Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics

Master's Thesis

Optimization of Shelf Space Allocation Considering Customer Behavior

Bc. David Tomáš

Supervisor: Ing. Ondřej Vaněk, PhD.

May 11, 2015

Aknowledgements

This thesis would not exist without support of my supervisor, Ing. Ondřej Vaněk, PhD., who gave me numerous advices during making practical part and writing of this thesis.

I would like to thank to my family for their endless support during my whole life.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 11. 5. 2015 [illegible]
....

Abstract

Shopping is an everyday part of our lives. People spend there lots of money every day. Supermarkets use this phenomenon and try to maximize their profit. They do researches focused on customer's behavior to influence his purchasing decisions. The goal of this thesis is to create supermarket designer, model customer's behavior and run simulation in simulation framework and provide optimization method for product placement. Problem is formed by agent-based simulation. Simulated annealing algorithm is the main component of the thesis, it is used for optimization. Results show dependencies between product popularity and its location in the supermarket.

Keywords: Agent-based model, simulation, shelf space allocation, optimization, supermarkets

Abstrakt

Nakupování je každodenní součastí našeho života. Lidé utrácí nakupováním spoustu peněz. Supermarkety toho využívají a snaží se maximalizovat jejich zisk. Zkoumají chování zákazníků a snaží se tak ovlivnit jejich rozhodování při nakupování. Cílem práce je vytvořit nástroj na tvorbu supermarketu, formulovat a vymodelovat chování zákazníků, spustit simulace v simulačním frameworku, který poskytuje optimalizaci rozložení produktů. Je vytvořena agetní simulace daného problému. Algoritmus Simulovaného žíhání je hlavním prvkem práce; slouží k optimalizaci alokace produktů. Výsledky ukazují závislosti mezi oblíbeností produktu a jeho umístení v supermarketu.

Klíčová slova: Agentní model, simulace, alokace regálového prostoru, optimalizace, supermarkety

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Computer Graphics and Interaction

DIPLOMA THESIS ASSIGNMENT

Student: **Bc. David Tomáš**

Study programme: Open Informatics
Specialisation: Software Engineering

Title of Diploma Thesis: **Optimization of Shelf Space Allocation Considering Customer Behavior**

Guidelines:

1. Study the problem of shelf space allocation (SSA) and customer behavior modeling
2. Create a SW tool for manual design of supermarket layout including SSA
3. Design and implement a set of customer shopping behavior models
4. Implement an agent-based simulation simulating customers with defined behaviors in supermarket layouts designed in (3)
5. Define a number of metrics for SSA
6. Propose and implement an algorithm optimizing SSA with respect to metrics defined in (6)
7. Create a scenario based on a real-world supermarket, model its its layout and SSA. Evaluate the quality of your algorithm w.r.t. the real-world SSA.

Bibliography/Sources:

1. Aloysius, George, and D. Binu. "An approach to products placement in supermarkets using PrefixSpan algorithm." Journal of King Saud University-Computer and Information Sciences 25.1 (2013): 77-87.
2. Chen, Mu-Chen, and Chia-Ping Lin. "A data mining approach to product assortment and shelf space allocation." Expert Systems with Applications 32.4 (2007): 976-986.
3. Hwang, Hark, Bum Choi, and Min-Jin Lee. "A model for shelf space allocation and inventory control considering location and inventory level effects on demand." International Journal of Production Economics 97.2 (2005): 185-195.
4. Klügl, Franziska, and Ana LC Bazzan. "Agent-based modeling and simulation." AI Magazine 33.3 (2012): 29.
5. Russell, Stuart, Peter Norvig, and Artificial Intelligence. "A modern approach." Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs 25 (1995).

Diploma Thesis Supervisor: Ing. Ondřej Vaněk, Ph.D.

Valid until the end of the summer semester of academic year 2015/2016

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
L.S.
-4-

[illegible]
prof. Ing. Jiří Žára, CSc.
Head of Department

[illegible]
prof. Ing. Pavel Ripka, CSc.
Dean

Prague, March 25, 2015

Contents

Chapter 1

Introduction

Supermarkets became an everyday part of our lives long time ago. People spend there a lot of time to buy everything they need. Nowadays, the shopping in supermarkets is not only about shopping, but also about marketing and psychology research conducted to influence customers' visits in the supermarkets. Shelf space allocation and product placement problem are issues in retailing which can have an affect on the customers' purchasing decisions. The goal of the supermarket is thus profit maximization through various means, including the two aforementioned problems.

We focus on optimization of product placement in supermarkets. Agent-based approach is used to capture decision-making process of customer.

We developed a tool for creating and customizing layouts, which can be subsequently used in simulation framework. It was necessary to make a list of products which will be placed in the supermarket shelves and be available for customers.

To substitute real customers' shopping lists we implemented *Shopping list generator*, which creates random list according to actual product properties - buying probability, correlation with other products.

Customer's activity is decomposed into a set of activities. We designed behavior for each this activity for different kinds of customer model. Models use different planners which characterize their behavior. We used A* algorithm to compute optimal route between 2 points.

To optimize product placement we defined number of metrics of shelf space allocation which were minimized and maximized by Simulated annealing algorithm which is based on Local Search algorithms and it uses swapping shelf contents to generate neighbour solution. This algorithm provides very good performance for big data.

The results show that there exists a connection between product buying probability and its location in supermarket. The products that people buy most are usually placed at the remote side of the supermarket which lengthen the total distance customers have to walk.

1.1 Structure of the Thesis

1. Study the problem of shelf space allocation (SSA) and customer behavior modeling - Chapter 2 focuses on the high-level problem description, i.e. supermarkets and their layouts. Section 4.1 describes product placement researches.

2. Create a SW tool for manual design of supermarket layout including SSA - In section 5.1 we aim on the Layout designer application with all its components and functions.

3. Design and implement a set of customer shopping behavior models - Section 5.3.2 describes three model implementations with their common behavior diagram.

4. Implement an agent-based simulation simulating customers with defined behaviors in supermarket layouts designed in (3) - We design simulation framework for executing customer simulations in created supermarket. Section 5.2 is focused on the simulation framework.

5. Define a number of metrics for SSA - Introduction and description of metrics is in Section 2.4. We propose with three metrics relevant with the problem of product placement.

6. Propose and implement an algorithm optimizing SSA with respect to metrics defined in (5) - We implemented a optimization algorithm based on Simulated annealing in Section 6.2

7. Create a scenario based on a real-world supermarket, model its its layout and SSA. Evaluate the quality of your algorithm w.r.t. the real-world SSA. - Chapter 7 is aimed on simulation results and their comparing.

Chapter 2

Domain Background

The aim of this thesis is to model customer agents with specific behavior, create supermarket by real template and run reliable simulation with these environment and apply optimization algorithm which relocate product locations in supermarket and return better product placement according to optimization criterion.

2.1 Supermarkets

Supermarkets are placed almost in each city. We can find all sizes of supermarkets which have with different shelf layout and different product offer. This offer is changing with followings factors:

- supermarket size - bigger supermarkets offer more products from more producers and give customers more options to choose.
- supermarket location - comparing big cities or cities against small villages
- customers' purchasing power - similar factor as supermarket location, in region with richer people we heavily finds cheap and not so quality products.

Markets are not placed anywhere, the location is usually strategic. The bigger ones we can find at the suburbs of cities where the traffic connection is the best (usually at the exits) and the number of people moving around it is big. There should be also enough space for a parking lot.

The smaller supermarkets can be placed in the centres, where people uses public transport and do shopping almost every day but with small shopping lists.

2.2 Layout

We can describe a supermarket layout as a two-dimensional grid which defines basic supermarket shape, mentioned in Figure 2.1. In supermarket there are three important types of objects (cells) of the grid determining paths where people can move - entrances, shelves and cash desks. The paths are defined by free cells between these three types of objects.

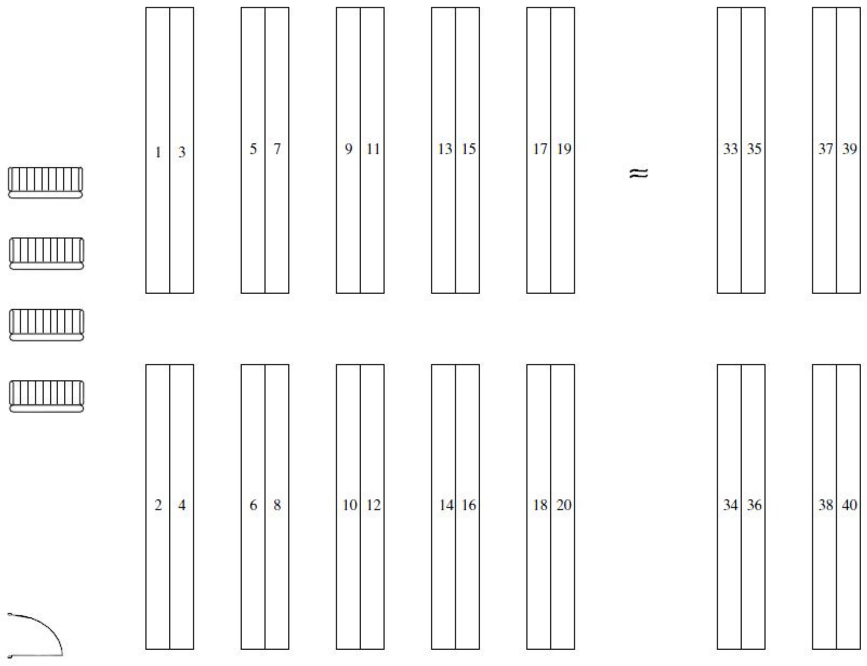

Figure 2.1: Grid allocation space in supermarkets

2.3 Product placement

Products in the shelves are not placed randomly. We can notice there are some basic rules supermarkets take into consideration while placing the products [5]. The most favourite products lie in the furthest locations from the entrances, If there is the possibility, the products which are somehow related (such as bread and the meat products) are not placed next to each other.

There is also difference in vertical product placement. Products have different popularity, according to the popularity products are placed in the shelf. The favourite ones can be found at the middle at the customer eye level. The less favourite lie at the top or at the bottom of the shelf.

2.4 Metrics

Metric is a property of the supermarket describing it layout. There are many metrics that can be used to characterization, but not all are useful for every situation. Layouts can be compared by their dimensions, number of entrances, shelves, space between shelves, etc. To describe layout for marketing purposes we define following metrics which will be computed by simulation:

- number of steps customers have to do to collect all items from their shopping list - we use maximization and minimization of this metric
- number of steps customers have to wait to avoid conflict with other customers

2.5 Optimization

Product placement optimization is a process which is used to maximize/minimize a supermarket layout metric 2.4. There are many algorithms which can be applied to solve the optimization process. Optimization is based on creating a neighbour solution from the current one and finding the optimum metric value.

2.6 Customer behaviour

Customers are not the same, we can distinguish customers by their age, gender, salary or their home place. These factors have influence on their behavior and shopping list. In the supermarkets there are many and many kinds of products of all categories and prices. We have for example rich scale of meats and every kind of this meat will find its buyer.

Chapter 3

Methods & Technology

This chapter contains description and formulation of the main algorithms and problems which are used in the thesis. At first we describe Local Search algorithm and its usage. We use optimization algorithm based on Local Search to find optimal solution of product placement. Next section is aimed on combinatorial problem called Travelling Salesman Problem which is used in customer path planner to optimize the shortest route. Planning and optimization algorithm is described in the next part this chapter. We introduce A* algorithm and Simulated Annealing algorithm. The last two sections are focused on JavaFX and its tools and on Business Intelligence tools for providing statistics.

3.1 Local Search

Local search is a method for solving hard optimization problems. It can be used when we want to find a solution with maximum or minimum criterion among many other possible solutions.

P. Moscato [12] describes basic intro for Local Search algorithm as follows: "A local search algorithm, starting from an initial solution $s_0 \in S$, iterates using the moves associated with the definition of neighborhood, such that it navigates the search space. At each step it makes a transition between one solution s to one of its neighbors s'. When the algorithm makes the transition from s to s', we say that the corresponding move t_m has been accepted and we also write that s' is equal to $s \oplus t_m$."

We can say Local search applies local changes on the current solution to create another solution from the search space. It proceeds from one solution to another until the solution is considered optimal or the time given to find solution is over.

Local Search algorithms are used with all kinds of problems, involving mathematics, bioinformatics, etc.

The most common usage of Local Search algorithm:

- Travelling salesman problem
- Boolean satisfiability problem

- Shifts scheduling problem

```
$S \leftarrow$ an arbitrary feasible solution in $S$
while $\exists S' \in B(S)$ such that $cost(S') < cost(S)$ do
    $S \leftarrow S'$
end
```

Algorithm 1: Pseudo code of Local Search algorithm

3.1.1 Local Search Techniques

This section introduce the three most favourite local search techniques. We describes Hill Climbing, Simulated Annealing and Tabu Search.

3.1.1.1 Hill Climbing

The simples technique of Local search. It is based on choosing a neighbor node with the highest value. This process is repeated until we have no nodes with higher value than the current node. The pros of this approach is time complexity, the cons is that the algorithm do not have to reach the global maximum.

3.1.1.2 Simulated Annealing

By Aarts [1] the main feature of Simulated Annealing algorithm is: "it accepts de deteriorations to a limited extent." That is difference compared to Hill Climbing approach which accepts only better solution, this approach can accept a worse solution in small range.

3.1.1.3 Tabu Search

P. Moscato [12] defines basic mechanism of Tabu Search as: "At each iteration a subset $Q \subseteq N(s)$ of the neightborhood of the current solution s is explored. The member of Q that gives the minimum value of the cost function becomes the new current solution independently of the fact that its value is better or worse than the value in s. To prevent cycling, there is a so-called *tabu list*, which is the list of moves which it is forbidden to execute. The tabu list comprises the last k moves, where k is a parameter of the method, and it is run as a queue; that is, whenever a new move is accepted as the new current, the oldest one is discarded."

3.2 Travelling Salesman Problem

Travelling salesman problem is hard optimization problem defines as follows:

```
Data: $iterations_{max}$, $temp_{max}$, $inputData$
$S_{current} \leftarrow CreateInitialSolution(inputData)$
$S_{best} \leftarrow S_{current}$ for $i$ = 1 to $iterations_{max}$ do
    $S_i \leftarrow CreateNeighborSolution(S_{current})$ $temp_{current} \leftarrow$
    $CalculateTemperature(i,temp_{max})$
    if $Cost(S_i) \leq Cost(S_{current})$ then
        $S_{current} \leftarrow S_i$
        if $Cost(S_i) \leq Cost(S_{best})$ then
            $S_{best} \leftarrow S_i$
        end
    end
    else if $\exp(\frac{CostS_{current} - CostS_i}{temp_{current}}) > Rand()$ then
        $S_{current} \leftarrow S_i$
    end
end
return $S_{best}$
```
Algorithm 2: Pseudo code of Simulated annealing algorithm for finding the best solution of product placement

"Let be C a set of cities and D matrix of distances between all the cities from C. Our goal is to visit each of given city from C and only once with the same starting and finishing city. We try to minimize total distance travelled on this route with distances from D."

3.2.1 Integer Linear Program

Mathematically TSP can be formulated as an Integer Linear Program. One of the earliest formulation of ILP is due to Dantzig, Fulkerson and Johnson [3]:

$$Minimize \sum_{i \neq j} c_{ij} x_{ij} \tag{3.1}$$

subject to

$$\sum_{j=1}^{n} x_{ij} = 1, i = 1, \dots, n \tag{3.2}$$

$$\begin{gathered}\sum_{i=1}^{n} x_{ij} = 1, j = 1, \dots, n \\ \sum_{i,j \in S}^{n} x_{ij} \leq |S| - 1\end{gathered} \tag{3.3}$$

$$\begin{gathered}S \subset V, 2 \leq |S| \leq n - 2 \\ x_{ij} \in \{0, 1\}\end{gathered} \tag{3.4}$$

$$i, j = 1, \dots, n, i \neq j \tag{3.5}$$

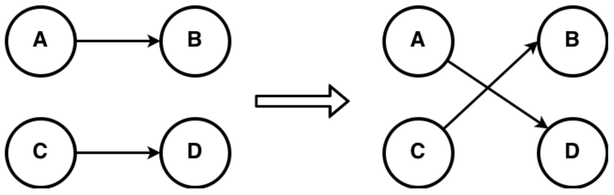

Figure 3.1: Demonstration of 2-opt algorithm.

TSP is $NP-complete$ and $NP-hard$ problem. This means there is no polynomial algorithm which can find a solution in polynomial time.

3.2.2 Approximation methods

We know many heuristics and approximation algorithms which quickly find a good solution. This solution does not have to be the best one, but it can be found quicker than the optimal solution.

3.2.2.1 r-opt heuristic

Laporte [8] defines r-opt heuristic in 2 steps:

1. Consider an initial tour.
2. Remove r arcs from the tour and tentatively reconnect the r remaining chains in all possible ways. If any reconnection yields a shorter tour, consider this tour as a new initial solution and repeat step 2. Stop when no improvement can be obtained.

Example of 2-opt algorithm is shown at Figure 3.1.

3.2.2.2 Nearest Neighbor

Nearest neighbor belongs to the easiest and fastest algorithm for TSP. Karkory [6] formulates this approach as follows: "Select a starting point, as long as there are cities that have not yet been visited, visit the nearest city that still has not appeared in the tour, finally, return to the first city." The time complexity of this approach is $O(n^2)$.

3.3 A*

A* [13] is a planning algorithm used for finding a path in a graph with given start and given end points. Best-first search algorithm is used to find an optimal path between points X and Y; *optimal* is meant as the shortest or cheapest, etc. We define function *f(x)* 3.6 which determines node value and with this value it sorts out the nodes to the right order in which these nodes will be visited and evaluated. Function *f(x)* is composed of these 2 fuctions [9, 11]:

- **g(x)** - *cost* function - distance between start and parameter point x

- **h(x)** - *heuristic* function - the best predicted distance to end, must be admissible (distance to target must not be overestimated)

and is given as:

$$f(x) = g(x) + h(x) \tag{3.6}$$

Pseudocode 3 below shows how A* works [1].

```
init Priority Queue;
init closed list;
put start point to the queue;
while open list is not empty do
    pop ($q$) off the queue;
    generate $q$'s successors and set their parents to $q$;
    foreach successor do
        if successor == goal then
            return;
        end
        successor.c = q.c + distance between successor and q;
        successor.h = distance from goal to successor;
        successor.f = successor.g + successor.h;
        if queue contains node with the same position and with lower f than
        successor then
            break;
        end
        if closed list contains node with the same position and with lower f than
        successor then
            break;
        end
        put the node to the queue;
    end
    push q on the closed list;
end
```

Algorithm 3: Pseudo code of A* algorithm

A* is complete algorithm; it will always find a solution if there is someone. It's also optimal if:

- closed list (list of visited nodes) is not used
- closed list is used and heuristic function *h(x)* is consistent; for any pair of nearby nodes *a*, *b*:

$$h(x) \leq h(y) + d(a, b) \tag{3.7}$$

where *d(a,b)* is distance between *a* and *b*.

[1] `http://web.mit.edu/eranki/www/tutorials/search/`

3.4 Java & JavaFX & JFXtras

JavaFX is a Java framework for making applications with rich and intuitive GUI. It was introduced in May 2007 at *JavaOne* conference. In the third quarter of 2008 JavaFX was published for desktop and web browsers. In the course of the years JavaFX officially replaced *Swing* library, which was considered to be an old technology lacking useful features (e. g. it does not support touch screens, animations, etc.).

Summary of JavaFX features:

- Scene Builder - Tool for creating GUI without writing code. It supports Drag&Drop technology to customizing the interface. It generates FXML code, which contains all interface properties with all components and their position and providing methods for interacting.
- FXML - language used only for defining application interface, based on XML
- Rich UI Controls - it provides charts generator, SVG shapes
- 2D/3D Graphic Support - it provides technology for animations and transitions. It is possible to manipulate with 3D objects.
- CSS-Like Styling - we can style component by setting css styles instead of updating objects one property by another property.

3.4.1 JFXtras

JFXtras[2] is a supporting library for JavaFX. It provides helper classes, new extended layouts, improved controls, menus and many others widgets not available in standard JavaFX. It is an open-source software under BSD License. It is available in maven repository and it is possible to download only a part of the components. JFxtras consists of the following parts which can be downloaded separately:

- jfxtras-common - layouts, utilities, etc.
- jfxtras-fxml - utilities for working with FXML file format
- jfxtras-controls - time and date pickers
- jfxtras-agenda - Google Calendar control
- jfxtras-window - non standard Window implementation
- jfxtras-menu - corner, circular or radial menu implementations
- jfxtras-labs - incubator classes

[2]`http://jfxtras.org/`

3.5 Business Intelligence Tool

Business Intelligence are experiences, knowledge, risks and methods in entrepreneurship which help for better understanding of business connections and market behavior. Business Intelligence collects data and makes analysis for better business decision making process.

Business Intelligence applications (tools) provides all kinds of data projection, reporting, corporate performance management, support for planning to the future, etc.

There are lot of corporate tools used by many companies. The most used are these ones:

- IBM Cognos [3]
- SAP Business Object [4]

For some little projects or small companies there is also a solution of open-source or freeware tools which have similar functions but they cannot be used for so many data.

- Tableau Software [5]
- QlikView [6]
- icCube [7]

[3] `http://www-01.ibm.com/software/analytics/cognos/`
[4] `http://go.sap.com/solution/platform-technology/business-intelligence.html`
[5] `http://www.tableau.com/`
[6] `http://www.qlik.com/`
[7] `http://www.iccube.com/`

Chapter 4

Related work

This chapter focuses on publications and papers with similar goals or methods. We introduce researches in product placement problem and Agent-based simulation approach for modeling reality.

4.1 Product Placement

Mu-Chen Chen in his article A data mining approach to product assortment and space allocation [2] studies problem with product placing to shelves according to their values and influence to profit. "This study developed a data mining approach to make decisions about which products to stock, how much shelf space allocated to the stocked products and where to display them."[2]. He propose a procedure of shelf space management which consists of three steps:

- Multi-level association rule mining - relations between product items, product subcategories and product categories
- Product assortment - estimation of the frequent item profits, resolving mathematical model of product assortment, generating basic and added products for store
- Shelf space allocation - allocation shelf space for each type of category, subcategory and items.

He cites Yang article about shelf space allocation algortihms [14]: "Product assortment and shelf space allocation are two important issues in retailing which can affect the customers' purchasing decisions. Through the proficient shelf space management, retailers can improve return on inventory and consumer's satisfaction, and therefore increase sales and margin profit".

According to M. Levy [10], retailers usually adopt the grid display to allocate the shelf space. The grid display has the longer demonstration shelf and walkway as shown in Figure 2.1. This layout is known from many supermarkets.

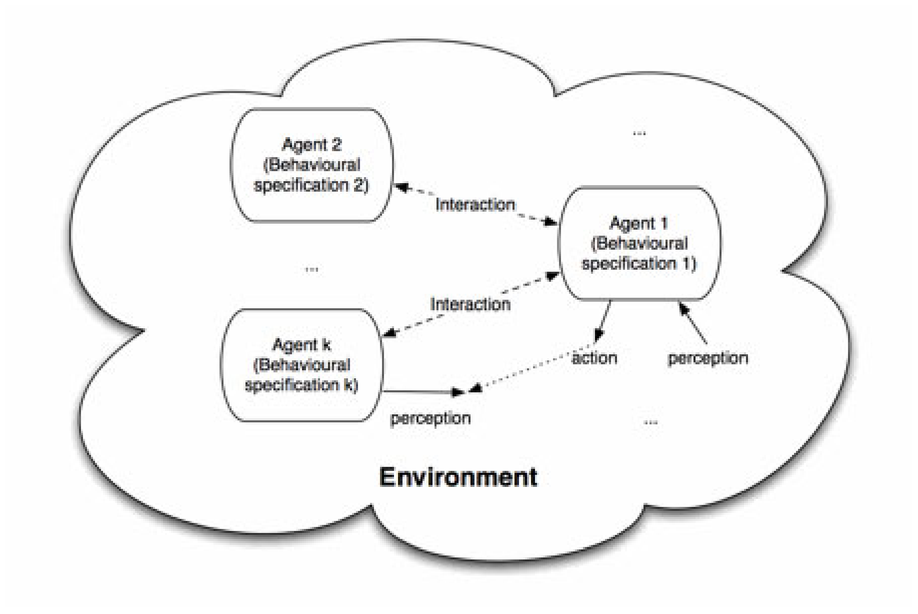

Figure 4.1: Illustration of agent-based model structure.

4.2 Agent-based Simulation

The best known paper about Agent-based Simulations is written by *Franzisca Klügl* and it is the *Agent-Based Modeling and Simulation* article [7]. She describes agent-based modeling and simulations as follow: "Multi-Agent Systems provide a description framework that is appropriate for many real world systems consisting of a set of interacting autonomously deciding actors. Human and animal societies form prominent and intuitive examples for real-world multi-agent systems." Basically we can transform real world to agent-based one. We define people, animals, etc. as agents who are interacting with environment or with other agents and who make sense of the real world. Figure 4.1 provides an illustration of models interacting[1].

4.2.1 Agent and Environment

Autonomous agent is described by Franklin and Graesser [4] as follows: "An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future." Environment in agent-based simulations is everything the agents can interact with.

4.2.2 Creating an Agent-based Model

According to Klügl [7] the creation of an *Agent-based model* is formed by these three elements which have to be defined:

[1]`http://jasss.soc.surrey.ac.uk/12/4/4/ref-model.gif`

- The set of agents is the most characteristic element. These agents are autonomous with respect to the other entities within the simulated environment.
- Next comes the specification of the interactions of the agents among themselves and with their shared environment.
- The third element, the simulated environment, contains all other elements. These may be resources, other objects without active behavior, as well as global properties.

Chapter 5

System Architecture

Following chapter describes the main components of the project. We introduce Layout designer - application for creating supermarket layout, Simulation framework - a set of components for running simulations with optimization goals, customers models which represent real customers in supermarkets and optimization process of product placement.

5.1 Layout designer

Layout designer is an application for modelling supermarket layouts 2.2 which can be used in Simulation framework. It allows us to create a new custom layout or update an existing one. Designer is fully accessible via UI to be easier for the usage.

It consists of two main cooperating components. The **main content component** is for the interaction with the user and the **info panel** is for navigating between windows and for displaying some useful information.

Layout modelling flow consists from 4 steps:

1. Setting supermarket's dimensions or loading existing one
2. Placement of supermarkets objects
3. Product placement on shelves
4. Saving layout

5.1.1 Wizard

Wizard component provides methods for navigating in this step flow. Steps (pages) are validated to offer only the correct solutions. Figure 5.1 captures wizard with its components.

5.1.1.1 Step 1

The initial step (shown at Figure 5.2 provides input fields for supermarket's dimensions and button for loading an existing supermarket. Input fields are validated to accept only positive values and multiples of fifteen (fifteen units are a supermarket cell).

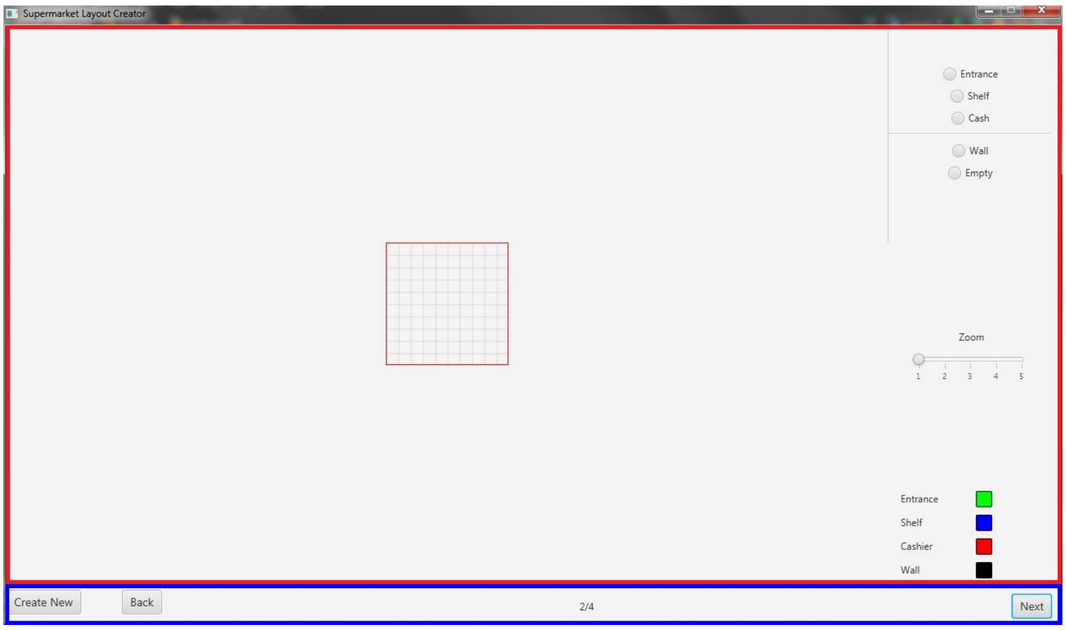

Figure 5.1: Wizard component description with its highlighted parts. Content - red, Controls - blue

Loading button reads the supermarket file and presets values to input fields which can be edited to customize the loaded supermarket.

5.1.1.2 Step 2

Figure 5.3 contains page with number two which shows supermarket layout in the big canvas covering most of the window and controls for customizing this layout, i.e. checkboxes in the right panel. Layout can be customized by clicking on it cells. The click places the selected value from checkbox to layout and shows the change on the canvas. Checkboxes are supplemented by image which describes their meaning.

Supermarket's layout is represented by two-dimensional array where each cell of this array is defined by one of these following objects:

- Entrance
- Shelf
- Cashier
- Wall
- Empty space

Step two provides zooming layout by slider component in the middle of right panel. We can apply zoom from 1 to 5 to get bigger layout projection and more precise customizing.

The last components is a small layout with legend which describe cell meaning by its color.

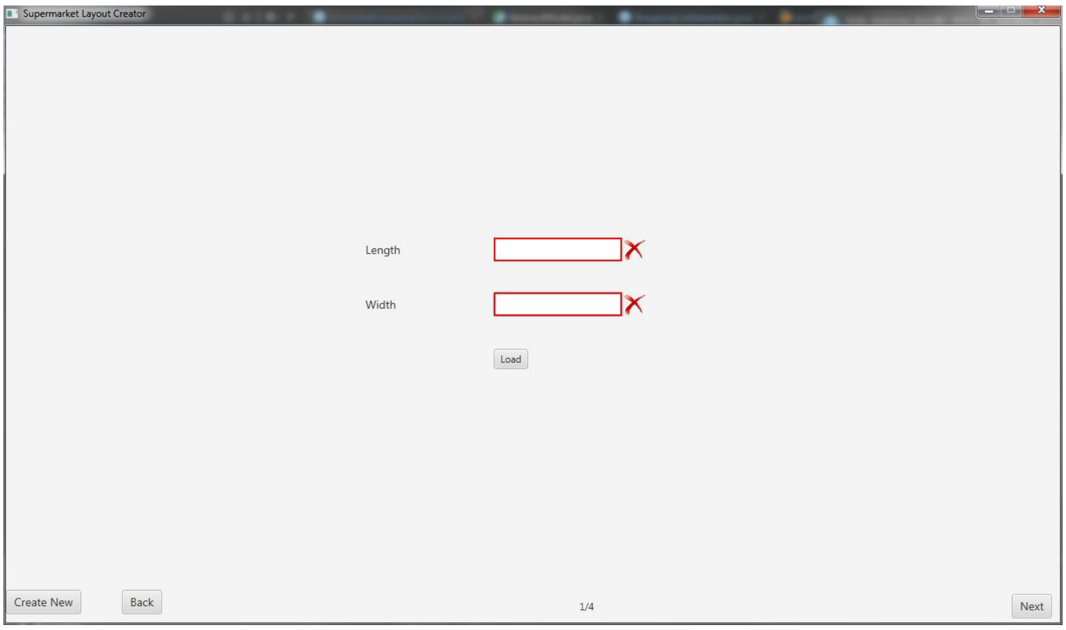

Figure 5.2: Step 1 for setting the supermarket's dimensions

To get to the next step layout has to contain at least one item from each of these types: entrance (green), shelf (blue) and cash desk (red).

5.1.1.3 Step 3

The third step is aimed on product placement; On this step we can put products to shelves. Figure 5.4 shows layout of the step 3 which is divided into 2 vertical parts. The left one contains the same canvas as the step 2; i.e. supermarket layout with all its objects. The right one shows shelf detail with product offer.

Shelf space used for products is inspired by typical shelf in supermarkets, a rectangular space with grid layout. A shelf usually contains one type of products from various producers.

To add a product to shelf it is necessary to perform these tasks:

1. Select shelf we want to modify - selected shelf is highlighted by light blue color.
2. Click on the one of the "Fill" buttons or click on the certain shelf space.
3. Select product from menu

Product selection uses a hierarchical menu as shown at Figure 5.5. Each menu level corresponds to product hierarchy levels:

1. level - category
2. level - subcategory
3. level - product

The menu remembers the last used product to provide its new quick usage. It helps the faster product selection.

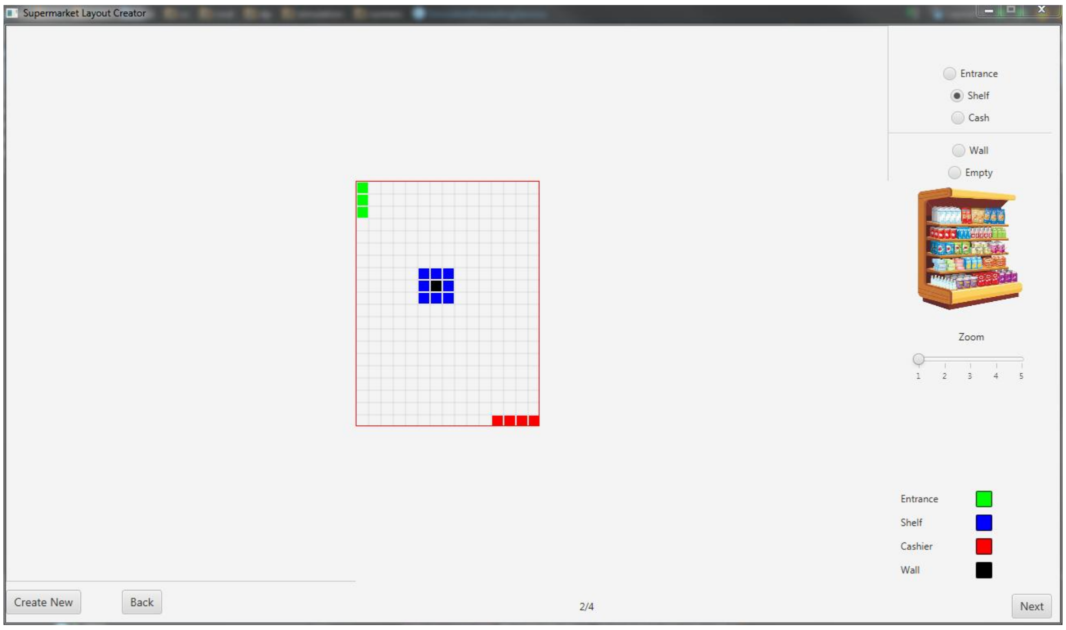

Figure 5.3: Model behavior diagram

5.1.1.4 Step 4

Last step of all with only one button for provoke the saving dialog. Supermarket's layout is stored in JSON format which is suitable for little corrections and modifications. In this layout, there are supermarket's dimensions, list of shelves with their product placement and location in the supermarket and list of entrances and cash desks with their locations. Demonstration of stored supermarket in JSON format is in Appendix B.

5.2 Simulation framework

Simulation is runnable in a framework which runs the evaluation using the supermarket and its defined models. Simulation is step-based, i.e., it runs in steps which define certain action of the model according to its state. Simulation starts with initializations of supermarket (loading layout and products properties) and models (starting positions and shopping list). Shopping lists are automatically generated by Shopping list generator 5.2.1

5.2.1 Shopping list generator

Some of the products are to be bought more frequently and some other products are purchased only irregularly. It can be said that some products have a higher probability or purchase, which signifies their popularity.

Another feature of the probability of the purchase of a product is a relation which can exist between a pair of products (P1, P2). It deals with the probability that if we buy a product P1 the we buy a product P2. The relation is symmetric; it is the same in both of the directions. All relations are stored in a correlation matrix. The matrix

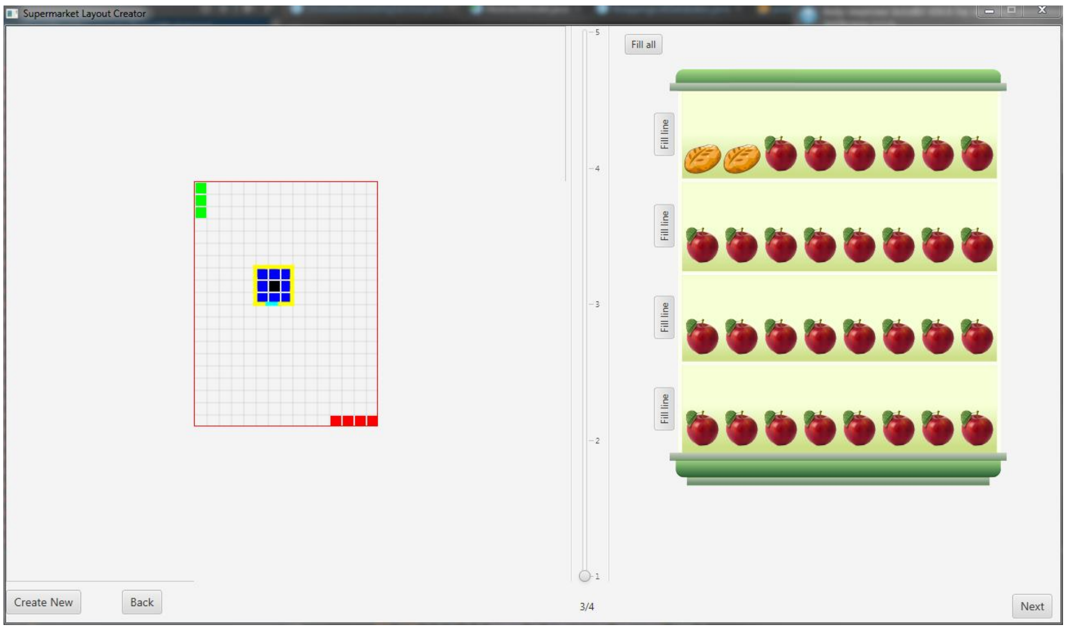

Figure 5.4: Model behavior diagram

contains all products values in the column and row labels and the values at a position told us the probability. Example of correlation matrix is in the Appendix C

5.3 Behaviour modelling

Customer models are big part of whole simulation framework and its evaluation. The model represents a real customer in a supermarket with defined shopping list and familiarity with the supermarket. The knowledge determines which planning algorithm will be used.

Each model is defined by these properties:

- Familiarity with the supermarket's layout
- Planning (behavior) algorithm
- Shopping list

5.3.1 Behavior diagram

Model's behavior is controlled by behavior state diagram 5.7. It defines states and transitions between them.

5.3.1.1 Wait

Model is created and waits for his starting position in the environment.

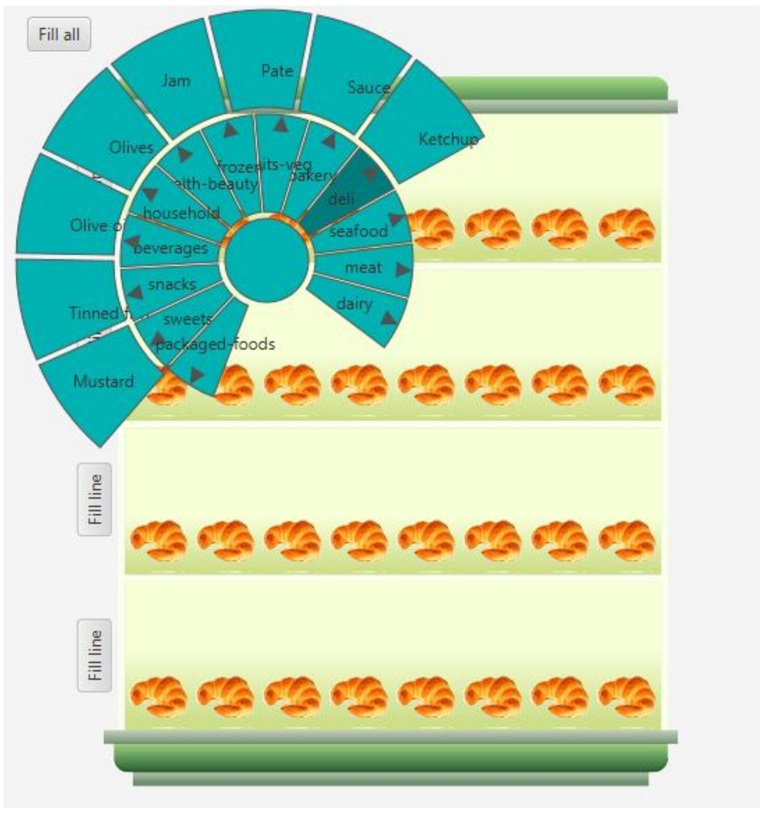

Figure 5.5: Menu for selecting products

5.3.1.2 Plan

This state contains creating route according to model type. It plans complete route from start to end.

5.3.1.3 Follow

Following state represents one model move. The move can be a step on another cell, waiting on the current cell to avoid conflicts with other models or waiting in sense of choosing the current item from list as described in Section 5.3.1.4.

5.3.1.4 Choose

It state is only virtual, is not implemented. It is used for describing customer behavior in real. In the code, this state is substitute by planning state which controls all the customer moves.

5.3.1.5 Leave

Leave state is also only virtual to distinguish 2 different types of customer moves:

- Move to next product in list
- Move to random cashdesk

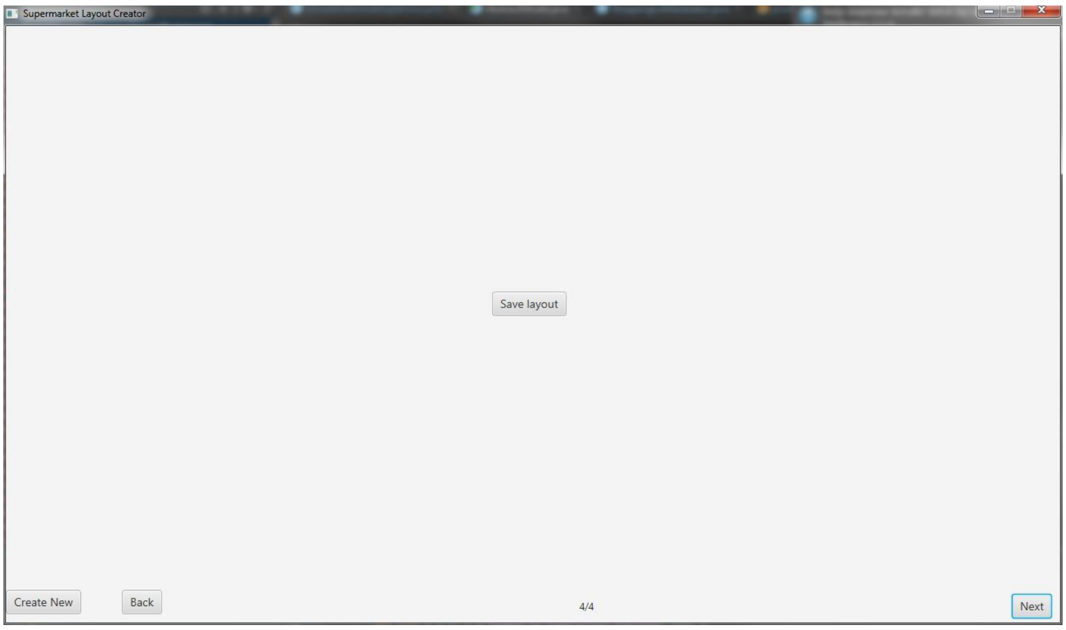

Figure 5.6: Window for storing supermarkets

5.3.1.6 Out

In this state model reach his destination (cashdesk) and is ready for leaving out of simulation.

5.3.1.7 Done

State Done means model is not act in the simulation. He bought all items from shopping list, paid for them at cashdesk and leave the supermarket.

5.3.2 Implemented models

In the simulation there are 3 types of customer models implemented.

5.3.2.1 Travelling Salesman Problem Model

So called TSP model represents a person who finds the shortest way to buy all of the products on the list and goes away. The model is well informed about the supermarket, he knows where all products are placed.

5.3.2.2 Random Model

Random model is the simplest model. From its shopping list it chooses always random item and go for it. It is also well informed about the supermarket as TSP model.

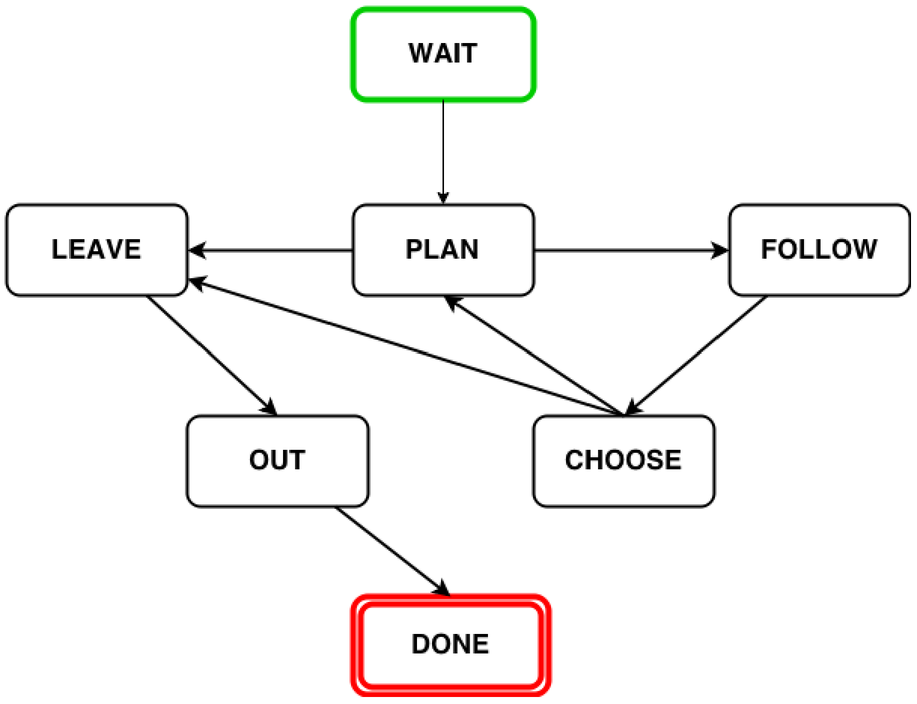

Figure 5.7: Model behavior diagram

5.3.2.3 Partial knowledge Model

Model with the partial knowledge of the supermarket is a model who knows positions of a given percentage of the products. The rest of the percentage is a knowledge about the product categories. This means that the model does not know the exact location of the product he wants, but he knows where to look for the category to which the product belongs.

The planner this model uses iterates all products from shopping list and finds the nearest product to him. If the product position is unknown, the planner use nearest point of the category the product belongs to. If there is a Partial Knowledge in the simulation the software initialize category info provider which is used by the planner a keep information about all categories and its products. Algorithm is described by pseudo-code 4

5.4 Placement optimizer

Placement optimizer used Simulated annealing algorithm to find an optimal solution of the metric in the selected supermarket.

The optimizer controls supermarket simulations and collects results to find the best the best supermarket configuration. Thirty simulations in each algorithm iteration are run to get relevant results. After each iteration the neighbour solution is generated by swapping shelves content. Number of swapped shelves are given by this equation:

$$swap = \min((temperature/50) + 1, 20); \tag{5.1}$$

```
product ← findNearestProduct()
if product position is known then
    getPathTo(productPosition)
end
else
    categoryPoints ← getCategoryPoints(product)
    nextPosition ← getNearestCategoryPoint() getPathTo(nextPosition)
    while current position != product position do
        nextPosition ← getNearestCategoryPoint()
        categoryPoints → remove(nextPosition)
        getPathTo(nextPosition)
        currentPosition ← nextPosition
    end
end
```

Algorithm 4: Pseudo code of Partial Knowledge model planner

Number of swapped shelves is not greater than twenty and greater than zero. This guarantees that the higher the temperature is the higher the number of swapped shelves is.

Chapter 6

Implementation

Chapter Implementation aims at particular components implementation. We introduce components from JavaFX library used for Layout designer, algorithm for path planning with no collisions and *Maven* tool for managing projects.

6.1 Layout designer

Layout designer is step-based flow creator. A wizard component for manipulating with the pages is the main part of designer GUI. Wizard comprises two parts as shown in 5.1:

- Content
- Controls

Content part displays page layout, component with controls is responsible for flow.

6.1.1 Products

Customer in simulation can buy lot of products from many categories. List in Appendix D shows all available products which represents set of typically bought products.

Products are divided into categories and subcategories to provide a simple and easy navigation in the designer product menu. Each of these products has its picture which is shown in the designer and is used for graphic shelf layout description.

For the products' selection, *Radial menu* from *JFXtras* library is used. *JFXtras* contains controls and add-ons for *JavaFX*. *Radial menu* is control component providing intuitive flow which makes the product selection easier. The flow for products with a subcategory is different from the flow without any subcategory.

6.2 Optimization

Product placement optimization uses Simulated annealing algorithm. Component *SimulatedAnnealingService* is responsible for running this algorithm with given parameters. This service implements *Runnable* interface to be run in separate thread. It

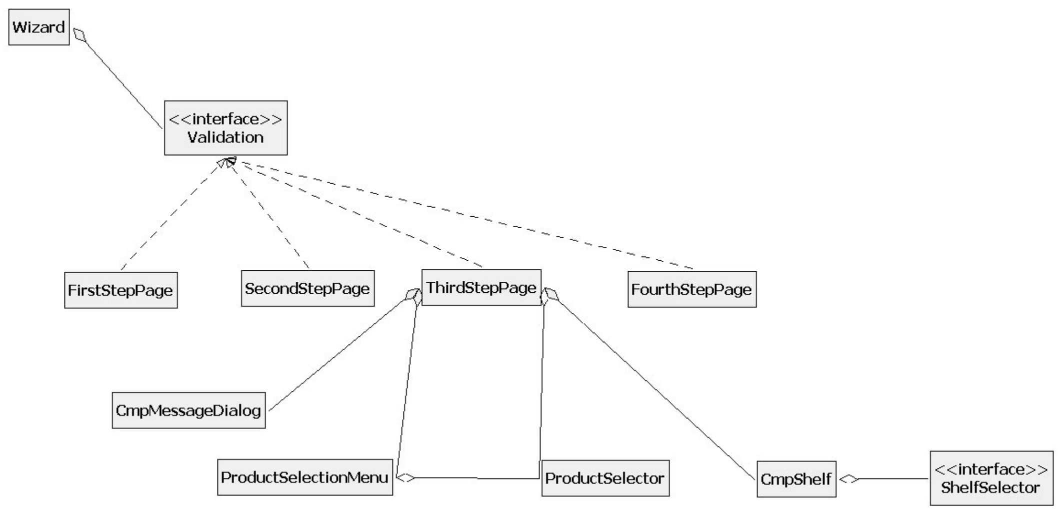

Figure 6.1: Simple class diagram of Layout module

provides method for run given number of simulations and returns the result to be processed by optimization algorithm.

The simulation parameters consist from *Supermarket* object and List with model definitions to be used in simulation. List of models is consist for the whole simulation run, the supermarket is changed after each temperature recomputing. *SimulatedAnnealingService* applies operator which swaps the shelves' content.

We define an equation for the temperature lowering:

$$temp = temp * (1 - coolingRate) \tag{6.1}$$

where *coolingRate* is equal 0.03 and initial temperature value is equal to 2000. These values provides 251 simulation runs.

6.3 Planning

Customer planners use A* algorithm on three-dimensional graph. Two of the dimensions are based on the supermarket layout (x and y coordinate). The third dimension is step (time) variable. This structure provides all the properties needed to find multiple paths which are not crossed (in reality there cannot be two people on the same place - their paths cannot be crossed).

Graph is implemented as *java.util.Map* with key of type *Long* that represents coordinate unique ID and value type *Coordinate* describing a location on graph. Coordinate ID is computed by this equation:

$$id = (y \cdot X + x) + (S - 1) \cdot X \cdot Z \tag{6.2}$$

where x and y are coordinate position on graph (layout), X and Y are layout dimensions and S describes coordinate step property (time variable).

Because of the big memory requirement, the graph is implemented as *lazy*, it adds it coordinates dynamically when they are required. This solution provides a fast access to a certain node and it is useful for a big layout with lots of steps.

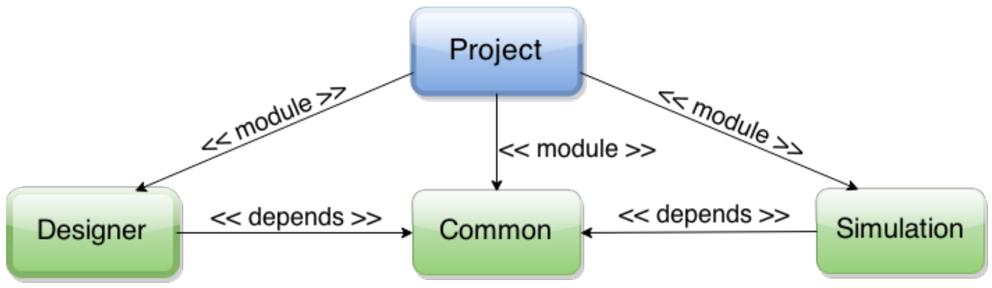

Figure 6.2: Maven module diagram describing project structure

```
cz.cvut.dp:dp:pom:1.0-SNAPSHOT
├── org.jfxtras:jfxtras-controls:jar:8.0-r2
│   └── org.jfxtras:jfxtras-fxml:jar:8.0-r2
├── org.jfxtras:jfxtras-common:jar:8.0-r2
├── org.jfxtras:jfxtras-menu:jar:8.0-r2
├── org.jfxtras:jfxtras-labs:jar:8.0-r1
└── com.google.code.gson:gson:jar:2.3
```

Figure 6.3: Maven tree of external library dependencies

6.4 Maven

Maven is a tool for managing, controlling and build automation. It is used mostly with *Java*. It describes software build properties - how the software is built and what external sources and libraries it depends on.

Project implementation is built on the *Maven* module structure to keep the management of dependencies and libraries for all the modules simple. Project structure is described by the diagram 6.2 which shows all of the used modules and the dependencies between them.

Chapter 7

Evaluation

We evaluated two real-life scenarios: one small supermarket, one bigger supermarket. Each time we ran these three optimization metrics:

- Minimization of customers' steps
- Maximization of customers' steps
- Minimization of customers' waiting time

To compare two different alternatives, we evaluated one customer as well as more customers. We use TSP model and model with partial knowledge to get the results. Random model is used only for making noise in more customers scenarios.

7.1 Layout designer use-case

At first we evaluated small scenario with small number of customers, than the bigger scenario with more customers.

7.1.1 Scenario 1 - small

The small scenario uses the supermarket 7.1 which is inspired by Tesco Express in Taborska street, Prague. It is small local market with limited range of products. It is designed for small a short shopping.

7.1.2 Scenario 2 - bigger

Next simulation scenario uses bigger supermarket. We use supermarket Tesco in Uničov. This supermarket is a center for about ten thousand people from Uničov and adjacent districts. It serves for all kinds of people so it offers many products of all prices.

Instead of one and ten people from the small scenario we use 30 customers to buy there with only one simulation run.

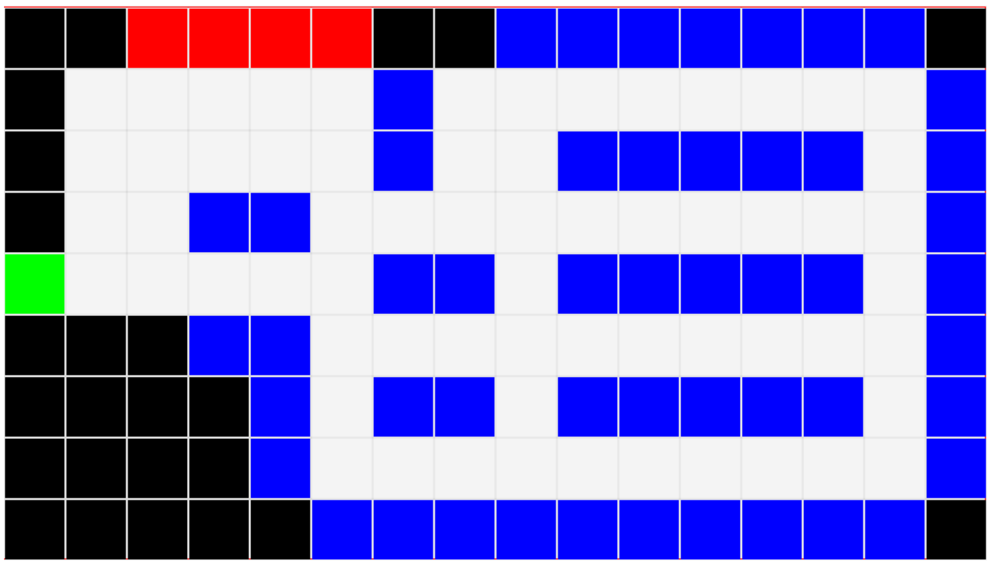

Figure 7.1: Small supermarket layout

7.2 Optimization

Simulated annealing algorithm which is used for all metrics. Simulation result is computed as arithmetic mean from 30 simulation executions with same settings to achieve relevant results. This simulated annealing process is run three times to verify similar results.

7.2.1 Scenario 1 - small

At first the small scenario is evaluated.

7.2.1.1 One customer

For the first scenario we have one customer with *Travelling Salesman Problem* planning strategy. On the Figure 7.3 there are progress of minimization and maximization of total steps customer has to do. We can see the difference between maximum and minimum value of 20. It is about 9-10 percent change compared to default layout.

Second evaluation uses one Partial Knowledge model with 50 percent familiarity with the supermarket. The result of minimization is the same is at TSP model. The maximum value is twice bigger compared to TSP (he has familiarity of 100 percent). The biggest weakness of this model the low value of familiarity with the supermarket. He has to do many steps to find a product. Bigger familiarity would cause less maximum value. This model path is very influenced by the familiarity. Figure 7.4 captures optimization process of a Partial Knowledge model.

The reason of minimization is that the favourite products were placed near entrance or cash desk and less favourite ones were placed on the other side of the entrances or cash desks.

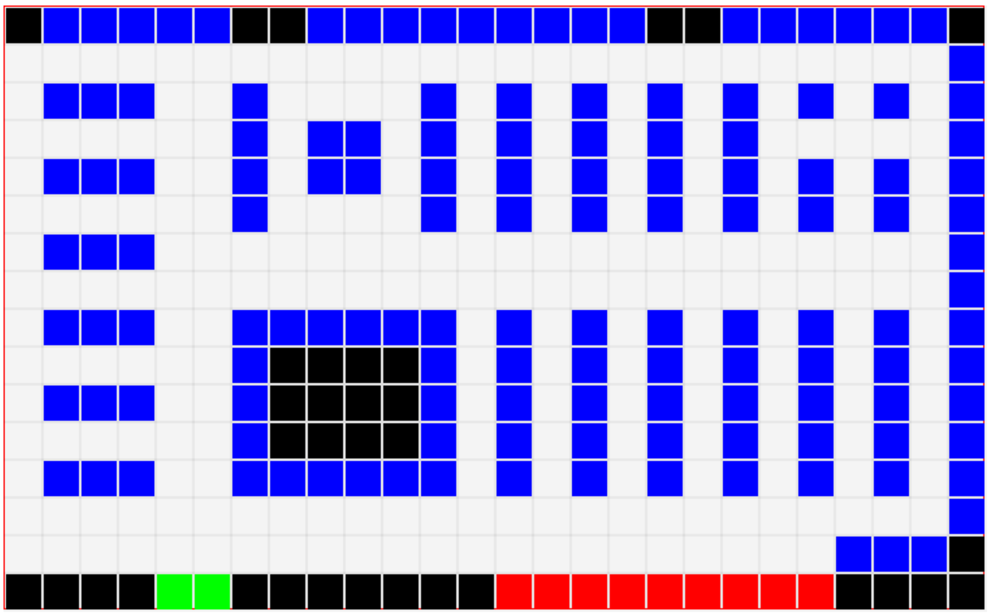
Figure 7.2: Big supermarket layout

7.2.1.2 More customers

Now we put 10 TSP customers to the small supermarket to do shopping. The difference of maximum values compared with scenario with one customer is about 10 percent (captured at figure 7.5 and it is given by more density of customer in the same area. Customers have to do more steps to avoid conflicts.

Ten Partial Knowledge customers have similar results (figure 7.6 as ten TSP customers. The maximum value is bigger about fifteen percent, the minimum value is equal to the one from the evaluation with one customer. The reason is the same - more density.

The last measuring was focus on waiting steps, i.e. how many time customers have to wait on its position to avoid conflicts with other customers. The result at Figure 7.7 of 20 steps is about 20 percent of done steps. The result is influenced by little space to move and high number of customers.

7.2.2 Scenario 2 - bigger

This section is focused on evaluation of the bigger scenario. We ran only one simulation to get results but with higher number of customers (30).

We put thirty TSP models to bigger supermarket. Figure 7.8 shows result that is relevant to the supermarket size. Customers do not have to avoid conflicts, there are lots of space between shelves and many shelves with the same product.

Big supermarkets shows the difference between TSP models (with 100 percent familiarity and Partial Knowledge models (with 50 percent familiarity) at figure 7.9. Knowledge of the supermarket strongly decreases the total steps needed to collect all items from list. Finding products through all shelves in the main reason of the high number (about 420 steps - twice as much compared with TSP). Figure 7.10 captures the progress

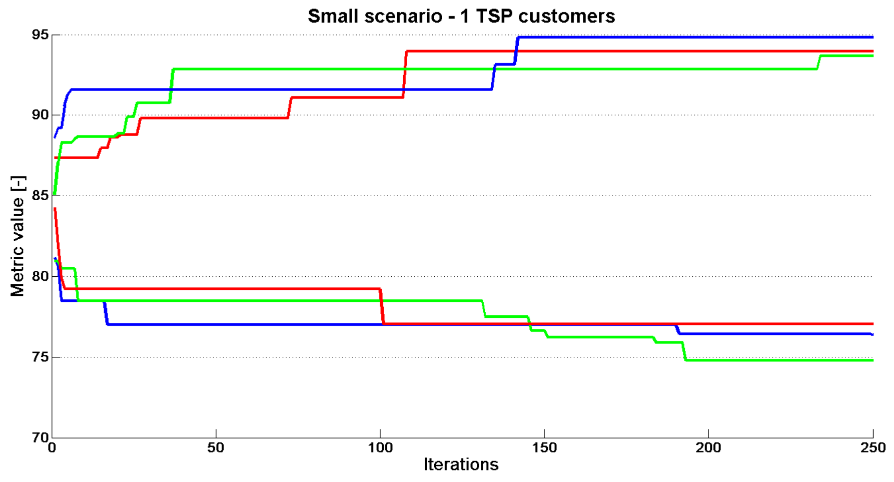

Figure 7.3: Optimization progress of 1 TSP customer in small supermarket

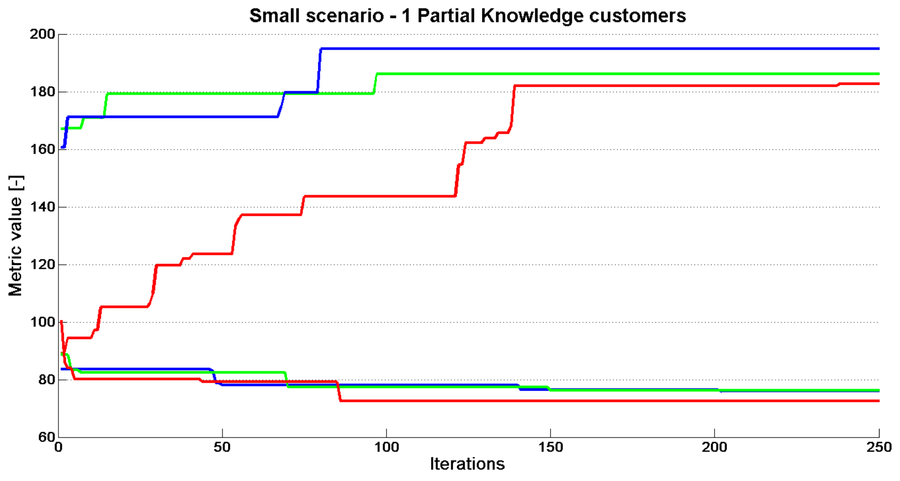

Figure 7.4: Optimization progress of 1 Partial Knowledge customer in small supermarket

of optimization the waiting steps.

The number of waiting steps is influenced by number of shelves with product category (customers do not have wait for customer who is choosing the same type of product) and the width of lane between shelves.

The evaluation of the wait steps at the big scenario is about 30 steps. That is about 10 steps higher than at small scenario. Big supermarkets have usually more spaces between shelves so customers can easily find a way without waiting for other customers.

7.2.3 BI Tool

We used Tableau software[1] to show statistics from simulations. Simulation saves statistics from the best solution in Microsoft Excel file which can be loaded by Tableau. Tableau can show paths of all models, usage of each cell, i.e. how many customers go

[1]http://www.tableau.com/

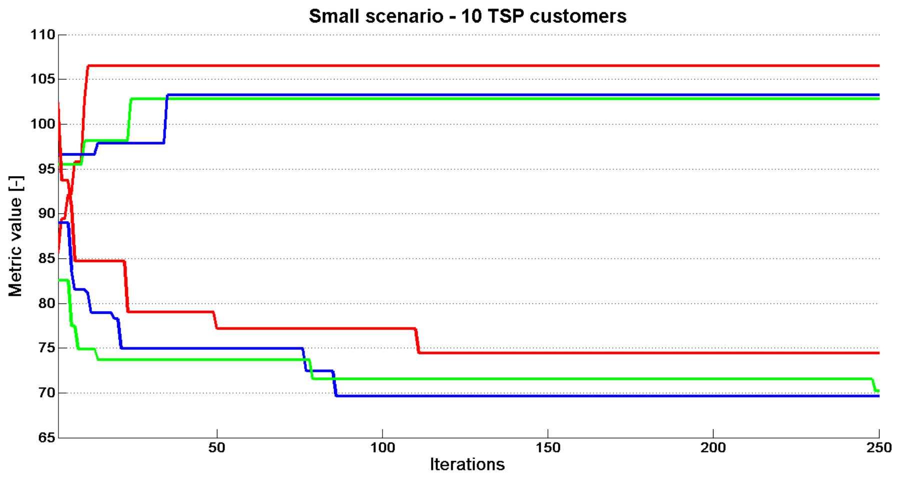

Figure 7.5: Optimization progress of 10 TSP customer in small supermarket

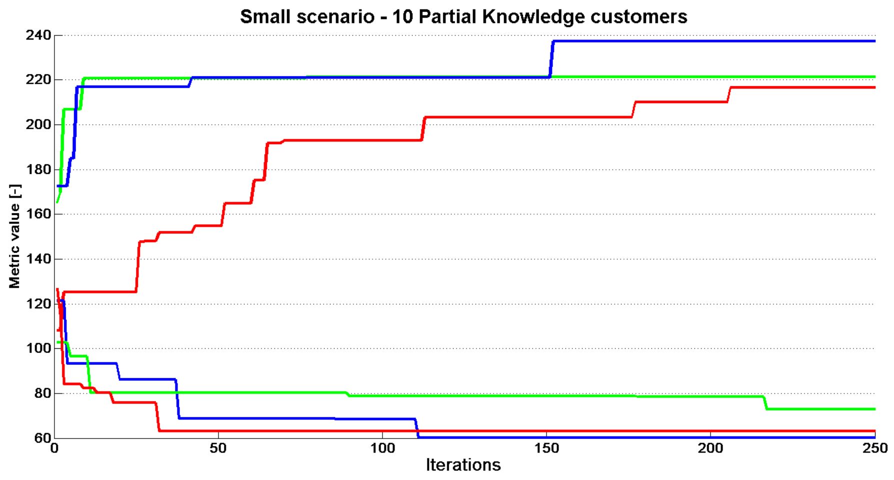

Figure 7.6: Optimization progress of 10 Partial Knowledge customers in small supermarket.

through each cell, etc. The statistic file contains number of wait steps of each model and their complete path.

A demonstration of BI tool we propose a figure 7.11 with small scenario layout which shows usage of each cell.

7.3 Summary

The goal of the thesis was to create supermarket creator, to model customer behavior models and to optimize shelf space allocation according to defined metrics. Optimization considers customer's behavior in specific environment.

I studied the problem of the shelf space allocation and proposed complex tools for demonstration this problem. In product placement there exists relation between product items, product subcategories and product categories. This relations then have effects

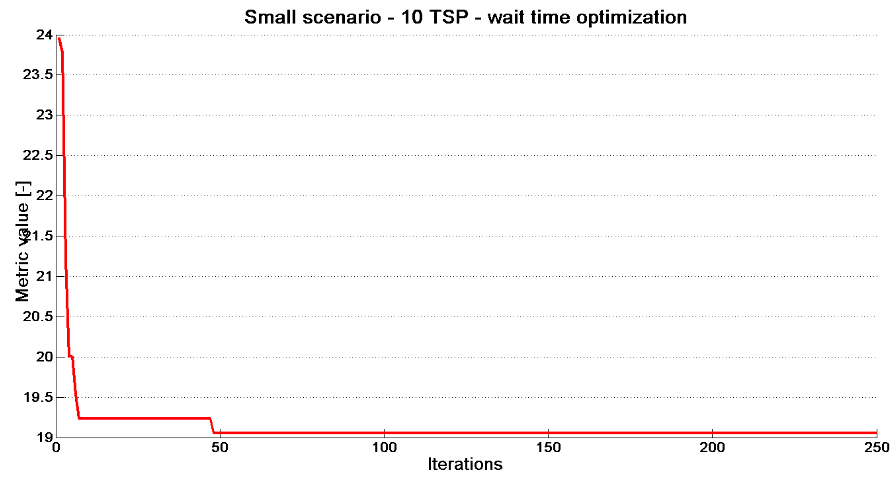

Figure 7.7: Optimization progress of Waiting steps of 10 TSP customers in small supermarket.

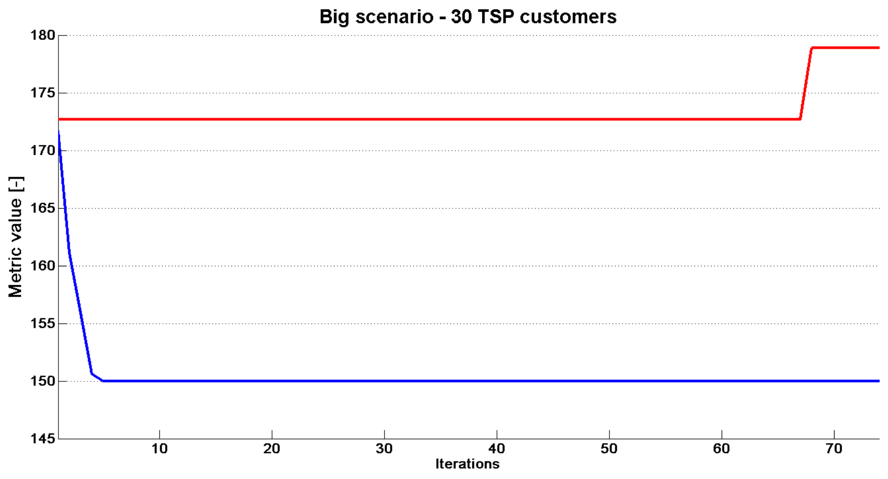

Figure 7.8: Optimization progress of 10 TSP customers in big supermarket.

on product placement.

I implemented a tool for creating supermarket layout which can be filled by products from available hierarchical list of product categories, subcategories and items. I modelled three customer behavior models which represents different kind of real customer. I used A* algorithm to solve problem with path planning in time. Next goal was to implemented simulation framework which takes the supermarket layout, set of customers and executes the Simulated annealing algorithm on that settings. Agent-based simulation is used for capturing the decision-making process.

I tested two scenarios with many different settings (number and type of customer figuring in simulation). The result of the thesis is that placing favourite products to the opposite ends of the supermarkets increases the total time customers stay inside the supermarket. This leads to bigger supermarket profits.

This work has certainly some limits; offering more products would bring more precise

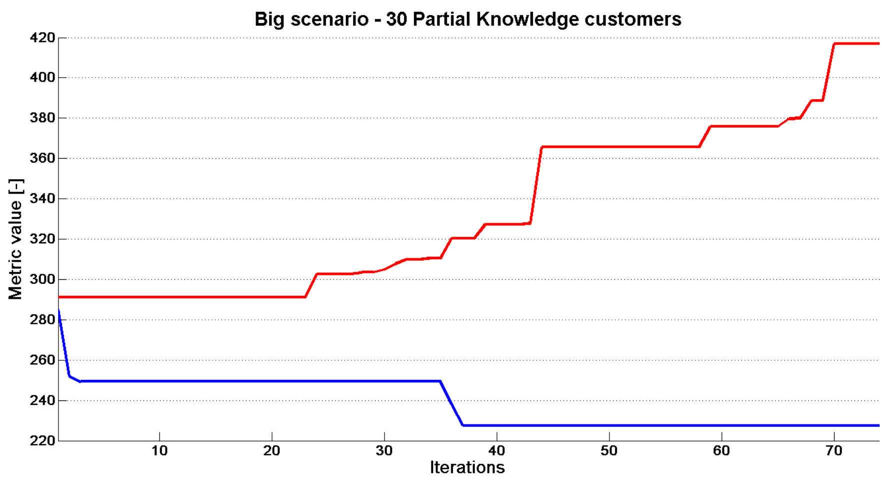

Figure 7.9: Optimization progress of 10 Partial Knowledge customers in big supermarket.

results. We would consider more customers with one shopping cart (a family). This factor would provide more accurate path planner. Next option is to implement cash desk queue.

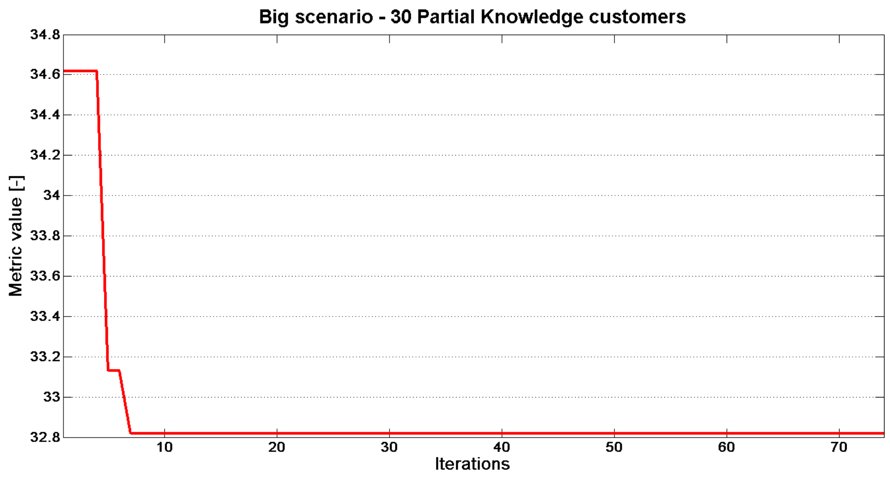

Figure 7.10: Optimization progress of Wait steps of 30 Partial Knowledge customers in big supermarket.

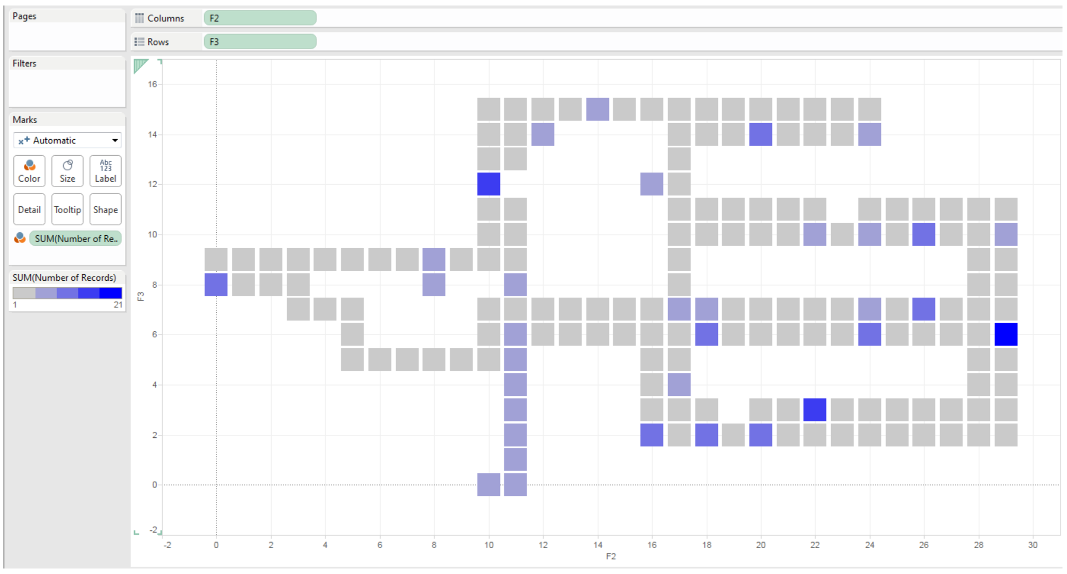

Figure 7.11: Statistic shown in BI Tool Tableau. Light blue - small usage, dark blue - big usage

Bibliography

[1] E. Aarts, J. Korst, and W. Michiels. Simulated Annealing. In E. . Burke and G. Kendall, editors, *Search Methodologies*, chapter 7, pages 187–210–210. Springer US, Boston, MA, 2005.

[2] M.-C. Chen and C.-P. Lin. A data mining approach to product assortment and shelf space allocation. *Expert Syst. Appl.*, 32(4):976–986, May 2007.

[3] G. B. Dantzig, D. R. Fulkerson, and S. M. Johnson. Solution of a large-scale traveling-salesman problem. *Operations Research*, 3, 1954.

[4] S. Franklin and A. Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Proceedings of the Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages*, ECAI '96, London, UK, UK, 1997. Springer-Verlag.

[5] H. Hwang, B. Choi, and M.-J. Lee. A model for shelf space allocation and inventory control considering location and inventory level effects on demand. *International Journal of Production Economics*, 97(2):185 – 195, 2005.

[6] F. A. Karkory and A. A. Abudalmola. *International Journal of Mathematical, Computational, Natural and Physical Engineering*, 7(10):987 – 997, 2013.

[7] F. Klügl. Agent-based modeling and simulation. *AI Magazine 33.3*, pages 29–40, 2012.

[8] G. Laporte. The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2):231–247, June 1992.

[9] S. LaValle. *Planning Algorithms.* Cambridge University Press, 2006.

[10] M. Levy and B. A. Weitz. *Retailing management.* Chicago: Irwin, 1995.

[11] G. Luger. *Artificial intelligence: structures and strategies for complex problem solving.* Pearson education. Addison-Wesley, 2005.

[12] A. S. P. Moscato. Local Search Techniques for Scheduling Problems. online. `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.123.2651&rep=rep1&type=ps`.

[13] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards. *Artificial intelligence: a modern approach*, volume 2. Prentice hall Englewood Cliffs, NJ, 1995.

[14] M.-H. Yang. An efficient algorithm to allocate shelf space. *European Journal of Operational Research*, 131(1):107 – 118, 2001.

Appendix A

CD Content

CD attached with the thesis contains:

- **Layout designer** — \Application\designer.jar
- **Simulation application** — \Application\simulation.jar
- **Product properties** — \Application\products_properties.xlsx
- **Designer runner** — \Application\runDesigner.bat
- **Simulation runner** — \Application\runSimulation.bat
- **Source code** — \Source code
- **Thesis in PDF** — \Thesis\tomasda2_DP.pdf

Appendix B

JSON

Example of JSON format of saved supermarket layout with all its properties

```
{
  "length": 240,
  "width": 135,
  "shelves": [
    {
      "layout": [
        [
          {
            "name": "Beet"
          },
          {
            "name": "Beet"
          },
          {
            "name": "Beet"
          }
        ],
        [
          {
            "name": "Beet"
          },
          {
            "name": "Beet"
          },
          {
            "name": "Beet"
          }
        ],
        [
          {
            "name": "Beet"
          },
          {
            "name": "Beet"
          },
          {
            "name": "Beet"
          }

        ],
        [
          {
            "name": "Beet"
          },
          {
            "name": "Beet"
          },
          {
            "name": "Beet"
          }
        ]
      ],
      "position": "BOTTOM",
      "layoutX": 0,
      "layoutY": 8
    }
    ],
  "entrances": [
    {
      "layoutX": 0,
      "layoutY": 4
    }
  ],
  "cashiers": [
    {
      "layoutX": 2,
      "layoutY": 0
    },
    {
      "layoutX": 3,
      "layoutY": 0
    }
  ],
  "walls": [
    {
      "layoutX": 7,
      "layoutY": 1
    },
    {
      "layoutX": 7,
      "layoutY": 2
    },
    {
      "layoutX": 4,
      "layoutY": 8
    },
    {
      "layoutX": 15,
      "layoutY": 8
    }
  ]
}
```

Appendix C

Correlation Matrix

Product correlation matrix example.

	Milks	Yoghurts	Cheese	Eggs	Butter	Sausages	Sliced meat
Sugar	0	0	0	0	0	0	0
Flour	0,1	0	0	0,1	0	0	0
Pasta	0	0	0	0	0	0,1	0
Spices	0	0	0	0	0	0	0
Soups	0	0	0	0	0	0	0
Rice	0	0	0	0	0	0	0
Biscuits	0,2	0,15	0	0	0	0	0

Appendix D

Categories and products

List of categories and products which can be placed into the supermarket shelves.

- Dairy
  - Milk
  - Yoghurt
  - Cheese
  - Egg
  - Butter
- Meat
  - Cooked Meat
    - Sausages
    - Sliced Meat
    - Oil
  - Fresh Meat
    - Bacon
    - Beef
    - Lamb
    - Pork
    - Poultry
    - Minced Meat
- Sea food
  - Fresh Fillets
  - Fish Fingers
  - Prawns and Shellfish
- Deli

  - Olive Oil
  - Olives
  - Pate
  - Tinned fish
  - Jam
  - Ketchup
  - Mustard
  - Sauce

- Bakery
  - Bread
  - Roll
  - Croissant

- Fruits and Vegetables
  - Fruits
    * Apple
    * Orange
    * Grapefruit
    * Banana
    * Grape
    * Strawberry
  - Vegetables
    * Potato
    * Tomato
    * Cucumber
    * Pepper
    * Lettuce
    * Cellery
    * Beet

- Frozen
  - Pizza
  - Ready Meals
  - Frozen Meats
  - Ice-cream
  - Frozen Fishes

  - French Fries

- Health and Beauty
  - Shampoo
  - Shower Gel
  - Soap
  - Sun care
  - Oral care
  - Perfume
- Household
  - Cleaning Stuff
  - Toilet Paper
  - Dishwasher Products
- Beverages
  - Alcoholic
    - Champagne
    - Red Wine
    - White Wine
    - Beer
    - Spirit
  - Non-alcoholic
    - Still Water
    - Sparkling Water
    - Flavoured Water
    - Lemonade
    - Syrup
    - Juice
    - Energy Drink
    - Tea
    - Coffee
- Snacks
  - Cracker
  - Crisps
  - Sticks
- Sweets

  - Chocolate
  - Wafer
  - Bar
  - Candy
  - Biscuits

- Packaged Foods
  - Rice
  - Soup
  - Spice
  - Pasta
  - Flour
  - Sugar