf{x}_k^m - \rho_k \alpha_k \nabla f^m(\mathbf{x}_k^m)$.

Table 6: Expected global optimization time of a capsule packing problem. Computing times are shown in a unit of million area calculations. The fastest algorithm for each problem is shown in bold.

Elevator size (mm)	Capacity ISO-std.	Capacity Simulations	Gradient method	BFGS	CPM (QLS)
1350 × 1400	10	15	136	**72.8**	73.2
2000 × 1400	17	22	31.9	17.3	**5.8**
2350 × 1700	26	32	410	648	**158**

The results are seen in Table 6. With the two larger elevators, the expected time for finding a global optimum is significantly faster with the CPM using the QLS compared to the other methods. Distributions of simulation results are shown in Figure 19, where we see that the CPM with QLS finds better optima noticeably more often compared

Algorithm 4 Gradient method with QLS, to be used with the CPM.

inputs: *iteration index* k, *capsule index* m, *initial placement* $\mathbf{x}_k^m$.
▷ Limit for step length
$M \leftarrow 0.5$
▷ Random search
Pick a random $\mathbf{x}_k^r$ uniformly from $B(\mathbf{x}_k^m, M)$
if $f^m(\mathbf{x}_k^r) \leq f^m(\mathbf{x}_k^m)$ **then**
    $\mathbf{x}_{k+1}^m \leftarrow \mathbf{x}_k^r$
    outputs: $\mathbf{x}_{k+1}^m$
    end algorithm
end if
▷ Gradient method and QLS
if $\nabla f^m(\mathbf{x}_k^m) = 0$ **then**
    $\mathbf{x}_{k+1}^m \leftarrow \mathbf{x}_k^m$
    outputs: $\mathbf{x}_{k+1}^m$
    end algorithm
end if
$h_k \leftarrow \dfrac{M}{\sqrt{k/n}\,\|\nabla f^m(\mathbf{x}_k^m)\|}$
$g_k(h_k) \leftarrow f^m(\mathbf{x}_k^m - h_k \nabla f^m(\mathbf{x}_k^m))$
$p_k''(0) \leftarrow \dfrac{2}{h_k^2}\big(g_k(h_k) - f^m(\mathbf{x}_k^m) + h_k\|\nabla f(\mathbf{x}_k^m)\|^2\big)$
if $p_k''(0) > 0$ **then**
    $\alpha_k \leftarrow \|\nabla f^m(\mathbf{x}_k^m)\|^2 / p_k''(0)$
else
    $\alpha_k \leftarrow h_k$
end if
▷ Acceleration step
Pick a random $\rho_k \in \{1, 2\}$, so that, $\mathbb{P}(\rho_k = 1) = 70\%$.
▷ Evaluate the new point
$\mathbf{x}_{k+1}^m \leftarrow \mathbf{x}_k^m - \rho_k \alpha_k \nabla f^m(\mathbf{x}_k^m)$
▷ Step limit
if $\|\mathbf{x}_{k+1}^m - \mathbf{x}_k^m\| > M$ **then**
    $\mathbf{d} \leftarrow (\mathbf{x}_{k+1}^m - \mathbf{x}_k^m)/\|\mathbf{x}_{k+1}^m - \mathbf{x}_k^m\|$
    $\mathbf{x}_{k+1}^m \leftarrow \mathbf{x}_k^m + M\mathbf{d}$
end if
outputs: $\mathbf{x}_{k+1}^m$

to *Matlab*'s exact line search method. The CPM with the QLS finds global optima less frequently than BFGS and gradient method at every elevator size. However, the expected global solving time for the CPM with the QLS is faster because the method is faster in solving local and global optima. With the smallest elevator size, we can see that all methods find the global solution rarely, which implies that the problem is challenging: global optima are sparse compared to local optima.

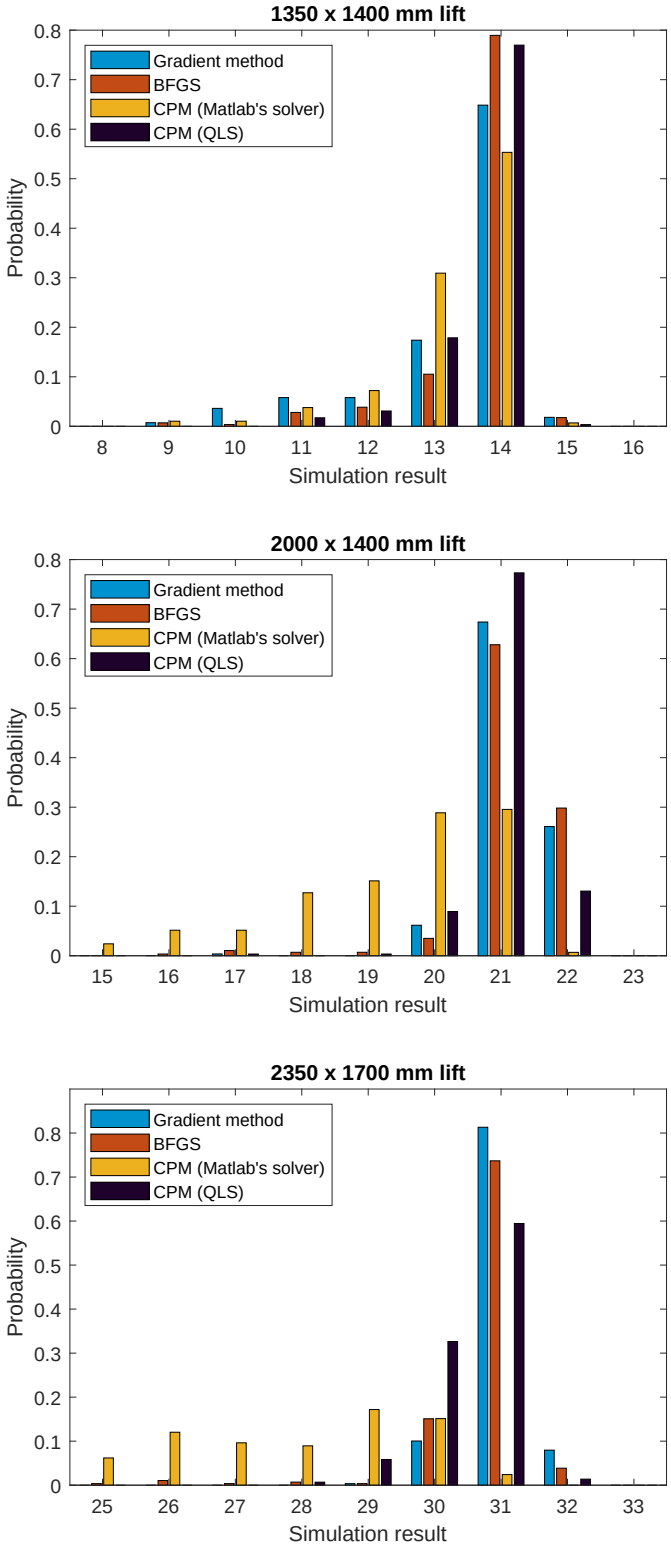


Figure 19: Distribution of the number of fitted objects.

5 Passengers with suitcases

In some scenarios, e.g., in hotel elevators, it may be necessary to simulate passengers carrying suitcases. In this Chapter, we present such a model, with a natural assumption: the suitcase and its owner are kept close to each other. The suitcase, modeled by a rectangle, is modeled to tangent the passenger, see Figure 20. Only the long edge of the rectangle is considered to tangent the capsule. When we proceed to minimize the overlapping, the idea is to let the optimizer also move the suitcases. We use the term *object* to refer to any capsules and capsule-rectangle combinations in the elevator.

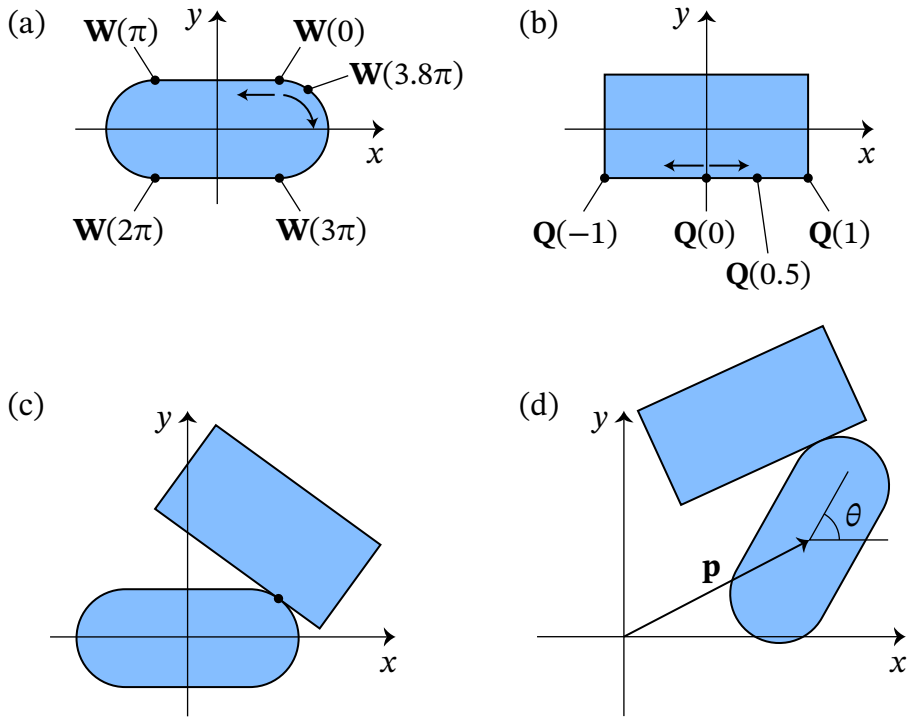


Figure 20: The suitcase model. (a)–(b): We first choose points **W** and **Q** on the capsule and rectangle, e.g., $\mathbf{W}(3.8\pi)$ and $\mathbf{Q}(0.5)$. We join the objects, by aligning these two points, and set the suitcase to such an angle that the two objects do not overlap. (d): Finally we displace the combination with displacement parameters $\mathbf{p}$ and θ.

We define minor modifications to the course of simulation described in Chapter 2. Every other arriving passenger carries a suitcase and every other does not carry a suitcase, see Figure 21. The first passenger carries a suitcase. Initially, the center point of the capsule is placed at the doorstep, as is seen in Figure 21. Similarly, as in Chapter 2, we keep adding objects to the elevator one by one, and overlapping is minimized in between. The procedure is continued until the optimizer cannot find a feasible configuration, i.e., overlapping remains.

We apply four optimization methods to this problem: gradient method, BFGS, CPM with *Matlab*'s solver, and CPM with gradient method and QLS. We use the same elevator sizes and capsule dimensions, as with the capsule packing problem. The dimensions of a suitcase are defined to be 364×220 mm. At the end of this Chapter,

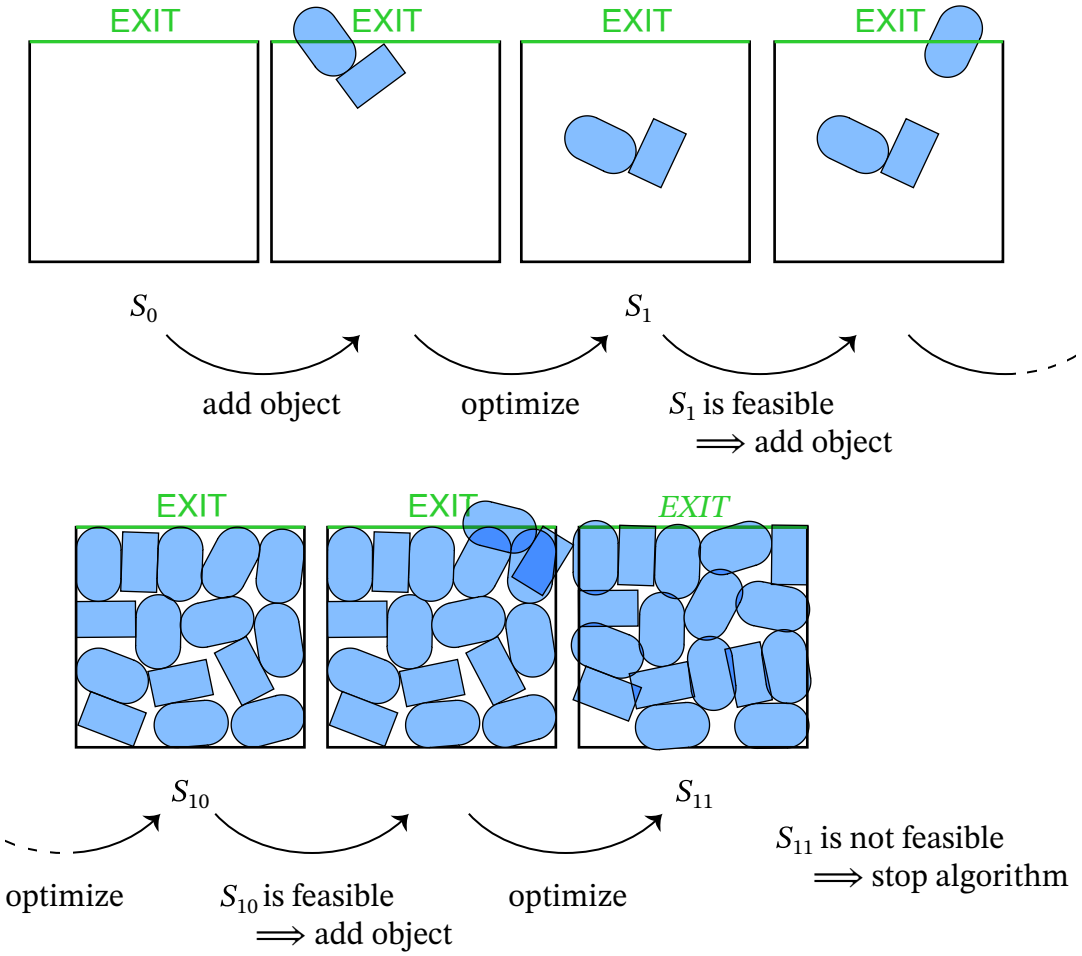


Figure 21: Flow chart of the algorithm.

we shall look at how the problem differs from a capsule packing problem, and how effective the methods are in this context.

5.1 The model

We define a capsule centered in the origin as in Chapter 2.1; in terms of $a, r > 0$, yielding the capsule rectangle corner points, $\mathbf{R}_i$, $i = 1, 2, 3, 4$, and capsule circle center points $\mathbf{C}_i$, $i = 1, 2$. The suitcase's corner points, in the origin, are defined analogously in terms of its dimensions $a', r' > 0$, as follows,

$$\mathbf{R}'_1 = \begin{bmatrix} a' \\ r' \end{bmatrix}, \ \mathbf{R}'_2 = \begin{bmatrix} -a' \\ r' \end{bmatrix}, \ \mathbf{R}'_3 = \begin{bmatrix} -a' \\ -r' \end{bmatrix}, \ \mathbf{R}'_4 = \begin{bmatrix} a' \\ -r' \end{bmatrix}.$$

We define two parameters, $q \in [-1, 1]$ and $\omega \in \mathbb{R}$, which we shall use to define the point of contact for the two objects. Varying the parameter values allows us to move the suitcase around the host capsule. The parameter q defines the contact point on the rectangle, and ω defines the contact point on the capsule. We define the rectangle

contact point $\mathbf{Q}(q)$, to lie on the rectangle's bottom edge (see Figure 20b),

$$\mathbf{Q}(q) = \begin{bmatrix} qa' \\ -r' \end{bmatrix}.$$

The point on the capsule, $\mathbf{W}(w)$, $0 \leq w < 4\pi$, is defined in a periodical manner from the up right corner point back to it in a period of 4π. With this curious parametrization we obtain tidy equations below. In addition, we use parameter ω defined as,

$$w = \omega - 4\pi \left\lfloor \frac{\omega}{4\pi} \right\rfloor$$

We now want the long edge of the rectangle to be aligned with the tangent at $\mathbf{W}(w)$, hence, the rectangle will not overlap with the capsule. We define the angle $\vartheta(w)$, to be the angle between this tangent and the x-axis. Table 7 contains the defining formulas.

Table 7: Definition of $\mathbf{W}(w)$ and $\vartheta(w)$.

w	$\mathbf{W}(w)$	$\vartheta(w)$
$0 \leq w < \pi$	$\begin{bmatrix} a - 2aw/\pi \\ r \end{bmatrix}$	0
$\pi \leq w < 2\pi$	$\begin{bmatrix} -a \\ 0 \end{bmatrix} + r \begin{bmatrix} \cos(w + \pi/2) \\ \sin(w + \pi/2) \end{bmatrix} = \begin{bmatrix} -r \sin w - a \\ r \cos w \end{bmatrix}$	$w - \pi$
$2\pi \leq w < 3\pi$	$\begin{bmatrix} -a + 2a(w - 2\pi)/\pi \\ -r \end{bmatrix}$	π
$3\pi \leq w < 4\pi$	$\begin{bmatrix} a \\ 0 \end{bmatrix} + r \begin{bmatrix} \cos(w + 3\pi/2) \\ \sin(w + 3\pi/2) \end{bmatrix} = \begin{bmatrix} r \sin w + a \\ -r \cos w \end{bmatrix}$	$w - 2\pi$

We now displace the rectangle's corner points by,

$$R(\vartheta(w))(\mathbf{R}'_i - \mathbf{Q}(q)) + \mathbf{W}(w),$$

which defines a rectangle seen in Figure 20c. $R(\cdot)$ is a rotation matrix, defined in Chapter 2.1. The rectangle tangents the capsule which is placed in the origin. Then, we displace both, the capsule and the rectangle, with $\mathbf{p} \in \mathbb{R}^2$ and $\theta \in \mathbb{R}$, yielding the final placement seen in Figure 20d,

$$\mathbf{r}'_i = R(\theta)\left[R(\vartheta(w))(\mathbf{R}'_i - \mathbf{Q}(q)) + \mathbf{W}(w)\right] + \mathbf{p},\ i = 1, 2, 3, 4,$$

$$\mathbf{r}_i = R(\theta)\,\mathbf{R}_i + \mathbf{p},\ i = 1, 2, 3, 4,$$

$$\mathbf{c}_i = R(\theta)\,\mathbf{C}_i + \mathbf{p},\ i = 1, 2.$$

5.2 Objective function

We need to modify the objective function presented in Chapter 2.3. The objective function is the same as with capsule packing problem, but with added optimization parameters, and exceptions in overlapping area and overlapping distance functions.

Define the number of capsules n, and the number of capsule-rectangle combinations n'. Define $N = \{1, \dots, n\}$, $N' = \{n+1, \dots, n+n'\}$, and the set of objects, $X = \{\mathbf{x}^1, \dots, \mathbf{x}^{n+n'}\}$, where,

$$\mathbf{x}^m = \begin{bmatrix} \mathbf{p}^m \\ \theta^m \end{bmatrix}, \ \forall m \in N,$$

$$\mathbf{x}^m = \begin{bmatrix} \mathbf{p}^m \\ \theta^m \\ q^m \\ \omega^m \end{bmatrix}, \ \forall m \in N'.$$

We redefine the functions E and A to handle overlapping of different types of objects. Define $\mathbf{k}^i$ to be the capsule of the capsule-rectangle combination, and $\mathbf{s}^i$ to be the rectangle of the object $\mathbf{x}^i$, $i \in N'$. Consider two any type of objects, $\mathbf{x}^i$, and $\mathbf{x}^j$. If $\mathbf{x}^i$ is a rectangle-capsule combination, and $\mathbf{x}^j$ is a capsule, we define,

$$E(\mathbf{x}^i, \mathbf{x}^j) = E(\mathbf{k}^i, \mathbf{x}^j) + A(\mathbf{s}^i, \mathbf{x}^j), \text{ for } i \in N', j \in N,$$
$$A(\mathbf{x}^i, \mathbf{x}^j) = A(\mathbf{k}^i, \mathbf{x}^j) + A(\mathbf{s}^i, \mathbf{x}^j), \text{ for } i \in N', \ j \in N.$$

If both $\mathbf{x}^i$ and $\mathbf{x}^j$ are capsule-rectangle combinations, we define,

$$E(\mathbf{x}^i, \mathbf{x}^j) = E(\mathbf{k}^i, \mathbf{k}^j) + A(\mathbf{k}^i, \mathbf{s}^j) + A(\mathbf{s}^i, \mathbf{k}^j) + A(\mathbf{s}^i, \mathbf{s}^j), \text{ for } i, j \in N',$$
$$A(\mathbf{x}^i, \mathbf{x}^j) = A(\mathbf{k}^i, \mathbf{k}^j) + A(\mathbf{k}^i, \mathbf{s}^j) + A(\mathbf{s}^i, \mathbf{k}^j) + A(\mathbf{s}^i, \mathbf{s}^j), \text{ for } i, j \in N'.$$

We define a barrier function to constrain $q^i \in [-1, 1], \forall i \in N'$. The barrier function is $\sum_{i \in N'} \gamma_k |q^i|^{\gamma_k}$, where parameter $\gamma_k \geq 1$ is increased as the iteration index k increases. We add the barrier function to the objective function. For the CPM methods, we define $\gamma_k = (k/(n+n')+1)^2$, and for the gradient method and BFGS, we define $\gamma_k = (k+1)^2$.

5.3 Results

Simulation results are shown in Table 8. The gradient method is left out of the table because it was unable to find any global optima to any problem. In addition, BFGS found global solutions only to the problem with the smallest elevator size. A non-global solution of BFGS is shown in Figure 22. We see that almost every object is surrounded from at least three different directions, locking the objects in their places and creating a local optimum. Rectangles seem to be far more exposed to form these local optima, compared to capsules. The varying of initial conditions will eventually lead to a global optimum, but it can consume much time, thus, different global solving techniques are useful here.

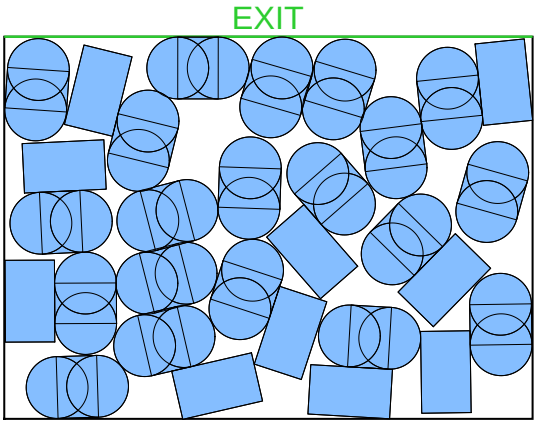


Figure 22: A local optimum found by the BFGS.

For the CPM with QLS, we implemented the random search which, according to the results, works very well. From Table 8 we see that the CPM with the QLS is very fast compared to other methods. With the largest problem, the CPM with the QLS is 38 times faster than the method with *Matlab*'s optimizer. From Figure 23 we see that with gradient method and BFGS there is a little probability of finding even near-global optima. The global optima are robustly only found by the CPM.

Table 8: Expected global optimization time of an elevator packing problem with passengers and suitcases. Computing times are shown in a unit of million area calculations. The fastest algorithm for each problem is shown in bold. A suitcase and the host capsule are counted as a single object.

Elevator size (mm)	Max capacity in simulations, $n + n'$	BFGS	CPM (*Matlab*'s optimizer)	CPM (QLS)
1350×1400	10	2.44	1.99	**0.878**
2000×1400	15	–	26.3	**6.42**
2350×1700	22	–	906	**23.6**

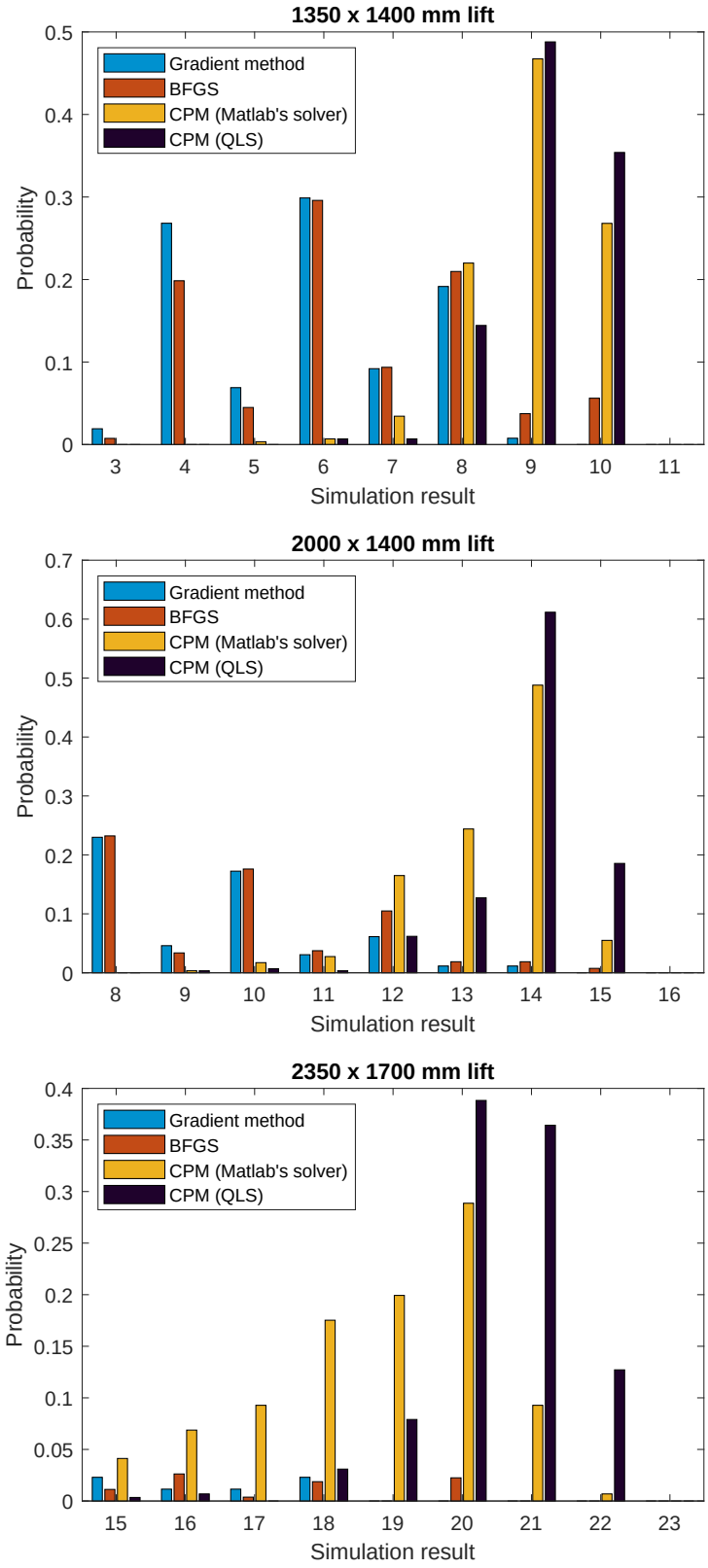


Figure 23: Distribution of the number of fitted objects.

6 Shopping cart problem

E.g., in shopping centers, elevators can be used to carry shopping carts. In this Chapter, we pack elevators with shopping carts and passengers. We first place the shopping cart(s) into the elevator and then add capsules using the same algorithm as in Chapter 2. As the optimization method, we use the CPM with QLS, described in Chapter 4.2. We use the same passenger dimensions, and three different elevator sizes, as in Chapter 2. A shopping cart is modeled by a 630×1180 mm rectangle.

First, we try few intuitive solutions with a single shopping cart fixed to a corner of the elevator, and then fill the available space with passengers. We try different shopping cart orientations. Our aim is to see how many passengers we can fit, and see if there is an optimal orientation. This method could work as a heuristic for finding sufficient solutions fast. In Chapter 6.2, we look for solutions thoroughly by placing shopping cart(s) randomly in the elevator, and optimizing its/their placement along with the passengers. With random placing, we handle 1–3 shopping carts. The aim here is to see if there are clever non-intuitive solutions.

6.1 One fixed shopping cart

We fix the shopping cart to a corner of an elevator (any of the four corners), with either horizontal or vertical orientation. By symmetry, we can reduce the problem to two cases, see Figure 24. With the shopping cart fixed, we shall fill the elevator with passengers, with the algorithm described in Chapter 2. The results are shown in Table 9 and Figure 25. We see that the shopping cart should be placed horizontally, with which we obtain more optimal packing. However, the results do not differ dramatically between the different orientations. Note that this result only applies to the three elevator sizes tested, and with constant capsule and shopping cart dimensions. The result is likely to change with different simulation setups.

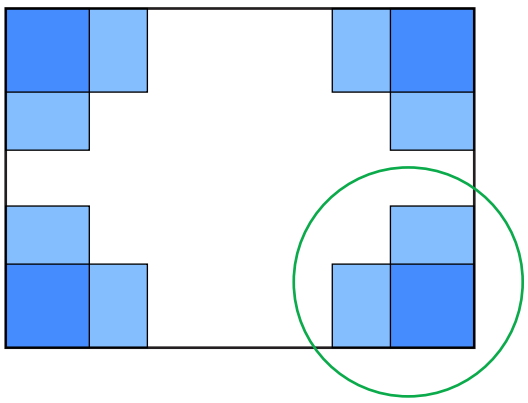


Figure 24: A shopping cart can be placed in a corner of an elevator in eight ways. Because of symmetry, we can reduce our study to two cases, which are highlighted with a circle.

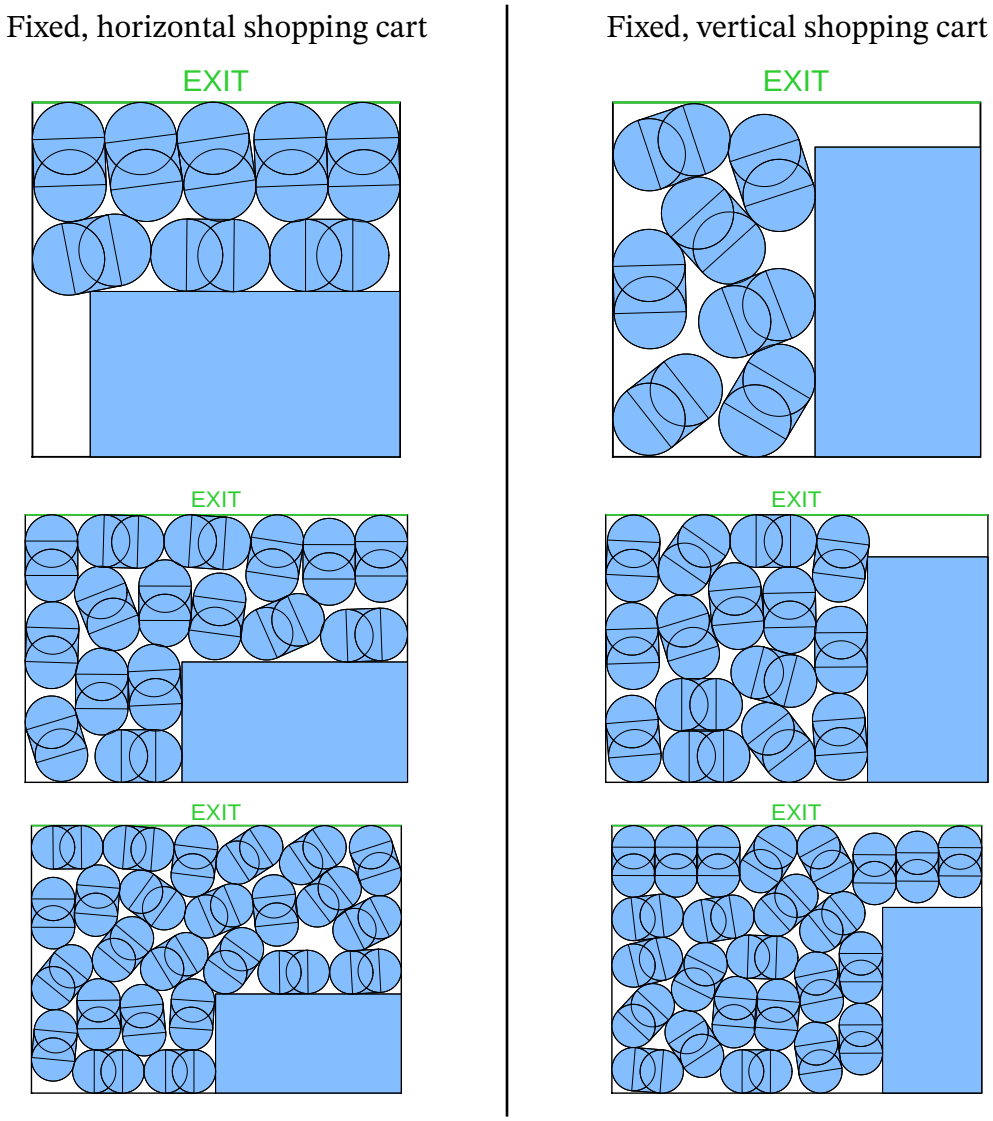


Figure 25: Results for the intuitive shopping cart arrangements. Number of passengers in each graph is listed in Table 9.

Table 9: Maximum number of passengers in an elevator, with a shopping cart fixed to a corner. The cart is placed to the corner either horizontally or vertically. Solutions with most passengers are in bold.

Box size (mm)	Shopping cart orientation	
	Horizontal	Vertical
1350 × 1400	**8**	7
2000 × 1400	**16**	15
2350 × 1700	**25**	**25**

6.2 One to three moving shopping carts

Now, we try and find optimal solutions randomly. We first place the shopping cart(s) randomly into the elevator, and we allow the optimizer to move its location, as if it was one of the capsules. Note, that the smallest elevator has capacity for two shopping carts, but then there is no room for any capsules. Similarly, the medium sized elevator has capacity for three shopping carts, but in that case no capsules can be fitted into the elevator. Excluding the infeasible cases, we only have 6 scenarios to handle.

Results are shown in Table 10 and Figures 26–28. With this rigorous testing, we could not find any better scenarios (e.g., with skewed shopping carts) than with the intuitive, horizontal placement. This means that a shopping cart at a corner works as a good heuristic, at least in these few cases. With the two smaller elevator sizes, we actually could not find as good solutions as with the horizontal intuitive case. For the large size, we found solutions with equal amount of capsules compared to the intuitive horizontal case. From the Figures 26–28, we see that it is commonplace for the shopping carts to lie against the box's edge. Also, horizontal and vertical shopping carts are common; skewed shopping carts are more rare. The algorithm may have a tendency to find non-skewed settings more often than skewed ones. On the other hand the non-skewed settings may very well be more optimal compared to skewed ones regardless of the algorithm.

Table 10: Maximum number of passengers using random placing for the shopping carts.

Box size (mm)	Number of shopping carts		
	1	2	3
1350 × 1400	7	0	–
2000 × 1400	15	8	0
2350 × 1700	25	19	12

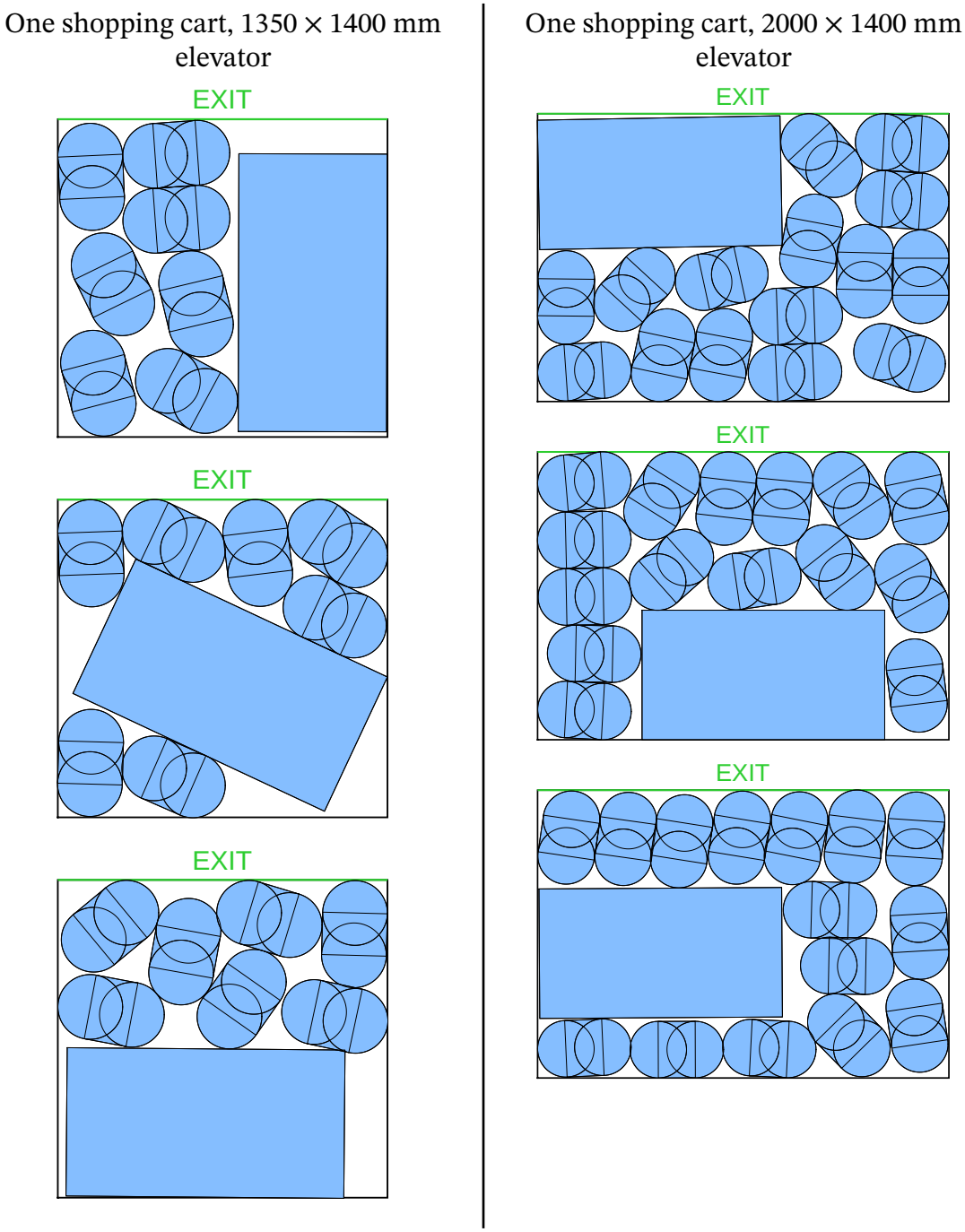


Figure 26: A sample of results for randomly searched arrangements.

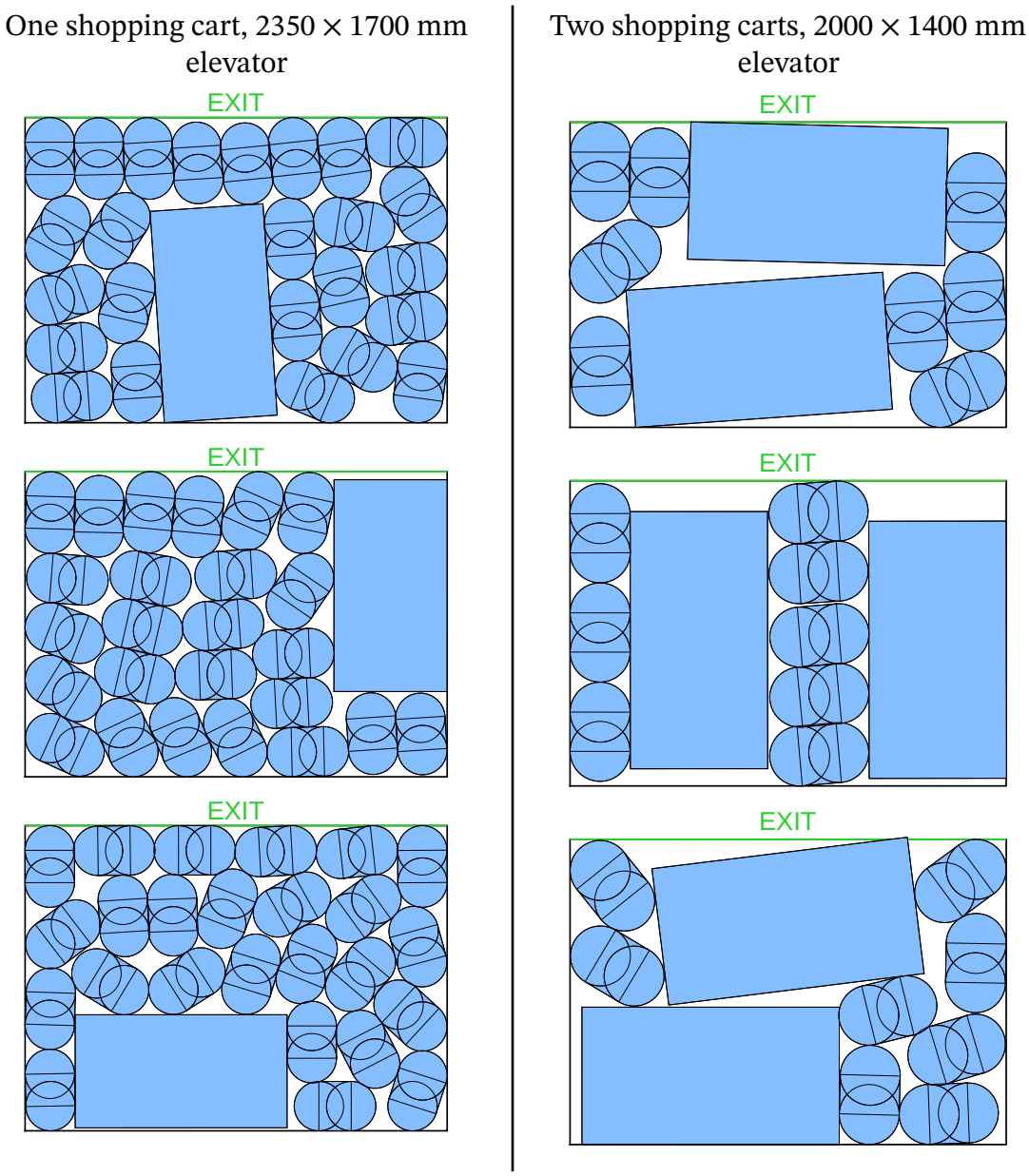


Figure 27: A sample of results for randomly searched arrangements.

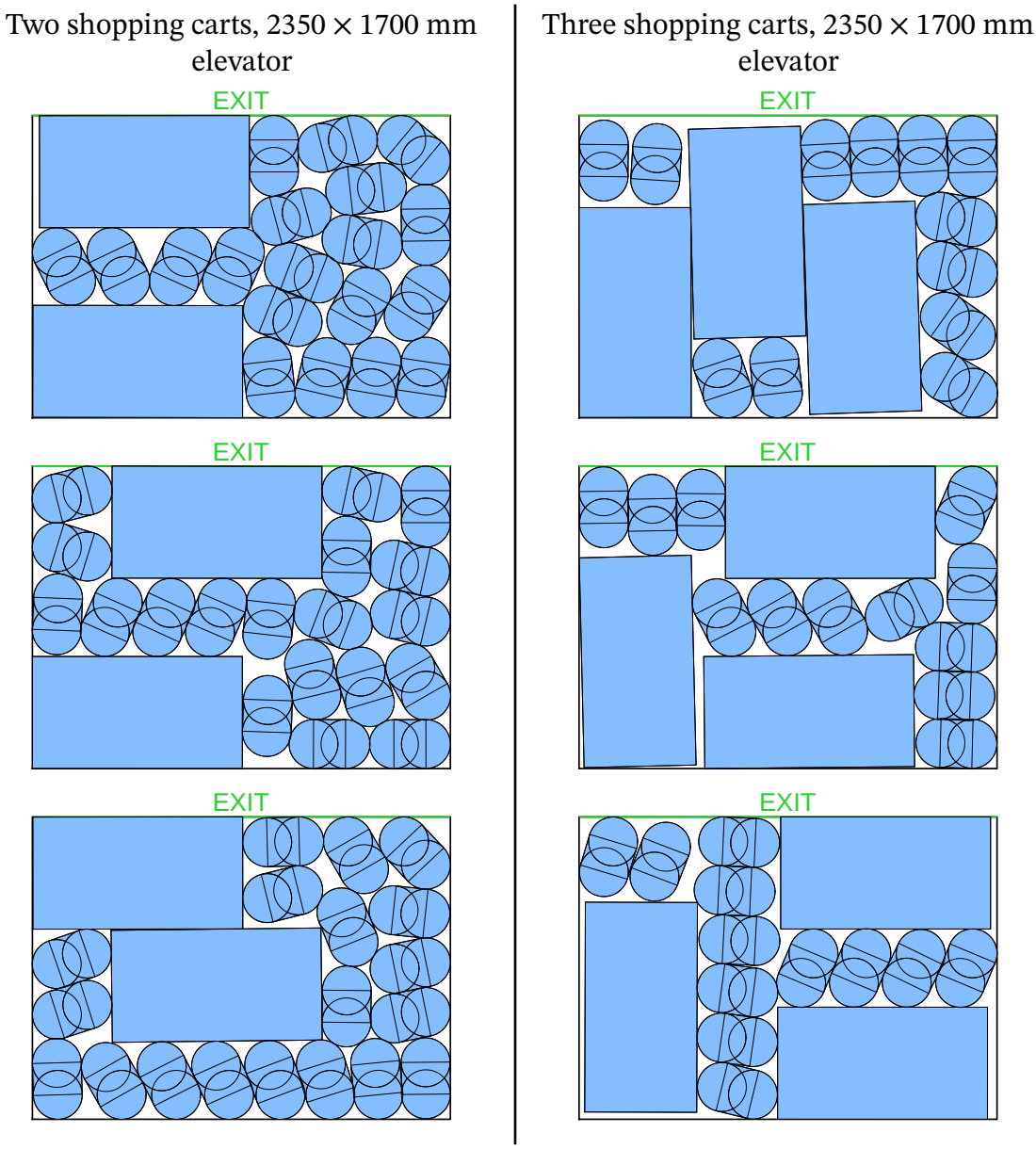


Figure 28: A sample of results for randomly searched arrangements.

7 Discussion and conclusions

In this Master's thesis we constructed a model for elevator packing algorithm, which relies on an optimization model; the minimization of overlapping between objects. We formulated the optimization model and solved it by using different non-linear optimization methods: gradient method, BFGS, and CPM. We also constructed a CPM with gradient method and QLS line search method. The latter method proved to be the best, based on a capsule packing problem and a problem where some of the passengers carry suitcases. Lastly, we applied the latter optimization method to an application involving also shopping carts.

This thesis is a continuum to the author's Bachelor's thesis [8]. The Bachelor's thesis handles pure capsule packing problem. The capsule shape was chosen because it somewhat resembles a person from above. The packing algorithm minimizes capsule overlapping area from a random initial configuration, where overlapping occurs. In the thesis, the algorithm for the overlapping area of two capsules or rectangles was developed, along with the CPM algorithm. Both methods were developed by the author. The gradient evaluation simplification emerges when CPM is formulated. In the B.Sc. thesis, the CPM was found to be much faster compared to a gradient method with no gradient simplification applied. In this Master's thesis, we used the simplified gradient evaluation method with the gradient method and BFGS, which made the methods much faster than in the B.Sc. thesis. The simplification of the gradient evaluation [8] is very effective and should be used always.

In this Master's thesis, we continue to refine the packing algorithm and making it more suitable for elevator packing applications. This application yields a different kind of packing approach: in the Bachelor's thesis, we looked for local optima, and we did not pay much attention to the feasibility of the resulting configurations. Here, the configurations must be feasible, which leads us to the model where we add objects one by one, and in between, to make sure that we can find a feasible configuration. This might seem like a minor change, but it affects the results: CPM with *Matlab*'s solver, which was used in the Bachelor's thesis, does not perform well here: the method solves local minima fast but does not find global minima too frequently.

In this thesis, we introduced a new concept, overlapping distance. It is useful when finding if, and how much, two capsules overlap. It is fast to compute, and the algorithm is fairly simple. It can be used with higher dimensions, by substituting the unit disk D with a unit sphere. Another new concept is the expected global solving time and the formula for it. This concept is on focus when we compare different optimizers. The implementation for QLS, where only one function evaluation is needed, was found by the author independently and was later found in the literature [10]. The method is rarely discussed along with other common line search methods, which is a pity, because it is a powerful method.

The visual appearance and clear information appearance were at a high priority level when making this thesis. The nature of the problem itself is visual, and so are the results.

7.1 Results and their reliability

We compared the methods by comparing their expected global solving times. The gradient method performs the worst of the methods: it is slow to find solutions to the capsule packing problem, and cannot find any global solution to any problem involving suitcases. BFGS is good at solving capsule packing problems but struggles with the problems with suitcases. The poor performance is explained by the non-differentiability of the objective function, and corners in the objective function. An addition of some kind of random search could potentially improve these methods.

The CPM was used with *Matlab*'s solver, and later we combined CPM with the QLS method. Both of these methods find solutions to problems with passengers and suitcases, but the CPM with QLS is much faster. The CPM with *Matlab*'s solver performed poorly with the capsule packing problem, whereas the BFGS, and CPM with QLS, performed rather well.

In the simulation results, we used a certain time unit (number of E and A function evaluations), which is not affected by computer hardware. The expected global solving time is a good measure for method performance in this context, however, the confidence interval of this measured estimate may be large due to small data sample sizes. This fact decreases the reliability of the results.

7.2 Future research

The elevator model is very simple and does not take into account, e.g., social factors. A passenger elevator is rarely packed to its full extent, so in this sense, the model does not reflect human behavior. At least some notion of social space should be considered. Also here, the passengers are assumed to be of the same size, which is a strong assumption. In this thesis, we aimed to analyze the methods, rather than build a realistic model. The algorithms described in this thesis can certainly be applied to more complex models, which take the above factors into account.

In Chapter 2.3, we found that the objective function is not differentiable everywhere, thus making the landscape challenging for the common non-linear optimization methods. If the objective function could be modified to a differentiable form, the optimization methods would probably perform better. Heuristics are not discussed much in this thesis. Heuristics could improve the algorithms' ability to find solutions, by e.g., providing more suitable initial conditions.

We aim to minimize the objective function, however, we know that at a feasible solution (if it exists), the objective function value is zero. Thus, we may be able to solve the problem with a multivariate root solving method, e.g., Broyden's method. This approach could potentially be faster than the minimization approach.

This thesis contains many parameters, whose values are not justified, and thus do not guarantee optimal performance of the methods used. These parameters include diagonal scaling factors and the stopping criteria, both discussed in Chapter 3. With the QLS, we also have the parameter h_k, and the step limit M, whose values may impact the performance of the method.

References

[1] International Organization for Standardization. ISO 8100-30:2019. Lifts for the transport of persons and goods — Part 30: Class I, II, III and VI lifts installation. 2019. URL: https://www.iso.org/standard/73082.html.

[2] Mirko Ruokokoski. Determining the number of passengers that can be fitted in a standard-sized lift car. Working paper, KONE Corporation, 2015.

[3] Mirko Ruokokoski, Janne Sorsa, and Marja-Liisa Siikonen. Human Body Size in Lift Traffic Design. In: *Proceedings of 4th Symposium on Lift and Escalator Technologies,* Northampton (2014), pp. 213–223.

[4] A. Pankratov, T. Romanova, and I. Subota. An Efficient Algorithm for the Ellipse Packing Problem. the National Academy of Sciences of Ukraine, 2013.

[5] Janne Sorsa. Optimization Models and Numerical Algorithms for an Elevator Group Control System, Aalto University publication series, Doctoral Dissertations, 139/2017. Aalto University, 2017.

[6] Mirko Ruokokoski et al. Assignment formulation for the Elevator Dispatching Problem with destination control and its performance analysis. In: *European Journal of Operational Research* 252.2 (2016), pp. 397–406.

[7] Gary B Hughes and Mohcine Chraibi. Calculating Ellipse Overlap Areas. In: *Computing and Visualization in Science* 15.5 (2012), pp. 291–301.

[8] Lauri Jokinen. Cyclic Placement Method for Capsule Packing Problem. B.Sc. thesis, Aalto University, Department of Mathematics and Systems Analysis, 2018.

[9] Amir Beck and Luba Tetruashvili. On the convergence of block coordinate descent type methods. In: *SIAM Journal on Optimization* 23.4 (2013), pp. 2037–2060.

[10] Jorge Nocedal and Stephen J. Wright. Numerical Optimization. Second Edition. Springer, 2006.

[11] Dimitri P. Bertsekas. Nonlinear Programming. Athena Scientific, 1995.

[12] M. Hifi and R. M'Hallah. A Literature Review on Circle and Sphere Packing Problems: Models and Methodologies. In: *Advances in Operations Research* 2009 (2009), 150624:1–150624:22.

[13] Dimitrii Ilin and Marc Bernacki. A new algorithm for dense ellipse packing and polygonal structures generation in context of FEM or DEM. MINES ParisTech, PSL-Research University, 2016.

[14] F. Carrabs, C. Cerrone, and R. Cerulli. A Tabu Search Approach for the Circle Packing Problem. In: *2014 17th International Conference on Network-Based Information Systems*. 2014, pp. 165–171. DOI: 10.1109/NBiS.2014.28.

[15] E. Hopper and B. Turton. A genetic algorithm for a 2D industrial packing problem. In: *Computers & Industrial Engineering* 37.1 (1999). Proceedings of the 24th international conference on computers and industrial engineering, pp. 375–378. URL: https://www.sciencedirect.com/science/article/pii/S0360835299000972.

[16] Mokhtar S. Bazaraa, Hanif D. Sherali, and C. M. Shetty. Nonlinear Programming, Theory and Algorithms. Wiley, 1993.

[17] Ion Necoara and Dragos Clipici. Parallel random coordinate descent method for composite minimization: Convergence analysis and error bounds. In: *SIAM Journal on Optimization* 26.1 (2016), pp. 197–226.

[18] Julie Nutini et al. Coordinate Descent Converges Faster with the Gauss-Southwell Rule Than Random Selection. 2018. URL: https://arxiv.org/pdf/1506.00552.pdf.

[19] Stephen J Wright. Coordinate descent algorithms. In: *Mathematical Programming* 151.1 (2015), pp. 3–34.

[20] Jie Chen et al. Optimal contraction theorem for exploration–exploitation tradeoff in search and optimization. In: *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 39.3 (2009), pp. 680–691.

[21] Shephen Pheasant and Christie M. Haslegrave. Bodyspace. Third Edition. Taylor & Francis Group, 2006.

[22] European Committee for Standardization. EN 81-1:1998 Safety rules for the construction and installation of lifts. 1998. URL: https://standards.iteh.ai/catalog/standards/cen/42876a79-9ff7-4af9-a835-39239dc8e18e/en-81-1-1998.

[23] Wolfram Mathworld. Circular Segment. URL: https://mathworld.wolfram.com/CircularSegment.html.

[24] ProofWiki. Reverse Triangle Inequality. URL: https://proofwiki.org/wiki/Reverse_Triangle_Inequality.

[25] Benjamin Zane Dunham. High-Order Automatic Differentiation of Unmodified Linear Algebra Routines via Nilpotent Matrices. 2017. URL: https://scholar.colorado.edu/cgi/viewcontent.cgi?article=1168&context=asen_gradetds.

[26] Mert Gurbuzbalaban et al. When cyclic coordinate descent outperforms randomized coordinate descent. In: *Advances in Neural Information Processing Systems*. 2017, pp. 6999–7007.

[27] Julie Nutini et al. Coordinate descent converges faster with the gauss-southwell rule than random selection. In: *International Conference on Machine Learning*. 2015, pp. 1632–1641.

[28] Wikipedia. Geometric series. URL: https://en.wikipedia.org/wiki/Geometric_series.

[29] Mark Schmidt. CPSC 540: Machine Learning, Convergence of Gradient Descent. 2017. URL: https://www.cs.ubc.ca/~schmidtm/Courses/540-W18/L4.pdf.

[30] Yun-Wei Shang and Yu-Huang Qiu. A note on the extended Rosenbrock function. In: *Evolutionary Computation* 14.1 (2006), pp. 119–126.

[31] Sigrún Andradóttir. An overview of simulation optimization via random search. In: *Handbooks in operations research and management science* 13 (2006), pp. 617–631.

A Intersection points of circles and line segments

Intersection point of two line segments

Define two line segments,

$$\begin{aligned} \mathbf{p} &= t\,\mathbf{p}_1 + (1-t)\,\mathbf{p}_2, \quad t \in [0,1], \quad \mathbf{p}_1, \mathbf{p}_2 \in \mathbb{R}^2, \\ \mathbf{q} &= s\,\mathbf{q}_1 + (1-s)\,\mathbf{q}_2, \quad s \in [0,1], \quad \mathbf{q}_1, \mathbf{q}_2 \in \mathbb{R}^2. \end{aligned} \tag{A1}$$

At the intersection point, it holds,

$$\begin{bmatrix} t \\ s \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1 - \mathbf{p}_2 & \mathbf{q}_2 - \mathbf{q}_1 \end{bmatrix}^{1} (\mathbf{q}_2 - \mathbf{p}_2).$$

If the inverse matrix does not exist (i.e., $\det\begin{bmatrix} \mathbf{p}_1 - \mathbf{p}_2 & \mathbf{q}_2 - \mathbf{q}_1 \end{bmatrix} = 0$), or the requirements $t, s \in [0,1]$ are not satisfied, the intersection point does not exists. If values of t and s are feasible, we shall substitute them to Equation (A1), which gives the coordinates of the intersection point.

Intersection points of two circles

Define two circles in terms of their center points, $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{R}^2$, and radii $r_1, r_2 > 0$. Define, $d = \|\mathbf{c}_1 - \mathbf{c}_2\|$. If $d = 0$ (even if the circles are identical), we don't return intersection points. Next define,

$$\mathbf{p}_{\pm} = \frac{1}{2d} \begin{bmatrix} d^2 + r_1^2 - r_2^2 \\ \pm\sqrt{2d^2(r_1^2 + r_2^2) - d^4 - (r_1^2 - r_2^2)^2} \end{bmatrix}.$$

If the above formula gives complex solutions, we reject them. Finally define,

$$\begin{aligned} \mathbf{w} &= \frac{\mathbf{c}_2 - \mathbf{c}_1}{d}, \\ \alpha &= \arctan(w_2/w_1), \\ R(\alpha) &= \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}, \\ \mathbf{q}_{\pm} &= R(\alpha)\mathbf{p}_{\pm} + \mathbf{c}_1, \end{aligned}$$

where $\mathbf{q}$ denotes the final coordinates of the intersection point(s).

Intersection points of a circle and a line segment

Consider a line segment,

$$\mathbf{r} = t\,\mathbf{r}_1 + (1-t)\,\mathbf{r}_2, \quad t \in [0,1], \quad \mathbf{r}_1, \mathbf{r}_2 \in \mathbb{R}^2,$$

and a circle defined in terms of its center point, $\mathbf{c} \in \mathbb{R}^2$ and radius $r > 0$. Define,

$$\begin{aligned}
\mathbf{w} &= \frac{\mathbf{r}_2 - \mathbf{r}_1}{\|\mathbf{r}_2 - \mathbf{r}_1\|}, \\
\beta &= \arctan(w_2/w_1), \\
\mathbf{r}'_1 &= R(-\beta)(\mathbf{r}_1 - \mathbf{c}), \\
\mathbf{r}'_2 &= R(-\beta)(\mathbf{r}_2 - \mathbf{c}), \\
\mathbf{p}_\pm &= \begin{bmatrix} \pm\sqrt{r^2 - (\mathbf{r}'_{1,y})^2} \\ \mathbf{r}'_{1,y} \end{bmatrix}.
\end{aligned}$$

We accept the points $\mathbf{p}_\pm$, if $\mathbf{p}_\pm$ is not complex, and

$$\min\{\mathbf{r}'_{1,x}, \mathbf{r}'_{2,x}\} \leq \mathbf{p}_{\pm,x} \leq \max\{\mathbf{r}'_{1,x}, \mathbf{r}'_{2,x}\}.$$

Finally define,

$$\mathbf{q}_\pm = R(\beta)\mathbf{p}_\pm + \mathbf{c},$$

which are our intersection points.

B Algorithm for a minimum distance between two line segments

Define two line segments,

$$\mathbf{p} = t\,\mathbf{p}_1 + (1-t)\,\mathbf{p}_2, \quad t \in [0,1], \quad \mathbf{p}_1, \mathbf{p}_2 \in \mathbb{R}^2, \quad \mathbf{p}_1 \neq \mathbf{p}_2,$$
$$\mathbf{q} = s\,\mathbf{q}_1 + (1-s)\,\mathbf{q}_2, \quad s \in [0,1], \quad \mathbf{q}_1, \mathbf{q}_2 \in \mathbb{R}^2, \quad \mathbf{q}_1 \neq \mathbf{q}_2.$$

Our aim is to solve $\min_{t,s\in[0,1]} \|\mathbf{p}(t) - \mathbf{q}(s)\|$, analytically. We define the set $[0,1]^2$ as a feasible set of the points (t,s). The solving is divided to interior point cases and boundary cases, listed in Table B1. Examples are shown in Figure B1. We solve each case, discard solutions which do not lie inside the feasible set, and choose the case where the minimum value is obtained.

Table B1: The interior point cases, and the boundary cases. Cases (i)–(ii) handle the interior points of the feasible set; cases (iii)–(vi) handle the boundary lines; cases (vii)–(ix) handle the boundary corner points.

(i)	$\min_{t,s\in\mathbb{R}} \|\mathbf{p}(t) - \mathbf{q}(s)\|$ and $\mathbf{p}(t) \nparallel \mathbf{q}(s)$
(ii)	$\min_{t,s\in\mathbb{R}} \|\mathbf{p}(t) - \mathbf{q}(s)\|$ and $\mathbf{p}(t) \parallel \mathbf{q}(s)$
(iii)	$\min_{t\in\mathbb{R}} \|\mathbf{p}(t) - \mathbf{q}(0)\|$
(iv)	$\min_{t\in\mathbb{R}} \|\mathbf{p}(t) - \mathbf{q}(1)\|$
(v)	$\min_{s\in\mathbb{R}} \|\mathbf{p}(0) - \mathbf{q}(s)\|$
(vi)	$\min_{s\in\mathbb{R}} \|\mathbf{p}(1) - \mathbf{q}(s)\|$
(vii)	$\|\mathbf{p}(0) - \mathbf{q}(0)\|$
(viii)	$\|\mathbf{p}(0) - \mathbf{q}(1)\|$
(ix)	$\|\mathbf{p}(1) - \mathbf{q}(0)\|$
(x)	$\|\mathbf{p}(1) - \mathbf{q}(1)\|$

The optimal value for case (i) is 0, because non-parallel lines always have an intersection point. The value of a norm is always non-negative, so zero is the minimum. The intersection point of line segments is defined in Appendix A.

The case (ii). When case (ii) produces a feasible solution to the minimization problem, it can be shown that an equivalent minimum can be obtained with the boundary cases as well. Some examples are shown in Figure B1. Thus, there is no need to handle the case (ii), since the same minimum value is obtained with cases (iii)–(x).

Cases (iii)-(vi) are analogous, so we shall solve (iii) as an example. The problem is

$$\min_{t\in\mathbb{R}} \left\{ \mathbf{p}_2 + t(\mathbf{p}_1 - \mathbf{p}_2) - \mathbf{q}(0) \right\}. \tag{B1}$$

The minimum can be obtained with projecting the point $\mathbf{q}(0)$ onto the line segment. The projected point can be found with the following value for parameter t,

$$t = \frac{(\mathbf{p}_1 - \mathbf{p}_2)^\top (\mathbf{q}(0) - \mathbf{p}_2)}{\|\mathbf{p}_1 - \mathbf{p}_2\|^2}.$$

If $t \notin [0,1]$, we discard the solution. Otherwise, the minimum length can be obtained with substituting the value of t to Equation (B1).

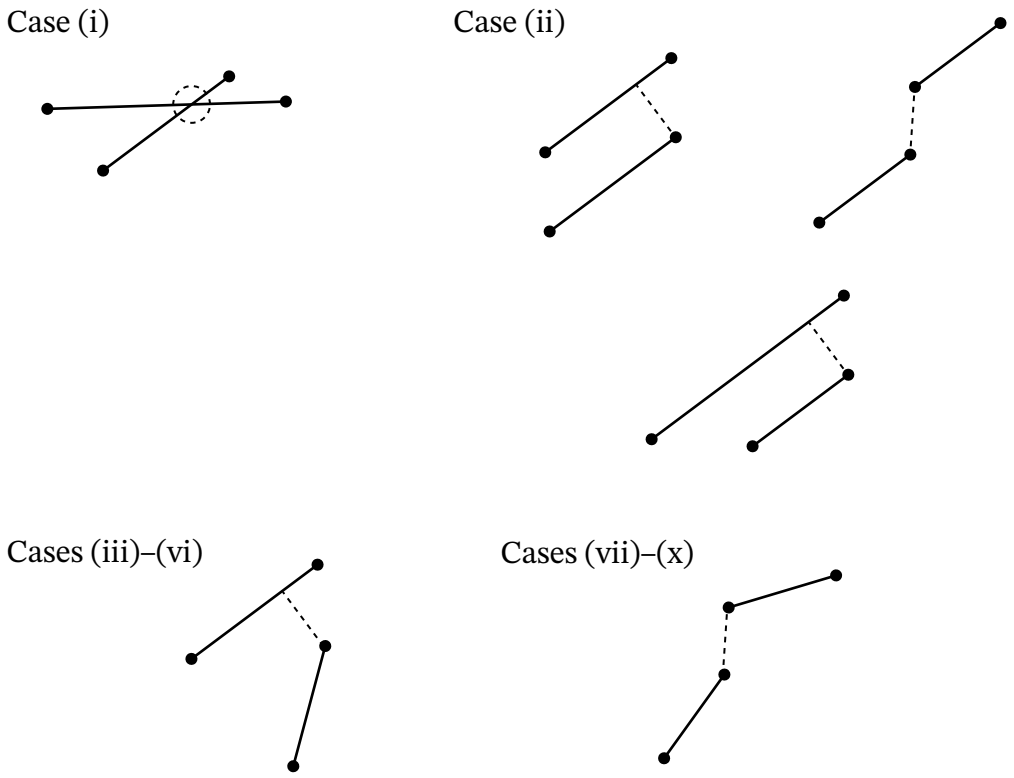


Figure B1: Feasible solutions for different cases. The dashed line shows the minimum distance between the two line segments. In the case (i), the minimum distance is zero.

C Simulation data

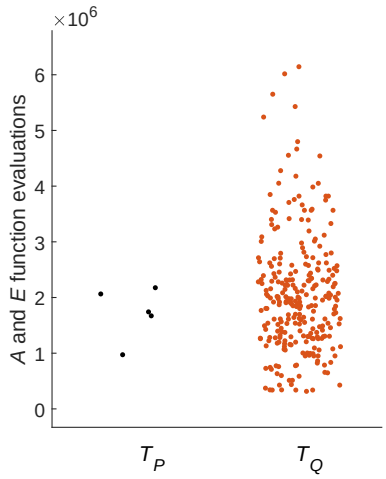
C.1 Global solving times

The following Figures contain the simulation time data, from which the expected global solving times can be calculated. We have separated the simulation times to two sets, T_P and T_Q, depending if the simulation outcome was a global maximum, or not the global maximum, respectively. Global maximum is defined as a configuration in which we have the largest possible number of objects in the elevator, with no overlapping between any of the objects. The horizontal coordinate in the Figures is varied at random, in order to spread the points a bit. The global solving formulas are found in Chapter 3.4. As a time unit, we use the number of A and E function evaluations, which is discussed in Chapter 3.

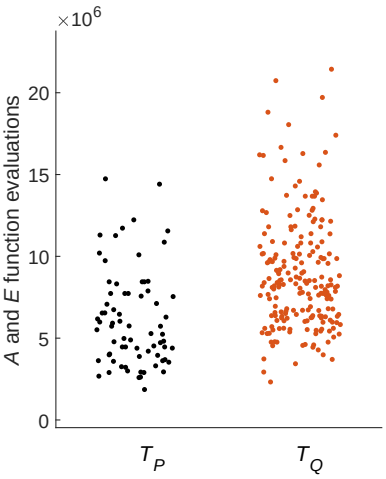
Gradient method

Capsule packing problem

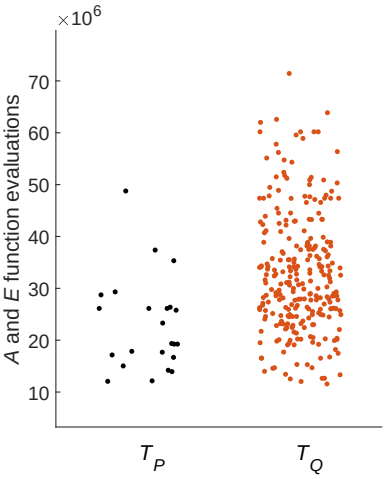
1350 × 1400 mm elevator



2000 × 1400 mm elevator

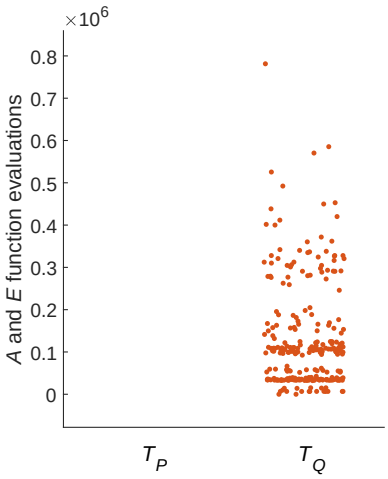


2350 × 1700 mm elevator

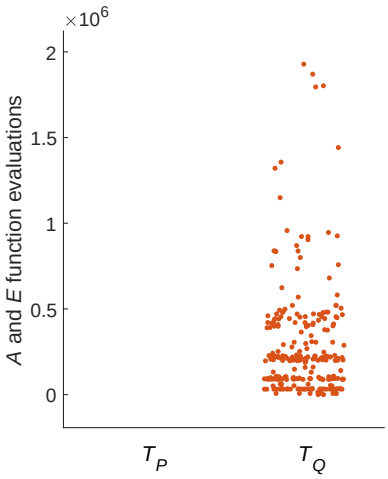


Passengers and suitcases

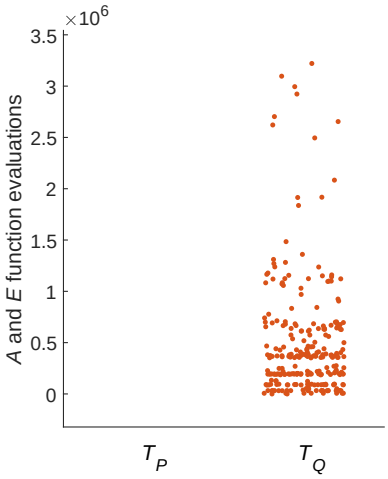
1350 × 1400 mm elevator



2000 × 1400 mm elevator



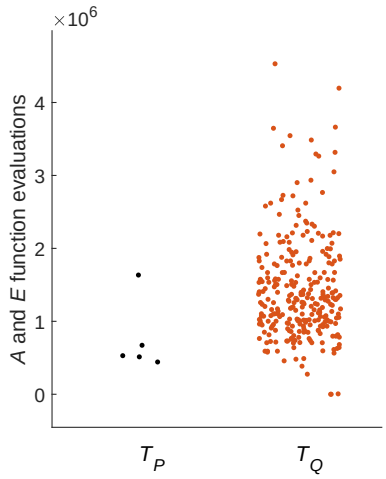
2350 × 1700 mm elevator



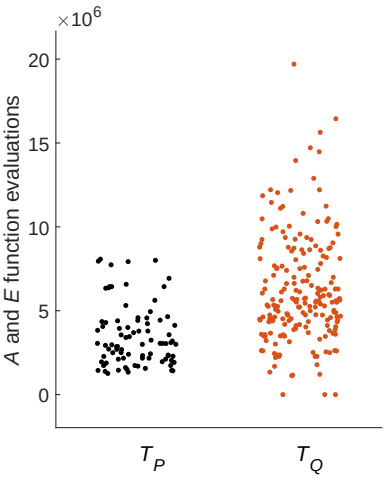
BFGS

Capsule packing problem

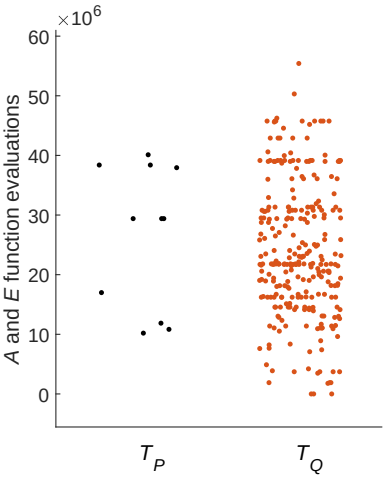
1350 × 1400 mm elevator



2000 × 1400 mm elevator

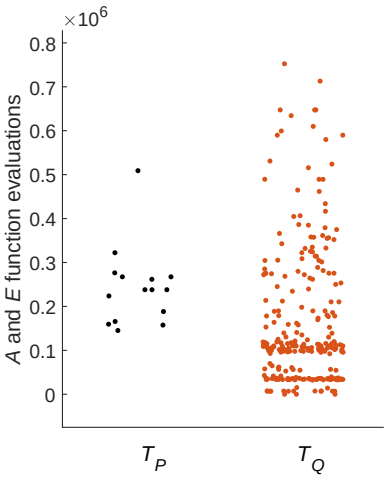


2350 × 1700 mm elevator

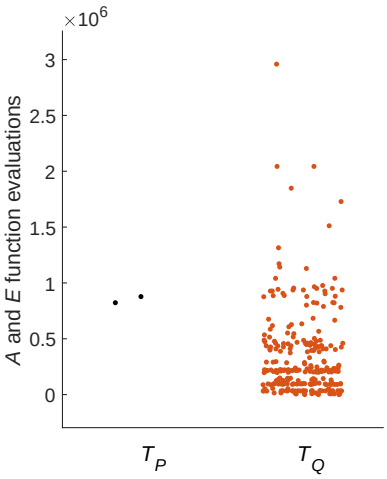


Passengers and suitcases

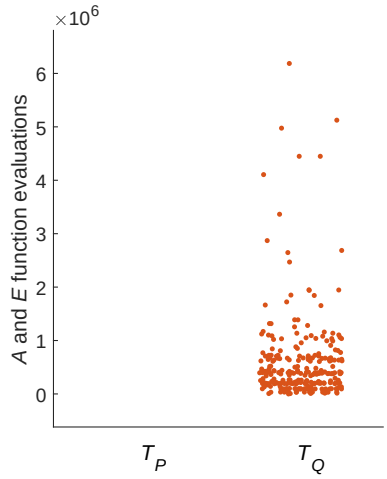
1350 × 1400 mm elevator



2000 × 1400 mm elevator



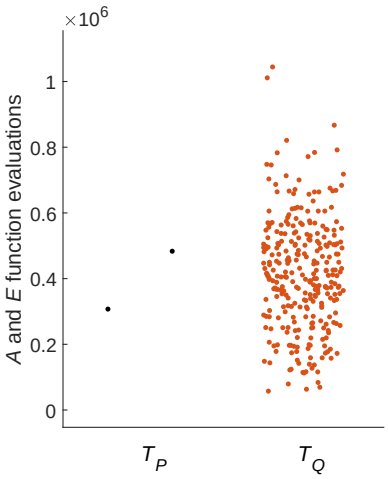
2350 × 1700 mm elevator



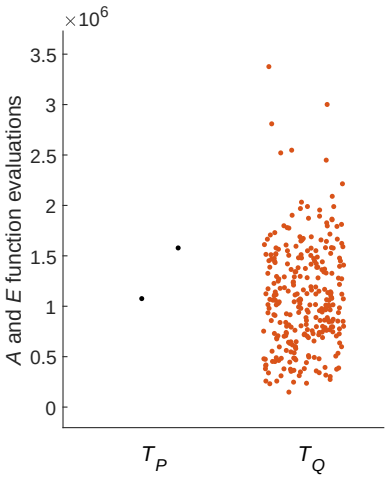
CPM with Matlab's solver

Capsule packing problem

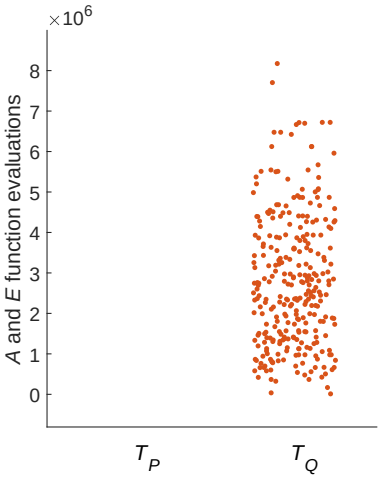
1350 × 1400 mm elevator



2000 × 1400 mm elevator

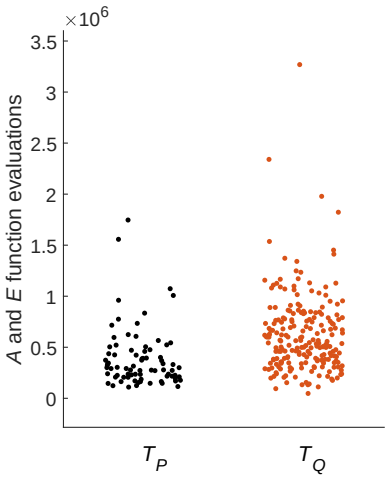


2350 × 1700 mm elevator

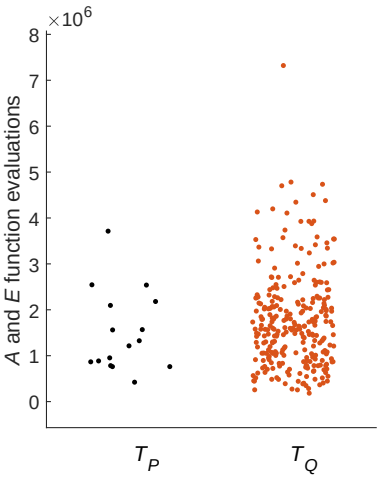


Passengers and suitcases

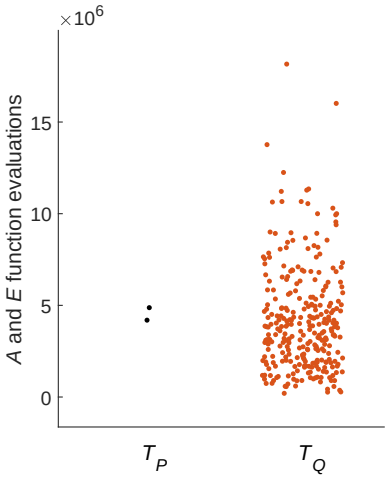
1350 × 1400 mm elevator



2000 × 1400 mm elevator



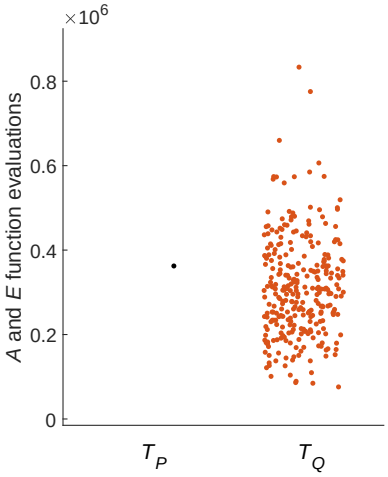
2350 × 1700 mm elevator



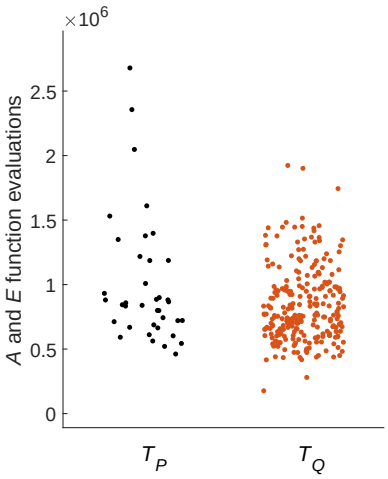
CPM with QLS

Capsule packing problem

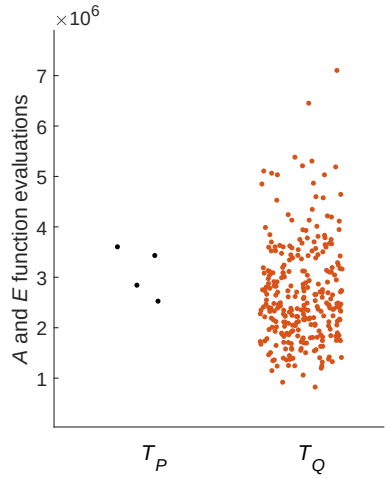
1350 × 1400 mm elevator



2000 × 1400 mm elevator

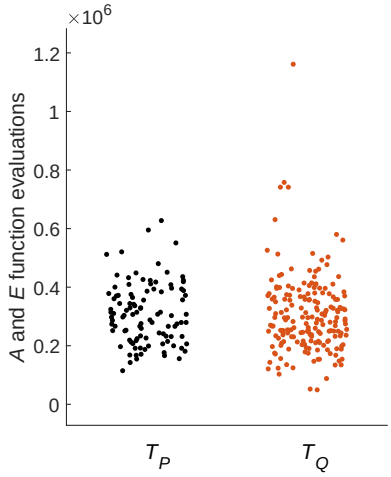


2350 × 1700 mm elevator

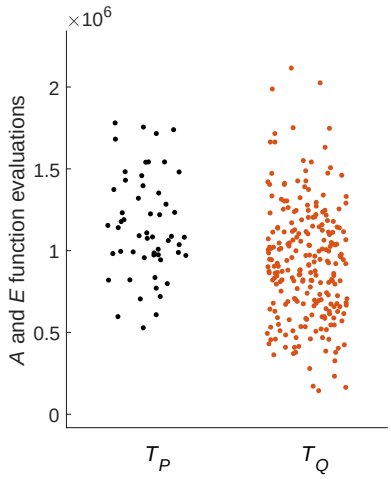


Passengers and suitcases

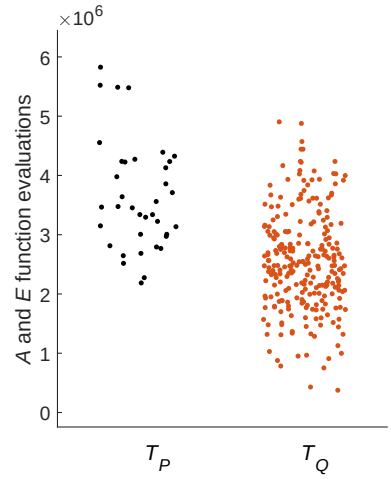
1350 × 1400 mm elevator



2000 × 1400 mm elevator



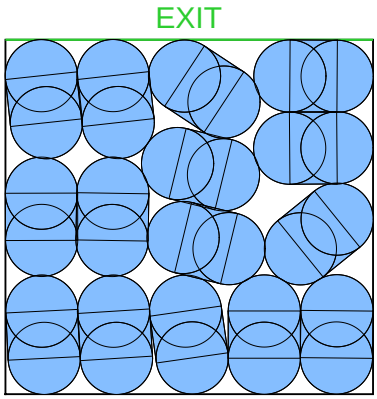
2350 × 1700 mm elevator



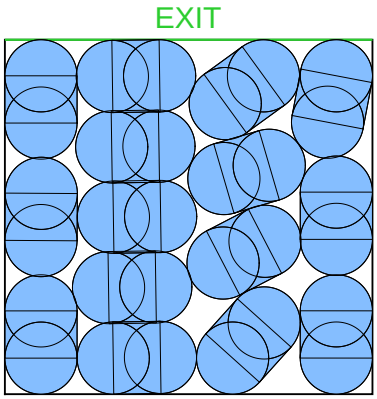
C.2 Example result for each scenario

Capsule packing problem, 1350 × 1400 mm elevator

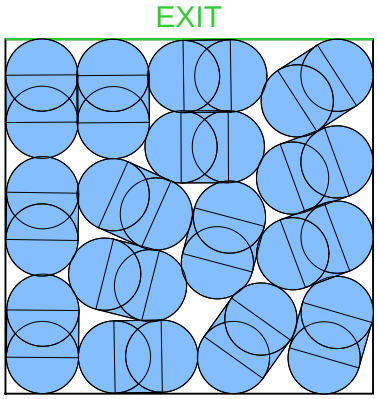
Gradient method



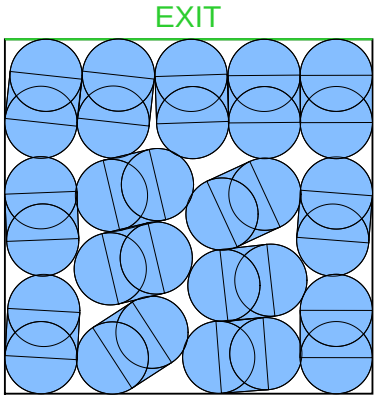
CPM with *Matlab*'s solver



BFGS

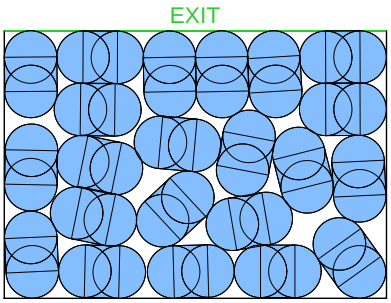


CPM with QLS

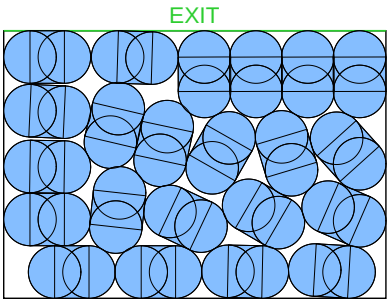


Capsule packing problem, 2000 × 1400 mm elevator

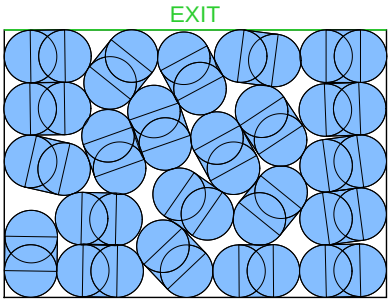
Gradient method



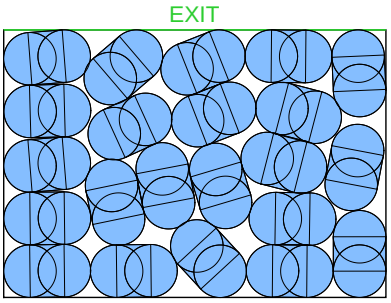
CPM with *Matlab*'s solver



BFGS

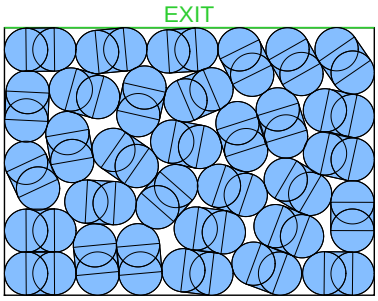


CPM with QLS

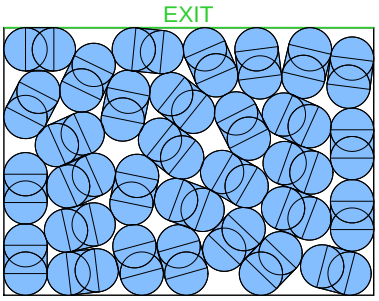


Capsule packing problem, 2350 × 1700 mm elevator

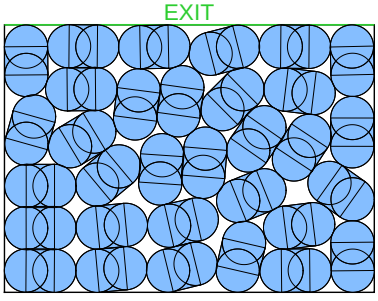
Gradient method



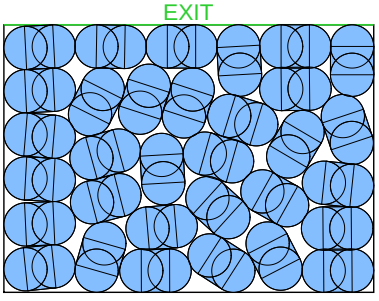
CPM with *Matlab*'s solver



BFGS

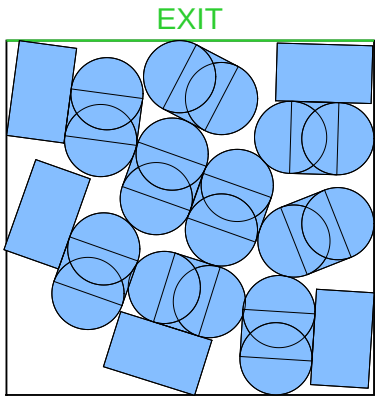


CPM with QLS

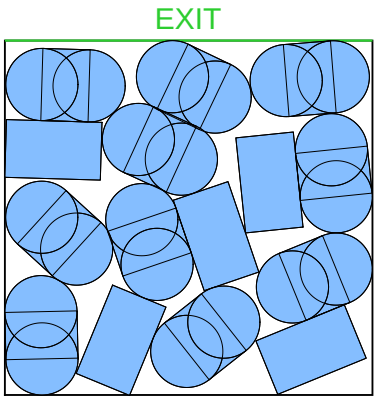


Passengers and suitcases, 1350 $\times$ 1400 mm elevator

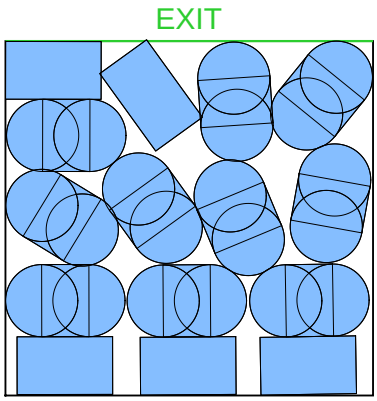
Gradient method



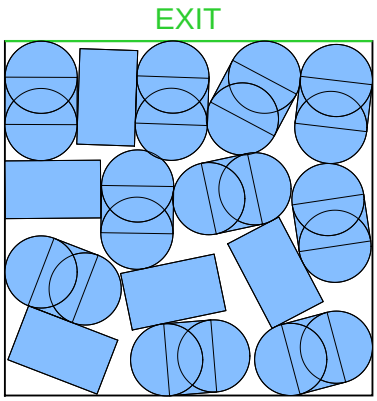
CPM with *Matlab*'s solver



BFGS

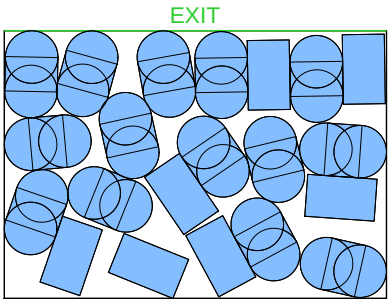


CPM with QLS

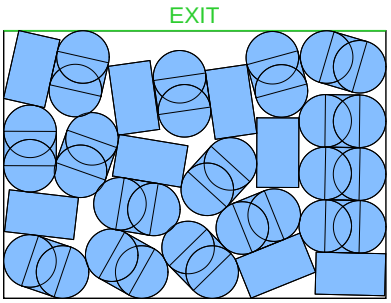


Passengers and suitcases, 2000 × 1400 mm elevator

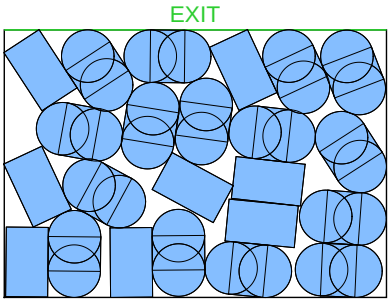
Gradient method



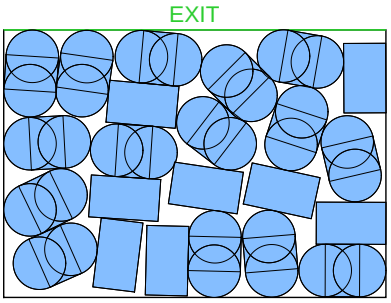
CPM with *Matlab*'s solver



BFGS

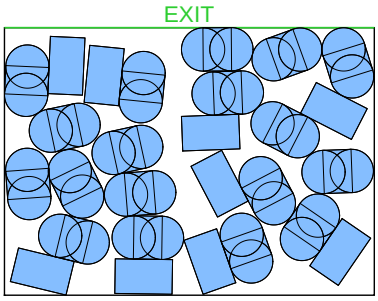


CPM with QLS

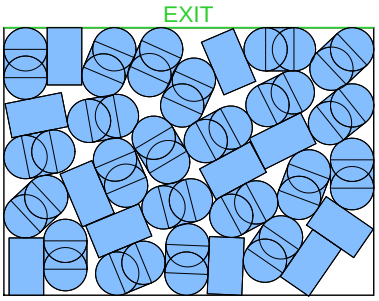


Passengers and suitcases, 2350 × 1700 mm elevator

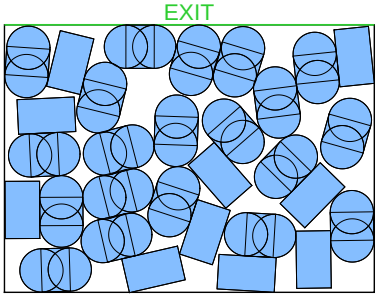
Gradient method



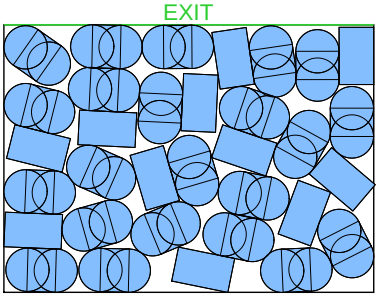
CPM with *Matlab*'s solver



BFGS



CPM with QLS



D Lemma 3

Lemma 3. *Define, $f : \mathbb{R}^n \mapsto \mathbb{R}$, $f \in C^1$. We define ∇f to be Lipschitz continuous, i.e., there is $L \in \mathbb{R}$, for which,*

$$\forall \boldsymbol{v}, \boldsymbol{w} : \ \|\nabla f(\boldsymbol{v}) - \nabla f(\boldsymbol{w})\| \leq L\|\boldsymbol{v} - \boldsymbol{w}\|. \tag{D1}$$

Define $k \in \mathbb{N}$, $\boldsymbol{x}_k \in \mathbb{R}^n$, $h_k \in \mathbb{R}$, $g_k(\alpha) = f(\boldsymbol{x}_k - \alpha \nabla f(\boldsymbol{x}_k))$, and,

$$p_k''(0) = \frac{2}{h_k}\big(g_k(h_k) - g_k(0) - h_k\, g_k'(0)\big). \tag{D2}$$

The inequality $p_k''(0) \leq L\|\nabla f(\boldsymbol{x}_k)\|^2$, holds.

Proof. With mean value theorem, there exists a $\xi_k \in [0, 1]$, for which,

$$g_k(h) = g_k(0) + h_k\, g_k'(0) + \frac{h_k^2}{2} g_k''(\xi_k h_k).$$

On the other hand, with Equation (D2),

$$g_k(h) = g_k(0) + h_k\, g_k'(0) + \frac{h_k^2}{2} p_k''(0),$$

hence,

$$\exists \xi_k \in [0, 1] : \ p_k''(0) = g_k''(\xi_k h_k). \tag{D3}$$

The Lipschitz condition (Equation (D1)) with Cauchy-Swartz inequality yields,

$$[\nabla f(\mathbf{v}) - \nabla f(\mathbf{w})]^\top (\mathbf{v} - \mathbf{w}) \leq L\|\mathbf{v} - \mathbf{w}\|^2.$$

Substituting $\mathbf{v} = \mathbf{x}_k - (\alpha + u)\nabla f(\mathbf{x}_k)$, and $\mathbf{w} = \mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)$, where $\alpha \in \mathbb{R}$, and $u \in \mathbb{R} \setminus \{0\}$, we get,

$$\big[\nabla f(\mathbf{x}_k - (\alpha + u)\nabla f(\mathbf{x}_k)) - \nabla f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k))\big]^\top (-u \nabla f(\mathbf{x}_k)) \leq L\|u \nabla f(\mathbf{x}_k)\|^2.$$

with Equation (16), we can rewrite the above inequality as,

$$g_k'(\alpha + u)\, u - g_k'(\alpha)\, u \leq L\|u \nabla f(\mathbf{x}_k)\|^2$$

$$\frac{g_k'(\alpha + u) - g_k'(\alpha)}{u} \leq L\|\nabla f(\mathbf{x}_k)\|^2.$$

Setting $u \to 0$, yields,

$$g_k''(\alpha) \leq L\|\nabla f(\mathbf{x}_k)\|^2. \tag{D4}$$

We prove the claim by setting $\alpha = \xi_k h_k$, and combining Equations (D3) and (D4),

$$p_k''(0) = g_k''(\xi_k h_k) \leq L\|\nabla f(\mathbf{x}_k)\|^2.$$

□