FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Prescriptive Analytics for Staff Scheduling Optimization in Retail

Catarina Alexandra Teixeira Ramos



Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Carlos Manuel Milheiro de Oliveira Pinto Soares

Second Supervisor: Yassine Baghoussi

July 3, 2019

Prescriptive Analytics for Staff Scheduling Optimization in Retail

Catarina Alexandra Teixeira Ramos

Mestrado Integrado em Engenharia Informática e Computação

July 3, 2019

Abstract

Human Resource Management (HRM) is an area where improving processes is important to achieve higher performance and profit in an organization. In recent years, this area has gain focus in research of data mining techniques, however, a particular sub-domain stands out: staffing, that deals with gathering, training, placing and retaining the best people for particular jobs or tasks in the organization.

In retail, shop floor employees have a great impact on sales since they interact directly with customers. Guaranteeing client coverage increases the possibility of clients being converted to sales and therefore, increase profit for the store. Ensuring client coverage could easily be done by allocating all employees to the store. However, allocation has costs and using all resources may lead to overstaffing. For this reason, optimization is needed to achieve the optimal number of shop floor employees in order to face staff demand.

Prescriptive analytics is a type of data mining that aims at prescribing the best decisions. For this, it combines data mining models with decision support techniques. This allows to prescribe solutions and provide support through a more complex process.

In this dissertation, the main goal is to build a prescriptive model that prescribes the best shifts and task allocation in order to maximize a store's profit. For this, a genetic algorithm was adapted and combined with one of two alternative sales forecasting architectures. One approach forecasts sales directly and the other forecasts three different targets: the number of transactions, the average number of products per each transaction and the average price of each product. In the last, the forecasted targets are multiplied by each other to calculate sales. In both architectures, the footfall of clients is previously forecasted and used as input for the final models. This way, the architectures follow a layered learning approach. Step-by-step instructions into implementing the proposed solution are properly described in this dissertation.

To validate the proposed solution, a case study was developed using real-world data about a fashion retailer. In the results we have observed an estimated increase of 3-9% in sales for the prescribed solution, in comparison with the one manually designed by the retailer. A further reflection on this kind of DM made us reach to the conclusion of the importance of the forecasting models. Further experiments have shown that the robustness of the prescriptive model relies on the quality of the forecasting models. This type of analytics evaluates the solutions by their impact in real-world, which is accessible through models. The less accurate these models are, the worse their ability to predict the real-world behaviour. This can cause the solutions to be misleadingly guided and to not have the estimated impact, leading to an unreliable prescriptive model.

Resumo

A gestão de recursos humanos é uma área em que a melhoria dos processos é muito importante para alcançar um desempenho e lucro maior. Nos últimos anos, esta área tem sido alvo de foco na investigação de técnicas de *data mining*. Porém, uma subárea tem demonstrado destaque: *staffing*, que lida com o recrutamento, treino, colocação e retenção dos melhores profissionais para determinados cargos ou tarefas numa organização.

Na área de retalho, os funcionários que trabalham diretamente na loja, têm um impacto elevado nas vendas, visto que interagem diretamente com os clientes. Garantir a cobertura de clientes aumenta a possibilidade de estes serem convertidos em vendas, traduzindo-se num aumento de lucro. Isto poderia ser facilmente conseguido ao alocar todos os funcionários para a loja. Non entanto, a alocação tem custos associados e a utilização de todos os recursos pode levar a um excesso desnecessário de funcionários. Por esta razão, a otimização é necessária para garantir o número ideal de funcionários em loja e enfrentar a demanda dos clientes.

Prescriptive Analytics é uma técnica de *data mining* que tem como objetivo prescrever as melhores decisões. Para isto, combina modelos de *data mining* com técnicas de suporte à decisão. Isto permite prescrever soluções e dar suporte através de um processo mais complexo.

Nesta dissertação, o principal é construir um modelo prescritivo que prescreva os melhores turnos e alocação de tarefas de forma a maximizar o lucro de uma loja. Para isso, um algoritmo genético foi adaptado e combinado com uma de duas alternativas de arquiteturas de previsão de vendas. Uma das arquiteturas visa prever as vendas diretamente e a outra através da previsão de três componentes diferentes: o número de transações, o número médio de produtos por transação e o preço médio por produto. Nesta última, as vendas são calculadas pela multiplicação dos diferentes componentes. Em ambas as arquiteturas, o número de entradas e saídas de clientes é previamente previsto e usado para a previsão final de vendas. Desta forma, as arquiteturas seguem uma estratégia de *layered learning*. Instruções passo-a-passo para implementar a solução proposta são devidamente descritas nesta dissertação.

Para validar a solução proposta, um caso de estudo foi desenvolvido usando dados reais sobre uma loja de roupa. Várias experiências foram conduzidas e os resultados mostraram que a solução prescrita previa um aumento de 3-9 % nas vendas, em comparação com a solução projetada pelo retalhista. Mais tarde, uma reflexão sobre este tipo de metodologia fez-nos chegar à conclusão da importância dos modelos de previsão. Uma das experiências mostraram que a robustez do modelo prescritivo depende da qualidade dos modelos de previsão. Este tipo de DM avalia as soluções pelo seu impacto no mundo real, acessível através de previsão. Quanto menos preciso forem estas previsões, pior é a sua habilidade em prever o comportamento do mundo real. Isto pode causar com que as soluções sejam otimizadas de forma errada e não ter o impacto estimado, levando a um modelo prescritivo duvidoso.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor Dr. Carlos Manuel Milheiro de Oliveira Pinto Soares, above all, for his guidance and support through the development of this project, but also for his motivation. His passion for data science inspire me to pursue a dissertation in this area and awaken an interest in me to want to have even more knowledge.

I also would like to acknowledge the Faculty of Engineering of the University of Porto for the resources and support in the research of scientific literature related to this dissertation. I would also like to thanks for the excellent facilities and access to the SONAE IM LAB@FEUP laboratory.

My special thanks to InovRetail and João Heitor Cunha Serra Guichard for providing real data about a fashion retailer. This enabled to build a case study for the solution developed and draw some conclusions from the conducted experiments.

I thank my fellow colleagues in for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had in the last five years. A special thanks to Inês Isabel Correia Gomes and Mário Gustavo Gomes Rosas de Azevedo Fernandes for they have been the most enthusiastic and admirable friends during my academic journey.

Finally, I must express my very profound gratitude to my parents and to my brothers for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of writing this dissertation. This accomplishment would not have been possible without them. Thank you.

Catarina Alexandra Teixeira Ramos

"Engineers like to solve problems.
If there are no problems handily available,
they will create their own problems."

Scott Adams

Contents

1 Introduction **1**
- 1.1 Motivation . . . 2
- 1.2 Goals . . . 2
- 1.3 Dissertation Structure . . . 3

2 State of Art **5**
- 2.1 Scheduling . . . 5
  - 2.1.1 Staff Scheduling . . . 5
  - 2.1.2 Scheduling in Retail . . . 7
- 2.2 Data Mining . . . 10
  - 2.2.1 Prescriptive Analytics . . . 12
- 2.3 Prescriptive Analytics for Staff Scheduling in Retail . . . 13

3 Problem Description **15**
- 3.1 Problem Formalization . . . 15
- 3.2 A Prescriptive Analytics Solution for Staff Scheduling . . . 16
  - 3.2.1 Predictive Modeling . . . 17
  - 3.2.2 Optimization Modeling . . . 17
  - 3.2.3 Prescriptive Modeling . . . 18
- 3.3 Data . . . 19
  - 3.3.1 Data Description . . . 19
  - 3.3.2 Data Cleaning . . . 22
  - 3.3.3 Data Preparation . . . 24

4 Implementation **27**
- 4.1 Forecasting . . . 27
  - 4.1.1 Forecasting Architecture . . . 27
  - 4.1.2 Feature Engineering . . . 30
  - 4.1.3 Data Dependency . . . 32
- 4.2 Optimization . . . 33
  - 4.2.1 Constraints . . . 34
  - 4.2.2 Representation . . . 35
  - 4.2.3 Initialization . . . 39
  - 4.2.4 Evaluation . . . 44
  - 4.2.5 Selection . . . 46
  - 4.2.6 Crossover . . . 46
  - 4.2.7 Mutation . . . 51

5 Results and Analysis **65**
5.1 Experimental Setup . . . 65
5.1.1 Data . . . 65
5.1.2 Experiments . . . 65
5.1.3 System Specifications . . . 69
5.2 Evaluation Metrics . . . 70
5.3 Results . . . 72
5.3.1 Forecasting . . . 72
5.3.2 Optimization . . . 78

6 Conclusion **89**
6.1 Future Work . . . 90
6.2 Contribution . . . 90
6.3 Final Conclusions . . . 91

A Data **97**
A.1 Examples . . . 97
A.1.1 Raw Data . . . 97
A.1.2 Prepared Data . . . 100
A.2 Data Analysis . . . 107
A.2.1 Events . . . 107
A.2.2 Footfalls . . . 109
A.2.3 Schedules . . . 112
A.2.4 Transactions . . . 114
A.2.5 Shifts . . . 118
A.2.6 Open Hours . . . 122
A.3 Data Cleaning . . . 123
A.4 Data Preparation . . . 127

B Implementation **135**
B.1 Forecasting . . . 135
B.2 Optimization . . . 137
B.2.1 Initialization Example . . . 137
B.2.2 Crossover Example . . . 144
B.2.3 Mutation Example . . . 155

C Results **167**
C.1 Forecasting . . . 167
C.1.1 Experiment F1 . . . 167
C.1.2 Experiment F3 . . . 169
C.1.3 Experiment F4 . . . 171
C.2 Optimization . . . 176
C.2.1 Optimization Runs . . . 176
C.2.2 Experiment O1 . . . 194
C.2.3 Experiment O2 . . . 201
C.2.4 Experiment O3 . . . 210
C.2.5 Experiment O4 . . . 215

List of Figures

2.1 Histogram of scientific publications collected from 1998 to 2018 about staff scheduling in retail with bins of a five-year interval. . . . 7
2.2 Data mining analytics work-flow. . . . 12

3.1 Example of a sales curve. . . . 17
3.2 Distribution of variables and objective function target across the planning horizon of a week for the optimization phase. . . . 18
3.3 Optimization iteration in prescriptive analytics. . . . 18
3.4 UML class diagram for the data provided. . . . 21

4.1 Indirect sales forecasting architecture. . . . 29
4.2 Direct sales forecasting architecture. . . . 29
4.3 Genetic algorithm architecture. . . . 33
4.4 Allocation structure. . . . 36
4.5 Allocation structure initialization. . . . 36
4.6 Availability structure. . . . 37
4.7 Availability structure initialization with *n* employees. . . . 37
4.8 Shift catalog structure. . . . 38
4.9 Shift catalog structure initialization. . . . 38
4.10 Example of the shift catalog structure with one shift. . . . 39
4.11 Example of the allocation structure with one shift. . . . 39
4.12 Example of the availability structure with one shift. . . . 39
4.13 Individual generation process. . . . 40
4.14 Shift creation process for gap coverage. . . . 42
4.15 Allocation structure generated by the initialization process. . . . 43
4.16 Availability structure generated by the initialization process. . . . 44
4.17 Shift catalog structure generated by the initialization process. . . . 44
4.18 Crossover slicing method and shift swap. Overall information (left image) and shifts boundaries (right image) representation. . . . 46
4.19 Crossover process. . . . 47
4.20 Crossover shift classification. . . . 48
4.21 Mutation *reduce start* operation. . . . 53
4.22 Mutation *reduce last* operation. . . . 54
4.23 Mutation *extend start* operation. . . . 55
4.24 Mutation *extend last* operation. . . . 56
4.25 Mutation *break* operation. Two possible outcomes. . . . 56
4.26 Mutation *join* operation. Two possible joining points for the same outcome. . . . 57
4.27 Mutation *remove* operation. . . . 58

4.28 Mutation *reallocate* operation. 59

5.1 Real versus forecasted *footfall in*. 73
5.2 Real versus forecasted footfall out. 73
5.3 Real versus the forecasted number of transactions. 73
5.4 Real versus forecasted average products per transaction. 74
5.5 Real versus forecasted average price per transaction. 74
5.6 Real versus forecasted direct sales. 74
5.7 Real versus direct and indirect forecasted sales. 75
5.8 MSE of the forecasted number of transactions with increasing *footfall in* error. . . 77
5.9 MSE of the forecasted average number of products per transactions with increasing *footfall in* error. 77
5.10 MSE of the forecasted average price per product with increasing *footfall in* error. 78
5.11 MSE of the forecasted direct and indirect sales with increasing *footfall in* error. . 78
5.12 Forecasted sales for the prescribed and manual solution in experiment $O1a$ in comparison with the real sales observed for the manual solution. 80
5.13 Forecasted sales for the prescribed and manual solution in experiment $O1b$ in comparison with the real sales observed for the manual solution. 80
5.14 Cumulative forecasted and real observed sales for the week before of planning in experiment $O2a$. 82
5.15 Cumulative forecasted and real observed sales for the week before of planning in experiment $O2b$. 82
5.16 Forecasted sales for the prescribed solution in run $R3a$ when evaluated with and without additional error in *footfall in*. 83
5.17 Forecasted sales for the prescribed solution in run $R3b$ when evaluated with and without additional error in *footfall in*. 84
5.18 Forecasted sales for the prescribed solution in run $R1a$ and solution in R3b, when evaluated without any additional error in *footfall in*. 84
5.19 Forecasted sales for the prescribed solution in run $R1$b and solution in R3b when evaluated without any additional error in *footfall in*. 85
5.20 Forecasted sales for the prescribed solution in run $R1a$ and solution in $R1b$, when evaluated through a direct sales forecasting architecture. 86
5.21 Forecasted sales for the prescribed solution in run $R1a$ and solution in $R1b$, when evaluated through an indirect sales forecasting architecture. 87

A.1 Histogram for the *start* attribute values of the raw events dataset in *Store_1*. . . . 107
A.2 Histogram for the *end* attribute values of the raw events datase in *Store_1*. 108
A.3 Histogram for the *type* attribute values of the raw events dataset in *Store_1*. . . . 108
A.4 Histogram for the *date* attribute values of the raw footfalls dataset in *Store_1*. . . 109
A.5 Histogram for the *time* attribute values of the raw footfalls dataset in *Store_1*. . . 110
A.6 Histogram for the *footfall in* attribute values of the raw footfalls dataset in *Store_1*. 110
A.7 Histogram for the *footfall out* attribute values of the raw footfalls dataset in *Store_1*. 111
A.8 Values of the *footfall in*, per day, through time for *Store_1*. 111
A.9 Values of the *footfall out*, per day, through time for *Store_1*. 112
A.10 Histogram for the *date* attribute values of the raw schedules dataset in *Store_1*. . 113
A.11 Histogram for the *start* attribute values of the raw schedules dataset in *Store_1*. . 113
A.12 Histogram for the *end* attribute values of the raw schedules dataset in *Store_1*. . . 114
A.13 Histogram for the *shift type* attribute values of the raw schedules dataset in *Store_1*. 114
A.14 Histogram for the *date* attribute values of the raw transactions dataset in *Store_1*. 115

A.15 Histogram for the *hour* attribute values of the raw transactions dataset in *Store_1*. Each bin represents a 3000 value range. 116
A.16 Histogram for the *qty* attribute values, of the raw transactions dataset in *Store_1*. . 116
A.17 Histogram for the *net sales* attribute values of the raw transactions dataset in *Store_1*.117
A.18 Values of the *qty*, per day, through time for *Store_1*. 117
A.19 Values of the *net sales*, per day, through time for *Store_1*. 118
A.20 Histogram for the *shift_order* attribute values of the shifts dataset in *Store_1*. . . 118
A.21 Histogram for the *shift_duration* attribute values of the shifts dataset in *Store_1*. . 119
A.22 Histogram for the *task_duration* attribute values of the shifts dataset in *Store_1*. . 119
A.23 Histogram for the *start_time* attribute values of the shifts dataset in *Store_1*. . . . 120
A.24 Histogram for the *end_time* attribute values of the shifts dataset in *Store_1*. . . . 120
A.25 Histogram for the *task_type* attribute values of the shifts dataset in *Store_1*. . . . 121
A.26 Comparison between the duration of shifts that have breaks and the break task duration. 121
A.27 Histogram for the *max_employees* attribute values of the open hours dataset in *Store_1*. 122
A.28 Histogram for the *open_hours* attribute values of the open hours dataset in *Store_1*. 122
A.29 Histogram for the *close_hours* attribute values of the open hours dataset in *Store_1*. 123

B.1 Day 0 allocation structure at iteration 0. 137
B.2 Day 0 availability structure at iteration 0. 138
B.3 Day 0 shifts catalog structure at iteration 0. 138
B.4 Day 0 allocation structure at iteration 1. 139
B.5 Day 0 availability structure in iteration 1. 139
B.6 Day 0 shifts catalog structure at iteration 1. 140
B.7 Day 0 allocation structure at iteration 2. 141
B.8 Day 0 availability structure at iteration 2 141
B.9 Day 0 shifts catalog structure at iteration 2. 141
B.10 Day 0 allocation structure in iteration 3. 142
B.11 Day 0 availability structure at iteration 3. 142
B.12 Day 0 shifts catalog structure at iteration 3. 143
B.13 Day 0 allocation structure at iteration 4. 144
B.14 Day 0 availability structure at iteration 4. 144
B.15 Day 0 shifts catalog structure at iteration 4. 144
B.16 Parent 1 allocation structure. 145
B.17 Parent 1 availability structure. 145
B.18 Parent 1 shift catalog structure. 146
B.19 Parent 1 shift visualization throughout the day. 146
B.20 Parent 2 allocation structure. 146
B.21 Parent 2 availability structure. 146
B.22 Parent 2 shift catalog structure. 147
B.23 Parent 2 shift visualization throughout the day. 147
B.24 Shift representation for parent 1 after the information is sliced. 147
B.25 Shift representation for parent 2 after the information is sliced. 148
B.26 Child 1 before the join of overlay shifts. 149
B.27 Child 2 before the join of overlay shifts. 149
B.28 Child 1 allocation structure. 153
B.29 Child 1 availability structure. 153
B.30 Child 1 shift catalog structure. 153

B.31 Child 1 shifts visualization throughout the day. 154
B.32 Child 2 allocation structure. 154
B.33 Child 2 availability structure. 154
B.34 Child 2 shift catalog structure. 154
B.35 Child 2 shifts visualization throughout the day. 155
B.36 Individual allocation structure. 155
B.37 Individual availability structure. 155
B.38 Individual shift catalog structure. 156
B.39 Individual allocation structure after the *reduce start* mutation on slot 8. 157
B.40 Individual availability structure after the *reduce start* mutation on slot 8. 157
B.41 Individual shift catalog structure after the *reduce start* mutation on slot 8. 158
B.42 Individual allocation structure after the *reduce last* mutation on slot 6. 158
B.43 Individual availability structure after the *reduce last* mutation on slot 6. 158
B.44 Individual shift catalog structure after the *reduce last* mutation on slot 6. 159
B.45 Individual allocation structure after the *extend start* mutation on slot 8. 160
B.46 Individual availability structure after the *extend start* mutation on slot 8. 160
B.47 Individual shift catalog structure after the *extend start* mutation on slot 8. 160
B.48 Individual allocation structure after the *extend last* mutation on slot 7. 161
B.49 Individual availability structure after the *extend last* mutation on slot 7. 161
B.50 Individual shift catalog structure after the *extend last* mutation on slot 7. 161
B.51 Individual allocation structure after the *break* mutation on slot 11. 162
B.52 Individual availability structure after the *break* mutation on slot 11. 162
B.53 Individual shift catalog structure after the *break* mutation on slot 11. 163
B.54 Individual allocation structure after the *join* mutation on slot 7. 164
B.55 Individual availability structure after the *join* mutation on slot 7. 164
B.56 Individual shift catalog structure after the *join* mutation on slot 7. 164
B.57 Individual allocation structure after the *remove* mutation on slot 10. 164
B.58 Individual availability structure after the *remove* mutation on slot 10. 165
B.59 Individual shift catalog structure after the *remove* mutation on slot 10. 165
B.60 Individual allocation structure after the *reallocate* mutation on slot 10. 165

C.1 Real versus forecasted number of transactions using accurate and forecasted footfall. 169
C.2 Real versus forecasted average products per transaction using accurate and forecasted footfall. 170
C.3 Real versus forecasted average price per product using accurate and forecasted footfall. 170
C.4 Real versus forecasted direct sales using accurate and forecasted footfall. 171
C.5 Real versus forecasted indirect sales using accurate and forecasted footfall. 171
C.6 Evolution of the best individual score through iterations in run $R1a$. 176
C.7 Forecasted sales curve for the prescribed solution in run $R1a$. 176
C.8 Allocation for the prescribed solution in run $R1a$. 177
C.9 Shifts boundaries for the prescribed solution in run $R1a$. 178
C.10 Evolution of the best individual score through iterations in run $R1b$. 179
C.11 Forecasted sales curve for the prescribed solution in run $R1b$. 179
C.12 Allocation for the prescribed solution in run $R1b$. 180
C.13 Shifts boundaries for the prescribed solution in run $R1b$. 181
C.14 Evolution of the best individual score through iterations in run $R2a$. 182
C.15 Forecasted sales curve for the prescribed solution in run $R2a$. 182
C.16 Allocation for the prescribed solution in run $R2a$. 183

C.17 Shifts boundaries for the prescribed solution in run *R*2*a*. 184
C.18 Evolution of the best individual score through iterations in run *R*2*b*. 185
C.19 Forecasted sales curve for the prescribed solution in run *R*2*b*. 185
C.20 Allocation for the prescribed solution in run *R*2*b*. 186
C.21 Shifts boundaries for the prescribed solution in run *R*2*b*. 187
C.22 Evolution of the best individual score through iterations in run *R*3*a*. 188
C.23 Forecasted sales curve for the prescribed solution in run *R*3*a*. 188
C.24 Allocation for the prescribed solution in run *R*3*a*. 189
C.25 Shifts boundaries for the prescribed solution in run *R*3*a*. 190
C.26 Evolution of the best individual score through iterations in run *R*3*b*. 191
C.27 Forecasted sales curve for the prescribed solution in run *R*3*b*. 191
C.28 Allocation for the prescribed solution in run *R*3*b*. 192
C.29 Shifts boundaries for the prescribed solution in run *R*3*b*. 193
C.30 Allocation for the solution solution in experiment *O*1 194
C.31 Shifts boundaries for the manual solution in experiment *O*1. 195
C.32 Allocation distances for the schedules in experiment *O*1*a* 196
C.33 Allocation distances for the schedules in experiment *O*1*b* 197
C.34 Forecasted sales for the prescribed and manual solution in experiment *O*1*a*, with the real sales observed for the manual solution. 198
C.35 Cumulative forecasted sales for the prescribed and manual solution in experiment *O*1*a*, with the real sales observed for the manual solution. 199
C.36 Forecasted sales for the prescribed and manual solution in experiment *O*1*b*, with the real sales observed for the manual solution. 199
C.37 Cumulative forecasted sales for the prescribed and manual solution in experiment *O*1*b*, with the real sales observed for the manual solution. 200
C.38 Allocation for the first week in advance in the prescribed solution of run *R*2. . . . 201
C.39 Shifts boundaries for the first week in advance in the prescribed solution for run *R*2. 202
C.40 Allocation distances for the schedules in experiment *O*2*a* 203
C.41 Allocation distances for the schedules in experiment *O*2*b* 204
C.42 Forecasted sales for the first week in advance of the prescribed solution in run *R*2*a*, and the real sales observed in *R*1*a*, in experiment *O*2*a*. 205
C.43 Cumulative forecasted sales for the first week in advance of the prescribed solution in run *R*2*a*, and the real sales observed in *R*1*a*, in experiment *O*2*a*. 206
C.44 Forecasted sales for the prescribed solution in run *R*2*a* and the solution in *R*1*a*, in experiment *O*2*a*. 206
C.45 Cumulative forecasted sales for the prescribed solution in run *R*2*a* and the solution in *R*1*a*, in experiment *O*2*a*. 207
C.46 Forecasted sales for the first week in advance, of the prescribed solution in run *R*2*b*, and the real sales observed in *R*1*b*, in experiment *O*2*b*. 207
C.47 Cumulative forecasted sales for the first week in advance, of the prescribed solution in run *R*2*b*, and the real sales observed in *R*1*b*, in experiment *O*2*b*. 208
C.48 Forecasted sales for the prescribed solution in run *R*2*b* and solution in *R*1*b*, in experiment *O*2*b*. 208
C.49 Cumulative forecasted sales for the prescribed solution in run *R*2*b* and solution in *R*1*b*, in experiment *O*2*b*. 209
C.50 Allocation distances for the schedules in experiment *O*3*a* 210
C.51 Allocation distances for the schedules in experiment *O*3*b* 211

C.52 Forecasted sales for the prescribed solution in run $R1a$ and solution in $R3a$, when evaluated with and without any additional error in footfall_in. 212
C.53 Cumulative forecasted sales for the prescribed solution in run $R1a$ and solution in $R3a$, when evaluated with and without any additional error in footfall_in. 213
C.54 Forecasted sales for the prescribed solution in run $R1b$ and solution in $R3b$, when evaluated with and without any additional error in footfall_in. 213
C.55 Cumulative forecasted sales for the prescribed solution in run $R1b$ and solution in $R3b$, when evaluated with and without any additional error in footfall_in. 214
C.56 Allocation distances for the schedules in experiment $O4$ 215
C.57 Forecated sales for the prescribing solution in run $R1a$ using a direct and indirect sales forecasting architecture. 216
C.58 Forecasted sales for the prescribed solution in run $R1a$ and solution in $R1b$, when evaluated through a direct sales forecasting architecture. 217
C.59 Cumulative forecasted sales for the prescribed solution in run $R1a$ and solution in $R1b$, when evaluated through a direct sales forecasting architecture. 217
C.60 Forecasted sales for the prescribed solution in run $R1b$ using a direct and indirect sales forecasting architecture. 218
C.61 Forecasted sales for the prescribed solution in run $R1a$ and solution in $R1b$, when evaluated through an indirect sales forecasting architecture. 218
C.62 Cumulative forecasted sales for the prescribed solution in run $R1a$ and solution in $R1b$, when evaluated through an indirect sales forecasting architecture. 219

List of Tables

2.1 Publishing year of the scientific publications collected. . . . 7
2.2 Optimization techniques used in the scientific publications collected. . . . 8
2.3 Scientific publications that used real-world data from retailers. . . . 8
2.4 Scientific publications that use real-world data per type of retailer. . . . 9
2.5 Categories of scientific publications collected according to Ernest et al. [8]. . . . 10
2.6 Special considerations in the collected scientific publications for schedule optimization. . . . 11

3.1 Raw events dataset information. . . . 19
3.2 Example of records in the events dataset. . . . 19
3.3 Raw footfall dataset information. . . . 20
3.4 Example of records in the footfall dataset. . . . 20
3.5 Raw schedules dataset information. . . . 20
3.6 Example of records from the schedules dataset. . . . 20
3.7 Raw transactions dataset information. . . . 21
3.8 Example of records from the transactions dataset. . . . 21
3.9 Problems identified in the events dataset and their respective cleaning solutions. . . . 22
3.10 Problems identified in the footfall dataset and their respective cleaning solutions. 22
3.11 Problems identified in the schedules dataset and their respective cleaning solutions. 23
3.12 Problems identified in the transactions dataset and their respective cleaning solutions. 23
3.13 Events dataset attributes after the first data preparation. . . . 24
3.14 Footfall dataset attributes after the first data preparation. . . . 25
3.15 Schedules dataset attributes after the first data preparation. . . . 25
3.16 Transactions dataset attributes after the first data preparation. . . . 25
3.17 Shifts dataset attributes. . . . 26
3.18 Open hours dataset attributes. . . . 26

4.1 Variables classification. . . . 28
4.2 Input and target variables of the different models. . . . 29
4.3 Time representation for each seasonality 31
4.4 Example of the time coordinates values for *datetime* 2016-01-02 10:00:00 31
4.5 Strict and soft constraints. . . . 34
4.6 Shift characteristics for a single shift gap. . . . 41
4.7 Shift characteristics for a multiple shift gap. . . . 41
4.8 Break characteristics. . . . 42
4.9 Constraints for an individual initialization example. . . . 43
4.10 Variable description for equations 4.8, 4.9, 4.10, 4.11 and 4.12. . . . 45
4.11 Priority rules for overlay shifts. . . . 49

4.12 Mutation types where adding a break might be necessary. 61
4.13 Mutation types where removing the break might be necessary. 61
4.14 Mutation types where extending the break's duration might be necessary. 62
4.15 Mutation types where reducing the break's duration might be necessary. 63

5.1 Default optimization parameters used in the experiments. 66
5.2 Random forest parameters values for the training of the models 66
5.3 Brief description of the forecasting experiments and the expected result. 67
5.4 Optimization runs characteristics. 67
5.5 Default constraints used in the experiments. 67
5.6 Information extracted from the default optimization parameters and constraints. . 68
5.7 Brief description of the optimization experiments and the expected result. 68
5.8 Research questions for the forecasting and optimization experiments. 69
5.9 Hardware specifications of the machine used for running the experiments. 69
5.10 Levels for allocation distance and similarity. 70
5.11 Evaluation metrics values for each target forecasted in experiment $F1$. 72
5.12 Evaluation metrics values for each sales forecast architecture in experiment $F2$. . 75
5.13 Evaluation metrics values for each target forecasted using accurate and forecasted *footfall in* for experiment $F3$. 76
5.14 Importance of *footfall in* attribute associated with each forecsating model. 76
5.15 Execution time for different optimization runs. 79
5.16 Solutions descriptions in experiment $O1$. 79
5.17 Forecasted sales for experiment $O1$. 80
5.18 Solutions descriptions in experiment $O2$. 81
5.19 Forecasted sales for experiment $O2$. 81
5.20 Solutions descriptions for experiment $O3$. 83
5.21 Forecasted sales for experiment $O3$. 83
5.22 Solutions descriptions for experiment $O4$. 85
5.23 Forecasted sales for experiment $O4$. 86
5.24 Distances and similarities between the prescribed solutions involved in the optimization experiments for different levels. 87

A.1 Raw events example for one year. 97
A.2 Raw footfall example for one day. 97
A.3 Raw schedule example for one day. 98
A.4 Raw transactions example for one day. 98
A.5 Prepared events example for one day. 100
A.6 Prepared footfalls example for one day. 101
A.7 Prepared schedules example for one day. 102
A.8 Prepared transactions example for one day. 103
A.9 Prepared shifts example. 105
A.10 Prepared open hours example. 105
A.11 Raw events dataset analysis. 107
A.12 Analysis of attributes in the raw events dataset. 107
A.13 Event type attribute values. 107
A.14 Raw footfalls dataset analysis. 109
A.15 Analysis of attributes in the raw footfalls dataset. 109
A.16 Raw schedules dataset analysis. 112
A.17 Analysis of attributes in the raw schedules dataset. 112

A.18 Schedule shift type attribute values. . . . 112
A.19 Raw transactions dataset analysis. . . . 114
A.20 Analysis of attributes in the raw transactions dataset. . . . 115
A.21 Problems identified in the data. . . . 123
A.22 Cleaning solution for the problems identified in the data and step-by-step explanation. . . . 124
A.23 Events dataset attributes after the first data preparation. . . . 128
A.24 Example of records in the events dataset after the first data preparation. . . . 128
A.25 Footfall dataset attributes after the first data preparation. . . . 129
A.26 Example of records in the footfall dataset after the first data preparation. . . . 129
A.27 Schedules dataset attributes after the first data preparation. . . . 130
A.28 Example of records in the schedules dataset after the first data preparation. . . . 130
A.29 Transactions dataset attributes after the first data preparation. . . . 131
A.30 Example of records in the transactions dataset after the first data preparation. . . . 131
A.31 Shifts dataset attributes. . . . 132
A.32 Example of records in the shifts dataset. . . . 132
A.33 Open hours dataset attributes. . . . 133
A.34 Example of records in the open hours dataset. . . . 133

B.1 Input attributes for the M1 and M2 forecasting models. . . . 135
B.2 Input attributes for the M3, M4, M5 and M6 forecasting models. . . . 136
B.3 Constraints for an initialization example. . . . 137
B.4 Gaps information at iteration 1. . . . 138
B.5 Shift created at iteration 1. . . . 139
B.6 Gaps information at iteration 2. . . . 140
B.7 Shift created at iteration 2. . . . 140
B.8 Gaps information at iteration 3. . . . 141
B.9 Shift created at iteration 3. . . . 142
B.10 Gaps information at iteration 4. . . . 143
B.11 Shift created at iteration 4. . . . 144
B.12 Constraints for a crossover example. . . . 145
B.13 Slicing slot. . . . 147
B.14 Shift classification for parent 1 after the information is sliced. . . . 148
B.15 Shift classification for parent 2 after the information is sliced. . . . 148
B.16 Priority of the overlay shift of child 1 of the join process 150
B.17 Join process iterations for child 1. . . . 150
B.18 Shifts after the join process for child 1. . . . 150
B.19 Shifts of child 1 without break insertion for the joined shifts. . . . 150
B.20 Priority of the overlay shift of child 2 of the join process 150
B.21 Join process iterations for child 2. . . . 151
B.22 Shifts after the join process for child 2. . . . 151
B.23 Shifts of child 2 without break insertion for the joined shifts. . . . 151
B.24 Final shifts of child 1. . . . 152
B.25 Final shifts of child 2. . . . 153
B.26 Valid mutation slots and their candidates for the different mutation types. . . . 156
B.27 Valid slots for the reallocate mutation. . . . 156
B.28 Characteristics of the *reduce start* mutation. . . . 157
B.29 Characteristics of the *reduce last* mutation. . . . 158
B.30 Characteristics of the *extend start* mutation. . . . 159

B.31 Characteristics of the *extend last* mutation. 160
B.32 Characteristics of the *break* mutation. 162
B.33 Characteristics of the *join* mutation. 163
B.34 Characteristics of the *remove* mutation. 164
B.35 Characteristics of the *reallocate* mutation. 165

C.1 Importance of the input attributes in the M1 and M2 forecasting models. 167
C.2 Importance of the input attributes in the M2, M4, M5 and M6 forecasting models. 168
C.3 Evaluation metrics values for the forecasted number of transactions with the increasing error in the footfall in. 171
C.4 Evaluation metrics values for the forecasted average products per transaction with the increasing error in the footfall in. 172
C.5 Evaluation metrics values for the forecasted average price per product with the increasing error in the footfall in. 173
C.6 Evaluation metrics values for the direct forecasted sales with the increasing error in the footfall in. 173
C.7 Evaluation metrics values for the indirect forecasted sales with the increasing error in the footfall in. 174
C.8 Solutions descriptions in experiment $O1$. 194
C.9 Distance and similarity between the prescribed and manual solutions in experiment $O1$. 198
C.10 Allocation distances between the prescribed solutions in experiment $O2$. 205
C.11 Solutions descriptions in experiment $O2$. 205
C.12 Allocation distances between the prescribed solutions in experiment $O3$. 212
C.13 Solutions descriptions for experiment $O3$. 212
C.14 Allocation distances between the prescribed solutions in experiment $O4$. 216
C.15 Solutions descriptions for experiment $O4$. 216

Abbreviations

CP	Constraint Programming
DOS	Days-off Scheduling
DM	Data Mining
DMM	Demand Modeling
EAV	Employee Availability
EPR	Employee Preference
ERL	Employee Role
ESA	Employee Salary
ESN	Employee Seniority
ESK	Employee Skill
EVS	Explained Variance Score
HRM	Human Resources Management
IP	Integer Programming
GA	Genetic Algorithm
LP	Linear Programming
LOW	Line Of Work
MAE	Mean Absolute Error
MDAE	Median Absolute Error
MER	Max Error
MILP	Mixed Integer Linear Programming
MIP	Mixed Integer Programming
ML	Machine Learning
MSA	Multiple Store Assignment
MSE	Mean Squared Error
PSO	Particle Swarm Optimization
RMSE	Root Mean Squared Error
SHS	Shift Scheduling
STA	Staff Assignment
TKA	Task Assignment

Chapter 1

Introduction

Modern business runs more intelligent and at a faster pace each passing day. With the increasing advancement of technology, procedures are becoming shorter, more efficient and automated which leads to better resource exploitation and profit growth [1]. More data is also being created and stored which opens a great set of possibilities, such as the use of Data Mining (DM) to extract knowledge from data.

In organizations, the key factor that instigates investing in DM solutions is the possibility of enhancing the company's performance. Descriptive and predictive analytics are two very well known types of DM that are increasingly being used in human resources (HRM), a sub-area of management that focuses on managing employees strategically in an organization. Optimizing the processes in this area has a direct impact on the performance of the company and therefore, in creating competitive advantage [2]. In HRM, descriptive models can provide information and statistics about employees, salaries, schedules, sales and others. On the other hand, a predictive model could be used to estimate employee's satisfaction, fatigue and productivity. These models allow to extract knowledge from data and support decision making. For example, we could predict the productivity of the employees and pair those with different levels of efficiency in order to boost the productivity of those who are less efficient. However, although these models provide information that can be used in decision making, normally, this process is done manually by a manager. So, in order to make this process automatic and more efficient, we make use of prescriptive analytics, a recent type of DM analytics that aims at prescribing the best decision by using descriptive and predictive models to extract knowledge [3]. This is achieved by analyzing how good the outcome of the available decisions is, considering a set of predefined goals and optimization metrics. Prescriptive analytics leads to faster and better-supported decisions that are well justified in why they are being recommended. Implementing prescriptive models in organizations can support decision making by closing the gap between data scientists and managers.

With the continuous exposure to internal and external changes, a company must choose the best course of action as fast as possible in order to gain competitive advantage [1]. Managers are

responsible for making decisions in a company. However, despite having access to DM insights, they are not able to fully use this information for decision making. This glitch could either be caused by a lack of knowledge or time [4], due to the speed of modern business. Not taking full advantage of DM insights leads to not fully supported decisions. This could compromise a profitable reaction to change. Taking this into account, we need to close the gap between managers, those who decide, and data scientists, those who build the DM models. The insights provided by the models should be clear and understandable in a way that knowledge about DM should not be required.

1.1 Motivation

In recent years, the interest in applying DM techniques to HRM has increased, with particular interest in staffing, which deals with gathering, training, placing and retaining the best people for particular jobs or tasks in the organization [5]. However, descriptive and predictive analytics in staffing management is not enough to fully aid the decision support since it only gives insights about data. When the time comes to make a decision the presence of a manager is always required since no decisions were prescribed by this type of DM techniques.

In retail, customers are always potential buyers, therefore, shop floor employees have a great impact on sales since they interact directly with customers. However, converting customers into sales is dependent on customer flow. Ensuring the optimal number of employees on the store guarantees excellent customer support, avoiding under-staffing, without spending money on extra employees, avoiding over-staffing. Thus, staff scheduling, a process of staff management, is crucial.

1.2 Goals

This project aims to build a prescriptive model for staff scheduling in retail. The goal is to prescribe the best shifts and task allocation for each time interval, that leads to higher profit. For this, we focus on the integration of a sales forecasting architecture and a genetic algorithm for optimization.

A case study is conducted to test the approach developed on real-world data from a fashion retailer. Results show that the prescribed solution estimates more 3-9% sales than the manually designed by the retailer. On other experiments, we observed that the more accurate is the data and the nearest the prescription is to the planning horizon, the more efficient is the prescriptive model.

With this dissertation we contribute with a genetic algorithm based on shift definition and task allocation and two new architectures for sales forecasting. In the first, we introduce new structures for the representation of an individual and new methods for crossover and mutation.

In the end of this project, we aim at understanding the potential of this type of analytics and its limits. The relation between the performance of the forecasting models and the efficiency of the prescriptive solution is also studied and discussed.

1.3 Dissertation Structure

This dissertation is structured in five chapters, apart from this one, and three appendixes. Chapter 2 presents a review on the *state of art* regarding the scheduling problem in retail. For this, a brief literature review was made in order to be familiarized with the main applications and techniques used in this domain. Some important concepts are also introduced to be acknowledged in further chapters. In chapter 3 the problem is formally described as a scheduling problem and a prescriptive approach is proposed to address the problem. This chapter also presents information about real-world data provided by a retailer, used in the case study, along with its cleaning and preparation processes. Further details about the data can be consulted in the appendix A. Chapter 4 describes the implementation process of the proposed solution. This is divided into two parts: predictive and optimization modeling, which together form the prescriptive solution. Through this chapter some processes are described where step-by-step examples can be found in the appendix B. Later in chapter 5 a set of experiments are presented, in order to evaluate the proposed solution, along with the results and a detailed analysis. The information generated from these experiments is available in appendix C. At the end, chapter 6 summarizes the implemented solution and the project results obtained. The scientific contribution of this project is also discussed along with some recommendations for future work.

Chapter 2

State of Art

2.1 Scheduling

According to Michael Pinedo, scheduling is a decision-making process that consists in organizing, controlling and allocating resources to tasks for a specific time interval [6]. This process usually has one or more optimization goals and multiple restrictions to be full-filled, which depends on the domain, application and objectives.

Finding a global optimal solution can be hard. As shown by Garey and Johnson, scheduling is a NP-Complete [7] problem.

2.1.1 Staff Scheduling

Staff scheduling is a particular scheduling problem, related to organizations, where human resources must be assigned to tasks. Apart from the restrictions related to the type of tasks to be carried out, there are also working policies and other organizational restrictions that have to be taken into account. The more restrictions, the harder is to find the optimal solution.

From a Human Resources Management (HRM) perspective, staffing is a sub-area that deals with gathering, training, placing and retaining the best people for particular jobs or tasks in the organization. Staff scheduling arises as placing the best employees in the best tasks, jobs and shifts.

Ernst et al. propose a generic classification for the staff scheduling process [8], [9]. Nevertheless, not all stages are mandatory or handled as separate processes. Many scheduling solutions only require a subset of the modules proposed to achieve a staff scheduling solution. Others even combine a set of stages into a single process. The proposed classification modules by Ernst et al. are:

1. ***Demand Modeling***: Determine how many staff members are needed, at each time interval for the planning horizon, by predicting patterns and translate them into duties. These

patterns can be, for example, related to customer flow which is the number of customers entering and exiting the store. The demand can be one of three types:

(a) *Task-based demand*: A set of tasks are established as mandatory. They can be completed within their time window or carried out during a specific time interval.

(b) *Flexible demand*: The request for services have irregular arrival rates as well as irregular service times. Demand patterns are usually obtained through predictive models.

(c) *Shift based demand*: The number of working staff members per shift is defined.

2. ***Days off Scheduling***: Assigning the days off for each employee over the planning horizon, rather than shift assignment on working days.

3. ***Shift Scheduling***: Selecting the set of best shifts and the number of required staff members on a single day. It aims at meeting the demand modeled in the first stage.

4. ***Line of Work Construction***: Creating the line of work/schedule to be assigned to a staff member that satisfies a set of constraints and ensures demand fulfillment. It defines the schedule of an employee over the planning horizon. The line of work is defined by a set of blocks placed over a period of time and can be one of the three types:

   (a) *Shift*: Represents a working day and is appointed to each assigned staff.

   (b) *Duty*: Responsibility for assigned tasks that can take more or less than one shift.

   (c) *Sting*: Sequence of shifts and rest days.

5. ***Task Assignment***: Assigning a set of task with specific start and end time to a set of shifts. Each task may have different types of staff requirements such as skill, experience and others.

6. ***Staff Assignment***: Assigning staff members to each line of work. Normally done in parallel with the fourth stage.

In research, to face the problem of staff scheduling, many techniques have been used. However, surveys from Bergh et al. and Ernst et al. demonstrate that, across several areas of application, the most popular techniques are: integer programming, mixed integer programming, constructive heuristics, simulation and genetic algorithms [8]–[10]. Health care and nursing are the areas of application more frequently used in research. These areas may arise more interest for its complexity, dynamics and real-world data availability.

Bergh et al. also shows that real-world data is being commonly used in research. Testing and modeling with real data is advantageous since researchers have the opportunity to implement their approaches and measure their performance in the real world. Therefore, research and publications that used real-world data are more appreciated and trustworthy.

2.1.2 Scheduling in Retail

Retail can be described as the selling of products or services in small quantities directly to the customer through one single point [11]. Selling can occur in a market, a mall, a shop or through e-commerce. One of the main common goals of retailers is to maximize profit and convert customers into sales.

Surveys from Ernst et al. and Bergh et al. suggested that, until a decade ago, staff scheduling did not receive much attention in retail [8]–[10]. To understand the current status and analyze the evolution of research in this area of application, scientific publications from the last two decades were collected and carefully reviewed. According to table 2.1, research on the first years was slow. Later on, research in retail started to appear more frequently until 2015 where, until now, it seems to have stabilized, although with less regularity. In figure 2.1, one may see that, with an almost ten-year interval in the last two decades, the number of scientific publications has increased. This indicates a growing interest in applying staff scheduling techniques to retail.

Table 2.1: Publishing year of the scientific publications collected.

Year	Literature
1998	[12] [13]
2007	[14]
2008	[15] [16] [17]
2010	[18] [19] [20]
2011	[21] [22]
2014	[23] [24]
2015	[25] [26] [27] [28]
2016	[29] [30]
2017	[31]
2018	[32] [33]

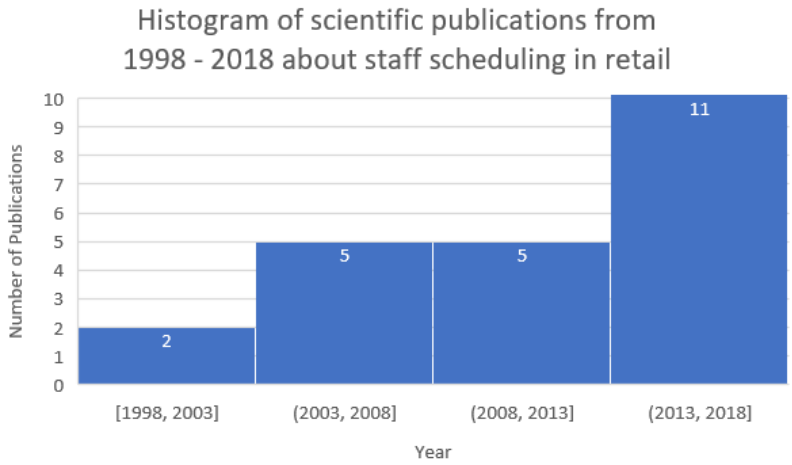


Figure 2.1: Histogram of scientific publications collected from 1998 to 2018 about staff scheduling in retail with bins of a five-year interval.

When comparing the techniques used to solve the staff scheduling problem in several areas [10], with the ones used in retail, we came to the conclusion that the most popular are not that different. Table 2.2 overviews the techniques used by the collected literature. Genetic Algorithms (GA), Constraint Programming (CP), Mixed Integer Programming (MIP) and Mixed Integer Linear Programming (MILP) are the most popular techniques.

Table 2.2: Optimization techniques used in the scientific publications collected.

Technique	**Literature**
Branch and Bound	[14] [19] [29]
Constraint Programming	[12] [14] [21] [23]
Constructive Heuristic	[22]
Genetic Algorithms	[18] [19] [22] [24] [33]
Integer Programming	[17] [20] [32]
Linear Programming	[14]
Mixed Integer Linear Programming	[16] [27] [26] [32]
Mixed Integer Programming	[15] [23] [25] [30] [31]
Other	[13] [28] [31]
Particle Swarm Optimization	[18] [24]
Queuing	[17]
Simulation	[15] [20] [32]

Scheduling problems seem easy to formulate under a MIP and MILP. Depending on the complexity and coverage of the constraints one may find it easier to formulate it as a CP problem, if too many restrictions define the problem. However, CP only finds a feasible solution which may not be optimal.

In the collected publications, Particle swarm optimization (PSO) was compared to GA [18], [24]. However, different publications favor different algorithms. GA was also compared with the branch and bound method and results demonstrated that in the majority of the cases, GA is more efficient [19]. The branch and bound method has produced bad solutions when the problem is highly constrained or with an objective function with several components. Contrarily, GA is very flexible in these conditions and can produce near-optimal solutions in any situation.

Real world data is commonly used to test and develop solutions, as can be seen in table 2.3. Data from fashion retailers are more commonly seen across publications. Some don't specify the source of the data and therefore, were classified as "other" in table 2.4.

Table 2.3: Scientific publications that used real-world data from retailers.

Real world data
[13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [31] [32] [33]

In table 2.5 the collected publications were categorized according to the classification in section 2.1.1. The names of the columns refer to demand modeling (DMM), days-off scheduling

Table 2.4: Scientific publications that use real-world data per type of retailer.

Supermarkets	Fashion	Home	Other
[14] [17] [20] [28]	[13] [15] [16] [18] [21] [22] [23] [25]	[26]	[19] [24] [27] [31] [32] [33]

(DOS), shift scheduling (SHS), line of work (LOW), task assignment (TKA) and staff assignment (STA). Overall, SHS and STA are the most common categories across publications.

Regarding DMM, some of the publications consider a flexible customer flow and therefore, the need for different number of store employees through the day. In some publications the staff demand is already provided, however, how they calculate the number of required employees through the planning horizon is not discussed in detail. Thus, in table 2.5 DMM was not contemplated for the publications in the situation described above. This may arise due to the fact that demand modeling is a complex problem and so, it is handled as a separate problem.

DOS and TKA are less frequent categories in comparison to others. TKA can be a complex problem if it involves considering employee skill, role and seniority, has suggested in table 2.6. In these cases, tasks have restrictions related to employee's characteristics, such as the ones motioned before, and can only accept staff assignment that full-fill all requirements. Normally the LOW blocks are shifts. However, in a few cases where tasks haven't the same time window of a shift, the blocks are duties.

In the staff scheduling optimization, some special considerations were identified in which most of them were shared across publications. Table 2.6 identifies the presence or not of these special considerations, such as employee preference (EPR), salary (ESA), availability (EAV), skills (ESK), roles (ERL), seniority (ESN) and application to multiple stores assignment (MSA). If these considerations are flexible, they can be part of the objective function. Otherwise, they must be expressed as restrictions.

From all the publications, ESA and EAV are the most common considerations in optimization. With ESA, employees are known as resources with cost and therefore, over-staffing can be avoided by introducing ESA in the objective function. EAV takes into account employee's vacations, illness and other days off. This is a special type of EPR, which considers the preference of employees to certain shifts. However, EAV is usually handled as a restriction where availability must be respected, contrarily, EPR is handled as part of the objective function where some flexibility is allowed.

EKS, ERL and ESN are characteristics that are normally used for task assignment or as restrictions for the team allocation to a shift. For example, a task may require a certain skill, role or seniority. On the other hand, the team in the store may also have restrictions that imply the presence of a certain number of employees with such expertise. For instance, the presence of a manager with at least two years of experience and with good leadership traits. ERL and ESN are characteristics that were rarely considered in past publications. Alternatively, ESK has been gaining more focus over the years.

MSA deals with staff scheduling in several stores and, according to the collected publications,

Table 2.5: Categories of scientific publications collected according to Ernest et al. [8].

Pub.	DMM	DOS	SHS	LOW	TKA	STA
[12]	•	•	•	shift		•
[13]	•			shift		
[14]	•		•	duty	•	•
[15]	•		•	shift		•
[16]			•	shift		•
[17]	•	•	•	shift		•
[18]			•	shift		•
[19]	•	•	•	shift	•	•
[20]	•		•	shift	•	•
[21]	•		•	shift	•	•
[22]			•	shift		•
[23]	•	•	•	shift	•	•
[24]			•	shift		•
[25]			•	shift	•	•
[26]			•	duty	•	•
[27]	•		•	shift		•
[28]			•	shift		•
[29]	•		•	shift		•
[30]	•	•	•	shift	•	•
[31]		•	•	shift	•	•
[32]			•	shift	•	•
[33]			•	shift	•	•
Total	12	6	21	-	11	21

is not being considered in the latest publications. Nevertheless, many publications that don't take this into consideration refer that they could easily adapt their solution to MSA, as long as employees can only be assigned to one particular store. The assignment of a single employee to more than one store has also been discussed as feature work. This particular problem as shown to be more complex since the spatial distribution of the employees and stores must be kept in mind.

Some publications include case studies estimating the impact on sales and profit of real retail stores. In 2011, Chapados et al. noticed that there would be an average sales increase of 7% across all stores and an operating income increase of 3% by implementing their solution [21]. Another solution was built in 2014 by the same authors with an estimated net profit increase of 2-3% [23].

2.2 Data Mining

Data Mining (DM) is the extensive analysis of data in order to extract knowledge and patterns that can be useful. A definition can be given by Hand et al. [34]:

Table 2.6: Special considerations in the collected scientific publications for schedule optimization.

Pub.	EPR	ESA	EAV	ESK	ERL	ESN	MSA
[12]							
[13]							
[14]	•	•	•	•			•
[15]		•	•				
[16]			•				
[17]	•	•	•	•			
[18]			•				•
[19]		•	•	•	•	•	•
[20]		•					
[21]	•	•					•
[22]			•	•	•		
[23]		•	•				•
[24]	•		•	•	•		•
[25]	•	•	•	•	•		
[26]		•		•			
[27]	•	•		•			
[28]		•					
[29]		•					
[30]			•	•			
[31]			•	•			
[32]	•		•	•			
[33]	•	•	•	•	•		
Total	8	13	14	12	5	1	6

"Data mining is the analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner."

Gudivada divides DM analytics into four functional categories: *Descriptive*, *Diagnostic*, *Predictive* and *Prescriptive* [35]. These categories are viewed as steps in the analytics work-flow and represent the evolution of the analytics domain, rather than a delimitation between functions. In fact, these types of analytics usually tend to overlap. A simple description of the different DM analytics and their purpose are:

- ***Descriptive Analytics***: Provides insights into the past. Exploratory analysis is done on past data in order to obtain relevant data and statistics. The goal is to summarize what happened in the past and make this information understandable.

- ***Diagnostic Analytics***: Provides explanatory insights about the events in the past. In contrast to descriptive analytics, that provides "what" happened on the past, diagnostic analytics tries to answer "why" it happened.

- ***Predictive Analytics***: Provides insights into the future based on past events. Predictive models are used to forecast what will probably happen in the future.
- ***Prescriptive Analytics***: Provides insights about the best course of action to be carried out to fulfill a maximization goal. This is done by evaluating the outcome of different possible actions, using predictive models and other insights.

Figure 2.2 pictures the different types of analytics as part of the analytics work-flow. Each step must provide insights that can answer different types of queries. The natural order and queries assigned to the different types of analytics are: Descriptive – *What happened?*, Diagnostic – *Why did it happen?*, Predictive – *What will happen?* and Prescriptive – *What should be done?*. The higher the step the higher the usefulness of the information provided and the more difficult is modeling.

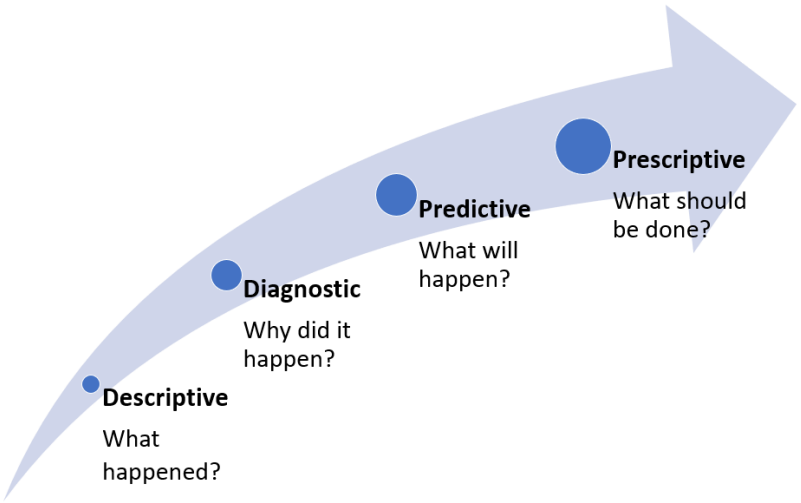


Figure 2.2: Data mining analytics work-flow.

In an organization, a descriptive analysis could provide information about its employees, customers, projects, capital, transactions and others, for example, the evolution of the organization's capital through the years. A diagnostic analysis could explain that evolution by tracing the data entries that affected the capital, like the selling and purchase of goods. A predictive model could forecast if a project will likely be successful or not. This model could avoid wasting money with projects that are likely to fail. On the other hand, it could encourage projects that help increase the profit. Given several models that, according to predictive models, will probably succeed there may not be enough resources to invest on all of them. A prescriptive model takes into account the predictions, the resources and the business goals to recommend the best decision.

2.2.1 Prescriptive Analytics

A prescriptive analysis can be used to implement a decision support system. With the use of descriptive and predictive models, it estimates the impact of different decisions and, according to a set of optimization measures and goals, it prescribes the best decision.

This method differs from other optimization methods because it uses forecast models to directly evaluate the solution at each iteration. This has the advantage of evaluating the current solution using a model trained on historical data. However, the performance of the prescriptive model is directly associated with the truth of the results forecasted by the predictive model. Thus, if this model isn't accurate, the schedules produced by the prescriptive model are not reliable.

The results achieved are evaluated by comparing the estimated impact of the prescribed solution with the impact that this had in the real world.

2.3 Prescriptive Analytics for Staff Scheduling in Retail

Prescriptive analytics is still at a very early stage. Regarding the staff scheduling problem applied to retail, from all the collected publications, only one implements a prescriptive solution. This publication does not even associate its approach to be a prescriptive analytics approach.

Chapman et al. suggested the use of a predictive model, later used as part of the objective function, that forecasts the operating income at a specific time t [21]. This model receives as input the number of selling employees, the store traffic and other explanatory variables at time t. This last is related to calendar and store events. This model can be cast into three separated forecasting models: customer flow forecasting, average price forecasting and sales volume forecasting at time t. With the main forecasting model, a sales curve can be built to understand the impact of changing the number of employees in the store. Therefore, the optimization model uses the output of the predictive model as a measure of quality in the objective function to maximize the profit.

Evaluating the results in this area is not simple. It implies applying the developed solution in a retail store and measure if the sales have improved or not. Nevertheless, customer flow has a great impact on sales and measuring the prescriptive model can be hard. During the test of the model in the store, more or fewer sales can be recorded, not because of the model's performance but because of the customer flow. It is important to make evaluations in long periods so that abnormal customer flow does not have a great weight in sales. Chapman et al. evaluate their results by comparing the hypothetical profit, associated with the schedules produced by its prescriptive model, with the hypothetical profit of the schedules used by the retail store at the same time period.

Chapter 3

Problem Description

3.1 Problem Formalization

In retail, shop floor employees have a great impact on sales since they interact directly with customers. Guaranteeing that all customers requests are full-filled, in the shortest time and best way possible, increases the possibility of customers being converted into sales. This could easily be accomplished by over-staffing, however, money and resources would be wasted on unnecessary staff. On the other hand, under-staffing could help save money on staff wages, but for lack of customer coverage, potential sales would not occur which would affect results negatively. For this reason, it is crucial to find a balance and get the best staff schedules that maximize profit, by assuring the optimal number of working employees through the planning horizon.

The problem is that customer flow isn't the same and also not easily calculated. This means that different optimal number of shop floor employees may be needed depending on the customer flow. If more clients arrive at the store in a small time period, more coverage is required and therefore, more floor shop staff. According to the demand modeling in section 2.1.1, retail can be categorized as flexible demand. This must be taken into consideration when dealing with scheduling since employee demand is not the same throughout the day.

Several factors can impact employee demand. They are mostly the ones that directly affect the customer behavior, such as:

- ***Time of the day***: Different times during the day means different levels of availability for different customers. Working hours is a time period where customers aren't available to shop. Although people have different working hours, there is a time interval where a lower customer flow can be observed. For example: from 9 am to 5 pm.

- ***Day of the week***: The days when customers are more often available tend to have higher customer flow. Working days have a huge impact since customers during their working hours are not available to enter the store. Although it's not the same for everyone, at the end

of the week it is normal to have higher customer flow since the majority of customers is not working. In contrast, at the start of the week there is a lower costumer flow since customers have probably full-filled their shopping needs recently and are busier.

- ***Seasonality***: There are products and services that are usually bought at a specific time period of the year. For example, in fashion retail, clothes are divided into winter, spring, summer and autumn collections. If a store only sells products of a particular collection then the customer flow will decrease in the other seasons.

- ***Calendar event***: Holidays and special dates attract more customers to retail stores. Depending on the type of event, the customer flow can increase almost proportionally as the event date arrives or the effect on the flow may be only on the day itself. Different types of retail stores may profit more or less depending on the type of event.

- ***Store event***: Special discounts or campaigns can attract more customers. These are viewed as opportunities by the customers and therefore, the higher the opportunity the higher the customer flow. This can have a lower effect if the competition offers a better opportunity.

The goal of this project is to propose a solution that deals with demand modeling and shift scheduling, phases described in section 2.1.1. Demand modeling will try to model the customer flow and determine the optimal number of employees throughout the day, for each task, in order to maximize profit. Shift scheduling deals with choosing the best shifts and build the lines of work for the planning horizon.

3.2 A Prescriptive Analytics Solution for Staff Scheduling

We propose a solution that prescribes shifts and task allocations to maximize store profit. The proposal is tested on real-world data from a fashion retailer.

This solution focus on the integration of a predictive model and an optimization algorithm to build a prescriptive solution. For this, out of the shelf techniques are used, such as genetic algorithms for optimization and random forest for forecasting. Details about the implementation of these techniques will be further described in chapter 4. In the following subsections, a generic description of each phase, prediction and optimization, is introduced to understand how both will be integrated to build a prescriptive model.

From a staff scheduling perspective, data can be split into two categories: uncontrollable and controllable variables. The former are those that can't be modified during the optimization process, such as, store information, working policies, products and others. The latter represent the decision variables related to the schedules and their characteristics, such as shift boundaries and the number of employees allocated to tasks.

3.2.1 Predictive Modeling

The main goal of the prescriptive model is to maximize a store's profit. However, in order to assess the quality of a solution, we need to estimate the impact it would have if it had been in place. This estimation is necessary because it is not possible to assess the real impact of a solution in practise (i.e. what would the sales have been if the staff schedule was the one under evaluation). Therefore, forecasting models are necessary to estimate the quality of a solution (i.e. the sales if that solution was implemented). The output of the model is later used as part of the objective function in optimization, as explained in section 3.2.3. The forecasting component does not need to be restricted to a single model (i.e. a model to predict sales). In fact, a composition of models, solving sub-tasks (e.g. predicting store footfall and average basket value), this combination leads to a more complex forecasting architecture (e.g. the sales forecasting model uses as input the predictions made by the footfall prediction and average basket value models, as well as other variables), as suggested by Chapados et al. [21].

With a sales forecasting model, a sales curve can be created representing the variation in sales through time. The area under this curve represents the total sales of a store and as expected, the higher the area under the curve, the higher the forecasted sales. Figure 3.1 shows an example of a sales curve.

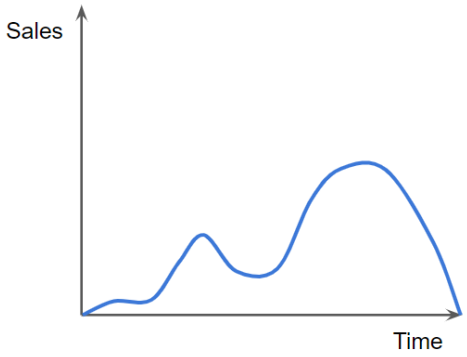


Figure 3.1: Example of a sales curve.

Since it's not possible to build a continuous sales curve, the sales must be predicted for a time interval. Hereby, we can divide time into slots, with the same time unit, and sales are estimated for each in order to build an approximated sales curve.

3.2.2 Optimization Modeling

The optimization model aims at prescribing the best solution according to a set of goals and metrics. In this project, the goal is to prescribe a set of employees schedules that together maximize the profit of a store. For this, the set of uncontrollable and controllable variables are associated to each planning day and the profit is calculated for each one (figure 3.2).

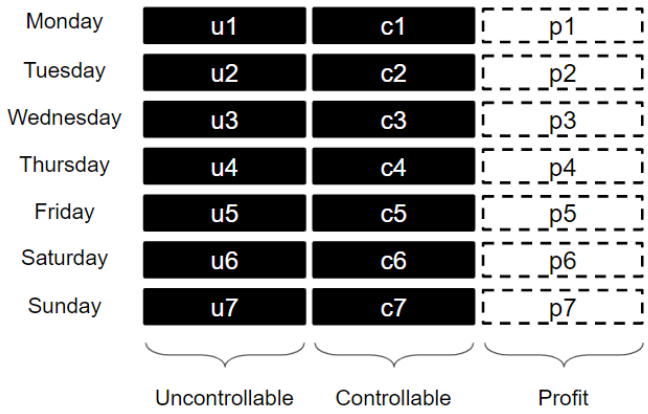


Figure 3.2: Distribution of variables and objective function target across the planning horizon of a week for the optimization phase.

In optimization, a set of predefined constraints are presented to guarantee the feasibility of a solution. For example, an upper manager may decide to limit the number of employees available for each day of the planning horizon. These restrictions are characterized as uncontrollable variables, since they are fixed and not modified during optimization.

In order to optimize a solution, at each iteration, different combinations of controllable variables are tested and each solution is evaluated by an objective function. This takes into account, the sales produced for the planning horizon minus the costs of the working employees. If any uncontrollable variable is violated, then the score can suffer a penalization or be completely discarded. Solutions with higher scores are naturally favored leading to better solutions through iterations. In general, the optimization stops after a selected number of iterations or after the best solution has stabilized and no further improvements were recorded.

3.2.3 Prescriptive Modeling

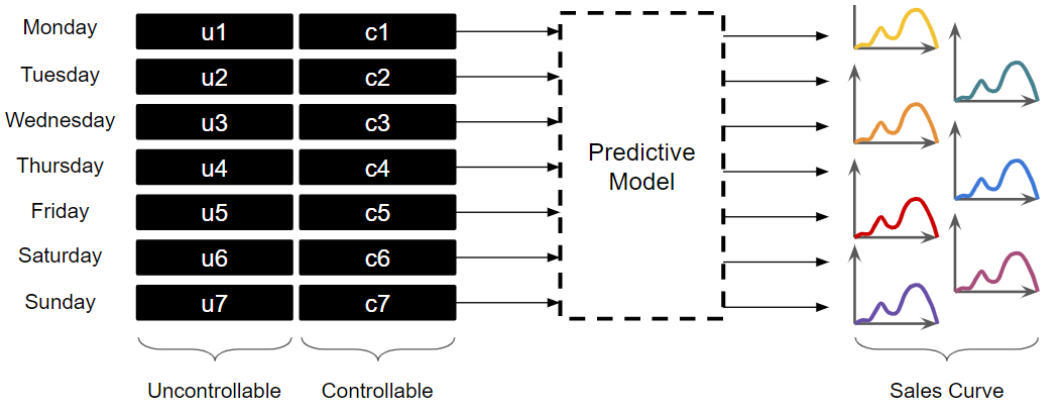


Figure 3.3: Optimization iteration in prescriptive analytics.

By combining a sales forecasting architecture and an optimization algorithm, a prescriptive model can be built to solve the staff scheduling problem. The objective function used in the

optimization process uses the sales predicted by the forecasting model to estimate the profit of the corresponding solution. Therefore, to evaluate a solution, a sales curve is predicted for each planning day and the area under all curves is summed up to estimated sales (figure 3.3).

During optimization, the values of the controllable variables provided by the solutions under analysis will lead to different sales curve. The goal is to find the sales curve that together with the costs of the allocated employees, allows achieving the maximum profit of the store.

3.3 Data

3.3.1 Data Description

For this project, real-world data was provided by a fashion retailer on four of its stores. For confidentiality reasons, the retailer can not be named. The data contains information that is directly related to the sale of products in store and is divided into four datasets, which contains historical data about:

- Events
- Footfall
- Schedules
- Transactions

The events dataset contains information about calendar and store events. For example, a calendar event can be related to a holiday and a store event to a promotion. Table 3.1 shows the attributes in this dataset, along with the description for each one, and table 3.2 some records from this dataset.

Table 3.1: Raw events dataset information.

Attributes	**Description**
location	Store id.
start	Start date of the event.
end	End date of the event.
type	Type of event.

Table 3.2: Example of records in the events dataset.

location	start	end	type
Store_1	20180601	20180601	EVENT-PUBLIC
Store_1	20180610	20180610	CALENDAR-PUBLIC
Store_1	20180624	20180624	CALENDAR-LOCAL
Store_1	20180715	20180915	PROMOTION-LOCAL

In the footfall dataset, we can access information about entries and exits of the store. This allows us to track and estimate customer flow, which has a direct impact on demand. In table 3.3 we can verify the attributes provided in this dataset. The attribute *time* is related to the start time of the footfall count. To obtain the time range associated with each footfall count, it's necessary to explore other records and see the difference between the *time* attribute. Table 3.4 shows some records from this dataset.

Table 3.3: Raw footfall dataset information.

Attributes	**Description**
location	Store id.
date	Date of the footfall count.
time	Start time of footfall count.
footfall in	Number of store entries.
footfall out	Number of exits in store.

Table 3.4: Example of records in the footfall dataset.

location	date	time	footfall in	footfall out
Store_1	20190110	90000	6	0
Store_1	20190110	100000	5.25	0
Store_1	20190110	110000	24.2591	0
Store_1	20190110	120000	27.2433	0

The schedules dataset provides us with information about the employee's allocation to shifts and tasks and their corresponding start and end time. Table 3.5 describes the information present in this dataset and table 3.6 shows some examples of records.

Table 3.5: Raw schedules dataset information.

Attributes	**Description**
location	Store id.
employee code	Employee id assigned to the shift.
date	Date of the shift.
start	Start time of the shift.
end	End time of the shift.
shift type	Types of task.

Table 3.6: Example of records from the schedules dataset.

location	employee code	date	start	end	shift type
Store_1	Employee_39	2019-01-10	10:00:00	15:00:00	SALES
Store_1	Employee_39	2019-01-10	15:00:00	16:00:00	BREAK
Store_1	Employee_39	2019-01-10	16:00:00	19:00:00	SALES
Store_1	Employee_47	2019-01-10	10:00:00	00:00:00	OTHERS

The transactions dataset contains information about the sales and returns of products. According to table 3.7, each transaction is associated with a product, a location and an employee. In table 3.8 we can see how records from this dataset look like.

Table 3.7: Raw transactions dataset information.

Attributes	**Description**
location	Store id.
product	Product id.
ticket code	Transaction id.
date	Date of transaction.
hour	Hour of transaction.
employee code	Employee responsible for the sale.
qty	Product quantity. Sales are represented with positive values and returns with negative values.
net sales	Total profit of the product's transaction.

Table 3.8: Example of records from the transactions dataset.

location	product	ticket code	date	hour	employee code	qty	net sales
Store_1	Product_21278	ticket_33527	20190110	115210	Employee_39	1	89
Store_1	Product_22267	ticket_33527	20190110	115210	Employee_39	1	63.2
Store_1	Product_25671	ticket_33527	20190110	115210	Employee_39	1	110
Store_1	Product_23520	ticket_21373	20190110	115621	Employee_6	1	25

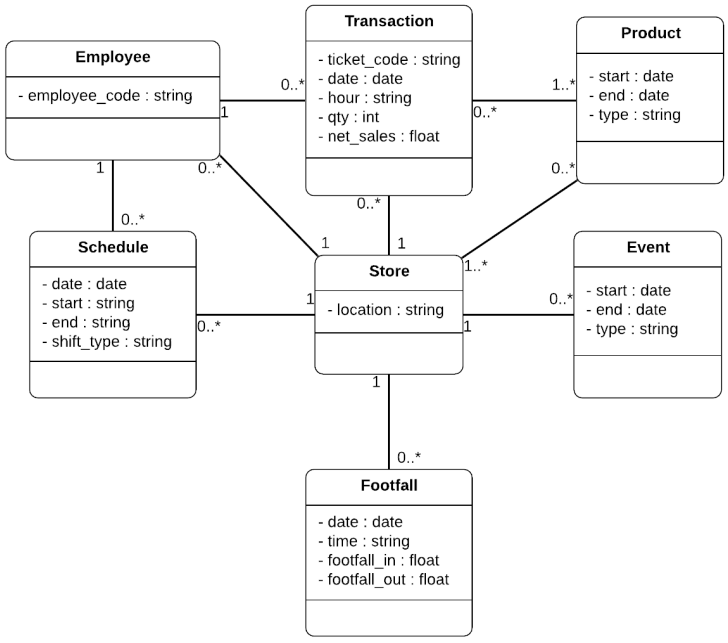


Figure 3.4: UML class diagram for the data provided.

From the data provided and described above, objects and their relationships can be identified and represented as a UML class diagram (figure 3.4). The objects identified are: employee, schedule, transaction, store, event and footfall. A store can be associated with every other type of objects but it is not mandatory. Conversely, every other type of objects must be associated with a store. An employee can be assigned to multiple schedules but a schedule can only be associated with one employee. Employees are also responsible for transactions and therefore, can be associated with multiple objects. Each transaction has to be at least associated with a product, regardless if it's a sale of a return. Products reference are shared between the same chain of stores thus, it can be related to more than one store.

More record examples of each dataset can be consulted in the appendix section A.1.1. A data analysis was also done for these datasets and is available in the appendix section A.2. The data contains information about four stores. However, this project only focus on prescribing the schedule for one store, therefore, only data about store *Store_1* is presented.

3.3.2 Data Cleaning

After carefully analyzing the data described in the previous section (appendix section A.2), some issues were identified in the data and gone through a cleaning process. Tables 3.9, 3.10, 3.11 and 3.12 contains information about the problems identified for the events, footfall, schedules and transactions datasets, respectively, along with a brief explanation of how they were handled. Step-by-step information and a more extensive explanation for each cleaning solution can be consulted in the appendix section A.3.

Table 3.9: Problems identified in the events dataset and their respective cleaning solutions.

Code	Problem	Solution
E1	The attributes *start* and *end* are represented by integers. Due to this, records between [00:00:00,09:59:59] are represented by only one digit and the others by 6 digits.	Convert to a 6 digits representation and then to integer.

Table 3.10: Problems identified in the footfall dataset and their respective cleaning solutions.

Code	Problem	Solution
F1	The attributes *date* and *time* are in the integer format (same situation described in problem E1 of table 3.9).	Convert to the correct format.

F2	Negative values in the *footfall in* attribute. It's impossible for the number of entries in the store to be less than zero.	Update the *footfall in* to zero when it's a negative value.
F3	Footfall is recorded for each hour, however, there are no records for some time intervals. For example, between 01:00h and 8:00h.	Insert new records in the dataset for the missing time intervals of each day with zero *footfall in* and *footfall out*.
F4	The attribute *footfall in* is not an integer.	Without knowing the source of this error or how the number of entries is calculated, we decided not to handle it.
F5	Anomaly in the values of the *footfall in* attribute after February in 2019.	Change the *footfall in* outliers with the average value.
F6	The attribute *footfall out* is always zero.	There is no solution for this case because we can't try to estimate the *footfall out* value since there is no historical data.

Table 3.11: Problems identified in the schedules dataset and their respective cleaning solutions.

Code	**Problem**	**Solution**
S1	The attributes *date*, *start* and *end* are in the string format.	Convert to the correct format.
S2	Employees with overlapping shifts.	Adjust the overlapping shift's boundaries so that they will be side by side instead of overlapping.
S3	Some BREAK shifts are not between two non-break shifts.	Remove the break shift if there is no shift before and after or adjust its *start* and *end* to be side by side with the previous and next shift.

Table 3.12: Problems identified in the transactions dataset and their respective cleaning solutions.

Code	**Problem**	**Solution**
T1	The *date* and *hour* attributes are in the integer format. Due to this (same situation described in problem E1 of table 3.9).	Convert to the correct format.

T2	Lack of information between the start of year 2017 until March 2018, in comparison with the past years.	Information is discarded until March 2018. Later in chapter 4 section 4.1, it's explained that each row of data will contain information about past values so that time patterns can be identified. This means that data from 2018 would rely on past information and, since it's not reliable, all information is discarded until then.

3.3.3 Data Preparation

After the cleaning process, the data is prepared so it can be used later in the predictive modeling and training. In this section, only a brief explanation and demonstration of the final attributes is presented. For more details of the preparations steps consult the appendix section A.4.

An important aspect of the data preparation is that the information is compacted into time intervals throughout the day, based on the same time unit. If no information is available for a specific time interval, then, in general, zero is assigned to the attributes. This is crucial because in optimization, employees will be allocated to time slots and not to a continuous space. Therefore, sales forecasting will also be related to a time interval. In this case, the information was compacted using a 30 minutes interval where the starting date is recorded in the attribute *datetime*. This means that the information in a record is associated to a time interval ranging between $[datetime, datetime + 00:30h[$.

Tables 3.13, 3.14, 3.15 and 3.16 contain information about the final attributes for the events, footfall, schedules and transactions datasets, respectively, after the data preparation process.

Table 3.13: Events dataset attributes after the first data preparation.

Attributes	Description
location	Store id.
datetime	Start date of the associated time interval.
event_calendar-local	Binary attribute corresponding to the presence or not of the event *calendal-local*.
event_calendar-public	Binary attribute corresponding to the presence or not of the event *calendal-public*.
event_event-public	Binary attribute corresponding to the presence or not of the event *event-public*.
event_promotion-local	Binary attribute corresponding to the presence or not of the event *promotion-local*.

Table 3.14: Footfall dataset attributes after the first data preparation.

Attributes	Description
location	Store id.
datetime	Start date of the associated time interval.
footfall_in	Number of store entries.
footfall_out	Number of exists in store.

Table 3.15: Schedules dataset attributes after the first data preparation.

Attributes	Description
location	Store id.
datetime	Start date of the associated time interval.
total_employees	Number of working employees.
n_at_sales	Number of employees allocated to 'SALES'.
n_at_others	Number of employees allocated to 'OTHERS'.
n_at_break	Number of employees allocated to 'BREAK'.

Table 3.16: Transactions dataset attributes after the first data preparation.

Attributes	Description
location	Store id.
datetime	Start date of the associated time interval.
n_transactions	Number of transactions (different *ticket code* values).
n_products	The sum of products transacted.
total_sales	Total sales.
avg_products_per_t	Average number of products transacted per each transaction.
avg_price_per_product	Average price per each product.
avg_sales_per_t	Average sales profit per each transaction.

After the data is cleaned, a new dataset was created containing the shifts that were extracted from the schedules dataset. A shift is defined by a set of non-overlapping tasks assigned to the same day, previously to one employee. Table 3.17 contains the attributes presented in this dataset along with the description for each one.

Table 3.17: Shifts dataset attributes.

Attributes	Description
location	Store id.
shift_id	Shift id.
shift_order	Order of task associated to the shift.
shift_duration	Total duration of the shift.
task_duration	Duration of the task.
start_time	Start time of the task.
end_time	End time of the task.
task_type	Type of task.

With the access to the shifts and footfall datasets, the open hours of the store were extracted from the data and a new dataset was created. We consider that the store is open at the earlier recorded footfall or sales task. For the closing of the store, the same logic is applied but for the latest recorded footfall or sales task. Table 3.18 summarizes the attributes of this dataset with a brief description.

Table 3.18: Open hours dataset attributes.

Attributes	Description
location	Store id.
datetime	Start date of the associated time interval.
close_hours	Close hours of the store.
max_employees	Maximum employees used.
open_hours	Open hours of the store.
weekday	Number of the weekday between [0,6] starting on Sunday.

For a better visualization of the contents of each prepared dataset, some record examples can be consulted in the appendix section A.1.2.

Chapter 4

Implementation

4.1 Forecasting

Sales forecasting is not a simple problem. There are many factors that can influence sales and using them can be tricky. Some of this information may not be accurately calculated or available beforehand. One approach is to develop and train models to previously forecast this information so it can be accessible for other forecasting tasks. In this case, the sales forecasting architecture may have a chained structure.

4.1.1 Forecasting Architecture

In the data provided and described in section 3.3, five major types of information can be identified: time, events, footfall, schedules and transactions (sales). The most significant dependencies to forecast sales for time t are: controllable and uncontrollable events e that affects footfall in fi and footfall out fo which, in turn, along with the employees schedules s, determines sales $sales$ (statement 4.1).

$$t \rightarrow e \rightarrow fi, fo, s \rightarrow sales \tag{4.1}$$

Information about time and events is available in advance, and the schedule is a decision variable, available at the moment of forecast, since it is determined by the store management or by the optimization algorithm. However, the remaining information can only be acquired at the moment it occurs. Without it, we are limiting the ability of a forecasting sales model, since essential information is not being fed to the model. One approach to address this obstacle is to train models to forecast the missing information as intermediate variables. This applies to the variables footfall in, footfall out, number of transactions, average products per transaction and average price per transaction. The last three variables are information that can be extracted from

the transactions information. Table 4.1 shows the variables classification as input, decision input, intermediate target and final target.

Table 4.1: Variables classification.

Variable	**Type**
Average Price per Transaction	Intermediate Target
Average Products per Transaction	Intermediate Target
Events	Input
Footfall In	Intermediate Target
Footfall Out	Intermediate Target
Number of Transactions	Intermediate Target
Sales	Final Target
Schedule	Decision Input
Time	Input

From here, there are two approaches to forecast sales:

1. Build a chain of models, some of which predict important attributes which are not known beforehand; then a final model predicts the target variable using attributes and the predictions of the intermediate models.

2. Predict the target variable directly from the attributes.

Taking into account the two approaches to forecast sales above, we have developed two different architectures. The first follows the first approach directly, by developing models for all intermediate variables. The second architecture is a mix of both: we develop intermediate models only for the footfall and predict sales directly from the attributes and the estimated footfall values. For this reason the first architecture is referenced as **indirect architecture** and the latter, the **direct architecture**, since one forecasts sales indirectly after the footfall is predicted and the other directly.

For the development of the two architecture six models were trained to forecast: footfall in (*M1*), footfall out (*M2*), number of transactions (*M3*), average products per transaction (*M4*), average price per transaction (*M5*) and sales (*M6*). Table 4.2 shows the target of each model along with the input information necessary.

Initially, both architectures follow the same logic to prepare the input data for the final models. For this, the information about the time and events are joined and data is prepared, according to section 4.1.2, to forecast footfall in and footfall out. Afterwards, the forecast data is joined with the events, time and schedule information and prepared for each model responsible for sales forecasting.

In the indirect architecture we decompose the problem of forecasting sales into three subproblems and try to forecast: the number of transactions, the average products per each transaction

Table 4.2: Input and target variables of the different models.

Model	Input	Target
M1	time, events	footfall in
M2	time, events	footfall out
M3	time, events, footfall in, footfall out, schedule	number of transactions
M4	time, events, footfall in, footfall out, schedule	average products per transaction
M5	time, events, footfall in, footfall out, schedule	average price per transaction
M6	time, events, footfall in, footfall out, schedule	sales

and the average price per each product. By decomposing the problem we try to specialize each model in solving a sub-problem so that when combined, the sub-problems can solve the original problem in an efficient way. Figure 4.1 represents the data flow for this architecture.

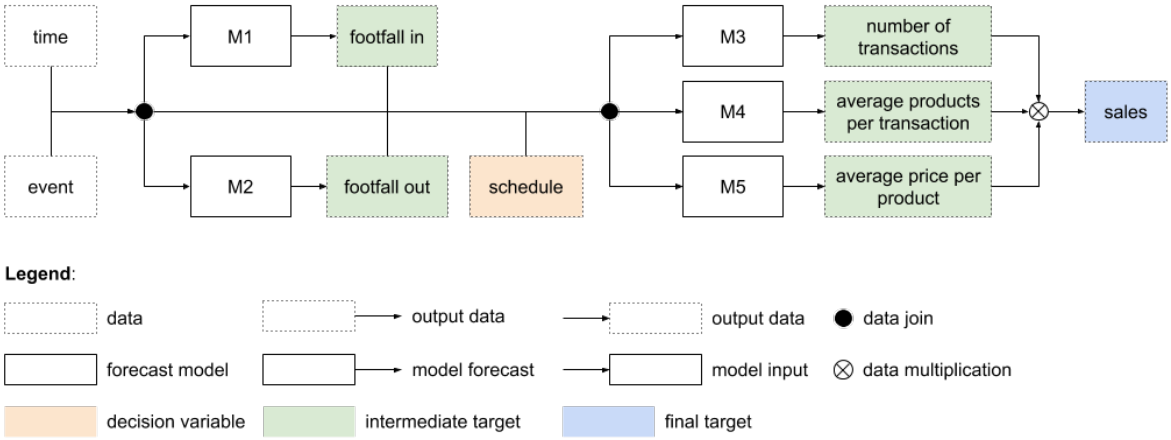


Figure 4.1: Indirect sales forecasting architecture.

The direct architecture is more straightforward and forecasts sales by using only one model. This architecture can be represented in figure 4.2.

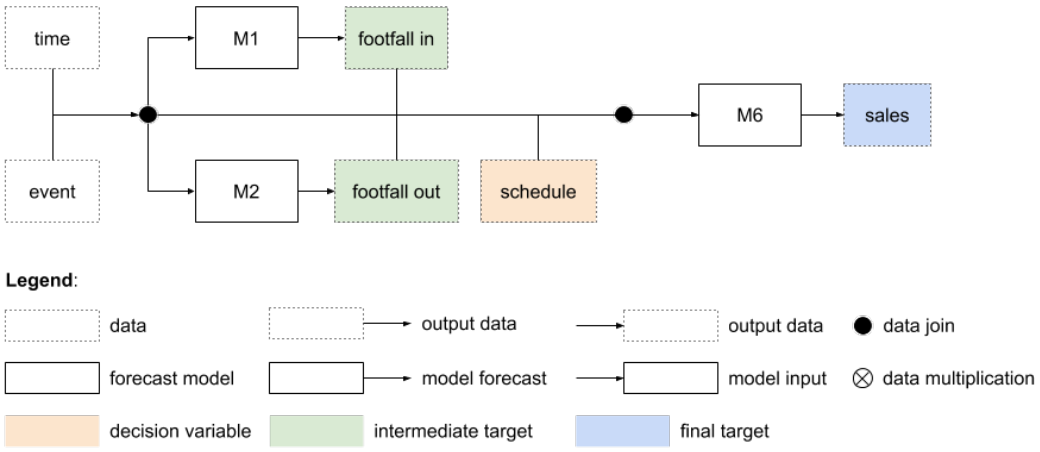


Figure 4.2: Direct sales forecasting architecture.

Depending on the architecture used, sales can be calculated by eq. 4.2. If a direct architecture is used, then the sales are the direct output of the sales model (*M6*). Otherwise, it's the multiplication of the *M3*, *M4* and *M5* model's output.

$$sales_{t,e,fi,fo,s} = \begin{cases} M6_{t,e,fi,fo,s}, & \textit{if direct_sales} \\ M3_{t,e,fi,fo,s} * M4_{t,e,fi,fo,s} * M5_{t,e,fi,fo,s}, & \textit{otherwise} \end{cases} \tag{4.2}$$

The random forest was used to learn the models. This algorithm was chosen by its popularity, which can be explained by consistently good results across a wide variety of problems. Although many other algorithms could be used (including time series algorithms), since this project focuses on building a prescriptive approach, which combines an optimization approach besides the ML models, we have decided to restrict the algorithms tries to random forest.

4.1.2 Feature Engineering

In this section, a second data preparation process is presented, specific to the predictive modeling approach described in the previous section. This process consists in building additional *time* and specific *target-oriented* attributes, for seasonality inference, and prepare the input data for each forecasting model.

To prepare the input data for any of the models, presented in the previous section, two main steps are required:

1. Gather the attributes associated to the target, according to table 4.2, and inner join the information on attribute *datetime*.

2. Create new attributes through feature engineering:

   (a) Extract different time attributes and represent each as cyclical.

   (b) Create attributes containing past values of the target.

After the data preparation, described in subsection 3.3.3, data is now a set of time series. Thus, seasonality, which is a common characteristic of time series which we also expect to exist in our problem, can be captured and may improve the forecasting models. However, as discussed in section 4.1.1, the models developed for forecasting, are traditional models and are not prepared to deal with time series specifically. For this reason, new attributes are created to infer seasonality and let regression deal with the problem. Given the context of this project, five types of seasonality are considered: **daily**, **weekly**, **monthly**, **quarterly** and **yearly**.

According to the types of seasonality listed above, time can be represented by the minutes of the day, day of the week, day of the month, day of the season and day of the year, respectively (table 4.3). This information allows to express the position of information in the cycle related to each seasonality. As an example, date 2019-01-01 10:33:00 corresponds to the 273^{th} minute of the day, 2^{nd} day of the week, 1^{st} day of the month, 11^{th} day of the season and 1^{st} day of the year.

Table 4.3: Time representation for each seasonality

Seasonality	**Time**	**Cycle Duration**
Daily	Minute of the day.	24*60
Weekly	Day of the week.	7
Monthly	Day of the month.	Number of days of the month.
Quarterly	Day of the season.	Number of days of the season.
Yearly	Day of the year.	366 if leap year, 365 otherwise.

Time is a cyclical attribute by nature. However, just using one attribute to represent this type of information is impossible. To solve this problem, the seasonality times are extracted from the *datetime* attribute and two new others are created through equations 4.3 and 4.4. This is a cyclic representation of time on a two-dimensional space through *sine* and *cosine* coordinates. Cycles like month, season and year don't have a fixed size. Therefore, it has to be specific to each time. Table 4.4 shows an example of the new created attributes for *datetime* 2016-01-02 10:00:00.

$$coord_x = \sin\left(\frac{2 * \pi * time}{cycle_duration}\right) \tag{4.3}$$

$$coord_y = \cos\left(\frac{2 * \pi * time}{cycle_duration}\right) \tag{4.4}$$

Table 4.4: Example of the time coordinates values for *datetime* 2016-01-02 10:00:00

Attribute	**Example**
coord_x_day_of_month	0.394356
coord_x_day_of_season	0.743145
coord_x_day_of_week	-0.974928
coord_x_day_of_year	0.034328
coord_x_min_of_day	0.500000
coord_y_day_of_month	0.918958
coord_y_day_of_season	0.669131
coord_y_day_of_week	-0.222521
coord_y_day_of_year	0.999411
coord_y_min_of_day	-0.866025

To infer seasonality, each record must contain information about the target's past values. Therefore, for time t, day d, week w, month m, season s and year y, new attributes are created with information about:

- Target values on the past minutes t: $target_{t-1}$, $target_{t-2}$, $target_{t-3}$, $target_{t-4}$, $target_{t-5}$;
- Target values on the past day d and past minutes t: $target_{d-1,t}target_{d-1,t-1}$, $target_{d-1,t-2}$, $target_{d-1,t-3}$, $target_{d-1,t-4}$, $target_{d-1,t-5}$;
- Target values on the past week w and past minutes t: $target_{w-1,t}target_{w-1,t-1}$, $target_{w-1,t-2}$, $target_{w-1,t-3}$, $target_{w-1,t-4}$, $target_{w-1,t-5}$;

- Target values on the past month m and past minutes t: $target_{m-1,t} target_{m-1,t-1}$, $target_{m-1,t-2}$, $target_{m-1,t-3}$, $target_{m-1,t-4}$, $target_{m-1,t-5}$;

- Target values on the past season s and past minutes t: $target_{s-1,t} target_{s-1,t-1}$, $target_{s-1,t-2}$, $target_{s-1,t-3}$, $target_{s-1,t-4}$, $target_{s-1,t-5}$;

- Target values on the past year y and past minutes t: $target_{y-1,t} target_{y-1,t-1}$, $target_{y-1,t-2}$, $target_{y-1,t-3}$, $target_{y-1,t-4}$, $target_{y-1,t-5}$;

Since each record represents information associated to a 30 minutes interval, the $target_{t-n}$ corresponds to the target's past value $30 * n$ minutes ago.

Eventually some records will have missing values due to the target's past values unavailability. To surpass this problem, the average of the attribute is calculated and inserted in the missing values.

We have assumed a simple times series perspective. Thus, attributes related to a target's past values are only used as input to models where the forecasting target is the same. Otherwise, these attributes are discarded. For example, the input information of the footfall models is used as input for the sales forecasting. However, since the footfall at the moment is available, there is no need on keeping the attributes related to the past values.

The input attributes for each model are represented in the appendix by table B.1, in case of footfall in or footfall out forecasting, or B.2 otherwise. In these tables the attributes are divided into three categories: time, target-oriented and others. According to the attributes created in this process or the already available after the data preparation (subsection 3.3.3).

4.1.3 Data Dependency

The input data for the models are dependent on the target's past values. This way the models can learn about different seasonality patterns, which is very important in this context. Although this can helps the model to forecast better, it also causes the forecast to be more troublesome.

When we want to make predictions for a time horizon greater than 1, i.e. predict $y_(t+h)$ with $h > 1$, the prediction will depend on the value of $y_(t+i)$, where $i < h$, which at time t, are unknown. Therefore, we need to predict the target values in the period within the horizon $y_(t+1), i = 1, ..., h-1$ and use them as attributes to calculate $y_(t+h)$. For this reason, to predict $t+h$, starting with $i = 0$, the following steps must be executed while $i \leq h$:

1. Gather the historical data until time $t+i$;

2. Prepare input attributes to predict $y_(t+i)$;

3. Predict $y_(t+i)$;

4. Increment i.

4.2 Optimization

As described in the previous chapter, in this staff scheduling optimization problem we aim at prescribing the best shifts and task allocation throughout the planning horizon that leads to the profit maximization. For this, a genetic algorithm is implemented in order to prescribe the best schedule. In one of the most cited books in computer science [36], David Goldberg explains the concept of genetic algorithms as the following:

> *"Genetic algorithms are search algorithms based on the mechanics f natural selection and natural genetics. They combine survival of the fittest among string structures with a structure yet randomized information exchange to form a search algorithm with some of the innovative flair of human search. In every generation, a new set of artificial creatures (strings) is created using bits and pieces of the fittest of the old; an occasional new part is tried for good measure. While randomized, genetic algorithms are no simple random walk. They efficiently exploit historical information to speculate on new search points with expected improved performance."*

Figure 4.3 represents the traditional logic behind this type of algorithm. Before any optimization is done, a set of initial solutions must be initialized. In this context, a solution is referred to as individual or chromosome and the batch of solutions known as the population. After the initial solutions are generated, the optimization process starts. At each iteration, each individual is evaluated by an objective function that expresses how good is the solution. Afterwards, a new population is created based on the old one. This process consists of selecting the individuals that will pass their information to the next generation and induce some variations in the genes by crossover and mutation. In this project, the selection methods used are elitism and the probability roulette wheel, two very well known methods [36].

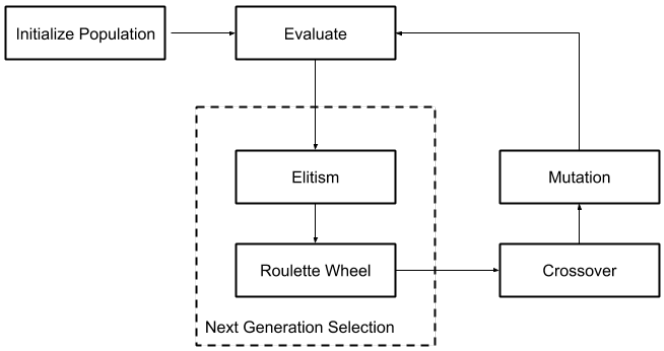


Figure 4.3: Genetic algorithm architecture.

In this project, an individual represents a feasible schedule for the planning horizon. For this, it makes use of three different structures for solution representation and aid in the constraint fulfillment. Upon the creation of an individual, the representation structures are first created with no shifts allocated. Afterwards, still in the initialization phase, each individual goes through a shift

creation process that guarantees an individual to represent a feasible solution. At each iteration, the crossover and mutation operators are applied, where new specific operators were created. All operators guarantee that the solution remains feasible.

4.2.1 Constraints

Scheduling optimization requires strict constraint fulfillment so that feasible solutions may be generated. Soft constraints are related to optimal characteristics if fulfilled. Table 4.5 shows the strict and soft constraints identified in our problem. The first row contains the specifications for shift definition and manipulation. The second row, constraints regarding the store operation. And the third, constraints related to shift assignment.

Table 4.5: Strict and soft constraints.

	Strict Constraints	**Soft Constraints**
Shift	- Maximum shift duration. - Maximum shift duration without breaks. - Minimum shift duration. - Zero or one break per shift.	- Break/shift duration ratio.
Store	- Number of employees. - One or more employees at sales between opening hours. - No allocation between closing hours.	- Employees contract weekly hours.
Assignment	- One employee assignment per shift. - Zero or one shift assignment per employee per day. - Tasks assigned to shifts must not overlap, be between the shifts time interval and as a whole, fill the shift completely.	

Shift constraints are important to define the possible range of values for the shifts characteristics. The whole shift duration is strictly restrained but there is some flexibility in the break insertion, although it is limited to one break per shift at maximum. The ratio between the break and shift duration is a soft constraint to facilitate the generation of feasible solutions. This constraint is only evaluated for shifts with duration above the maximum shift duration without breaks. Apart from the characteristics of the shift, the number of shifts is also highly constraint, since there can't exist more shifts than employees.

Store constraints are related to the store's working policy. This defines the allocation space for shifts, depending on the operating hours, and how many employees the store has. There is also another constraint that requires the presence of at least one employee at sales. This is a strict constraint to enforce minimum client coverage, which is crucial for sales and therefore, a step

ahead into the optimization process. The employee's contract weekly hours is not handled by optimization. This allows evaluating if the store is overstaffed or understaffed.

Due to resource scarcity, mainly time, the assignment of employees to shifts and of shifts to tasks is not handled by the genetic algorithm. If the assignment task was included in the algorithm, the space of feasible solutions would be bigger since each shift could be assign to any employee and each task to any shift. For this reason, finding a desirable solution would take longer since it would require more time to run the algorithm and tune the parameters. Even though the assignment is not handled, it's important to be aware of the constraints involved for optimization modeling.

4.2.2 Representation

In genetic algorithms, an individual represents a solution for the optimization problem. In this case, the number of allocated employees to the different types of shifts through the planning horizon. Choosing a good representation is very important to ensure and facilitate constraint fulfillment and also to simplify the development of operators that promote a more efficient and effective search.

For representation purposes, the concept of time slot is used so that information can be associated to a time interval. This way, a day can be represented by a set of time slots of equal duration, which depends on the time unit used. Eq. 4.5 shows how to calculate the number of time slots of a day for a specific time unit *tu* (in minutes). To calculate the time slot number *s* associated to the minutes of the day *m* eq. 4.6 is used. Since a time slot is associated to a time interval, different times of the day can be represented by the same time slot number. Eq. 4.7 shows how to calculate the time interval represeted by a particular time slot. For example, for a 30 minutes time unit, 10h:00m (600 minutes) corresponds to the 20^{th} slot. Following the previous example, the time interval for the 20^{th} slot is $[600, 629]$ minutes, equal to $[10h:00m, 11h:29m]$.

$$day_slots = \frac{24 * 60}{tu} \tag{4.5}$$

$$time_slot_m = \left\lfloor \frac{m}{tu} \right\rfloor \tag{4.6}$$

$$day_minutes_s \in [s * tu, (s+1) * tu - 1] \tag{4.7}$$

Each individual is characterized by a main structure, for representation, and two others for constraint fulfillment. The name of these structures are: **Allocation** (main), **Availability** (auxiliary) and **Shift Catalog** (auxiliary).

4.2.2.1 Allocation Structure

The allocation structure records the number of employees allocated to each task on each time slot of the planning horizon. It is considered the main structure because it contains all the information needed for evaluation.

This structure is a three dimensional matrix with shape $(n_tasks, n_planning_days, n_time_slots)$, as represented by figure 4.4. Each cell represents the number of allocated employees and it can range between zero and the number of maximum employees, except during the store's close hours where is always zero. Access to information can be done by task, planning day and/or time slot.

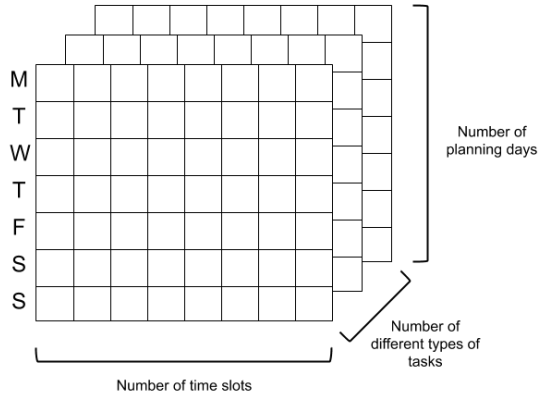


Figure 4.4: Allocation structure.

In the beginning, before initializing an individual, this structure is filled with zeros, since no allocations were yet made. Figure 4.5 shows the initialization of this structure.

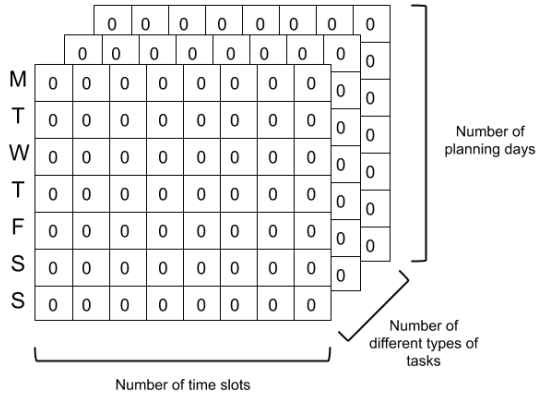


Figure 4.5: Allocation structure initialization.

4.2.2.2 Availability Structure

This structure records the number of available employees for allocation through the planning horizon. It helps to check if the number of shifts respects the maximum number of employees, avoiding constraint violation.

This structure is a two-dimensional matrix, as represented in figure 4.6, with the shape $(n_planning_days, n_time_slots)$. Therefore, the possible value range of each cell is between 0 and the number of employees. Access to information can be done by day and/or time slot.

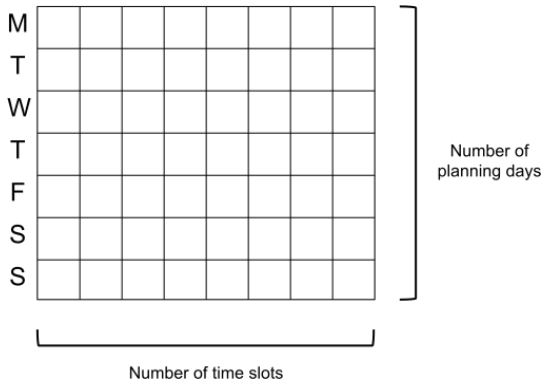


Figure 4.6: Availability structure.

To initialize this structure, the slots where the store is open are filled with the number of maximum employees for allocation, the rest are filled with zero. Figure 4.6 shows the initialization of this structure.

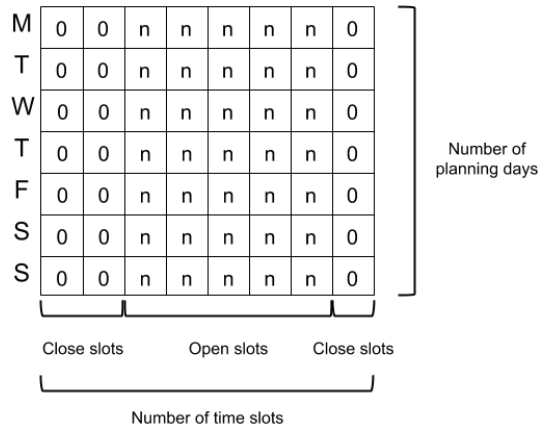


Figure 4.7: Availability structure initialization with n employees.

4.2.2.3 Shift Catalog Structure

This structure records the shifts created on each day of the planning horizon. Since the allocation structure only provides information about employee allocation to tasks, this structure aids in fulfilling shift characteristics constraints.

This structure is a two-dimensional matrix of dictionaries with shape $(n_planning_days, 2)$, as represented by figure 4.8. The keys of the dictionaries are time slot numbers and time value is a list of shifts. The dictionaries in the first columns catalogs shifts by their starting slot and the second column, shifts by their last slot. A shift is defined by their starting slot, last slot and, if any break exists, a start and last break slot.

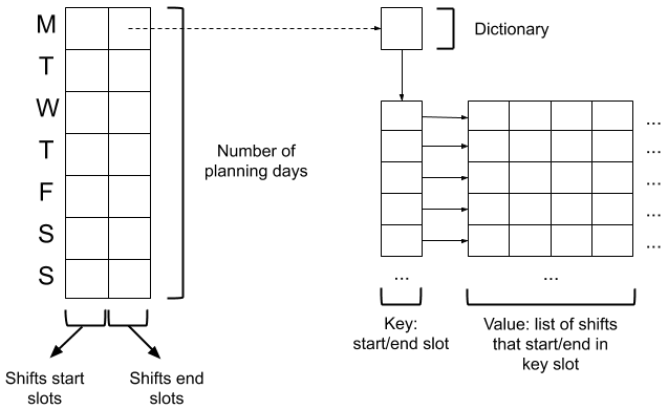


Figure 4.8: Shift catalog structure.

At first, the two-dimensional matrix is created with empty dictionaries as cell elements. Figure 4.9 shows the structure when initialized.

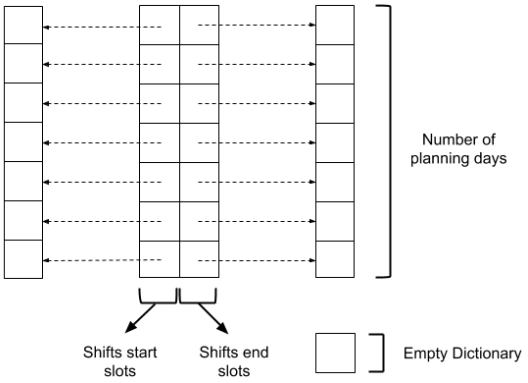
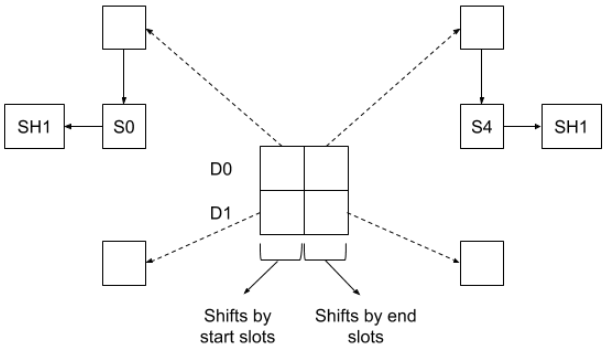


Figure 4.9: Shift catalog structure initialization.

4.2.2.4 Example

For better understanding, we will demonstrate how these structures are updated upon a shift creation. For example, for 2 planning days with 8 time slots, 3 tasks, 3 employees and opening hours between slot 0 and 6, a shift is created with: start slot 0, break slot 2 and end slot 4 for day 0. Consequently, the allocation, availability and shift catalog structures have the following shape: $(3,2,8)$, $(2,8)$ and $(2,2)$.

First, the shift must be added to the shift catalog for day zero, by its start and end slot, as demonstrated in figure 4.10. In the start slot dictionary, since there was no key with value 0, a new one was created with a list containing the shift as value, otherwise, the shift would have been appended to the existing list. The same situation occurred for the end slot dictionary with key 4.



Shift	Day	Start slot	Last slot	Break start slot	Break last slot
SH1	0	0	4	2	2

Figure 4.10: Example of the shift catalog structure with one shift.

Afterwards, for each time slot in the allocation structure, the allocation of one employee is made to a task. Figure 4.11 demonstrates a possible allocation for the created shift.

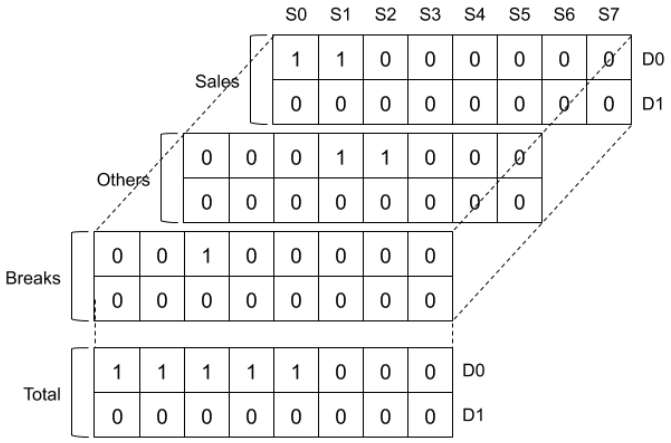


Figure 4.11: Example of the allocation structure with one shift.

Since employees were already allocated to tasks, the availability structure must be updated. This is done by removing one resource at each slot between slot 0 and 4. Figure 4.12 represents the availability structure after being updated.

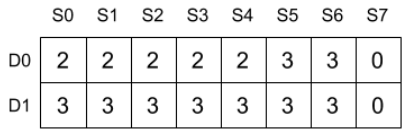


Figure 4.12: Example of the availability structure with one shift.

4.2.3 Initialization

The first step of a genetic algorithm requires having a population of individuals, in other words, a set of initial solutions. Guaranteeing that the solutions are feasible means that all constraints are fulfilled and the optimization will be carried out inside the space of possible solutions. In these

conditions, the optimization process will be much faster and efficient. For this reason, initializing and generating individuals must be handled very carefully.

Figure 4.13 represents the implemented strategy to generate a valid individual. It consists in generating a set of valid shifts per day and guaranteeing that the defined constraints are not violated. Some randomness is introduced in shift generation which leads to different solutions. Randomness may also lead to the need of creating more shifts than the number of employees. In that case, the shifts for that day are discarded and the process of initialization for that day is restarted.

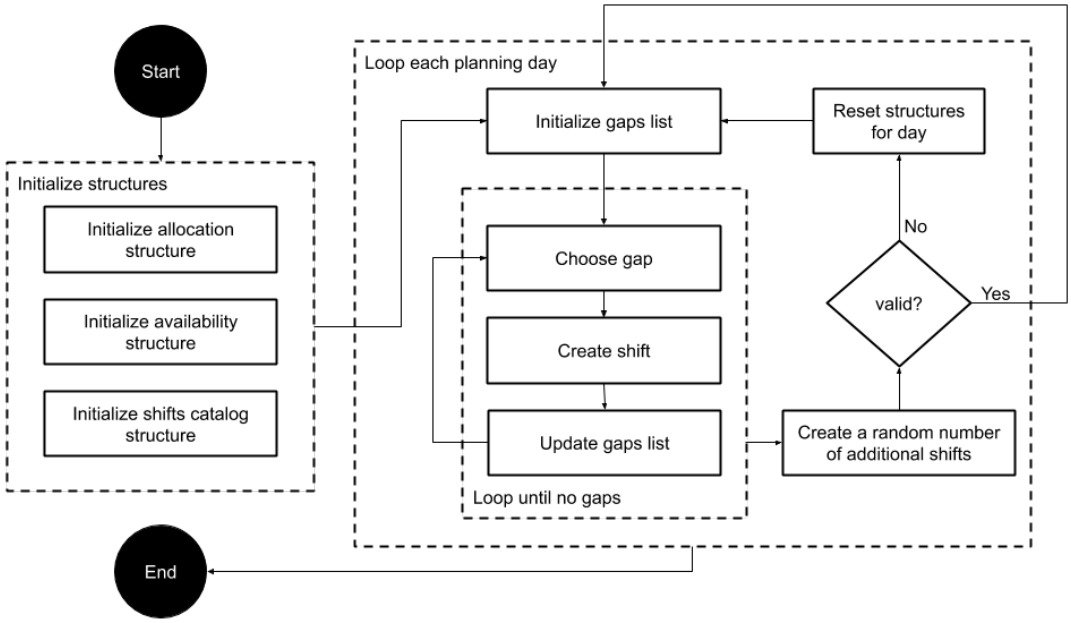


Figure 4.13: Individual generation process.

First, the structures that represent the individual are initialized, according to section 4.2.2. Afterwards, valid shifts are created to ensure client coverage on the store during opening hours. An allocation gap is a period with no employees assigned to sales, which violated a strict constraint (see section 4.2.1). To control coverage on the store, a list is used to record the gaps that must be integrated into working shifts. A gap is represented by a list of two elements: the starting slot and the last slot of the gap range. For each planning day, before shifts are created, the gaps list is initialized with $[store_{open_slot}, store_{close_slot} - 1]$, which is associated with the store's operating hours. A gap can be classified as: **single shift** where one shift is enough to cover the gap, or, **multiple shift** where more than one is needed.

Different types of gaps can lead to shifts with different possible values for the shift's attributes. In table 4.6 we can see how a shift's attributes valid range can be calculated for a single shift gap. Table 4.7 shows how these can be done for a multiple shifts gap.

Given the gaps for a specific day, one is randomly picked and the process of shift creation starts. This process is represented by figure 4.14 and begins by identifying the type of gap and calculating the possible range of values for the shift's attributes. Afterwards, the shift duration and

Table 4.6: Shift characteristics for a single shift gap.

Single Shift Gap Condition
$gap_{duration} \leq constraints_{shift_max_duration}$
Shift Duration
$shift_{duration} \in [min_duration, max_duration]$ $min_duration = \max(gap_{duration}, constraints_{shift_min_duration}$ $max_duration = \min(\max(gap_{duration}, constraints_{shift_max_duration}), store_{open_duration})$
Shift Start Slot
$shift_{start_slot} \in [min_start, max_start]$ $min_start = \begin{cases} store_{close_slot} - shift_{duration}, & \text{if } gap_{start_slot} + shift_{duration} - 1 \geq store_{close_slot} \\ gap_{start_slot}, & \text{otherwise} \end{cases}$ $max_start = \begin{cases} store_{open_slot}, & \text{if } gap_{last_slot} + 1 - shift_{duration} < store_{open_slot} \\ gap_{last_slot} + 1 - shift_{duration}, & \text{otherwise} \end{cases}$
Shift Last Slot
$shift_{last_slot} = shift_{start_slot} + shift_{duration} - 1$

Table 4.7: Shift characteristics for a multiple shift gap.

Multiple Shift Gap Condition
$gap_{duration} > constraints_{shift_max_duration}$
Shift Duration
$shift_{duration} \in [min_duration, max_duration]$ $min_duration = constraints_{shift_min_duration}$ $max_duration = \min\left(constraints_{shift_max_duration}, store_{open_duration}\right)$
Shift Start Slot
$shift_{start_slot} \in [min_start, max_start]$ $min_start = gap_start_slot$ $max_start = gap_{last_slot} + 1 - shift_{duration}$
Shift Last Slot
$shift_{last_slot} = shift_{start_slot} + shift_{duration} - 1$

Table 4.8: Break characteristics.

Ideal Break Duration
$break_{duration} = \begin{cases} \lfloor constraints_{break_ratio} * shift_{duration} \rfloor, & \text{if } shift_{duration} > constraints_{shift_max_duration_no_break} \\ 0, & \text{otherwise} \end{cases}$
Break Start Slot
$break_{start_slot} \in [slot_a, slot_b]$ $slot_a = \lfloor shift_{duration} * 1/3 \rceil + shift_{start_slot}$ $slot_b = shift_{duration} - \lfloor shift_{duration} * 1/3 \rceil - break_{duration} + shift_{start_slot}$
Break Last Slot
$break_{last_slot} = break_{start_slot} + break_{duration} - 1$

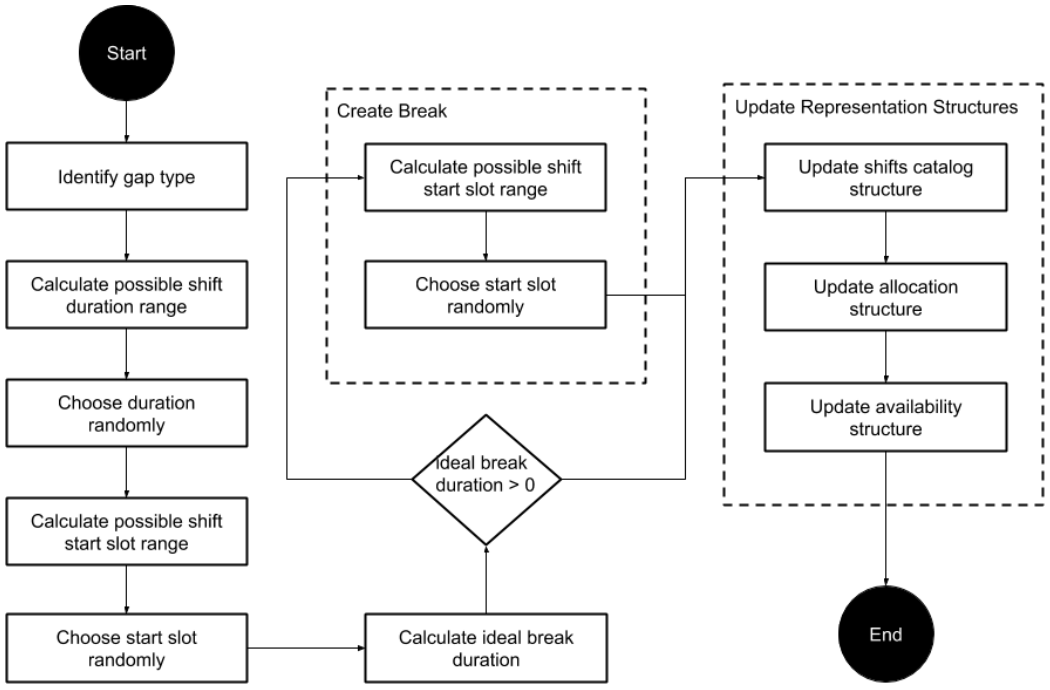


Figure 4.14: Shift creation process for gap coverage.

Table 4.9: Constraints for an individual initialization example.

Optimization Constraints	**Value**
Number of planning days	1 day
Day slots	16 slots

Shift Constraints	**Value**
Tasks	[SALES, OTHERS, BREAK]
Max. shift duration	8 slots
Max. shift duration without break	4
Min. shift duration	2 slots
Break/shift duration	1/4 slots

Store Constraints	**Value**
Number of employees	5
Cost of 1 worker per slot	4
Opening hours	Day 0: [3,15]

start slot are randomly selected, by this order, and if needed, a break is randomly inserted. Table 4.8 shows how to determine if a break is required and what are the possible range of values for its start and last slot.

After all the shift characteristics are specified, the shift is created and the representation structures are properly updated. The allocation between the shift's working hours to tasks is made randomly but always prioritizing sales when no employee is assigned. In the end, the list of gaps is updated and the process restarts until no gap is found. Once coverage is guaranteed, extra shifts can be created. The number of extra shifts is randomly picked and it ranges between $[0, constraints_{max_employees} - n_shifts_created]$.

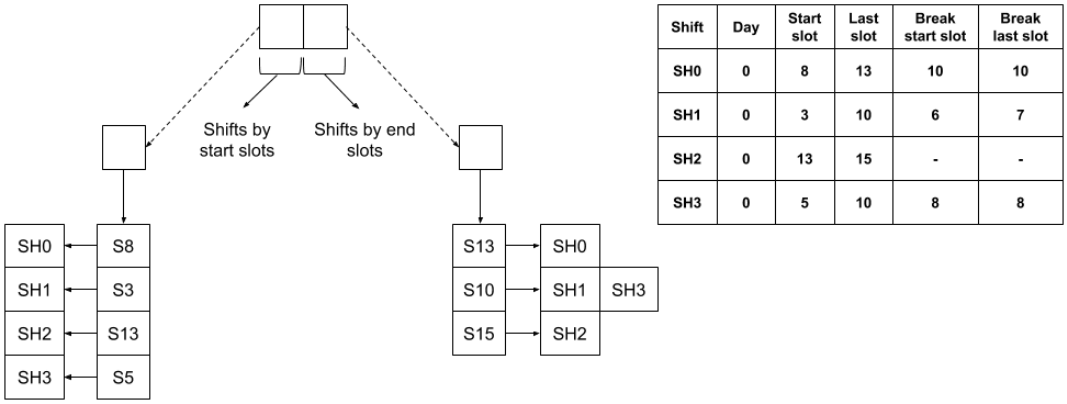
Figures 4.15, 4.16 and 4.17 shows the content of the representation structures of a generated individual, under the conditions in table 4.9. The initialization process took four iterations to produce this valid solution. Step by step details about the generation of this solution can be consulted in the appendix section B.2.1.

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
Sales	0	0	0	1	1	2	1	1	1	2	1	1	1	1	1	1
Others	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0
Breaks	0	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0
Total	0	0	0	1	1	2	2	2	3	3	3	1	1	2	1	1

Figure 4.15: Allocation structure generated by the initialization process.

S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
0	0	0	4	4	3	3	3	2	2	2	4	4	3	4	4

Figure 4.16: Availability structure generated by the initialization process.



Shift	Day	Start slot	Last slot	Break start slot	Break last slot
SH0	0	8	13	10	10
SH1	0	3	10	6	7
SH2	0	13	15	-	-
SH3	0	5	10	8	8

Figure 4.17: Shift catalog structure generated by the initialization process.

4.2.4 Evaluation

At each iteration, of the optimization algorithm, each individual is evaluated so that one may know how good is each solution. In this project, the goal is to maximize the sales for the planning horizon and also take into account the cost of working employees. A possible objective function is to maximize the sum of the difference between the sales and costs (eq. 4.8). Taking this into account, the score of an individual can be calculated through eq. 4.9.

The sales of a day can be calculated by the sales forecast sum at each time slot (eq. 4.10). For faster evaluations, we assume that there are no sales during closing hours and so, the forecast is only made for the store open slots. The sales can be forecast by using architecture 4.2 if direct sales are allowed, or 4.1 otherwise. The sales are then provided through eq. 4.2, which can be reformulated by eq. 4.11. For this, the necessary information regarding the schedules is extracted from the representation structures and the input data for each model is prepared (section 4.1.2).

The cost of a working employee is used in the objective function to introduce a penalization for the usage of employees. Otherwise, over-staffing could not be avoided, since assigning all resources would maximize customer coverage and could help increase sales as well. The cost of one employee, at each time slot, is constant over time and equal for all employees. The total costs of a day are then calculated through eq. 4.12.

For a better interpretation of equations 4.8, 4.9, 4.10, 4.11 and 4.12 please consult table 4.10 with the description of the variables.

$$\max \left(\sum_{d=0}^{n_planning_days-1} (day_sales_{i,d} - cost_{i,d}) \right) \tag{4.8}$$

$$score_i = \sum_{d=0}^{n_planning_days-1} (day_sales_{i,d} - cost_{i,d}) \tag{4.9}$$

$$day_sales_{i,d} = \sum_{s=0}^{day_time_slots-1} slot_sales_{i,d,s} \tag{4.10}$$

$$slot_sales_{i,d,s} = \begin{cases} total_sales_{i,d,s}, & if\ direct_sales \\ n_trans_{i,d,s} * avg_prods_t_{i,d,s} * avg_price_p_{i,d,s}, & otherwise \end{cases} \tag{4.11}$$

$$day_cost_{i,d,s} = \sum_{s=0}^{day_time_slots-1} \sum_{t=0}^{n_tasks-1} slot_cost * allocated_employees_{i,d,s,t} \tag{4.12}$$

Table 4.10: Variable description for equations 4.8, 4.9, 4.10, 4.11 and 4.12.

Variable	Description
d	planning day $\in [0, n_planning_days - 1]$
i	individual $\in [0, population_{size} - 1]$
s	time slot $\in [0, day_time_slots - 1]$
t	task $\in [0, n_tasks - 1]$
day_time_slots	total slots of a day
n_planning_days	number of days to plan
n_tasks	number of tasks types
allocated_employees$_{i,d,s,t}$	number of allocated employees in individual *i*, for day *d*, slot *s* and task *t*
avg_prods_t$_{i,d,s}$	predicted average products per transaction for individual i, for day *d* in slot *s*
avg_price_p$_{i,d,s}$	predicted average price per product for individual i, for day *d* in slot *s*
cost$_{i,day}$	cost of allocated employees in individual *i* on day *d*
day_costi,d,s	cost of allocated employees in individual *i*, for day *d* and slot *s*
day_sales_i,d	sales of day *d* for individual *i*
n_trans$_{i,d,s}$	predicted transactions for individual i, for day *d* in slot *s*
score$_i$	score of individual *i*
slot_cost	slot cost of one working employee
slot_sales$_{i,d,s}$	sales of individual *i*, for day *d* on time slot *s*

4.2.5 Selection

To build the next population generation, individuals are selected to pass their information to their descendants. Thus, two traditional strategies are used: elitism and probability roulette wheel.

Elitism assures that the best n individuals are automatically passed to the next generation. With this, the best solution in the population is always equal or better than the one in the previous generation.

The roulette wheel is a strategy used to randomly select and pair individuals for crossover, also known as parents. The probability of an individual i being chosen can be calculated through eq. 4.13, where $score_x$ is the score of an individual x (eq. 4.9) and j an individual of the population with size $population_size$. The higher an individual's score, the higher the probability of being picked. In the end, the number of pairs are equal to $\left\lceil \frac{population_size - elitist_size}{2} \right\rceil$. After crossover, the population size must be assured to be the same as in the previous generation. Thus, some children can be discarded for this purpose.

$$probability_chosen_i = \frac{score_i}{\sum_{j=0}^{population_size-1} score_j} \tag{4.13}$$

4.2.6 Crossover

This operator allows building two new individuals, also known as children, by swapping their parent's information. After the crossover, the children replace their parents in the population.

To perform crossover, random planning days are selected and crossover is done for each one. For this, a slicing strategy is used, where a time slot is randomly chosen, as the slicing slot, to swap the information that comes after. This slicing method is illustrated in figure 4.18, where at the left we can picture the information being sliced and swapped and the corresponding shifts at right.

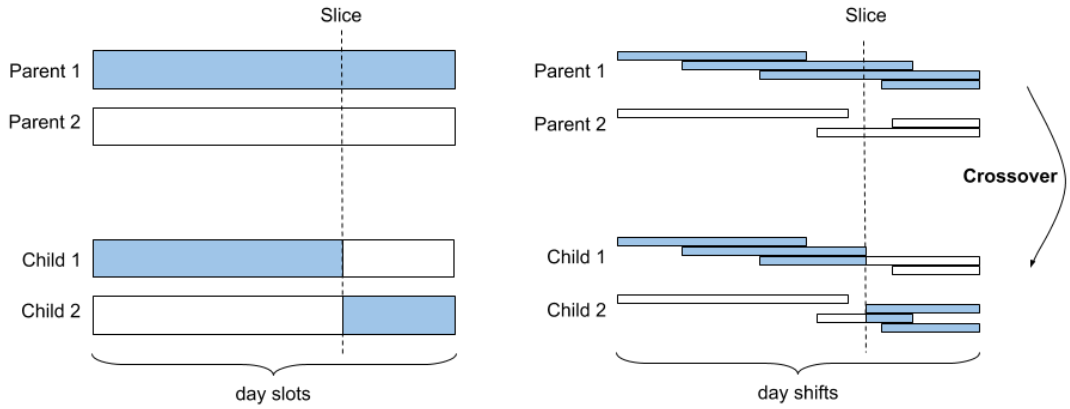


Figure 4.18: Crossover slicing method and shift swap. Overall information (left image) and shifts boundaries (right image) representation.

This slicing method doesn't guarantee that the information to join, after the swap, is compatible. Hence, a trial and error strategy is applied and the original information of the parents is

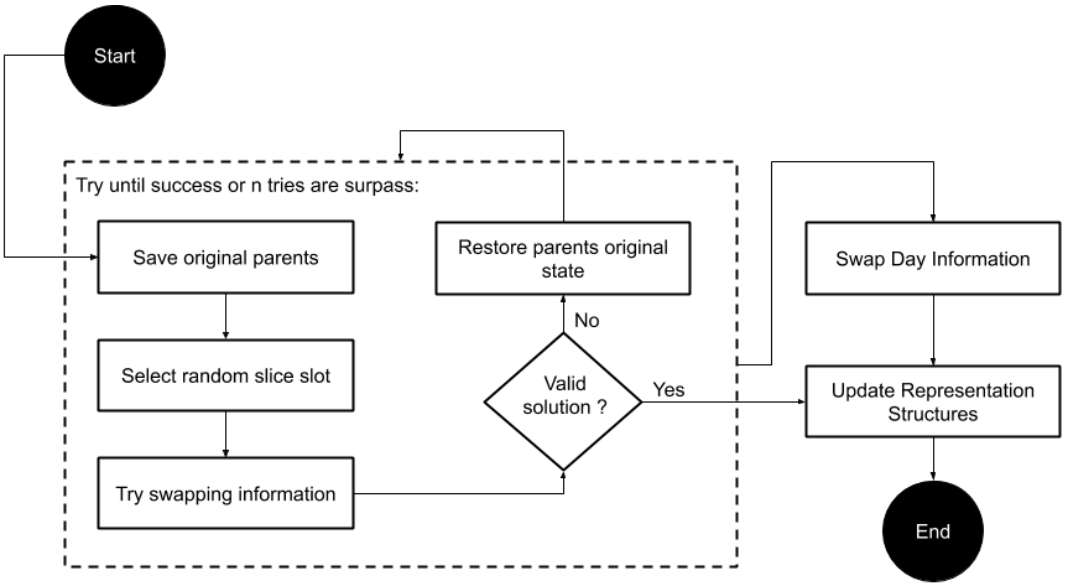


Figure 4.19: Crossover process.

restored to its original state at each trial. For every attempt in the slicing strategy, a new slicing slot is picked so that other shifts combinations can be tested. If the swap does not succeed in any of the attempts, the whole day is swapped. This option is always valid but it does not allow the combination of different shifts for the same day, instead, it takes information of only one parent. For this reason, day swap is used as a last resource. Figure 4.19 represents the crossover process described above.

Every time a slicing slot is chosen, an algorithm is executed to try and swap the information on both parents. For this, the algorithm must identify and try to join some of the shifts so that the solution remains valid. Selecting which ones can be joined depends on the original shifts characteristics and the slicing slot. Therefore, the algorithm classifies the shifts as:

- **Fixed Shift** - Shift that does not overlap with the slicing slot nor it starts in the slot after (condition 4.14).

$$\begin{aligned}&(shift_{start_slot}, shift_{last_slot} \in [store_{open_slot}, slicing_slot - 1]) \\ \vee &(shift_{start_slot}, shift_{last_slot} \in [slicing_slot + 2, store_{close_slot} - 1])\end{aligned} \tag{4.14}$$

- **Overlay Shift** - Shift that overlaps with the slicing slot or starts in the slot after (condition 4.15).

$$\begin{aligned}&(shift_{start_slot} \in [store_{open_slot}, slicing_slot + 1]) \\ \wedge &(shift_{last_slot} \in [slicing_slot, store_{close_slot} - 1])\end{aligned} \tag{4.15}$$

The main difference between the fixed and overlay shifts is that the overlay shifts participates in the join process and the fixed shifts do not. Since the fixed shifts do not overlap with the slicing slot, the shift information is kept intact and is inherited by only one child. In the overlay shifts, the slice may cause the shift information to be divided and, if it is the case, two new shifts are created ranging between $[shift_{start_slot}, slicing_slot]$ and $[slicing_slot + 1, shift_{last_slot}]$. This occurs if the starting slot of the shift is before $slicing_slot + 1$ and the end slot after $slicing_slot$. Now, an individual's shifts can be represented by: left fixed shifts, left overlay shifts, right fixed shifts and right overlay shifts, where the left shifts are those at the left of the slicing slot and the right shifts the ones at the right (figure 4.20). The overlay shifts are candidates to the join process because, after the slice, the right overlay shifts from a parent are at 1 slot distance from the left overlay shifts of the other, and vice versa. This means that if they are joined, the employees allocation (allocation and availability structures) will remain the same.

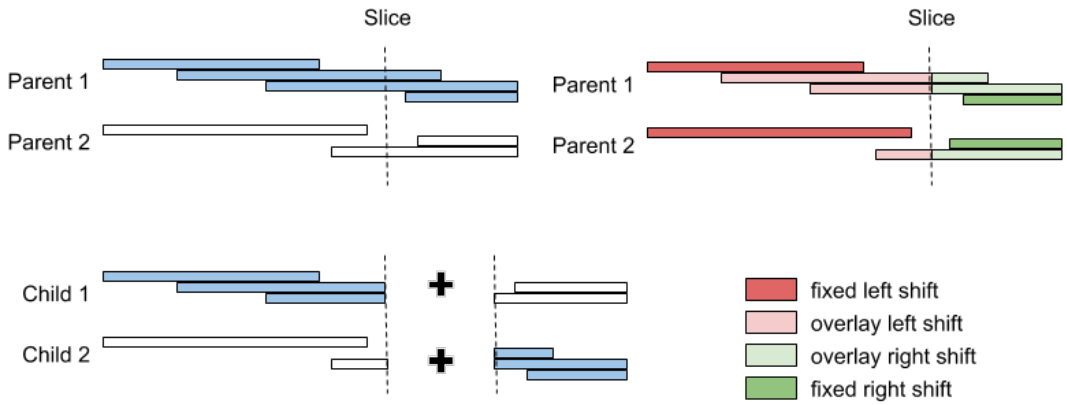


Figure 4.20: Crossover shift classification.

The information is then swapped and each child is the result of joining the left information of a parent, with the right information of the other (equations 4.16). Here, the left and right overlay shifts are gathered and a join process starts for each child. As a precaution, the breaks of all overlay shifts are removed and only inserted after a successful join. For this, the employees that were assigned to the breaks, on those time slots, are randomly distributed by the remaining tasks.

$$\begin{aligned} child_1 &\leftarrow parent_{1,left_fixed} + parent_{left_overlay} + parent_{2,right_overlay} + parent_{2,right_fixed} \\ child_2 &\leftarrow parent_{2,left_fixed} + parent_{2,left_overlay} + parent_{1,right_overlay} + parent_{1,right_fixed} \end{aligned} \tag{4.16}$$

Before starting the join process, three types of information are recorded for each overlay shift:

- **Possible Extend** - The number of slots that can still be added to the shift without violating any constraints (eq. 4.17).

$$possible_extend = constraints_{shift_max_duration} - shift_{duration} \tag{4.17}$$

- **Mandatory Extend** - The number of slots that must definitely be added to the shift in order to fulfill the shift's constraints (eq. 4.18).

$$mandatory_extend = \begin{cases} constraints_{shift_min_duration} - shift_{duration}, \\ if\ shift_{duration} < constraints_{shift_min_duration} \\ 0, \qquad\qquad\qquad\qquad otherwise \end{cases} \tag{4.18}$$

- **Compatible Shifts** - The overlay shifts, from the opposite direction, that are suitable for the shift's possible and mandatory extend (condition 4.19).

$$shift_{i,duration} \in [mandatory_extend_j, possible_extend_j] \tag{4.19}$$

Overlay shifts have different priorities in the join process. Table 4.11 shows the priority rules ordered by their preference. Rule (a) guarantees that shifts with mandatory slots have priority when the number of compatible shifts is reduced to one. Rule (b) prioritizes shifts with a higher number of mandatory slots. Rule (c) favors shifts with fewer compatible shifts (i.e., shifts with fewer opportunities to join). Lastly, rule (d), prioritizes shifts with fewer possible slots to extend.

Table 4.11: Priority rules for overlay shifts.

Rule	**priority(a) > priority(b)**
(a)	$len(a.compatible_shifts) == 1 \wedge a.mandatory_extend > 0$
(b)	$a.mandatory_extend > b.mandatory_extend$
(c)	$len(a.compatible_shifts) < len(b.compatible_shifts) \wedge len(a.compatible_shifts) \neq 0$
(d)	$a.possible_extend < b.possible_extend$

In the join process, two lists are used to record the shifts that will be processed and those that are the result of a join. In the beginning, the first list is initialized with the overlay shifts and the second is empty. At each iteration, the first is sorted, according to rules 4.11, and the shift with the highest priority is chosen. The same is done for its compatible shifts. Then, these shifts are used to create a new one that starts at the same slot of the earliest shift and lasts until the last slot of the later. The shifts that were not yet processed must also update their compatible shifts, since these may only include those that were not processed. This means that shifts that were joined won't be

processed again by the algorithm. This process is carried out until all shifts are processed or the one with the highest priority does not have compatible shifts. In the end, the fixed shifts and the ones from the join process, whether they are joined or not, represent the child. The strategy behind the join process is presented in algorithm 1.

Algorithm 1 Join overlay shifts

```
1:  procedure JOINSHIFTS(overlay_shifts)
2:      to_process ← overlay_shifts
3:      joined_shifts ← empty_list
4:      while True do
5:          to_process ← sort(to_process)
6:          shift1 ← shifts[0]                                  ▷ Highest Priority
7:          compatible_shifts ← compatible shifts of shift1
8:          if length(compatible_shifts) > 0 then               ▷ There are compatible shifts
9:              compatible_shifts ← sort(compatible_shifts)
10:             shift2 ← compatible_shifts[0]                   ▷ Highest Priority
11:             new_shift ← create a shift with the joined information of shift1 and shift2
12:             joined_shifts ← append new_shift to joined_shifts
13:             to_process ← remove shift1 and shift2 from to_process
14:         else                                                ▷ No more possible shifts to join
15:             break
          return to_process, joined_shifts
```

To validate the swap, the shifts from both children must fulfill the established constraints. Hereby, the shift's minimum and maximum duration and the number of store employees. There are two reasons that can lead to the failure in the join: there are overlay shifts still to process with mandatory extend and no compatible shifts, or, in the end, there are more shifts than employees. This allows us to check if a child is successful or not, after the join process, and avoid starting this process on the other child if it already fails in the first.

If both children are valid, breaks are inserted in the shifts that need them and the representation structures are updated. The slot range for the break starting slot of a shift, calculated through table 4.8, might include some slots where the allocation to sales is only assured by the shift in question. Being the only shift that assures the allocation of an employee to sales, introducing a break in such slots would violate a constraint. Therefore, breaks can't be inserted in these slots. To surpass this, the slots between that range are analyzed and, from the valid ranges, the most suitable slot is chosen, according to the ideal break duration. It may also happen that a shift does not have any feasible slots to insert a break and no break is actually inserted. So, in these cases, no break is inserted. As the optimization proceeds, a break will eventually be inserted in these shifts, since crossover and mutation checks if a break must be inserted or, in the latter case, adjusted (see section 4.2.7.9).

After inserting the missing breaks, the crossover for that day is finished and this process is done for the next selected day. After crossover is applied for every day chosen, the children replace their parents in the new population and the process ends.

For a better understanding of how this operator is applied consult the appendix section B.2.2. This section shows an example of applying crossover to two different individuals where step-by-step details are described and explained.

4.2.7 Mutation

Each new generation can have characteristics that were not inherited by their ancestors. This is caused by mutation and is applied randomly, depending on a predefined probability. In a population with a high level of likelihood, this operator generates solutions with a new combination of features and therefore, avoids population stagnation. Mutation may also lead to better solutions since modifying a set of small aspects can make all the difference.

Algorithm 2 describes the process of mutating an individual. It starts by iterating each time slot, for each planning day, and randomly apply a mutation. At each slot, a test is conducted in order to evaluate if a mutation will occur or not. This test consists in generating a random number between $[0, 100]$ and check if it's below the mutation probability percentage. If the test is positive, then the slot is chosen for mutation and the process begins.

There are different types of mutation, however, not all are guaranteed to be possible. For this reason, when a time slot is chosen, the prerequisites for each type of mutation are analyzed and a random mutation is selected, from the feasible options. If no mutation is possible, the mutation process is carried out to the next time slot and so on, until is executed. More details about the types of mutations are described in later subsections but a small introduction is given below:

- **Reduce Start** - Reduces the duration of a shift by changing the start slot.
- **Reduce Last** - Reduces the duration of a shift by changing the last slot.
- **Extend Start** - Increases the duration of a shift by changing the start slot.
- **Extend Last** - Increases the duration of a shift by changing the last slot.
- **Break** - Breaks a shift in two smaller shifts.
- **Join** - Joins two shifts in one.
- **Remove** - Removes a shift.
- **Reallocate** - Reallocates employees between tasks.

When evaluating if a type of mutation is suitable, the algorithm accesses the characteristics of the individual and records pieces of information where the mutation can be applied as candidates. For instance, it can record part of the allocation structure as to identify where to apply a mutation. After the evaluation, the algorithm checks which mutations have candidates, and from these, one is randomly picked along with a candidate. All feasible mutations have probability $\frac{1}{length(feasible_mutations)}$ except the *remove* mutation. If the number of shifts per day is high, this

type of mutation will have a higher probability of being chosen, which would lead to a great decrease in shifts. To avoid this, a limit is imposed on the maximum probability of this kind of mutation. Hence, the probability of choosing the *remove* mutation, if it's feasible, is equal to $\min(remove_max_probability, \frac{1}{length(feasible_mutations)})$.

After some types of mutations are applied, some breaks of the individual's shift might need to be updated. Therefore, after a shift is mutated, its break is adjusted. Details about how this is done can be found in section 4.2.7.9.

Algorithm 2 Individual mutation algorithm.

```
1:  mutation_probability ∈ [0,1]                                   ▷ Float values
2:  remove_probability ∈ [0,1]                                     ▷ Float values
3:  procedure MUTATE(individual,mutation_probability,remove_probability)
4:      mutate_next_slot ← False
5:      for all day in individual.planning_days do                 ▷ Iterate days
6:          open_slots ← Open store slots of day
7:          for all slot in open_slots do                          ▷ Iterate open slots
8:              random ← random(0,100)                             ▷ For precision 2
9:              if (random ≤ mutation_probability*100) ∨(mutate_next_slot == True) then
10:                 feasible_mutations ← feasible types of mutation
11:                 random_mutation ← None
12:                 if remove_mutation ∈ feasible_mutations ∧ 1/length(feasible_mutations) >
    remove_probability then
13:                     random ← random(0,100)
14:                     if random ≤ remove_probability*100 then
15:                         feasible_mutations ← list with only the remove mutation
16:                     else
17:                         feasible_mutations ← feasible_mutations - remove mutation
18:                 if feasible_mutation ≠ empty then              ▷ Possible Mutation
19:                     mutate_next_slot ← False
20:                     mutation ← random(feasible_mutations)
21:                     candidates ← get_candidates(individual,random_mutation)
22:                     candidate ← random(candidates)
23:                     individual ← apply_mutation(individual,mutation,candidate,slot)
24:                     individual ← adjust_break(individual)
25:                 else                                           ▷ Mutation Fail
26:                     mutate_next_slot ← True
        return individual
```

Mutation is applied to all individuals in the population. The result of a mutation on a solution which was selected with the elitism mechanism is only accepted if it leads to a better solution (algorithm 3). For this, the elite members are mutated and then evaluated, according to section 4.2.4. If a solution is worse than the original, it is discarded and the individual goes back to its original state. Otherwise, the individual accepts its mutated form and a better solution is provided. This strategy allows a faster optimization since it ensures that the best solution is as good or better

than the last, and by testing if variations of the best solution, are in fact, even better.

For a better understanding of how each type of mutation is applied and how breaks are adjusted, some examples are presented in the appendix section B.2.3.

Algorithm 3 Population mutation algorithm.

```
1:  mutation_probability ∈ [0,1]                                   ▷ Float values
2:  remove_probability ∈ [0,1]                                     ▷ Float values
3:  procedure MUTATEPOPULATION(population,mutation_probability,remove_probability)
4:      for i in range(0,length(population)) do
5:          individual ← population[i]
6:          if is_elite(individual) then                           ▷ Mutate elite
7:              original_score ← score of individual
8:              mutated_individual ← mutate(individual,mutation_probability,remove_probability)
9:              mutated_score ← score of mutated_individual
10:             if mutated_score > original_score then             ▷ Elite mutated is better
11:                 population[i] ← mutated_individual
12:         else                                                   ▷ Mutate regular
13:             population[i] ← mutate(individual,mutation_probability,remove_probability)
        return population
```

4.2.7.1 Mutation: Reduce Start

This type of mutation consists in reducing the duration of a shift by increment the start slot in one unit (figure 4.21). This is only feasible when the shift starts at the selected slot, its duration is above the minimum allowed and more than one employee is allocated to sales (condition 4.20).

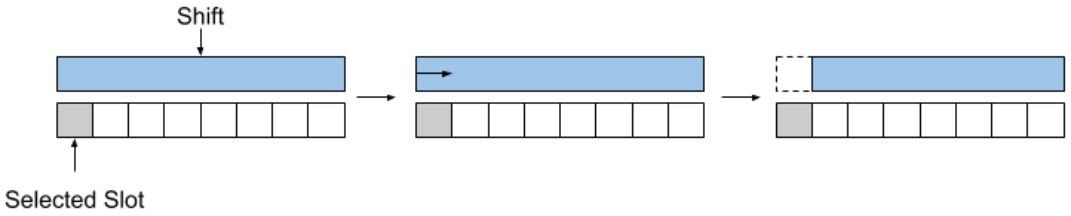


Figure 4.21: Mutation *reduce start* operation.

$$\begin{aligned} &(shift_{start_slot} == mutation_slot) \\ \wedge &(shift_{duration} > constraints_{shift_min_duration}) \\ \wedge &(allocation_{sales,mutation_slot} \neq 1) \end{aligned} \tag{4.20}$$

To apply this mutation, the three structures must be updated, for that specific day:

- **Shift Catalog**

1. Update the shift's start slot to $shift_{start_slot} + 1$;
2. Remove the shift from the starting slot dictionary and add it again with the new key;

- **Availability**

1. Increase by one the number of available employees in the selected slot for mutation.

- **Allocation**

1. Check which non-break tasks can remove an employee.
2. Randomly choose one valid task, in the selected slot, and remove one allocated employee.

4.2.7.2 Mutation: Reduce Last

This type of mutation consists in reducing the duration of a shift by reducing the last slot in one unit (figure 4.22). For this to be possible, the shift's last slot must be the same as the selected slot, the duration has to be above the minimum allowed and more than one employee is allocated to sales (condition 4.21).

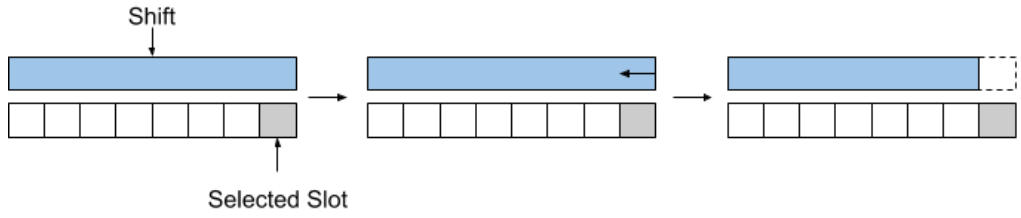


Figure 4.22: Mutation *reduce last* operation.

$$\begin{aligned} &(shift_{last_slot} == mutation_slot) \\ \wedge &(shift_{duration} > constraints_{shift_min_duration}) \\ \wedge &(allocation_{sales,mutation_slot} > 1) \end{aligned} \tag{4.21}$$

To apply this mutation, the three structures must be updated, for that specific day, as follows:

- **Shift Catalog**

1. Update the shift's last slot to $shift_{last_slot} - 1$;
2. Remove the shift from the dictionary that catalogs the last slots and add it again with the new key;

- **Availability**

1. Increase by one the number of available employees in the selected slot for mutation.

- **Allocation**

  1. Check which non-break tasks can remove an employee.
  2. Randomly choose one valid task, in the selected slot, and remove one allocated employee.

4.2.7.3 Mutation: Extend Start

This type of mutation consists in extending the duration of a shift by decrease the start slot in one unit (figure 4.23). For this to be possible, this shift must start on the slot after the selected slot for mutation and have duration below the maximum allowed (condition 4.22).

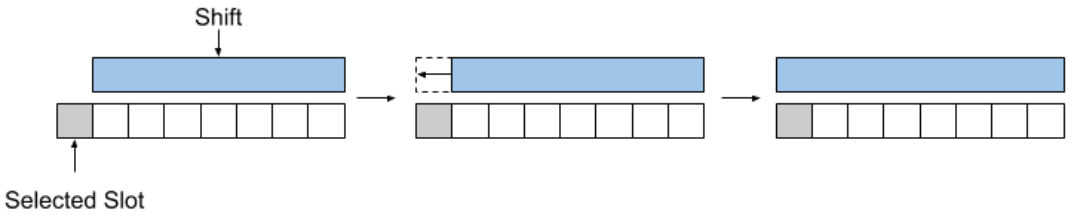


Figure 4.23: Mutation *extend start* operation.

$$\begin{aligned}&(shift_{start_slot} == mutation_slot + 1)\\ &\wedge (shift_{duration} < constraints_{shift_max_duration})\end{aligned} \tag{4.22}$$

To apply this mutation, the three structures must be updated, for that specific day:

- **Shift Catalog**

  1. Update the shift's start slot to $shift_{start_slot} - 1$;
  2. Remove the shift from the dictionary that catalogs the starting slots and add it again with the new key;

- **Availability**

  1. Decrease by one the number of available employees in the selected slot for mutation.

- **Allocation**

  1. Allocate one employee to a non-break task in the selected slot for mutation.

4.2.7.4 Mutation: Extend Last

This type of mutation consists in extending the duration of a shift by increment the last slot in one unit (figure 4.24). This is only feasible if the shift's last slot is immediately before the selected slot for mutation and shift's duration is lower than the maximum allowed (condition 4.23).

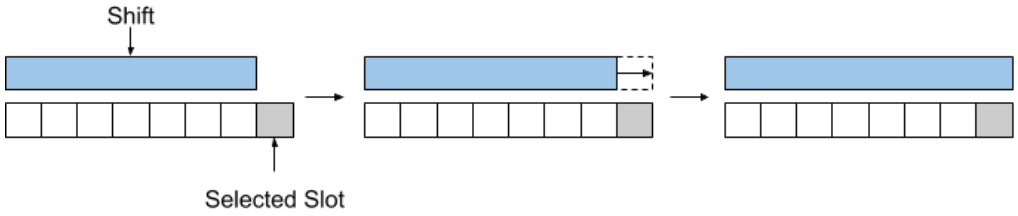


Figure 4.24: Mutation *extend last* operation.

$$\begin{aligned}&(shift_{last_slot} == mutation_slot - 1)\\ \wedge&(shift_{duration} < constraints_{shift_max_duration})\end{aligned} \tag{4.23}$$

To apply this mutation, the three structures must be updated, for that specific day:

- **Shift Catalog**
  1. Update the shift's last slot to $shift_{last_slot} + 1$;
  2. Remove the shift from the dictionary that catalogs the last slots and add it again with the new key;
- **Availability**
  1. Decrease by one the number of available employees in the selected slot for mutation.
- **Allocation**
  1. Allocate one employee to a non-break task in the selected slot for mutation.

4.2.7.5 Mutation: Break

This type of mutation breaks a shift into two smaller ones (figure 4.25). The break can be before or after the selected slot, however, the shifts after the break must respect the minimum shift duration constraint. Condition 4.24 states the prerequisite of the shift for a break before the selected slot and condition 4.25 for the break after it.

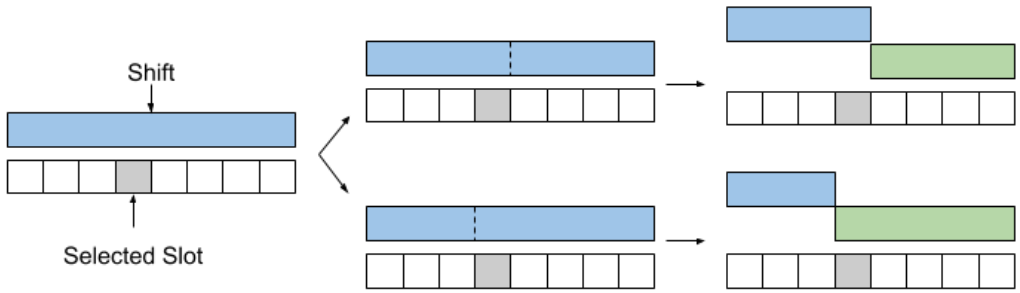


Figure 4.25: Mutation *break* operation. Two possible outcomes.

$$(shift_{start_slot} <= mutation_slot - constraints_{shift_min_duration}) \\ \wedge (shift_{last_slot} > mutation_slot + constraints_{shift_min_duration}) \tag{4.24}$$

$$(shift_{start_slot} <= mutation_slot + 1 - constraints_{shift_min_duration}) \\ \wedge (shift_{last_slot} > mutation_slot + 1 + constraints_{shift_min_duration}) \tag{4.25}$$

To apply this mutation, only one structure must be updated, for that specific day:

- **Shift Catalog**
  1. Remove the break start and last slot information for the shift.
  2. Create two new shifts with start and last slot:
     (a) $[shift_{start_slot, break_slot-1}]$ and $[break_slot, shift_{last_slot}]$ if condition 4.24 holds
     (b) $[shift_{start_slot, break_slot}]$ and $[break_slot + 1, shift_{last_slot}]$ if condition 4.25 holds
  3. Remove the old shift from the dictionaries for that day.
  4. Add the two newly created shifts to the shift catalog, by their start and last slot.
- **Allocation**
  1. For each slot range between the old $[shift_{break_start_slot}, shift_{break_last_slot}]$, remove one allocated employee from the break task and assign it to a random non-break task.

4.2.7.6 Mutation: Join

This type of mutation joins two shifts into one at the selected slot (figure 4.26). The shifts must be side by side, which means that the slot after the last slot of one of the shifts must be equal to the starting slot of the other. The sum of the shift's duration must also be less or equal to the maximum shift duration allowed. Condition 4.26 represents the prerequisites for the feasibility of this type of mutation.

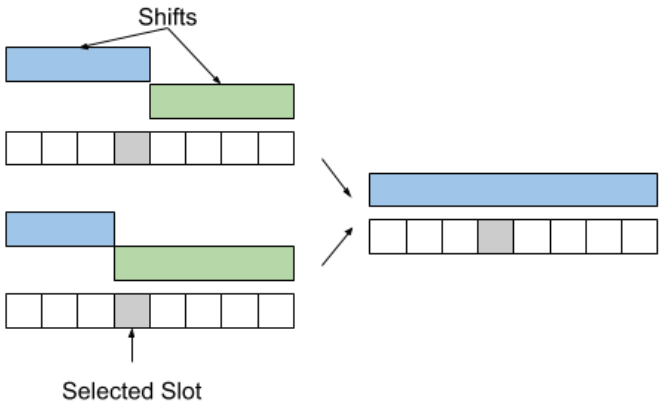


Figure 4.26: Mutation *join* operation. Two possible joining points for the same outcome.

$$\begin{aligned}
&(shift_{1,last_slot} \in [mutation_slot - 1, mutation_slot]) \\
\wedge &(shift_{2,start_slot} \in [mutation_slot, mutation_slot + 1]) \\
\wedge &(shift_{1,last_slot} - shift_{2,start_slot} == 1) \\
\wedge &(shift_{1,duration} + shift_{2,duration} \leq constraints_{shift_max_duration})
\end{aligned} \tag{4.26}$$

To apply this mutation, only one structure must be updated, for that specific day:

- **Shift Catalog**
  1. Remove the break start and last slot information for the shift.
  2. Create a new shift with the information from the two selected, resulting in a start slot at $\min(shift1_{start_slot}, shift2_{start_slot})$ and last slot at $max((shift1_{last_slot}, shift2_{last_slot})$.
  3. Remove the old shifts from the dictionaries for that day.
  4. Add the newly created shift to the shift catalog by its start and last slot.
- **Allocation**
  1. For each slot range between the old $[shift_{break_start_slot}, shift_{break_last_slot}]$, remove one allocated employee from the break task and assign it to a random non-break task.

4.2.7.7 Mutation: Remove

This type of mutation removes a shift from the planning (figure 4.27). For this to be acceptable, the shift must overlap with the selected slot and sales coverage must be guaranteed if the shift is removed (condition 4.27).

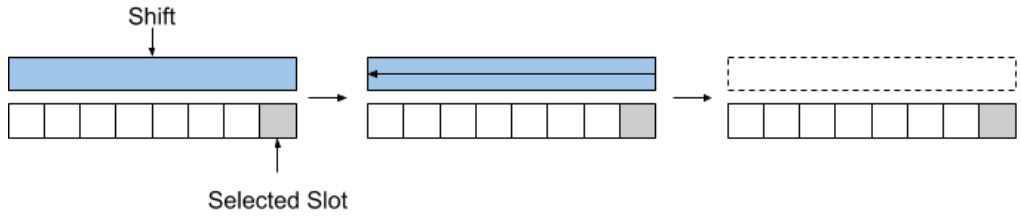


Figure 4.27: Mutation *remove* operation.

$$\begin{aligned}
&(shift_{start_slot} \leq mutation_slot) \\
\wedge &(shift_{last_slot} \geq mutation_slot) \\
\wedge &(sales_{slot} > 1, \forall slot \in [shift_{start_slot}, shift_{last_slot}])
\end{aligned} \tag{4.27}$$

To apply this mutation, the three structures must be updated, for that specific day:

- **Shift Catalog**

  1. Remove the shift from both dictionaries.

- **Availability**

  1. For all slots between the beginning and end of the shift, increase the number of available employees in one.

- **Allocation**

  1. For all slots between the start and the last of the shift, choose a random non-break task and remove one allocated employee. Tasks can only be chosen if the number of allocated employees is above zero or above one, in the case of sales task. In the slots that corresponded to the shift's break, the employee must be removed from the break task.

4.2.7.8 Mutation: Reallocate

This type of mutation reallocates a random number of employees from a random task to another (figure 4.28). This mutation is only possible if another task, apart from break, has employees allocated in the selected slot (condition 4.28).

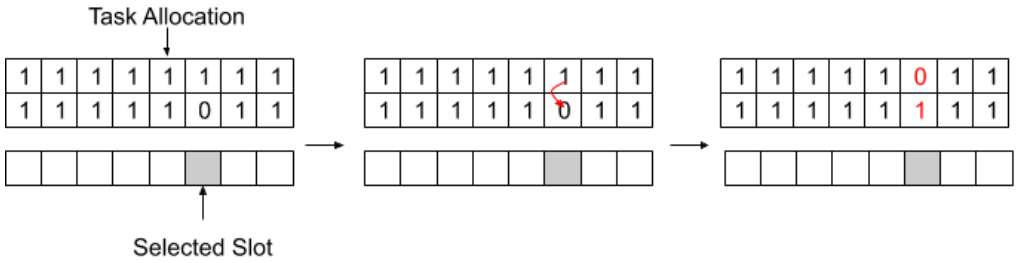


Figure 4.28: Mutation *reallocate* operation.

$$\begin{aligned}&(allocation_{task,mutation_slot} > 0 \wedge task \notin [SALES, BREAK])\\ \vee &(allocation_{sales,mutation_slot} > 1)\end{aligned} \tag{4.28}$$

To apply this mutation, only one structure must be updated, for that specific day:

- **Allocation**

  1. Choose a random non-break task where the number of allocated employees is above zero or above one, in case of sales task (condition 4.28).
  2. Choose a random non-break task different from the one chosen in step 1.

3. Randomly choose a number of employees to reallocate between $[1, allocated_{task}]$ for the chosen task in step 1. If the sales task is chosen, this can only range between $[1, allocated_{task} - 1]$.
4. Remove the employees chosen in step 3, from the selected task in step 1, and allocated them to the task selected in step 2.

4.2.7.9 Break Adjustment

Some types of mutation change directly the characteristics of a shift. It might happen that, after a mutation is applied, the ideal break duration of a shift is not the same as the one that actually exists. Due to this, it is important to adjust the breaks so that the break ratio restriction, although a soft constraint, is violated the fewest possible number of times.

There can be up to four situations for break adjustment:

- **Add** - The ideal duration break is above zero and the shift does not have any break (condition 4.29). Table 4.12 shows the types of mutation where this type of adjustment may be necessary. To add a break, the following steps must be performed:

  1. Calculate the start and last slot range, according to table 4.8 in section 4.2.3, for the ideal break duration.
  2. Extract the list of feasible slot ranges, between the slot range calculated in step 1, where sales allocation is above one. This way, we guarantee that the break is positioned only on valid slots.
  3. If no slot range, in the list of step 2, allows to fulfilling the ideal break duration, than the one that allows the longest break is selected. Otherwise, one is randomly picked.
  4. Randomly choose the start slot between the slot range calculated through eq. 4.30. From the valid slot range chosen in step 3, $range[0]$ represents the first slot of the range and $range[1]$ the last.
  5. Calculate the last slot of the break through eq. 4.31.
  6. Update the representation structures with the new break.
     (a) In the allocation structure, for each slot between the break starting slot and last slot, an employee is removed from a random non-break task and allocated to the break. The number of employees on sales must be above 1, to allow choosing from that task, and above 0 in the others.
     (b) In the shift catalog, update the shift's break start slot and last slot.

$$\begin{aligned} &(ideal_break_duration > 0) \\ \wedge &(shift_{break_duration} == 0) \end{aligned} \tag{4.29}$$

$$break_{start_slot} \in [range[0], \max(range[0], range[1] - ideal_break_duration))] \tag{4.30}$$

$$break_{last_slot} = break_start_slot + \min(ideal_break_duration - 1, range[1]) \tag{4.31}$$

Table 4.12: Mutation types where adding a break might be necessary.

Mutation Types
Extend Start
Extend Last
Join

- **Remove** - The shift has a break inserted but ideally, no break is needed (condition 4.32). Table 4.13 shows the types of mutation where this type of adjustment may be necessary. To remove a break, the following steps must be performed:

  1. In the allocation structure, reallocate one employee from the break task to a random non-break task. This must be done for all time slots between the break start slot and the last slot.
  2. In the shift catalog, the shift must update its break start and last slot to -1 (no break).

$$\begin{aligned} &(ideal_break_duration == 0) \\ \wedge &(shift_{break_duration} > 0) \end{aligned} \tag{4.32}$$

Table 4.13: Mutation types where removing the break might be necessary.

Mutation Types
Reduce Start
Reduce Last
Break
Remove

- **Extend** - The ideal duration break is higher than the current break duration (condition 4.33). Table 4.14 shows the types of mutation where this type of adjustment may be necessary. To extend the duration of a break, the following steps are performed:

1. Calculate the number of slots to extend (eq. 4.34).
2. For each time slot to add:
   (a) Check if the $break_{start_slot} - 1$ and $break_{last_slot} + 1$ have more than one employee allocated to sales or above zero on the other tasks, apart from break. From those who fulfill this condition, select the one that is farther away from the shift's start or end. If none of these slots fulfill the condition, the break adjustment stops and no more breaks are added.
   (b) In the allocation structure, randomly choose an employee from a non-break task that has allocated employees above zero or one, in the case of sales, and reallocate one employee to the break task.
   (c) Update the shift's break start and last slot in the shift catalog.

$$ideal_break_duration > shift_{break_duration} > 0 \tag{4.33}$$

$$n_slots_extend = ideal_break_duration - shift_{break_duration} \tag{4.34}$$

Table 4.14: Mutation types where extending the break's duration might be necessary.

Mutation Types
Extend Start
Extend Last

- **Reduce** - The ideal duration is lower than the current break duration (condition 4.35). Table 4.15 shows the types of mutation where this type of adjustment may be necessary. To reduce a break, the following steps must be performed:

1. Calculate the number of slots to reduce (eq. 4.36).
2. For each time slot to reduce:
   (a) Select the slot between $break_{startslot}$ and $break_{last_slot}$ that is closest to the shift's start or end.
   (b) In the allocation structure, reallocate one employee from the break task to a random non-break.
   (c) Update the shift's break start and last slot on the shift catalog.

$$ideal_break_duration < shift_{break_duration} > 0 \tag{4.35}$$

$$slots_to_reduce = shift_{break_duration} - ideal_break_duration \tag{4.36}$$

Table 4.15: Mutation types where reducing the break's duration might be necessary.

Mutation Types
Reduce Start
Reduce Last

Chapter 5

Results and Analysis

5.1 Experimental Setup

In this section, a set of experiments and the conditions in which they were conducted are presented and described, both for forecasting and optimization. Each experiment aims to evaluate different aspects of the implemented solution and draw conclusions from their results.

5.1.1 Data

To implement the solution proposed, real-world data was used to train the forecasting models and extract useful information for the optimization process. This data was previously described in chapter 3.3 along with the cleaning and data preparation process. A more specific feature engineering process was presented later, in section 4.1.2, to prepare the input data for each forecasting model.

5.1.2 Experiments

Experiments are divided into forecasting and optimization. The former evaluate the quality of the individual models and the sales forecasting architectures (section 4.1.1). The latter assess, the schedules prescribed by the optimization algorithm based on different characteristics.

Both forecasting and optimization experiments require that each forecasting model, described in section 4.1.1, must be trained using data that is collected and prepared before the moment a prediction is made. In order to fulfill both requirements and later combine results, all models are trained until the day before the first day of planning and are evaluated during the planning horizon. According to table 5.1, models M1 and M2 are trained on data between [2016-01-02 10:00:00,2019-01-31 23:30:00], representing about 3 years, and models M3,M4,M5 and M6 between [2018-03-11 10:00:00, 2019-01-31 23:30:30], representing about 10 months. This leaves

us with one month and a half with historical data for evaluation. The models are trained using the random forest regressor from the *scikit-learn* API [37] with the parameters defined in table 5.2.

Table 5.1: Default optimization parameters used in the experiments.

Parameters	**Value**
Start planning date	2019-02-01 00:00:00 (Friday)
End planning date	2019-02-07 23:30:00 (Thursday)
Time unit	30 minutes
N. iterations	75
Max. tries for slicing crossover	5
Population size	80
Elitist percentage	10 %
Mutation probability	5 %
Remove mutation probability	20 %

Table 5.2: Random forest parameters values for the training of the models

Parameters	**Value**
Number of trees	100
Criterion	MSE
Max. depth	30
Min. samples to split	2
Min. samples on leaf	1
Max. features	Number of features
Min. impurity decrease	0
Min. impurity split	$1e^{-7}$
Bootstrap	True

To evaluate the performance of the trained models, different experiments are conducted as described in table 5.3. The forecasting experiments are represented by Fx, where x is the number of the experiment. Although they were designed to test different aspects, the main steps for evaluation are the same: forecast each target for a time horizon and compare these values with the ground truth. The first two experiments focus on evaluating the individual performance of models and the sales forecasting architectures, suggested in section 4.1.1. The last two experiments study how the error in footfall is propagated to later models. In the third experiment the performance of the models is compared when accurate (i.e. observed) and forecasted footfall is used as input. Later, in the last experiment, an increasing error is added to the accurate footfall and the performance is measured for each error in each model. The last two experiments are unrealistic, however, it enables us to understand the robustness of the prediction in sales depending on the accuracy of the predictions of *footfall in*.

Table 5.3: Brief description of the forecasting experiments and the expected result.

Exp.	Description	Expected Results
$F1$	Evaluate the performance of each model on unseen data.	Recognition of daily and weekly seasonality trends.
$F2$	Compare a direct sales forecasting architecture with an indirect architecture.	
$F3$	Compare the use of accurate versus forecasted *footfall in*, as input in the models responsible for sales forecasting.	The performance is higher when the real footfall is used.
$F4$	Evaluate the impact of footfall error, in the sales forecasting architectures, varying it between [0%,300%].	The higher the error in *footfall in*, the lower the performance of the models.

To evaluate the implemented optimization solution, in different conditions, six different runs were executed (table 5.4). The default optimization parameters can be consulted in table 5.1 and the values of the constraints in table 5.5 (additional information in table 5.6). Each run is characterized by Rxy were x is the type of run and y if a direct (a) or indirect (b) sales forecasting architecture was used. The first run, $R1y$, is a simple run without any special parameters. The second run, $R2y$, optimizes schedules for one week of planning but one week in advance. This means that the solutions are evaluated with forecasted information from the week before, using the schedule defined by the retailers. In the last run, $R3y$, an additional error of 20% is inflicted in the forecasted *footfall in* attribute.

Table 5.4: Optimization runs characteristics.

Run	Sales Architecture	Special Parameters
$R1a$	Direct	
$R1b$	Indirect	
$R2a$	Direct	Planning with one week in advance with a pre-defined schedule for the first week.
$R2b$	Indirect	
$R3a$	Direct	Forecasted *footfall in* with 20% additional error
$R3b$	Indirect	

Table 5.5: Default constraints used in the experiments.

Constraints	Value
N. employees	8
Cost employee/hour	5
Store operating hours	[10:00:00h, 00:00:00h]
Max. shift duration	09:00:00h
Max. shift duration (no breaks)	05:00:00h
Min. shift duration	04:00:00h
Break/shift duration	01:00:00h / 08:00:00h

Table 5.6: Information extracted from the default optimization parameters and constraints.

Constraints	**Value**
Slots per day	48
Total slots	336
Open slots per day	28
Total open slots	196
Avg. time slots mutated at each iteration per individual	10
Max. shift slots duration	18
Max. shift slots duration (no breaks)	10
Min. shift slots duration	8
Break/shift slots duration	1/4

Some optimization parameters and constraints were extracted from historical data, after the data cleaning and analysis processes, so that we could deal with real values rather that arbitrary ones. Since the chosen planning horizon is in the past, historical data is available to fill some parameters such as the operating hours and the number of employees, extracted from the open hours dataset. The characteristics of the shifts were extracted from the shifts dataset, by analyzing the histogram containing the shift's duration and the relation between a shift and its break duration.

Table 5.7: Brief description of the optimization experiments and the expected result.

Exp.	**Runs**	**Description**	**Expected**
$O1$	$R1a$ ($O1a$) $R1b$ ($O1b$)	Comparison between the prescribed solution with the retailer's solution.	The prescribed solution is better than the one applied by the retailer.
$O2$	$R1a$ x $R2a$ ($O2a$) $R1b$ x $R2b$ ($O2b$)	Comparison between two prescribed solutions in which one was planned with one week in advance, using a pre-defined schedule for the first week.	Optimization in advance is more inaccurate due to the increased error in forecasting.
$O3$	$R1a$ x $R3a$ ($O3a$) $R1b$ x $R3b$ ($O3b$)	Comparison between two optimization runs in which one has an additional 20% error in the forecasted *footfall in*.	The prescribed solution, with error in footfall, is estimated to have lower sales than the ones estimated during optimization.
$O4$	$R1a$ x $R1b$	Comparison between prescribed solutions using a direct and indirect sales forecasting architecture.	

Table 5.7 presents a brief description and what is expected from each optimization experiment, along with the associated prescribed solutions from each run. Each optimization experiment is

represented by *Ox*, where *x* is the number of the experiment. However, since different sales forecasting architectures can be used, each experiment is decomposed into two sub-experiments represented by *Oxy*, where *y* the type of architecture used. For a direct architecture, the sub-experiment is represented by *Oxa* and *Oxb* if an indirect architecture is used.

The experiments proposed and described above were designed to answer to some particular research questions, related to the implemented solution. Table 5.8 shows and formulates the question associated to each experiment.

Table 5.8: Research questions for the forecasting and optimization experiments.

Experiment	Research Question
$F1$	Do the forecasting models detect seasonal patterns?
$F2$	Does dividing the sales forecasting problem into sub-problem leads to better forecasts?
$F3$	If *footfall in was available would the estimated sales be more accurate?*
$F4$	If we increase the error of *footfall in* would the error in the forecasted sales increase as well?
$O1$	Is the prescribed solution better than the manually designed by the retailer?
$O2$	Does prescribing closer to the planning horizon leads to better solutions?
$O3$	Does prescribing with more accurate data leads to better solutions?
$O4$	Are the prescribed solutions adapted to the forecasting architecture used for optimization?

5.1.3 System Specifications

The running time of the optimization runs is highly dependent on the machine used to run the experiments. For this reason, the machine specifications are presented in table 5.9.

Table 5.9: Hardware specifications of the machine used for running the experiments.

Brand	TOSHIBA
Model	SATELLITE P50-B-10V
CPU	Intel(R) Core(TM) i7-4710HQ CPU @ 2.50GHz, 2501 Mhz, 4 Core(s), 8 Logical Processor(s)
GPU	2.00 GB AMD Radeon™ R9 M265X, Intel® HD Graphics 4600
RAM	8.00 GB DDR3L-1600
System Type	x64-based PC
Operating System	Ubuntu 18.04.2

For faster optimization, computational parallelization is used for individual evaluation, crossover and mutation. In this case, all cores of the machine are used to process information in parallel. Information about the necessary time to run each experiment and the average time per optimization iteration will also be presented in the results.

5.2 Evaluation Metrics

To measure and evaluate the results of the experiments, different metrics are calculated and analyzed. In the forecasting experiments, the important is to measure model performance and therefore, a comparison between the forecasts and ground truth values is done. In the optimization experiments, a valuable metric is to measure the distance and similarity between two schedules.

The forecasting metrics used to evaluate a model are traditional regression metrics available in the API used [38], such as:

- **MSE** - Mean Squared Error
- **RMSE** - Root Mean Squared Error
- **MAE** - Mean Absolute Error
- **MDAE** - Median Absolute Error
- **EVS** - Estimated Variance Score
- **MER** - Maximum Residual Error
- **R2** - R2 Score

In order to compare two schedules, in the optimization experiments, different levels of employee assignment is examined (table 5.10). In level zero, allocation to particular tasks is not taken into account and therefore, only the number of total working employees, by time slot, is considered. Level one and two, acknowledge task allocation but in the first, only non-break tasks are of interest.

Table 5.10: Levels for allocation distance and similarity.

Level	Description
0	Only the total employee allocated are considered at each time slot. No particular task allocation is taken into consideration.
1	Only allocations to non-break tasks are considered.
2	All tasks are considered.

To measure the distance between two solutions, the allocation structures are analyzed and two types of distances are calculated:

- **Unit Distance** - Number of employees re-allocations, to add or remove from each time slot and task, so that a schedule can be transformed into the other (equation 5.1).

$$unit_distance(s1,s2) = \sum_{task}^{tasks} \sum_{day}^{planning_days} \sum_{slot}^{open_slots} abs(s1_{task,day,slot} - s2_{task,day,slot}) \quad (5.1)$$

- **Euclidean Distance** - Euclidean distance between the allocation of employees through the planning horizon (equation 5.2).

$$euclidean_distance(s1,s2) = \sqrt{\sum_{task}^{tasks} \sum_{day}^{planning_days} \sum_{slot}^{open_slots} (s1_{task,day,slot} - s2_{task,day,slot})^2} \quad (5.2)$$

From the distance measures presented above, the similarity can be calculated by comparing the distance of two schedules with the maximum distance possible. Since the possible employee allocation per time slot is between $[1, n_employees]$, the maximum allocation difference is the number of employees minus one. Allocation during the store's closed hours is not relevant, because it's always the same and the distance is always zero. With this in mind, the maximum distance between schedules can be calculated by $(n_employees - 1) * total_open_slots$, for unit distance, and by $\sqrt{(n_employees - 1)^2 * total_open_slots}$, for Euclidean distance. Thus, the similarity can be calculated using equations 5.3 and 5.4, for each type of distance.

$$unit_similarity(s1,s2) = 1 - \frac{unit_distance(s1,s2)}{(n_employees - 1) * total_open_slots} \quad (5.3)$$

$$euclidean_similarity(s1,s2) = 1 - \frac{euclidean_distance(s1,s2)}{\sqrt{(n_employees - 1)^2 * total_open_slots}} \quad (5.4)$$

The similarity between two schedules only evaluates how much more distant they could be, regarding employee allocation to tasks on time slots. No similarity is measured in terms of the shifts created and their specifications. For this reason, and given the meaning of similarity, this metric is a weak evaluation metric since it does not consider all aspects of the two solutions. To measure the distance, we are more interested in measuring the difference between the allocation structures because this is the information that will be used to forecast sales. Due to this, unlike the similarity metric, the distance is a strong measure for evaluation.

5.3 Results

In this section, the results of the conducted experiments are presented and properly analyzed. Due to the great amount of data produced by each experiment, only the most important information is presented. The rest of the information can be found in appendix C, which has the same structure as this one.

5.3.1 Forecasting

5.3.1.1 Experiment F1

"Do the forecasting models detect seasonal patterns?"

From this experiment, we can analyze the quality of the models and if they can recognize seasonal patterns. In this experiment we ignore the *footfall out* results since the historical data for this target is always zero, due to the target unavailability, and won't produce any results of interest. The evaluation metrics values for each model can be consulted in table 5.11.

Table 5.11: Evaluation metrics values for each target forecasted in experiment $F1$.

Target	Models	MSE	RMSE	MAE	MDAE	EVS	MER	R2
footfall_in	M1	40.1557	0.0778	3.56087	1.0552	0.7793	29.3794	0.7537
footfall_out	M2	0.0	0.0	0.0	0.0	1.0	0.0	1.0
n_transactions	M1,M2,M3	2.1618	1.0809	0.9220	0.4050	0.5467	5.31	0.5215
avg_products_per_t	M1,M2,M4	0.5154	0.2577	0.3567	0.0769	0.3419	5.2536	0.3400
avg_price_per_product	M1,M2,M5	619.2167	309.6083	15.3697	6.2276	0.5471	89.1466	0.5469
total_sales	M1,M2,M6,	16496.0391	8248.0195	81.1688	44.3931	0.4627	463.3963	0.4372

By analyzing the ground truth values for each model evaluation (figures 5.1, 5.2, 5.3, 5.4, 5.5 and 5.6), we can clearly conclude that the *footfall in* is the one with least abrupt changes and variations through time. Maybe this can explain why, visually, the model seems to identify a better daily and weekly seasonality pattern that the others.

The model that forecasts the number of transactions seems to detect the weekly seasonality, however, during the day, even though the forecasted values seem to vary with the same frequency as the ground truth, the variations of the latter are much bigger (figure (5.3)). This may lead to spikes in the day that the model is unable to forecast, which leads to a greater mean squared error. The models that forecast the average number of products and the average price also suffer from this problem. Howbeit, these don't demonstrate any weekly nor specific daily patterns through the forecasted values (figures 5.4 and 5.5).

The model that forecasts the total sales, although it has a high MSE, it seems to follow a weekly trend by having the first three days of the time horizon, Friday, Saturday and Sunday, with higher sales than the rest of the week. Given the context of this project, assuming that sales will be higher at the end of the week, is an acceptable assumption. The high MSE is due to the sudden change in sales in a short interval and the incapacity of the model to keep up.

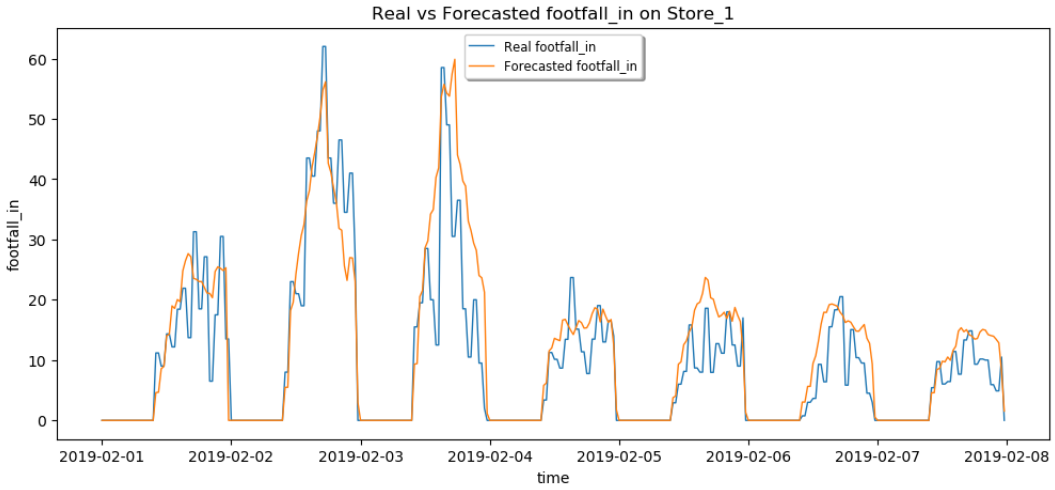


Figure 5.1: Real versus forecasted *footfall in*.

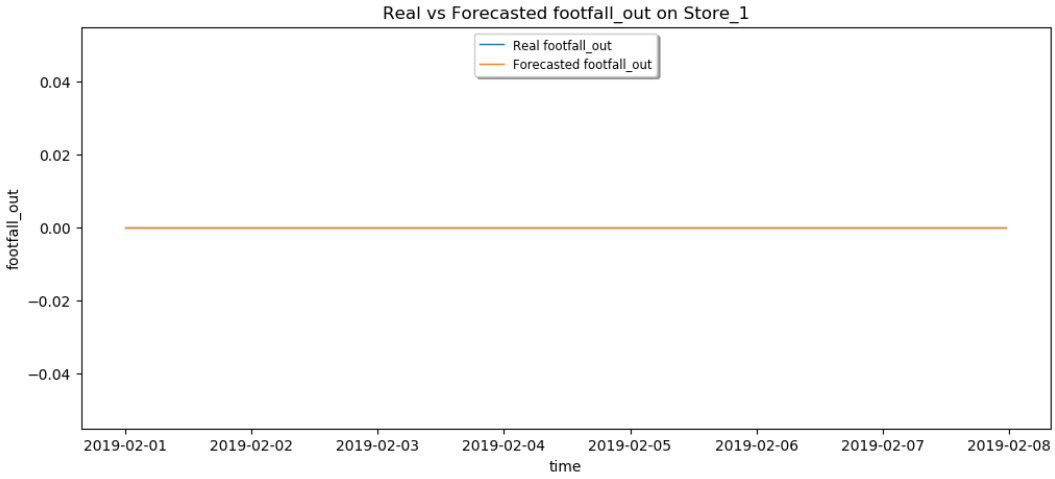


Figure 5.2: Real versus forecasted footfall out.

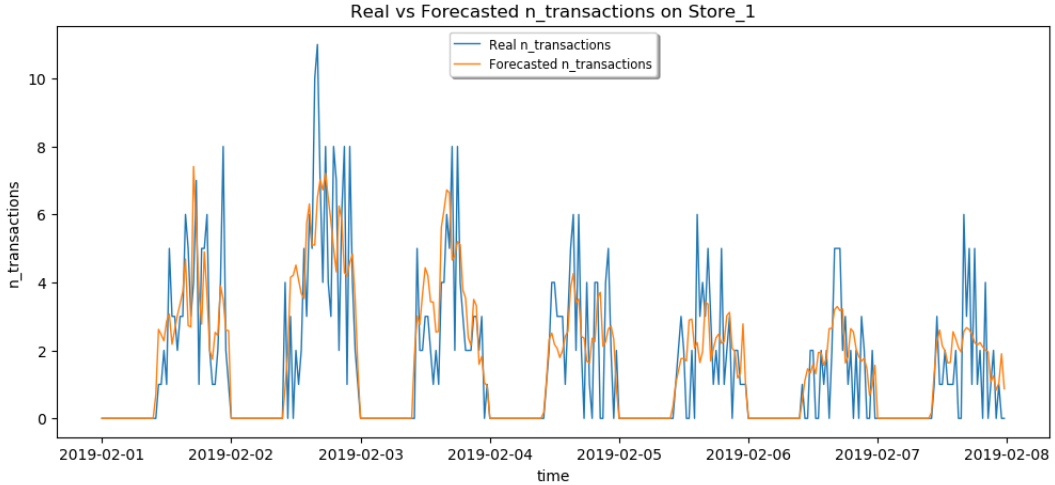


Figure 5.3: Real versus the forecasted number of transactions.

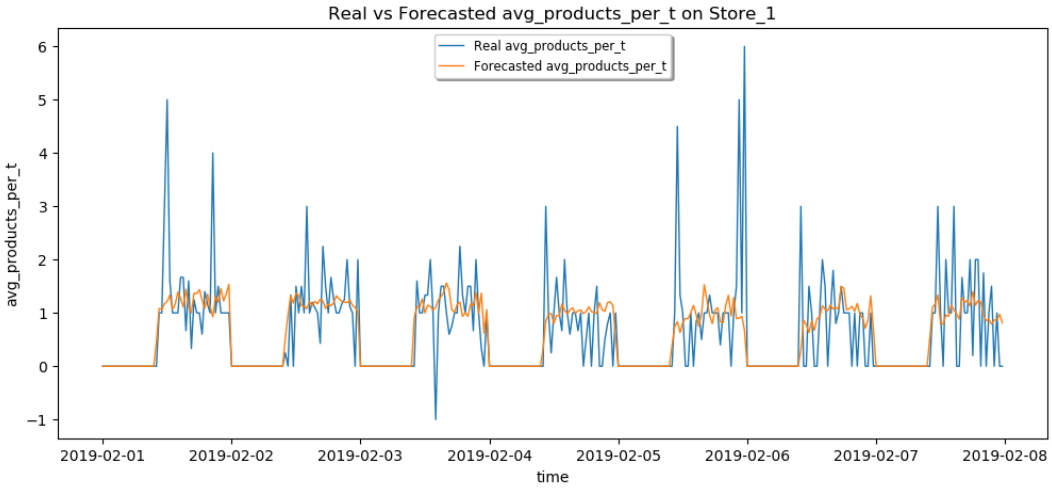


Figure 5.4: Real versus forecasted average products per transaction.

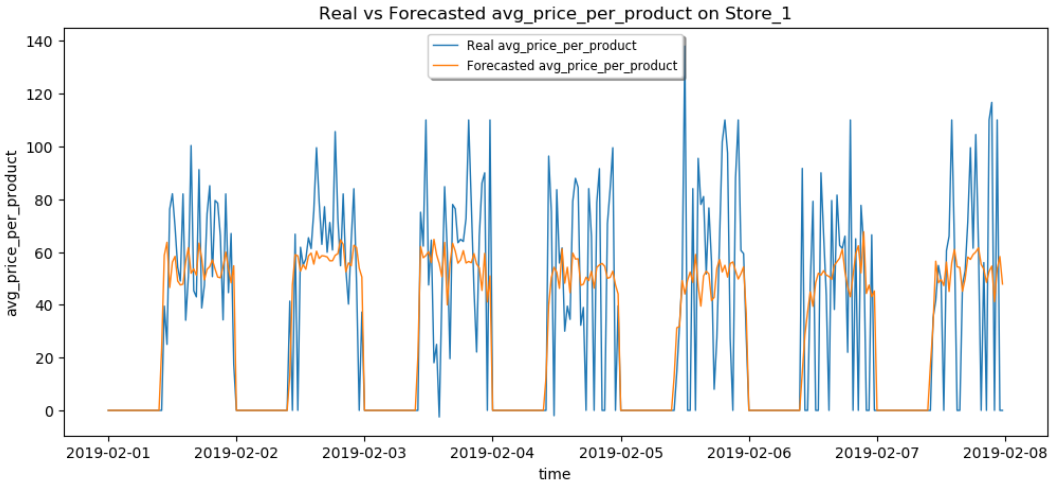


Figure 5.5: Real versus forecasted average price per transaction.

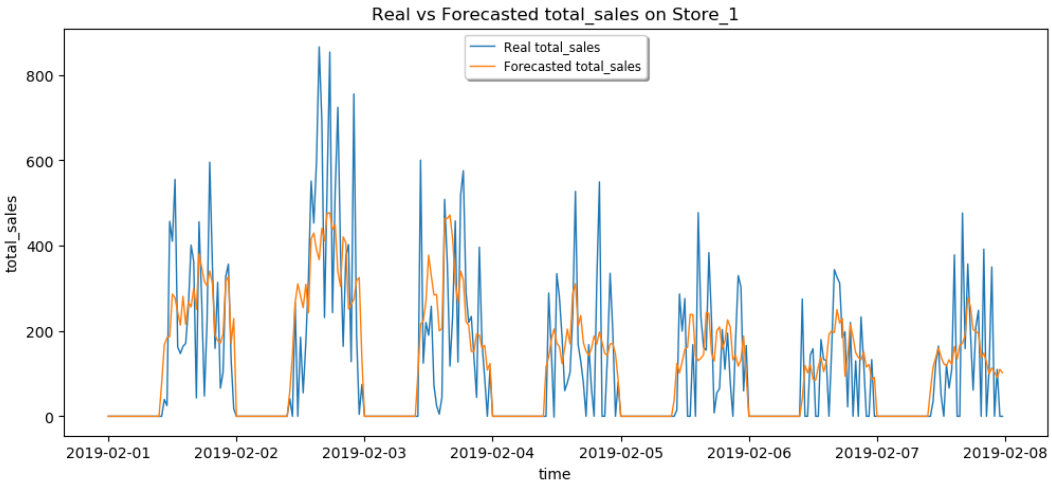


Figure 5.6: Real versus forecasted direct sales.

5.3.1.2 Experiment F2

"Does dividing the sales forecasting problem into sub-problems leads to better forecasts?"

When comparing the sales forecasts through an indirect or direct architecture, for the same forecasting horizon, we can analyze that the errors associated with each architecture are not significantly different (table 5.12). However, overall, the indirect sales forecasting architecture present lower error values.

Table 5.12: Evaluation metrics values for each sales forecast architecture in experiment $F2$.

Architecture	MSE	RMSE	MAE	MDAE	EVS	MER	R2
Direct Sales	16496.0391	8248.0195	81.1688	44.3931	0.4627	463.3963	0.4372
Indirect Sales	16263.4957	8131.7479	75.1551	20.5672	0.4469	495.1852	0.4452

By analyzing the forecasted sales of each architecture with the ground truth values, in figure 5.7, we can see that the indirect architecture is better at forecasting the spikes in sales. The direct architecture, on the other hand, has more stable forecasts between short time intervals. Since the indirect architecture combines the predictions from three models, it seems that errors from one model are being somewhat compensated by the errors from the others. For example, if the number of transactions is forecasted with the double of the ground truth value and the average price per product with the half, they both have a 50% error but in the end, the sales are forecasted as if none of these errors exists. Although, at first sight, this may seem positive, it makes the indirect sales architecture unpredictable. For instance, replacing one of the internal models with another with higher accuracy, which is a reasonable thing to do, may lead to a decrease in the forecasting ability of the whole system.

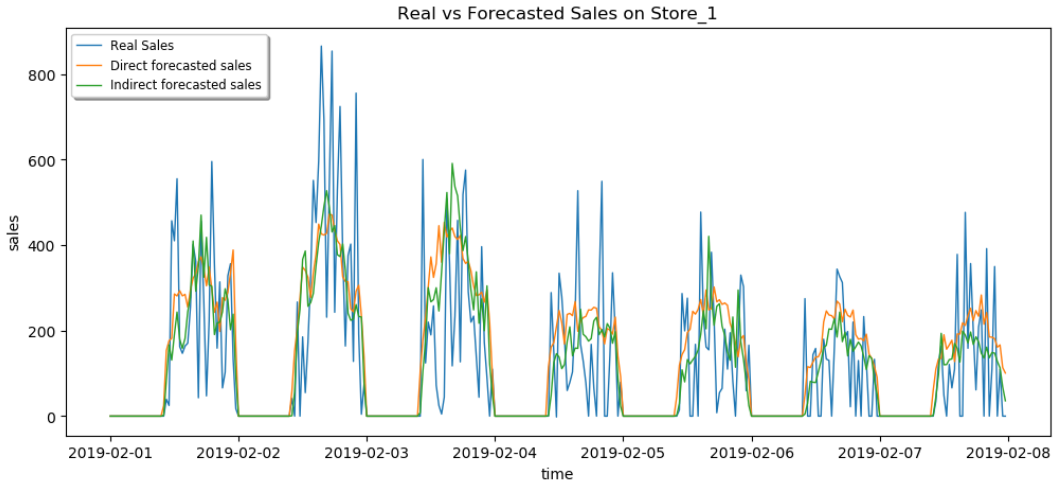


Figure 5.7: Real versus direct and indirect forecasted sales.

5.3.1.3 Experiment F3

"If footfall in was available would the estimated sales be more accurate?"

In this experiment, a comparison is made between using real or forecasted *footfall in* the models that depend on this attribute as input. As it's demonstrated by table 5.13, models forecast their target better when real footfall is used. The main reason for this is that sales are highly correlated with footfall. However, this is not possible in practice footfall data is collected shortly before the corresponding sales take place and the latter must be predicted several days in advance. However, these experiments confirm that, as expected, the error associated with footfall is propagated to the next models, affecting their performance.

Although overall, models produce better forecasts with accurate footfall, the percentage of improvement is not the same for all. According to table 5.14, the *footfall in* attribute has a higher importance in the models responsible for forecasting the number of transactions and direct sales. This explains why the errors are considerably different when real or forecasted footfall is used.

When comparing the sales forecasting architectures, direct sales would be a better approach if real footfall was available. However, the current error in footfall has a higher impact on this type of architecture leading to an error greater than the one in the indirect architecture, even though the difference is not significant.

Table 5.13: Evaluation metrics values for each target forecasted using accurate and forecasted *footfall in* for experiment $F3$.

Target	Footfall	MSE	RMSE	MAE	MDAE	EVS	MER	R2
n_transactions	Accurate	1.7977	0.8988	0.8232	0.3199	0.6057	4.9	0.6021
	Forecasted	2.1618	1.0809	0.9220	0.4050	0.5467	5.31	0.5215
avg_products_per_t	Accurate	0.5102	0.2551	0.3467	0.0767	0.3468	5.3245	0.3467
	Forecasted	0.5154	0.2577	0.3567	0.0769	0.3419	5.2536	0.3400
avg_price_per_product	Accurate	630.6505	315.3252	15.4555	4.8687	0.5392	93.9348	0.5385
	Forecasted	619.2167	309.6083	15.3697	6.2276	0.5471	89.1466	0.5469
total_sales (direct)	Accurate	13626.5075	6813.2538	71.3033	29.8749	0.5368	498.7829	0.5351
	Forecasted,	16496.0391	8248.0195	81.1688	44.3931	0.4627	463.3963	0.4372
total_sales (indirect)	Accurate	14705.7691	7352.8845	70.3948	25.6049	0.5034	508.4782	0.4983
	Forecasted	16263.4957	8131.7479	75.1551	20.5672	0.4469	495.1852	0.4452

Table 5.14: Importance of *footfall in* attribute associated with each forecsating model.

	n_transactions	avg_products_per_t	avg_price_per_product	total_sales (direct)
Importance	0.579904	0.006793	0.007208	0.483238

5.3.1.4 Experiment F4

"If we increase the error of footfall in would the error in the forecasted sales increase as well?"

From this experiment, we can further evaluate how the performance of each model, or architecture, is affected by increasing the error in the *footfall in*.

As expected, the error in the forecasted number of transactions and sales, using a direct and indirect architecture, is strongly correlated with the error in footfall. As it can be seen in figures 5.8 and 5.11, the trend is to increase the error when the error in footfall also increases. Additional evidence is given by the importance that each model assign to this input attribute (table 5.14). On the other hand, the error of the models that forecast the average number of products and average price, are rather unpredictable, given the error in the *footfall in* model and exhibit chaotic changes in error without any particular trend (figures 5.10 and 5.9).

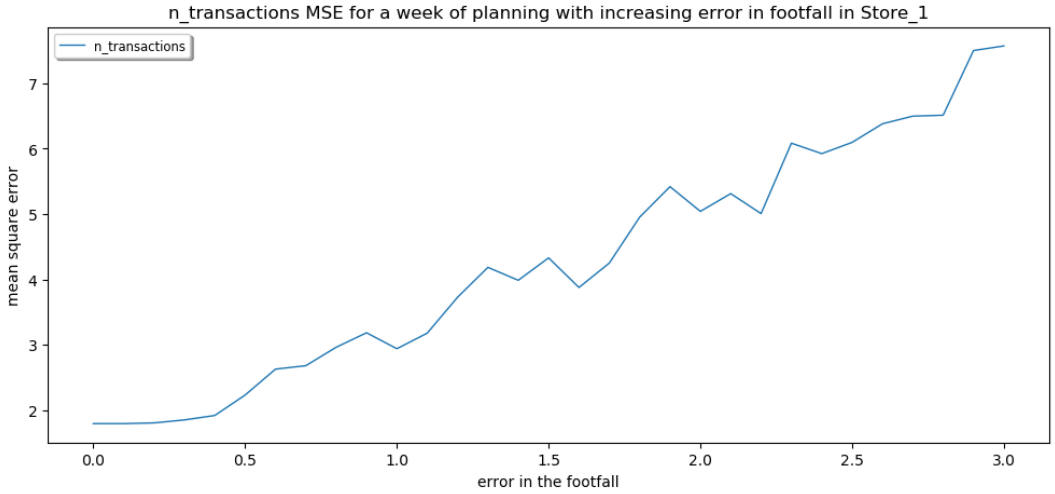


Figure 5.8: MSE of the forecasted number of transactions with increasing *footfall in* error.

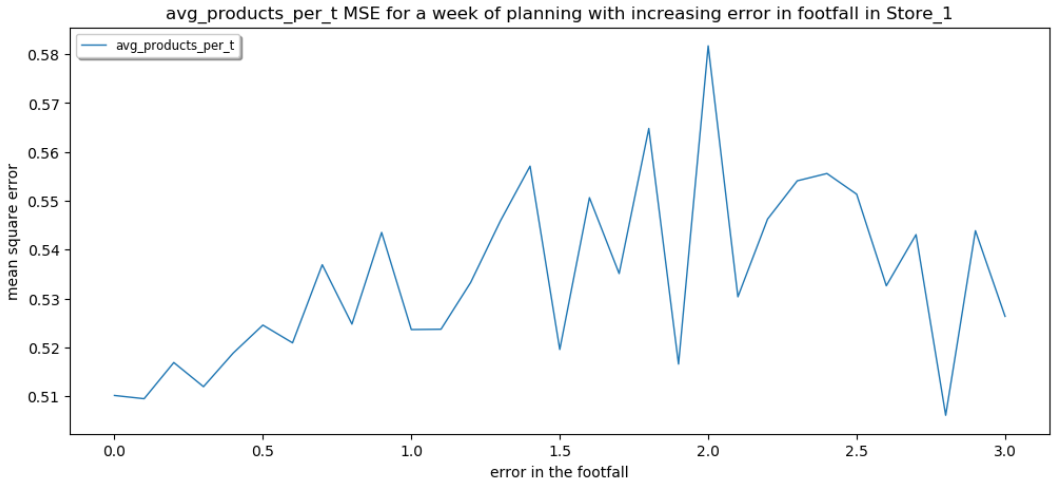


Figure 5.9: MSE of the forecasted average number of products per transactions with increasing *footfall in* error.

When analyzing the behavior of the direct and indirect forecasting sales architectures, both follow an increasing error trend as footfall error increases. Beyond this, they also seem to vary MSE in a synchronized way. However, differently from the results in table 5.11 from the first experiment, the direct architecture systematically obtains a lower error than the indirect one (figure

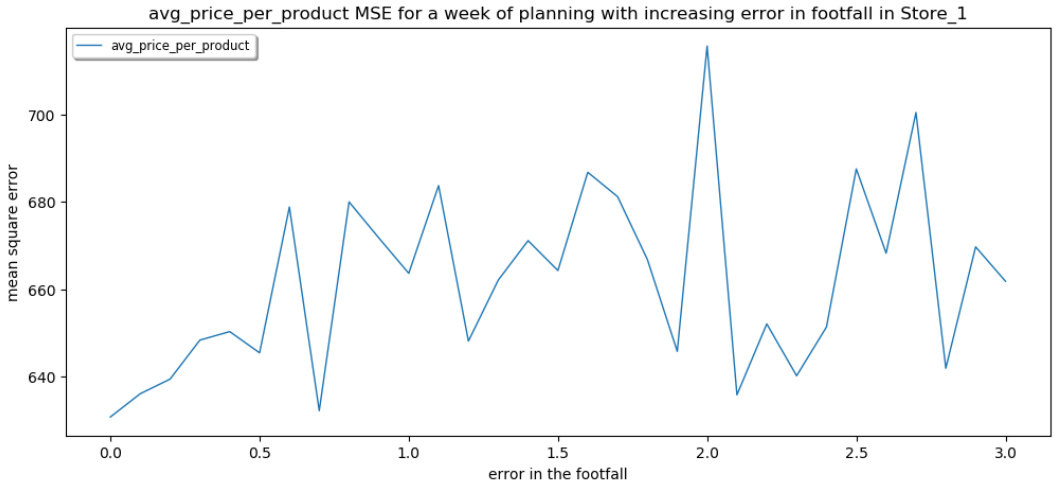


Figure 5.10: MSE of the forecasted average price per product with increasing *footfall in* error.

5.11). This may be due to the combined errors of the models *M3*, *M4* and *M5* in the indirect architecture that leads to a higher error in sales. Therefore, from this experience, we can conclude that the direct architecture is more robust to error in the *footfall in* predictions, which makes it a more suitable choice for our project.



Figure 5.11: MSE of the forecasted direct and indirect sales with increasing *footfall in* error.

5.3.2 Optimization

In the next subsections, the main results of each optimization experiment are provided and discussed. Details about each experiment or optimization run, such as similarity and distance between schedules, can be consulted in the appendix C.2.

The total time to execute each optimization run is presented in table 5.15. As expected, optimization through an indirect sales forecasting architecture takes more time to execute, since there is the triple number of models to forecast, in comparison with a direct architecture. Overall, even

with parallelization, the execution times are high due to the data dependency between each time slot forecast (section 4.1.3).

Table 5.15: Execution time for different optimization runs.

Run	Total Time (m)	Time/Iteration (m)
*R*1*a*	196	2.61
*R*1*b*	389	5.19
*R*2*a*	450	6
*R*2*b*	1255	17
*R*3*a*	193	3.22
*R*3*b*	582	7.75

5.3.2.1 Experiment O1

"Is the prescribed solution better than the manually designed by the retailer?"

From this experiment, we can analyze if the implemented strategy to optimize staff schedules, prescribes better solutions than the schedules used by the retailer (table 5.16). For this, the schedules are evaluated, using the same architecture used in the optimization run, and a comparison is made between each forecasted sales. Information about the schedule suggested by the retailer can be found in the appendix C.2.2.

By the results of this experiment, exhibited in table 5.17, we can verify that the sales forecasted for the prescribed solution are higher than the ones for the manual schedule. This validates the advantage of using the suggested strategy to optimize and build schedules that leads to higher sales. Nonetheless, this is only true if we consider that the forecasting models are accurate enough to lead to this conclusion. Therefore, in figures 5.12 and 5.13, the forecasted sales curve is plotted against the real sales observed when the manual solution was applied. The first figure is related to the sub-experiment *O*1*a*, which uses a direct forecasting sales architecture, and the second to *O*1*b*, which uses an indirect architecture. In these figures, we can see that the real sales oscillate very often and by a large value. In the end, with the total real and forecasted sales for the planning week, we conclude that the forecasting models are over-predicting sales. This makes us believe that if the prescribed solution was applied, the real sales would be lower than the estimated.

Table 5.16: Solutions descriptions in experiment *O*1.

Solution Name	Description
Prescribed Solution	Prescribed solution from run *R*1*a* or *R*1*b*.
Manual Solution	Solution designed by the retailer for the same planning horizon as *R*1*a* and *R*1*b*.

Table 5.17: Forecasted sales for experiment $O1$.

	Manual Solution	**Prescribed Solution**	**Prescribed/Manual**
Sales *O1a*	46304	47819	103.27%
Sales *O1b*	40348	43759	108.56%
Difference	5956	4060	5.29pp

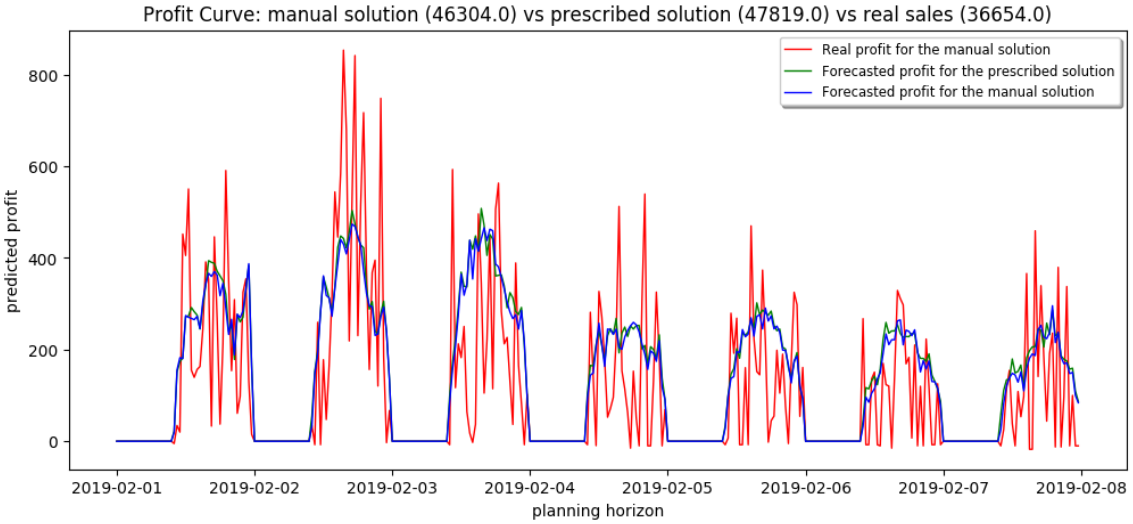


Figure 5.12: Forecasted sales for the prescribed and manual solution in experiment $O1a$ in comparison with the real sales observed for the manual solution.

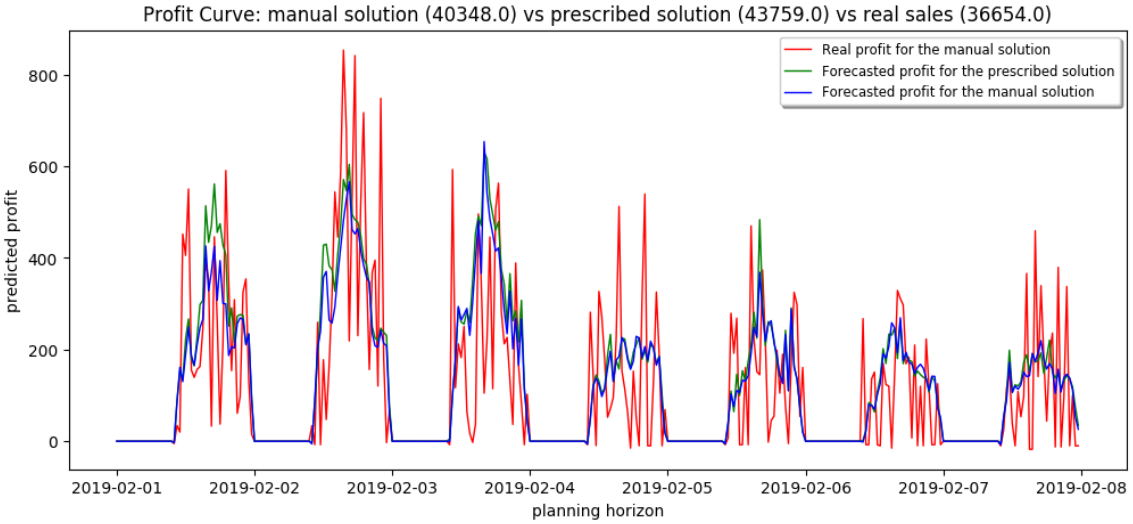


Figure 5.13: Forecasted sales for the prescribed and manual solution in experiment $O1b$ in comparison with the real sales observed for the manual solution.

5.3.2.2 Experiment O2

"Does prescribing closer to the planning horizon leads to better solutions?"

In this experiment, for the same planning horizon, we compare the solution prescribed one week ahead with the solution prescribed immediately before the first day of planning. In the first case, during the week ahead, the schedule is already defined, although sales must still be forecasted in order to optimize the schedule for the planning horizon (the week after). For an easier reference to each solution, the names presented in table 5.18 are used in the legend of the presented figures and tables.

Table 5.18: Solutions descriptions in experiment $O2$.

Solution Name	**Description**
Solution 1	Prescribed solution in run $R2a$ or $R2b$, with one week in advance.
Solution 1a	Solution 1 evaluated with one week in advance.
Solution 1b	Solution 1 evaluated with no time in advance.
Solution 2	Prescribed solution from run $R1a$ or $R1b$

According to table 5.19, the estimated sales for the prescribed solution, with one week in advance, are lower than the ones reestimated one week later. When analyzing figures 5.14 and 5.15, associated with each sub-experiment, we note that the observed sales for the week before planning, were even lower than the expected. Curiously, even though the sales were overpredicted for the first week, the forecasted sales for the prescribed solution increased when evaluated with more recent data.

Table 5.19: Forecasted sales for experiment $O2$.

	Solution 1a	**Solution 1b**	**Solution 2**	**S1b/S1a**	**S2/S1b**
Sales *O2a*	43636	47385	47819	108.59%	100.92%
Sales *O2b*	42582	42645	43759	100.14%	102.61%
Difference	1054	4740	4060	8.45 %	1.69%

When evaluating the prescribed solution one week ahead, after one week, in comparison with the solution immediately before the planning horizon, we can inspect that the latter forecasts higher sales than the former. This may be due to the fact that this solution was prescribed using more recent data, since it was prescribed closer to the planning horizon, which allows it to have more accurate insights and better forecasts. With this, the optimization was more efficiently guided, leading to a better solution.

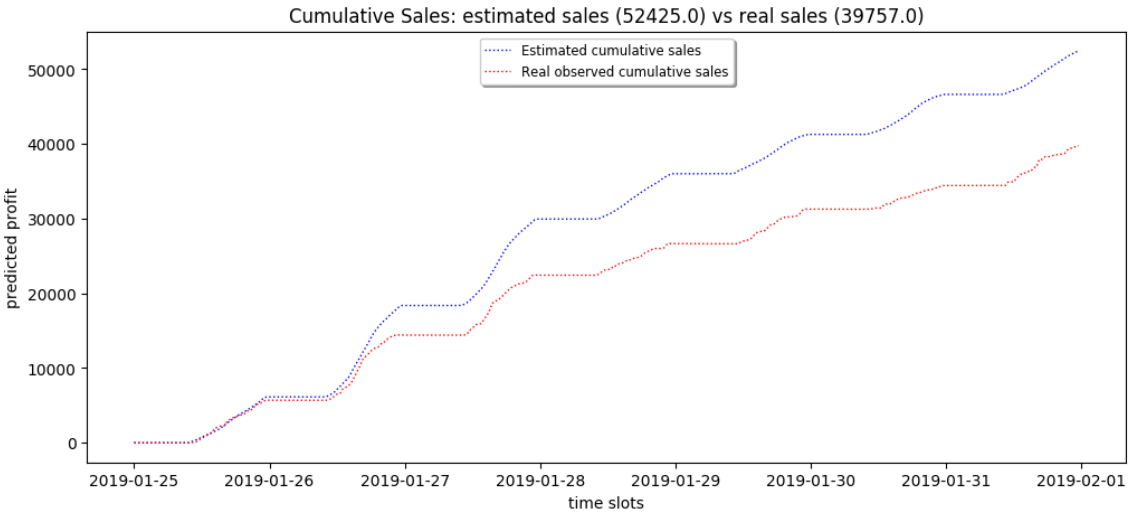


Figure 5.14: Cumulative forecasted and real observed sales for the week before of planning in experiment *O*2*a*.

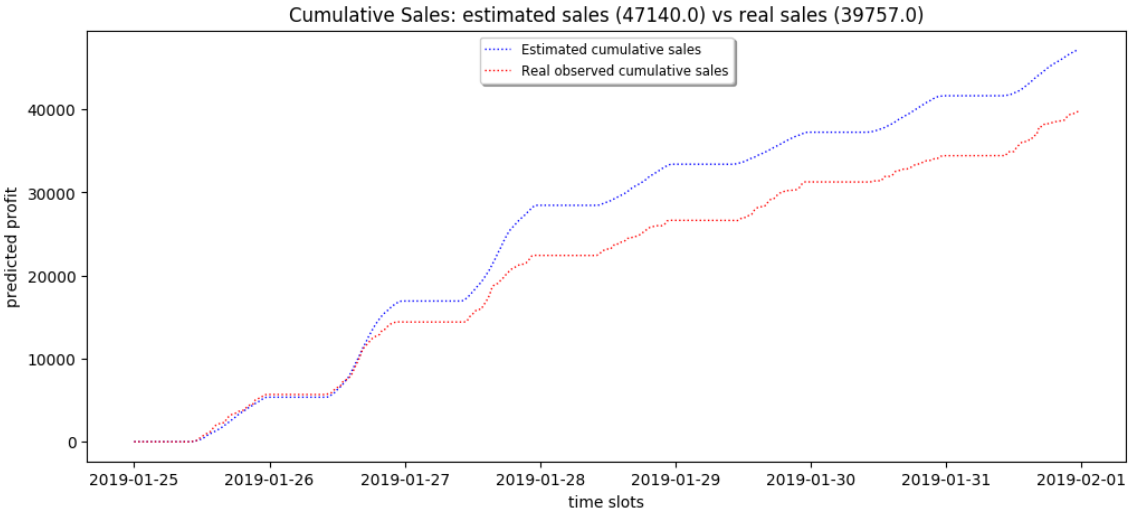


Figure 5.15: Cumulative forecasted and real observed sales for the week before of planning in experiment *O*2*b*.

5.3.2.3 Experiment O3

"Does prescribing with more accurate data leads to better solutions?"

In this experiment, the goal is to evaluate the impact of optimizing schedules with an additional error in the forecasted *footfall in*, which is used to evaluate the individuals in the genetic algorithm. For an easier reference to each solution, on the figures and tables presented below, the names described in table 5.20 are used.

Table 5.20: Solutions descriptions for experiment $O3$.

Solution Name	Description
Solution 1	Prescribed solution from run $R1a$ or $R1b$.
Solution 2	Prescribed solution from run $R3a$ or $R3b$ evaluated with additional error in *footfall in*.
Solution 2a	Solution 2 evaluated with an additional 20% error in the forecasted *footfall in* attribute.
Solution 2b	Solution 2 evaluated with no additional error in the forecasted *footfall in* attribute.

The results in table 5.21 show that when the prescribed solution, optimized through an additional error, is evaluated without error, it has lower forecasted sales than the estimated. This result was already expected since schedules are optimized to forecast higher sales, however, the higher the error in an input attribute, the higher the error in sales (experiment $F3$). This leads to schedules that are misleadingly evaluated and may not be the best solution when evaluated with more accurate data. This explains why the prescribed solution, without any additional error, has higher forecasted sales even when the other solution is evaluated with the same data.

Table 5.21: Forecasted sales for experiment $O3$.

	Solution 1	Solution 2a	Solution 2b	S2b/S2a	S1/S2b
Sales *O3a*	47819	48048	47300	98.44%	101.10%
Sales *O3b*	43759	43136	42779	99.17%	102.29%
Difference	4060	4912	4521	0.73%	1.19%

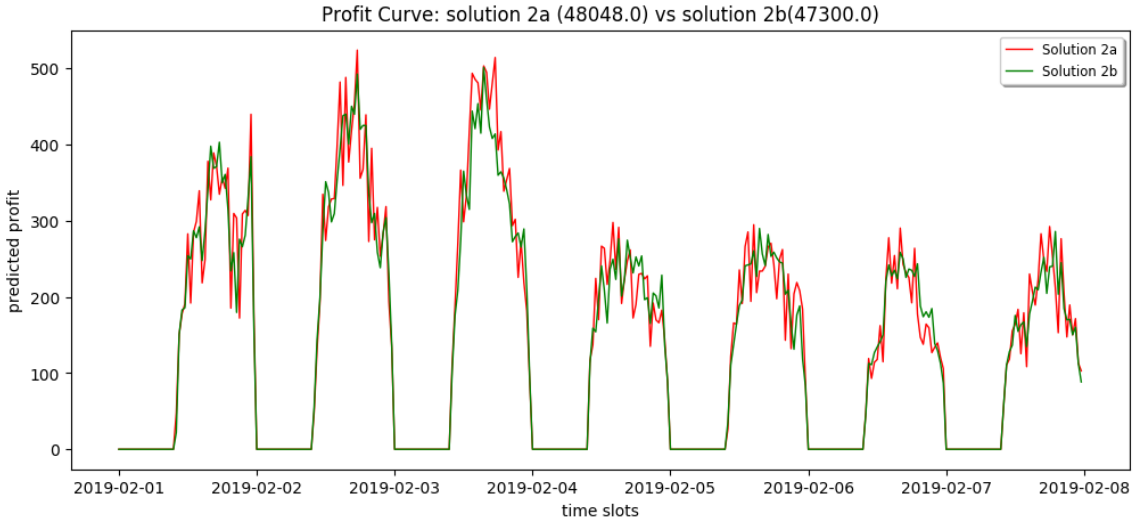


Figure 5.16: Forecasted sales for the prescribed solution in run $R3a$ when evaluated with and without additional error in *footfall in*.

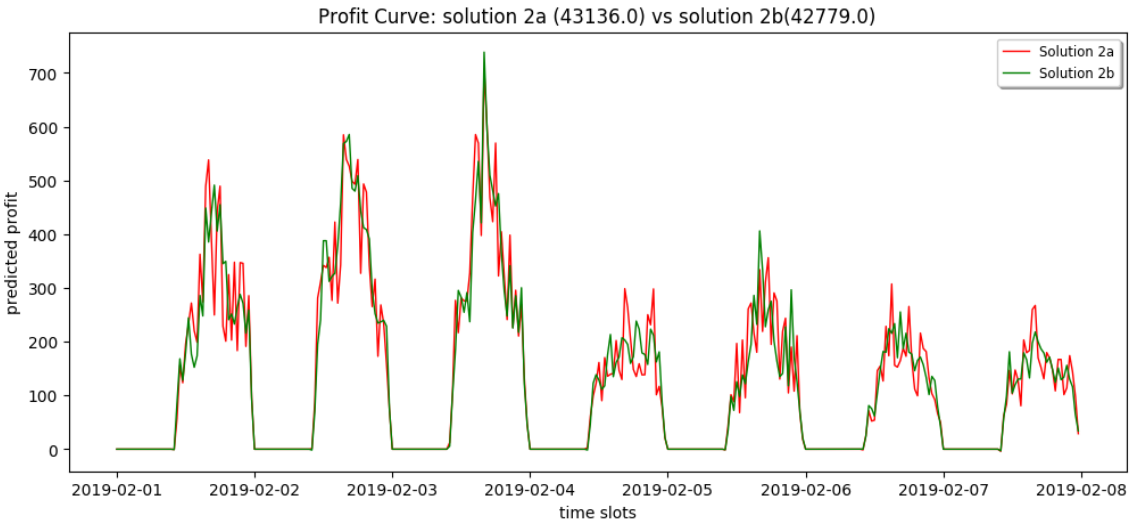


Figure 5.17: Forecasted sales for the prescribed solution in run *R*3*b* when evaluated with and without additional error in *footfall in*.

Figures 5.16 and 5.17 shows the difference between the forecasted sales for the prescribed solution, optimized through error, when evaluated with and without the additional error in footfall. Although the forecasted sales with the error, follow the same weekly trend has the one evaluated without it, the error causes more abrupt and sudden changes in the forecasts. When comparing the two prescribed solution evaluated equally, we can notice that in some parts of the day, the solution optimized with more accurate data forecasts higher sales (figures 5.18 and 5.19).

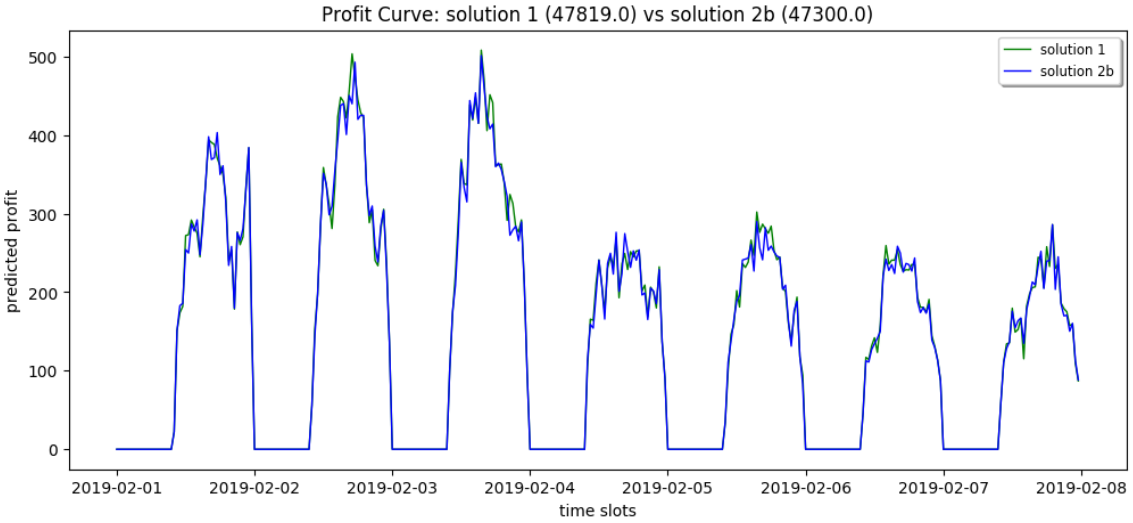


Figure 5.18: Forecasted sales for the prescribed solution in run *R*1*a* and solution in R3b, when evaluated without any additional error in *footfall in*.

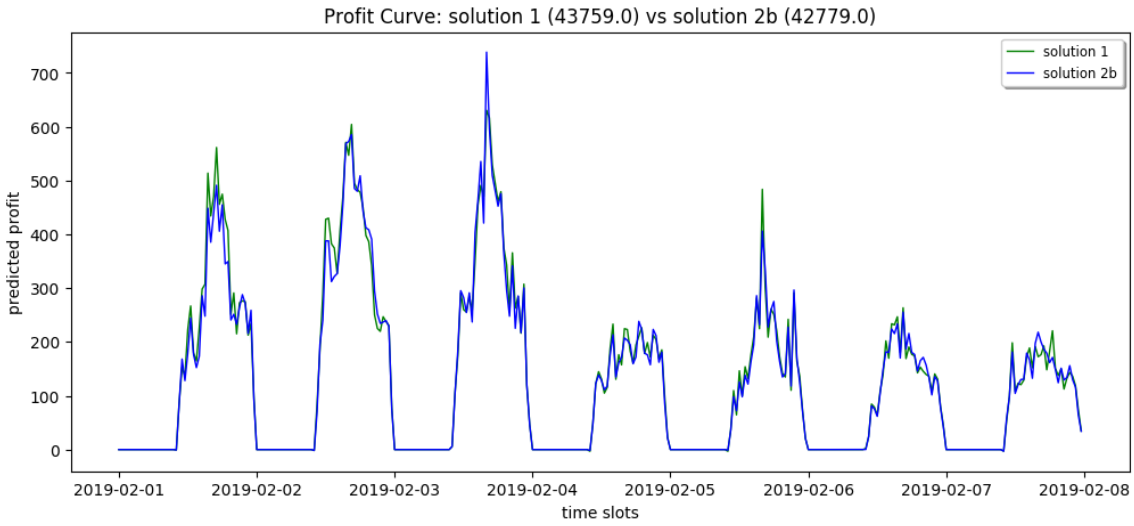


Figure 5.19: Forecasted sales for the prescribed solution in run *R*1b and solution in R3b when evaluated without any additional error in *footfall in*.

5.3.2.4 Experiment O4

"Are the prescribed solutions adapted to the forecasting architecture used for optimization?"

In this experiment, we compare the prescribed solution optimized through different sales forecasting architecture. For this, both solutions are evaluated by the two architectures and the results are compared. In the following figures and tables, each solution is referenced as in table 5.22.

Table 5.22: Solutions descriptions for experiment *O*4.

Solution Name	**Description**
Solution 1	Prescribed solution from run *R*1*a* (direct sales forecasting architecture).
Solution 2	Prescribed solution from run *R*1*b* (indirect sales forecasting architecture).

As presented by table 5.23, when the two solutions are evaluated through different forecasting architectures, the solution with the highest sales is the one that was optimized using the same architecture for evaluation. Therefore, the best solution evaluated through a direct architecture is the solution of run *R*1*a* and for the indirect architecture, it is the solution of run *R*1*b*. This was already expected because solutions are prescribed to maximize sales forecasted using a particular architecture. Thus, an optimized solution is always more adapted to the architecture in which it was evaluated, during optimization, than others.

Table 5.23: Forecasted sales for experiment $O4$.

	Solution 1	**Solution 2**	**S1/S2**
Direct Sales Forecast	47819	46736	102.32%
Indirect Sales Forecast	42110	43759	96.23%
Direct/Indirect	113.56%	106.80%	

There is also a considerable difference between the values forecasted by each architecture. According to the results, the direct architecture forecasts higher sales than the indirect. However, in experiments $O1$ and $O2$, the forecasted sales are compared with the real observed sales for the retailer's schedule, and the total sales were being over-predicted by both architectures. We can say that the indirect architecture would be the safest choice when choosing an architecture for optimization. Withal, according to figure 5.11 in experiment $F4$, the indirect architecture only pays off, for very specific footfall error. If the models in the architectures are improved or more data is used in training, the best architecture to use in optimization might be different.

Figures 5.20 and 5.21 show the forecasted sales curve for both solutions, when evaluated through a direct and indirect architecture, respectively. From the information presented, we can identify several spikes in the forecasts of the indirect architecture, represented by a sudden increase and decrease in sales. Although the direct sales architecture also presents some spikes, they are less abrupt and the forecasts are more stable through time.

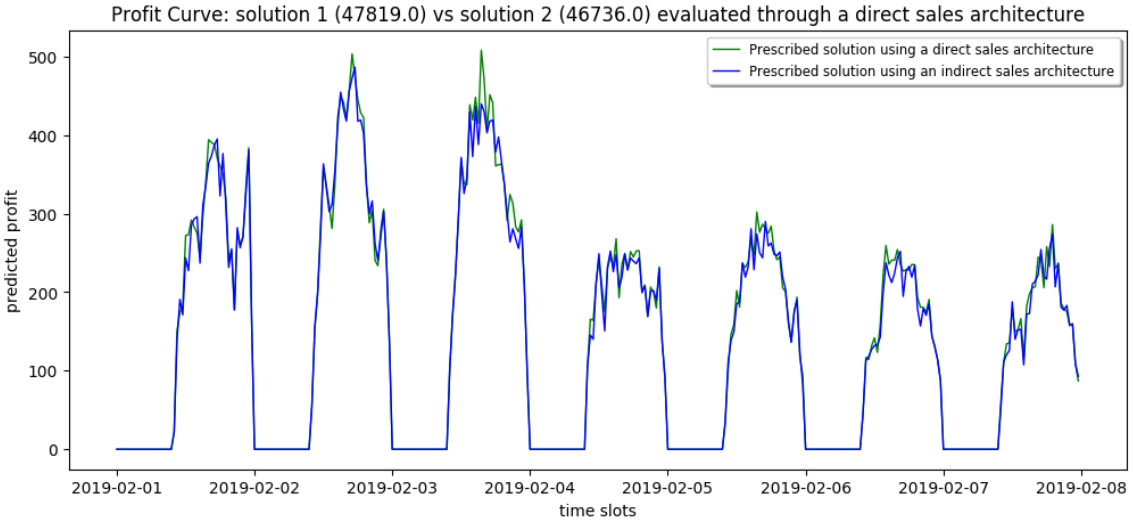


Figure 5.20: Forecasted sales for the prescribed solution in run $R1a$ and solution in $R1b$, when evaluated through a direct sales forecasting architecture.

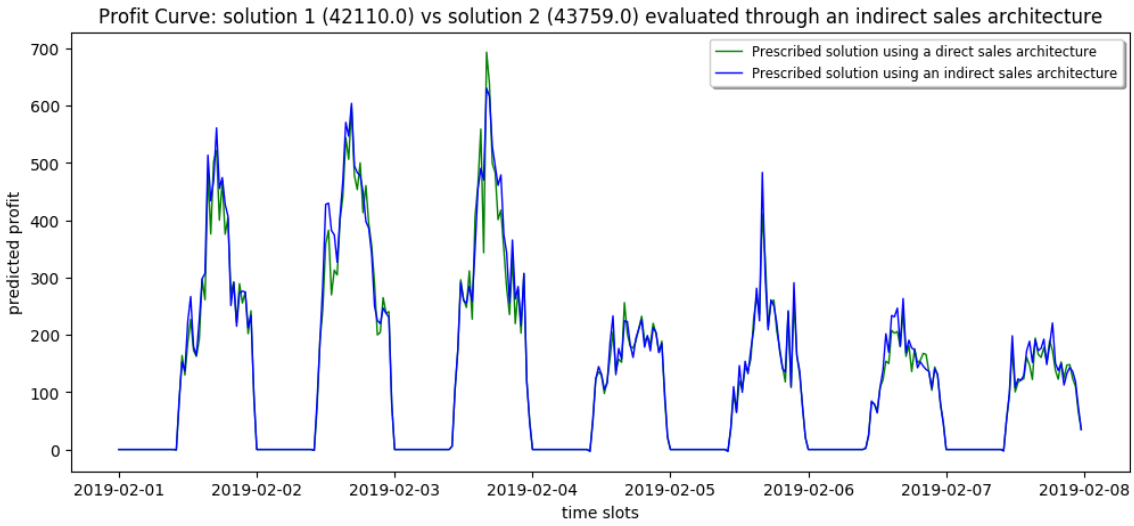


Figure 5.21: Forecasted sales for the prescribed solution in run $R1a$ and solution in $R1b$, when evaluated through an indirect sales forecasting architecture.

5.3.2.5 Distance and Similarity

For every pair of solutions that participates in each experiment and sub-experiment (table 5.7), the distance and similarity between both was calculated according to section 5.2. Table 5.24 shows the values of these measures for the three different levels.

Table 5.24: Distances and similarities between the prescribed solutions involved in the optimization experiments for different levels.

Level	Metric	O1a	O2a	O3a	O4	O1b	O2b	O3b
0	Unit Distance	433	311	273	320	327	293	260
0	Euclidean Distance	36.92	31.38	28.41	29.33	27.95	30.02	24.21
0	Unit Similarity	68.44%	77.33%	80.10%	76.68%	76.17%	78.64%	81.05%
0	Euclidean Similarity	62.33%	67.97%	71.01%	70.08%	71.48%	69.37%	75.30%
1	Unit Distance	568	375	433	478	500	476	490
1	Euclidean Distance	37.26	30.25	33.78	34.21	33.41	35.13	35.52
1	Unit Similarity	58.60%	72.67%	68.44%	65.16%	63.56%	65.31%	64.29%
1	Euclidean Similarity	61.98%	69.13%	65.53%	65.10%	65.91%	64.15%	63.75%
2	Unit Distance	649	437	499	558	591	565	570
2	Euclidean Distance	38.74	31.67	35.06	36.00	35.17	37.05	37.28
2	Unit Similarity	52.70%	68.15%	63.63%	59.33%	56.92%	58.82%	58.45%
2	Euclidean Similarity	60.47%	67.68%	64.23%	63.27%	64.11%	62.19%	61.96%

As expected, in the majority of the cases, the distance between two solutions increases as the level increases. This happens because the higher the level the higher the detail in which the algorithm calculates the distance. However, there might be some exceptions such as the Euclidean distance measured for *O2a* along the three levels. The same case is applied for the similarity but instead of increasing with the level it decreases.

Regarding experiment *O1*, we can observe that, for every type of distance and level, the distance between the prescribed solution and the retailer's, in the sub-experiment *O1a* is clearly

higher that the others experiments. The same applies to the similarity, which is also lower. This means that the solutions prescribed in different runs, although with different configurations, are still closer to each other, than the retailer's solution.

Chapter 6

Conclusion

In this project we developed a prescriptive solution for staff scheduling in retail in order to maximize the profit of stores. The developed solution combines a genetic algorithm with one of two possible sales forecasting architectures, for solution evaluation. The optimization algorithm is based on shift definition and task allocation, where a solution contains information about the boundaries of the shifts created and the number of allocated employees, for each task on each time interval. During optimization, a score is given to each solution based on the sum of the forecasted sales and the total cost of the allocated employees. This enforces the optimization goal leading to solutions that maximize the sales but also avoid over-staffing by associating a cost to the allocated employees. To forecast sales, both architectures require information about time, events and the employees schedules. First, the *footfall in* and *footfall out* is predicted, using two different models, and then the sales are forecasted through one or more models. The first architecture uses one model to predict sales (direct) and the other uses three models with different targets (indirect). In the latter architecture the sales are obtained by multiplying the different targets, which are: the *number of transactions*, *average products per transaction* and *average price per product*.

In order to test the implemented solution, a case study was carried out using real data from a fashion retailer. Results have shown that the prescribed solution could improve sales in 3-9% when compared with the forecasted sales for the solution designed by the retailer. Solutions developed ahead of time also demonstrated to be less efficient than others, optimized closer to the planning horizon, as could be expected. In the conducted experiments, a prescribed solution optimized with no time ahead presented more 1-3% estimated sales than the solution prescribed with one week in advance. Other experiments were also conducted to understand the impact on the error in sales forecasts of varying the error of the forecasted *footfall in*, which is used as input in the former models. Results showed that for an error of 20% in the forecasted footfall in, the error in the final sales would be around 1-2%. As expected and demonstrated by other experiments, the higher the error in footfall in, the higher the error in the forecasted sales.

When analyzing the two possible architectures used in optimization, we observed that the

direct architecture forecasts more 6-14% sales than the indirect architecture. As demonstrated in other experiments, both architectures overpredict the total sales of the planning horizon. This could lead us to the conclusion that the indirect architecture is a better choice since the error is lower. However, when analyzing how different errors in footfall in affects sales predictions, we concluded that the indirect architecture is rarely better than the other. This makes the direct architecture the safest choice.

A particularly interesting observation, during the experiments, showed that poor performance in the models of the indirect sales architecture does not necessarily leads to bad sales forecast. In this architecture the output of three models is combined to calculate the forecasted sales. However, even if the models have low performance, individual errors seem to compensate each other. This leads to forecasted sales with lower error than the direct architecture.

6.1 Future Work

As future work, we suggest some issues that should be addressed in order to improve the current solution. First of all, the reliability of the evaluation should be improved. It may be interesting to apply the developed solution in a real store and compare the sales with another similar store so that some real results can be discussed. The forecasting models should also be improved to, in the end, forecast sales more accurately. This would increase the value of the prescriptive model greatly since it evaluates the solutions based on their application in the real world. Regarding optimization, more constraints should be enforced to fit other working policies of a store, for example, the minimum and maximum task duration. The cost of an employee should also be modeled by a function that translates the real cost, depending on the shift's duration and other factors. It would also be interesting to explore other crossover and mutation strategies and see how different parameters of the optimization algorithm affects the prescribed solution. As the last step, take advantage of the prescribed solution by the model and deal with shift employee assignment and task assignment to shifts.

6.2 Contribution

Research in this type of data analytics, prescriptive analytics, is very recent and not much work can be found. Concerning retail, even less related projects could be found. For this reason, we contribute with a prescriptive solution based on a genetic algorithm and two suggested forecasting sales architectures. For the optimization algorithm, new structures were introduced for the representation of an individual, based on shift definition and task allocation. The crossover and mutation operations were also fully adapted to guarantee the feasibility of the solutions, introducing new strategies for these operations.

6.3 Final Conclusions

With the conclusion of this dissertation, we can understand the power and promising results of prescriptive analytics. However, the efficiency is highly dependent on the quality of the predictive models, since they are used to estimate the impact of the solutions in the real world. If the behavior is not well modeled, then the optimization may be misleadingly guided and the prescribed solutions may not have the estimated impact. In retail, sales are very hard to model for short time intervals. Many factors affect sales and choosing the best can be tricky. Some are not even available at the moment of forecast and models are used to predict their values, propagating the error to the next models. For these reasons, we conclude that data quality and quantity is the key ingredient to enable the full potential of a prescriptive solution, since they help building more accurate models.

References

[1] N. A. E. Ata and M. J. Perks, "Solving the dynamic complexity dilemma predictive and prescriptive business management: Answering the need for a new paradigm", *Solving the Dynamic Complexity Dilemma: Predictive and Prescriptive Business Management: Answering the Need for a New Paradigm*, vol. 9783642543, pp. 1–281, 2014. DOI: 10.1007/978-3-642-54310-4.

[2] J. B. Barney and P. M. Wright, "On becoming a strategic partner: The role of human resources in gaining competitive advantage", 37(1), 31–46, 1998. DOI: 10.1002/(sici)1099-050x(199821)37:1<31::aid-hrm4>3.0.co;2-w.

[3] B. Light, "Business Analytics: Research and Teaching Perspectives", in *Proceedings of the ITI 2013 35th International Conference on INFORMATION TECHNOLOGY INTERFACES*, 2013, ISBN: 978-953-7138-30-1. DOI: 10.2498/iti.2013.0589.

[4] S. Ransbotham, D. Kiron, and P. K. Prentice, "Minding the Analytics Gap", *MIT Sloan Management Review*, vol. 56, no. Spring, 2015, pp. 63–68, 2015, ISSN: 1532-9194. DOI: 10.1098/rspa.2011.0114.

[5] S. Strohmeier and F. Piazza, "Domain driven data mining in human resource management: A review of current research Keywords: Data mining HRM Literature review Domain driven data mining HR data mining Electronic human resource management (e-HRM)", *Expert Systems With Applications*, vol. 40, pp. 2410–2420, 2013. DOI: 10.1016/j.eswa.2012.10.059.

[6] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*. Prentice Hall, 2002, ISBN: 9780130281388.

[7] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979, ISBN: 0716710447.

[8] A. Ernst, H. Jiang, M. Krishnamoorthy, B. Owens, and D. Sier, "An annotated bibliography of personnel scheduling and rostering", English, *Annals of Operations Research*, vol. 127, no. 1-4, pp. 21–144, 2004, ISSN: 0254-5330.

[9] A. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sie, "Staff scheduling and rostering: A review of applications, methods and models", *European Journal of Operational Research*, vol. 153, no. 1, pp. 3–27, Feb. 2004, ISSN: 03772217. DOI: 10.1016/S0377-2217(03)00095-X.

[10] J. V. den Bergh, J. Beliën, P. D. Bruecker, E. Demeulemeester, and L. D. Boeck, "Personnel scheduling: A literature review.", *European Journal of Operational Research*, vol. 226, no. 3, pp. 367–385, 2013.

[11] P. Juneja, *Understanding retail - what is retail ?*, https://www.managementstudyguide.com/what-is-retail.htm, (Accessed on 20/06/2019).

[12] R. N. Burns, R. Narasimhan, and L. D. Smith, "A set-processing algorithm for scheduling staff on 4-day or 3-day work weeks", *Naval Research Logistics*, vol. 45, no. 8, pp. 839–853, 1998, ISSN: 0894069X. DOI: 10.1002/(SICI)1520-6750(199812)45:8<839::AID-NAV5>3.0.CO;2-R.

[13] S. Lam, M. Vandenbosch, and M. Pearce, "Retail sales force scheduling based on store traffic forecasting", *Journal of Retailing*, vol. 74, no. 1, pp. 61–88, 1998, ISSN: 00224359. DOI: 10.1016/S0022-4359(99)80088-8.

[14] S. K. Mirrazavi and H. Beringer, "A web-based workforce management system for Sainsburys Supermarkets Ltd", *Annals of Operations Research*, vol. 155, no. 1, pp. 437–457, Aug. 2007, ISSN: 0254-5330. DOI: 10.1007/s10479-007-0204-2.

[15] Ö. Kabak, F. Ülengin, E. Aktaş, Ş. Önsel, and Y. I. Topcu, "Efficient shift scheduling in the retail sector through two-stage optimization", *European Journal of Operational Research*, vol. 184, no. 1, pp. 76–90, 2008, ISSN: 03772217. DOI: 10.1016/j.ejor.2006.10.039.

[16] R. Pastor and J. Olivella, "Selecting and adapting weekly work schedules with working time accounts: A case of a retail clothing chain", *European Journal of Operational Research*, vol. 210, no. 3, pp. 467–473, 2008, ISSN: 03772217. DOI: 10.1016/j.ejor.2006.10.028.

[17] E. Melachrinoudis and H. Min, "The hybrid queuing and bi-objective integer programming model for scheduling frontline employees in a retail organisation", *International Journal of Services Technology and Management*, vol. 9, no. 1, p. 33, 2008, ISSN: 1460-6720. DOI: 10.1504/IJSTM.2008.016810.

[18] V. Nissen and M. Günther, "Automatic generation of optimised working time models in personnel planning", in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6234 LNCS, 2010, pp. 384–391, ISBN: 3642154603. DOI: 10.1007/978-3-642-15461-4_35.

[19] S. Zolfaghari, V. Quan, A. El-Bouri, and M. Khashayardoust, "Application of a Genetic Algorithm to staff scheduling in retail sector", *Int. J. Industrial and Systems Engineering*, vol. 5, no. 1, pp. 20–47, 2010. DOI: 10.1504/IJISE.2010.029755.

[20] K. Miwa and S. Takakuwa, "Optimization and analysis of staffing problems at a retail store", in *Proceedings - Winter Simulation Conference*, 2010, pp. 1911–1923, ISBN: 9781424498666. DOI: 10.1109/WSC.2010.5678879. arXiv: arXiv:1011.1669v3.

[21] N. Chapados, M. Joliveau, and L. M. Rousseau, "Retail store workforce scheduling by expected operating income maximization", in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer, Berlin, Heidelberg, 2011, pp. 53–58, ISBN: 9783642213106. DOI: 10.1007/978-3-642-21311-3_7.

[22] V. Nissen, M. Günther, and R. Schumann, "Integrated generation of working time models and staff schedules in workforce management", in *Applications of Evolutionary Computation*, C. Di Chio, A. Brabazon, G. A. Di Caro, R. Drechsler, M. Farooq, J. Grahl, G. Greenfield, C. Prins, J. Romero, G. Squillero, E. Tarantino, A. G. B. Tettamanzi, N. Urquhart, and A. Ş. Uyar, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 491–500, ISBN: 978-3-642-20520-0.

[23] N. Chapados, M. Joliveau, P. L'Ecuyer, and L. M. Rousseau, "Retail store scheduling for profit", *European Journal of Operational Research*, vol. 239, no. 3, pp. 609–624, 2014, ISSN: 03772217. DOI: 10.1016/j.ejor.2014.05.033.

[24] M. Günther and V. Nissen, "A comparison of three heuristics on a practical case of sub-daily staff scheduling", *Annals of Operations Research*, vol. 218, no. 1, pp. 201–219, 2014, ISSN: 15729338. DOI: 10.1007/s10479-012-1259-2.

[25] D. Lin, Y. Tong, G. Niu, Y. Xue, X. Shi, C. Ren, and Z. Zhang, "Scheduling workforce for retail stores with employee preferences", in *10th IEEE Int. Conf. on Service Operations and Logistics, and Informatics, SOLI 2015 - In conjunction with ICT4ALL 2015*, 2015, ISBN: 9781467384803. DOI: 10.1109/SOLI.2015.7367407.

[26] C. A. Henao, J. C. Munoz, and J. C. Ferrer, "The impact of multi-skilling on personnel scheduling in the service sector: A retail industry case", *Journal of the Operational Research Society*, vol. 66, no. 12, pp. 1949–1959, 2015, ISSN: 14769360. DOI: 10.1057/jors.2015.9.

[27] A. Parisio and C. Neil Jones, "A two-stage stochastic programming approach to employee scheduling in retail outlets with uncertain demand", *Omega (United Kingdom)*, vol. 53, pp. 97–103, 2015, ISSN: 03050483. DOI: 10.1016/j.omega.2015.01.003.

[28] L. Talarico and P. A. Maya Duque, "An optimization algorithm for the workforce management in a retail chain", *Computers and Industrial Engineering*, vol. 82, pp. 65–77, 2015, ISSN: 03608352. DOI: 10.1016/j.cie.2015.01.014.

[29] M. Defraeye and I. Van Nieuwenhuyse, "A branch-and-bound algorithm for shift scheduling with stochastic nonstationary demand", *Computers and Operations Research*, vol. 65, pp. 149–162, 2016, ISSN: 03050548. DOI: 10.1016/j.cor.2015.06.016.

[30] R. Cuevas, J. C. Ferrer, M. Klapp, and J. C. Muñoz, "A mixed integer programming approach to multi-skilled workforce scheduling", *Journal of Scheduling*, vol. 19, no. 1, pp. 91–106, 2016, ISSN: 10946136. DOI: 10.1007/s10951-015-0450-0.

[31] M. Mac-Vicar, J. C. Ferrer, J. C. Muñoz, and C. A. Henao, "Real-time recovering strategies on personnel scheduling in the retail industry", *Computers and Industrial Engineering*, vol. 113, pp. 589–601, 2017, ISSN: 03608352. DOI: 10.1016/j.cie.2017.09.045.

[32] R. Bürgy, H. Michon-Lacaze, and G. Desaulniers, *Employee scheduling with short demand perturbations and extensible shifts*, 2018. DOI: 10.1016/j.omega.2018.10.009.

[33] N. Xue, D. Landa-Silva, I. Triguero, and G. P. Figueredo, "A Genetic Algorithm with Composite Chromosome for Shift Assignment of Part-time Employees", in *2018 IEEE Congress on Evolutionary Computation, CEC 2018 - Proceedings*, 2018, ISBN: 9781509060177. DOI: 10.1109/CEC.2018.8477818.

[34] D. J. Hand, P. Smyth, and H. Mannila, *Principles of Data Mining*. Cambridge, MA, USA: MIT Press, 2001, ISBN: 0-262-08290-X, 9780262082907.

[35] V. N. Gudivada, "Data Analytics", in *Data Analytics for Intelligent Transportation Systems*, vol. 147, Elsevier, 2017, pp. 31–67, ISBN: 9780128098516. DOI: 10.1016/B978-0-12-809715-1.00002-X.

[36] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989, ISBN: 0201157675.

[37] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: Experiences from the scikit-learn project", in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[38] *3.3. model evaluation: Quantifying the quality of predictions — scikit-learn 0.21.3 documentation*, https://scikit-learn.org/stable/modules/model_evaluation.html#regression-metrics, (Accessed on 20/06/2019).

Appendix A

Data

A.1 Examples

A.1.1 Raw Data

Table A.1: Raw events example for one year.

location	start	end	type
Store_1	20180101	20180101	CALENDAR-PUBLIC
Store_1	20180213	20180213	CALENDAR-PUBLIC
Store_1	20180214	20180214	EVENT-PUBLIC
Store_1	20180308	20180308	EVENT-PUBLIC
Store_1	20180319	20180319	EVENT-PUBLIC
Store_1	20180330	20180330	CALENDAR-PUBLIC
Store_1	20180401	20180401	CALENDAR-PUBLIC
Store_1	20180425	20180425	CALENDAR-PUBLIC
Store_1	20180501	20180501	CALENDAR-PUBLIC
Store_1	20180506	20180506	EVENT-PUBLIC
Store_1	20180531	20180531	CALENDAR-PUBLIC
Store_1	20180601	20180601	EVENT-PUBLIC
Store_1	20180610	20180610	CALENDAR-PUBLIC
Store_1	20180624	20180624	CALENDAR-LOCAL
Store_1	20180715	20180915	PROMOTION-LOCAL
Store_1	20180815	20180815	CALENDAR-PUBLIC
Store_1	20181005	20181005	CALENDAR-PUBLIC
Store_1	20181101	20181101	CALENDAR-PUBLIC
Store_1	20181123	20181123	PROMOTION-LOCAL
Store_1	20181201	20181201	CALENDAR-PUBLIC
Store_1	20181208	20181208	CALENDAR-PUBLIC
Store_1	20181225	20181225	CALENDAR-PUBLIC
Store_1	20181226	20181226	EVENT-PUBLIC
Store_1	20181228	20190228	PROMOTION-LOCAL
Store_1	20181231	20181231	EVENT-PUBLIC

Table A.2: Raw footfall example for one day.

location	date	time	footfall in	footfall out

Store_1	20190110	90000	6	0
Store_1	20190110	100000	5.25	0
Store_1	20190110	110000	24.2591	0
Store_1	20190110	120000	27.2433	0
Store_1	20190110	130000	19.453	0
Store_1	20190110	140000	21.5464	0
Store_1	20190110	150000	33.3766	0
Store_1	20190110	160000	49.6265	0
Store_1	20190110	170000	36.614	0
Store_1	20190110	180000	27.6293	0
Store_1	20190110	190000	20.2175	0
Store_1	20190110	200000	39.2	0
Store_1	20190110	210000	44.7406	0
Store_1	20190110	220000	38.1524	0
Store_1	20190110	230000	14	0

Table A.3: Raw schedule example for one day.

location	employee code	date	start	end	shift type
Store_1	Employee_4	2019-01-10	12:00:00	16:00:00	SALES
Store_1	Employee_4	2019-01-10	16:00:00	17:00:00	BREAK
Store_1	Employee_4	2019-01-10	17:00:00	21:00:00	SALES
Store_1	Employee_6	2019-01-10	10:00:00	15:00:00	SALES
Store_1	Employee_12	2019-01-10	10:00:00	00:00:00	BREAK
Store_1	Employee_19	2019-01-10	10:00:00	14:00:00	SALES
Store_1	Employee_19	2019-01-10	14:00:00	15:00:00	BREAK
Store_1	Employee_19	2019-01-10	15:00:00	19:00:00	SALES
Store_1	Employee_26	2019-01-10	15:00:00	17:00:00	SALES
Store_1	Employee_26	2019-01-10	17:00:00	18:00:00	BREAK
Store_1	Employee_26	2019-01-10	18:00:00	00:00:00	SALES
Store_1	Employee_31	2019-01-10	10:00:00	00:00:00	OTHERS
Store_1	Employee_38	2019-01-10	10:00:00	00:00:00	BREAK
Store_1	Employee_39	2019-01-10	10:00:00	15:00:00	SALES
Store_1	Employee_39	2019-01-10	15:00:00	16:00:00	BREAK
Store_1	Employee_39	2019-01-10	16:00:00	19:00:00	SALES
Store_1	Employee_47	2019-01-10	10:00:00	00:00:00	OTHERS
Store_1	Employee_48	2019-01-10	15:00:00	18:00:00	SALES
Store_1	Employee_48	2019-01-10	18:00:00	19:00:00	BREAK
Store_1	Employee_48	2019-01-10	19:00:00	00:00:00	SALES
Store_1	Employee_51	2019-01-10	19:00:00	00:00:00	SALES

Table A.4: Raw transactions example for one day.

location	product	ticket code	date	hour	employee code	qty	net sales
Store_1	Product_24165	ticket_39161	20190110	105540	Employee_39	1	66
Store_1	Product_6036	ticket_21632	20190110	105838	Employee_6	-1	-100
Store_1	Product_25257	ticket_41413	20190110	105948	Employee_6	1	120
Store_1	Product_6036	ticket_57106	20190110	111508	Employee_39	-1	-100
Store_1	Product_23636	ticket_17413	20190110	111638	Employee_39	1	110
Store_1	Product_20960	ticket_24935	20190110	112452	Employee_6	1	66
Store_1	Product_11772	ticket_33527	20190110	115210	Employee_39	1	35

Store_1	Product_11776	ticket_33527	20190110	115210	Employee_39	1	63.2
Store_1	Product_17015	ticket_33527	20190110	115210	Employee_39	1	79
Store_1	Product_19276	ticket_33527	20190110	115210	Employee_39	1	52
Store_1	Product_21278	ticket_33527	20190110	115210	Employee_39	1	89
Store_1	Product_22267	ticket_33527	20190110	115210	Employee_39	1	63.2
Store_1	Product_25671	ticket_33527	20190110	115210	Employee_39	1	110
Store_1	Product_23520	ticket_21373	20190110	115621	Employee_6	1	25
Store_1	Product_24230	ticket_21373	20190110	115621	Employee_6	1	69.3
Store_1	Product_9158	ticket_61144	20190110	122000	Employee_6	1	79
Store_1	Product_6036	ticket_53966	20190110	122234	Employee_4	1	100
Store_1	Product_25388	ticket_58403	20190110	123552	Employee_6	-1	-99
Store_1	Product_25399	ticket_54420	20190110	123636	Employee_6	1	99
Store_1	Product_15588	ticket_9009	20190110	123911	Employee_39	1	55
Store_1	Product_24193	ticket_9601	20190110	131115	Employee_39	1	14.5
Store_1	Product_25080	ticket_24949	20190110	131205	Employee_4	-1	-45
Store_1	Product_25054	ticket_55006	20190110	131304	Employee_4	1	45
Store_1	Product_25154	ticket_42028	20190110	131507	Employee_4	1	17.4
Store_1	Product_12665	ticket_27567	20190110	134311	Employee_39	1	99
Store_1	Product_10488	ticket_54853	20190110	135524	Employee_39	1	69.3
Store_1	Product_23566	ticket_4260	20190110	141349	Employee_6	1	59.4
Store_1	Product_14742	ticket_40628	20190110	143201	Employee_4	1	19.99
Store_1	Product_18269	ticket_29716	20190110	143337	Employee_6	1	20
Store_1	Product_23461	ticket_16368	20190110	153037	Employee_4	1	52
Store_1	Product_25151	ticket_16368	20190110	153037	Employee_4	1	17.4
Store_1	Product_24493	ticket_19694	20190110	153907	Employee_26	1	48.3
Store_1	Product_25060	ticket_39849	20190110	155017	Employee_26	1	56
Store_1	Product_23601	ticket_10029	20190110	155428	Employee_26	1	59.4
Store_1	Product_24940	ticket_10029	20190110	155428	Employee_26	1	84
Store_1	Product_16694	ticket_30857	20190110	161149	Employee_26	1	88
Store_1	Product_23461	ticket_57441	20190110	163953	Employee_38	-1	-52
Store_1	Product_23461	ticket_23473	20190110	164018	Employee_38	1	52
Store_1	Product_24613	ticket_37745	20190110	164009	Employee_4	1	66
Store_1	Product_25717	ticket_54064	20190110	170823	Employee_26	1	25
Store_1	Product_6035	ticket_48883	20190110	171335	Employee_39	-1	-10
Store_1	Product_6036	ticket_48883	20190110	171335	Employee_39	-1	-100
Store_1	Product_6037	ticket_48883	20190110	171335	Employee_39	-1	-20
Store_1	Product_23281	ticket_49062	20190110	171406	Employee_39	1	99
Store_1	Product_24071	ticket_49062	20190110	171406	Employee_39	1	78
Store_1	Product_22952	ticket_47934	20190110	173207	Employee_26	1	77
Store_1	Product_23427	ticket_62270	20190110	174602	Employee_26	1	110
Store_1	Product_24930	ticket_23538	20190110	175330	Employee_26	1	84
Store_1	Product_25721	ticket_23538	20190110	175330	Employee_26	1	25
Store_1	Product_6036	ticket_32865	20190110	175438	Employee_4	-1	-100
Store_1	Product_17211	ticket_64100	20190110	175458	Employee_4	1	280
Store_1	Product_21393	ticket_51140	20190110	181346	Employee_48	-1	-47.2
Store_1	Product_6041	ticket_36588	20190110	181440	Employee_48	1	50
Store_1	Product_18270	ticket_18779	20190110	181920	Employee_39	1	20
Store_1	Product_24193	ticket_18779	20190110	181920	Employee_39	1	14.5
Store_1	Product_25153	ticket_18779	20190110	181920	Employee_39	1	17.4
Store_1	Product_11788	ticket_30112	20190110	185315	Employee_39	-1	-20
Store_1	Product_17769	ticket_23514	20190110	185403	Employee_39	1	29
Store_1	Product_24091	ticket_32086	20190110	190034	Employee_39	1	21.99
Store_1	Product_19218	ticket_32858	20190110	190354	Employee_51	1	35
Store_1	Product_23325	ticket_60596	20190110	194005	Employee_26	1	23.2
Store_1	Product_20961	ticket_60633	20190110	195527	Employee_4	1	66
Store_1	Product_20043	ticket_39843	20190110	201323	Employee_51	1	77
Store_1	Product_24834	ticket_57959	20190110	202326	Employee_26	1	80

Store_1	Product_22515	ticket_61222	20190110	203208	Employee_4	1	23.2
Store_1	Product_22609	ticket_61222	20190110	203208	Employee_4	1	23.2
Store_1	Product_20195	ticket_21213	20190110	203521	Employee_51	-1	-110
Store_1	Product_20022	ticket_54879	20190110	203544	Employee_51	1	110
Store_1	Product_23271	ticket_54879	20190110	203544	Employee_51	1	110
Store_1	Product_146	ticket_17011	20190110	205701	Employee_26	-1	-89
Store_1	Product_16725	ticket_65546	20190110	205731	Employee_26	1	99
Store_1	Product_16694	ticket_23480	20190110	212237	Employee_12	-1	-110
Store_1	Product_16695	ticket_60241	20190110	212415	Employee_12	1	110
Store_1	Product_25510	ticket_45380	20190110	212856	Employee_38	1	130
Store_1	Product_24435	ticket_21301	20190110	213132	Employee_51	1	56
Store_1	Product_22868	ticket_50755	20190110	213251	Employee_26	1	66
Store_1	Product_8966	ticket_16934	20190110	215146	Employee_26	1	20
Store_1	Product_11790	ticket_19153	20190110	220903	Employee_47	-1	-20
Store_1	Product_11786	ticket_28482	20190110	220949	Employee_47	1	20
Store_1	Product_24419	ticket_40017	20190110	222407	Employee_51	1	27.3
Store_1	Product_24928	ticket_45413	20190110	230737	Employee_26	1	84
Store_1	Product_25186	ticket_21479	20190110	234023	Employee_26	1	140
Store_1	Product_23466	ticket_30103	20190110	234815	Employee_51	1	25
Store_1	Product_23467	ticket_30103	20190110	234815	Employee_51	1	25

A.1.2 Prepared Data

Table A.5: Prepared events example for one day.

location	datetime	event_calendar-local	event_calendar-public	event_event-public	event_promotion-local
Store_1	2019-01-10 00:00:00	0	0	0	1
Store_1	2019-01-10 00:30:00	0	0	0	1
Store_1	2019-01-10 01:00:00	0	0	0	1
Store_1	2019-01-10 01:30:00	0	0	0	1
Store_1	2019-01-10 02:00:00	0	0	0	1
Store_1	2019-01-10 02:30:00	0	0	0	1
Store_1	2019-01-10 03:00:00	0	0	0	1
Store_1	2019-01-10 03:30:00	0	0	0	1
Store_1	2019-01-10 04:00:00	0	0	0	1
Store_1	2019-01-10 04:30:00	0	0	0	1
Store_1	2019-01-10 05:00:00	0	0	0	1
Store_1	2019-01-10 05:30:00	0	0	0	1
Store_1	2019-01-10 06:00:00	0	0	0	1
Store_1	2019-01-10 06:30:00	0	0	0	1
Store_1	2019-01-10 07:00:00	0	0	0	1
Store_1	2019-01-10 07:30:00	0	0	0	1
Store_1	2019-01-10 08:00:00	0	0	0	1
Store_1	2019-01-10 08:30:00	0	0	0	1
Store_1	2019-01-10 09:00:00	0	0	0	1
Store_1	2019-01-10 09:30:00	0	0	0	1
Store_1	2019-01-10 10:00:00	0	0	0	1
Store_1	2019-01-10 10:30:00	0	0	0	1
Store_1	2019-01-10 11:00:00	0	0	0	1
Store_1	2019-01-10 11:30:00	0	0	0	1
Store_1	2019-01-10 12:00:00	0	0	0	1
Store_1	2019-01-10 12:30:00	0	0	0	1
Store_1	2019-01-10 13:00:00	0	0	0	1
Store_1	2019-01-10 13:30:00	0	0	0	1

Store_1	2019-01-10 14:00:00	0	0	0	1
Store_1	2019-01-10 14:30:00	0	0	0	1
Store_1	2019-01-10 15:00:00	0	0	0	1
Store_1	2019-01-10 15:30:00	0	0	0	1
Store_1	2019-01-10 16:00:00	0	0	0	1
Store_1	2019-01-10 16:30:00	0	0	0	1
Store_1	2019-01-10 17:00:00	0	0	0	1
Store_1	2019-01-10 17:30:00	0	0	0	1
Store_1	2019-01-10 18:00:00	0	0	0	1
Store_1	2019-01-10 18:30:00	0	0	0	1
Store_1	2019-01-10 19:00:00	0	0	0	1
Store_1	2019-01-10 19:30:00	0	0	0	1
Store_1	2019-01-10 20:00:00	0	0	0	1
Store_1	2019-01-10 20:30:00	0	0	0	1
Store_1	2019-01-10 21:00:00	0	0	0	1
Store_1	2019-01-10 21:30:00	0	0	0	1
Store_1	2019-01-10 22:00:00	0	0	0	1
Store_1	2019-01-10 22:30:00	0	0	0	1
Store_1	2019-01-10 23:00:00	0	0	0	1
Store_1	2019-01-10 23:30:00	0	0	0	1

Table A.6: Prepared footfalls example for one day.

location	datetime	footfall_in	footfall_out
Store_1	2019-01-10 00:00:00	0.0	0.0
Store_1	2019-01-10 00:30:00	0.0	0.0
Store_1	2019-01-10 01:00:00	0.0	0.0
Store_1	2019-01-10 01:30:00	0.0	0.0
Store_1	2019-01-10 02:00:00	0.0	0.0
Store_1	2019-01-10 02:30:00	0.0	0.0
Store_1	2019-01-10 03:00:00	0.0	0.0
Store_1	2019-01-10 03:30:00	0.0	0.0
Store_1	2019-01-10 04:00:00	0.0	0.0
Store_1	2019-01-10 04:30:00	0.0	0.0
Store_1	2019-01-10 05:00:00	0.0	0.0
Store_1	2019-01-10 05:30:00	0.0	0.0
Store_1	2019-01-10 06:00:00	0.0	0.0
Store_1	2019-01-10 06:30:00	0.0	0.0
Store_1	2019-01-10 07:00:00	0.0	0.0
Store_1	2019-01-10 07:30:00	0.0	0.0
Store_1	2019-01-10 08:00:00	0.0	0.0
Store_1	2019-01-10 08:30:00	0.0	0.0
Store_1	2019-01-10 09:00:00	3.0	0.0
Store_1	2019-01-10 09:30:00	3.0	0.0
Store_1	2019-01-10 10:00:00	2.625	0.0
Store_1	2019-01-10 10:30:00	2.625	0.0
Store_1	2019-01-10 11:00:00	12.1295	0.0
Store_1	2019-01-10 11:30:00	12.1295	0.0
Store_1	2019-01-10 12:00:00	13.6216	0.0
Store_1	2019-01-10 12:30:00	13.6216	0.0
Store_1	2019-01-10 13:00:00	9.7265	0.0
Store_1	2019-01-10 13:30:00	9.7265	0.0
Store_1	2019-01-10 14:00:00	10.7732	0.0
Store_1	2019-01-10 14:30:00	10.7732	0.0
Store_1	2019-01-10 15:00:00	16.6883	0.0

Store_1	2019-01-10 15:30:00	16.6883	0.0
Store_1	2019-01-10 16:00:00	24.8132	0.0
Store_1	2019-01-10 16:30:00	24.8132	0.0
Store_1	2019-01-10 17:00:00	18.3070	0.0
Store_1	2019-01-10 17:30:00	18.3070	0.0
Store_1	2019-01-10 18:00:00	13.8146	0.0
Store_1	2019-01-10 18:30:00	13.8146	0.0
Store_1	2019-01-10 19:00:00	10.1087	0.0
Store_1	2019-01-10 19:30:00	10.1087	0.0
Store_1	2019-01-10 20:00:00	19.6	0.0
Store_1	2019-01-10 20:30:00	19.6	0.0
Store_1	2019-01-10 21:00:00	22.3703	0.0
Store_1	2019-01-10 21:30:00	22.3703	0.0
Store_1	2019-01-10 22:00:00	19.0762	0.0
Store_1	2019-01-10 22:30:00	19.0762	0.0
Store_1	2019-01-10 23:00:00	14.0	0.0
Store_1	2019-01-10 23:30:00	0.0	0.0

Table A.7: Prepared schedules example for one day.

location	datetime	total_employees	n_at_sales	n_at_others	n_at_break
Store_1	2019-01-10 00:00:00	0.0	0.0	0.0	0.0
Store_1	2019-01-10 00:30:00	0.0	0.0	0.0	0.0
Store_1	2019-01-10 01:00:00	0.0	0.0	0.0	0.0
Store_1	2019-01-10 01:30:00	0.0	0.0	0.0	0.0
Store_1	2019-01-10 02:00:00	0.0	0.0	0.0	0.0
Store_1	2019-01-10 02:30:00	0.0	0.0	0.0	0.0
Store_1	2019-01-10 03:00:00	0.0	0.0	0.0	0.0
Store_1	2019-01-10 03:30:00	0.0	0.0	0.0	0.0
Store_1	2019-01-10 04:00:00	0.0	0.0	0.0	0.0
Store_1	2019-01-10 04:30:00	0.0	0.0	0.0	0.0
Store_1	2019-01-10 05:00:00	0.0	0.0	0.0	0.0
Store_1	2019-01-10 05:30:00	0.0	0.0	0.0	0.0
Store_1	2019-01-10 06:00:00	0.0	0.0	0.0	0.0
Store_1	2019-01-10 06:30:00	0.0	0.0	0.0	0.0
Store_1	2019-01-10 07:00:00	0.0	0.0	0.0	0.0
Store_1	2019-01-10 07:30:00	0.0	0.0	0.0	0.0
Store_1	2019-01-10 08:00:00	0.0	0.0	0.0	0.0
Store_1	2019-01-10 08:30:00	0.0	0.0	0.0	0.0
Store_1	2019-01-10 09:00:00	0.0	0.0	0.0	0.0
Store_1	2019-01-10 09:30:00	0.0	0.0	0.0	0.0
Store_1	2019-01-10 10:00:00	5.0	3.0	2.0	0.0
Store_1	2019-01-10 10:30:00	5.0	3.0	2.0	0.0
Store_1	2019-01-10 11:00:00	5.0	3.0	2.0	0.0
Store_1	2019-01-10 11:30:00	5.0	3.0	2.0	0.0
Store_1	2019-01-10 12:00:00	6.0	4.0	2.0	0.0
Store_1	2019-01-10 12:30:00	6.0	4.0	2.0	0.0
Store_1	2019-01-10 13:00:00	6.0	4.0	2.0	0.0
Store_1	2019-01-10 13:30:00	6.0	4.0	2.0	0.0
Store_1	2019-01-10 14:00:00	6.0	3.0	2.0	1.0
Store_1	2019-01-10 14:30:00	6.0	3.0	2.0	1.0
Store_1	2019-01-10 15:00:00	7.0	4.0	2.0	1.0
Store_1	2019-01-10 15:30:00	7.0	4.0	2.0	1.0
Store_1	2019-01-10 16:00:00	7.0	4.0	2.0	1.0

Store_1	2019-01-10 16:30:00	7.0	4.0	2.0	1.0
Store_1	2019-01-10 17:00:00	7.0	4.0	2.0	1.0
Store_1	2019-01-10 17:30:00	7.0	4.0	2.0	1.0
Store_1	2019-01-10 18:00:00	7.0	4.0	2.0	1.0
Store_1	2019-01-10 18:30:00	7.0	4.0	2.0	1.0
Store_1	2019-01-10 19:00:00	6.0	4.0	2.0	0.0
Store_1	2019-01-10 19:30:00	6.0	4.0	2.0	0.0
Store_1	2019-01-10 20:00:00	6.0	4.0	2.0	0.0
Store_1	2019-01-10 20:30:00	6.0	4.0	2.0	0.0
Store_1	2019-01-10 21:00:00	5.0	3.0	2.0	0.0
Store_1	2019-01-10 21:30:00	5.0	3.0	2.0	0.0
Store_1	2019-01-10 22:00:00	5.0	3.0	2.0	0.0
Store_1	2019-01-10 22:30:00	5.0	3.0	2.0	0.0
Store_1	2019-01-10 23:00:00	5.0	3.0	2.0	0.0
Store_1	2019-01-10 23:30:00	5.0	3.0	2.0	0.0

Table A.8: Prepared transactions example for one day.

location	datetime	n_transactions avg_products_per_t	n_products avg_price_per_product	total_sales avg_sales_per_t
Store_1	2019-01-10 00:00:00	0	0	0.0
		0.0	0.0	0.0
Store_1	2019-01-10 00:30:00	0	0	0.0
		0.0	0.0	0.0
Store_1	2019-01-10 01:00:00	0	0	0.0
		0.0	0.0	0.0
Store_1	2019-01-10 01:30:00	0	0	0.0
		0.0	0.0	0.0
Store_1	2019-01-10 02:00:00	0	0	0.0
		0.0	0.0	0.0
Store_1	2019-01-10 02:30:00	0	0	0.0
		0.0	0.0	0.0
Store_1	2019-01-10 03:00:00	0	0	0.0
		0.0	0.0	0.0
Store_1	2019-01-10 03:30:00	0	0	0.0
		0.0	0.0	0.0
Store_1	2019-01-10 04:00:00	0	0	0.0
		0.0	0.0	0.0
Store_1	2019-01-10 04:30:00	0	0	0.0
		0.0	0.0	0.0
Store_1	2019-01-10 05:00:00	0	0	0.0
		0.0	0.0	0.0
Store_1	2019-01-10 05:30:00	0	0	0.0
		0.0	0.0	0.0
Store_1	2019-01-10 06:00:00	0	0	0.0
		0.0	0.0	0.0
Store_1	2019-01-10 06:30:00	0	0	0.0
		0.0	0.0	0.0
Store_1	2019-01-10 07:00:00	0	0	0.0
		0.0	0.0	0.0
Store_1	2019-01-10 07:30:00	0	0	0.0
		0.0	0.0	0.0
Store_1	2019-01-10 08:00:00	0	0	0.0
		0.0	0.0	0.0

Store_1	2019-01-10 08:30:00	0	0	0.0
		0.0	0.0	0.0
Store_1	2019-01-10 09:00:00	0	0	0.0
		0.0	0.0	0.0
Store_1	2019-01-10 09:30:00	0	0	0.0
		0.0	0.0	0.0
Store_1	2019-01-10 10:00:00	0	0	0.0
		0.0	0.0	0.0
Store_1	2019-01-10 10:30:00	3	1	86.0
		0.3333	86.0	28.6666
Store_1	2019-01-10 11:00:00	3	1	76.0
		0.3333	76.0	25.3333
Store_1	2019-01-10 11:30:00	2	9	585.6999
		4.5	65.07777	292.8499
Store_1	2019-01-10 12:00:00	2	2	179.0
		1.0	89.5	89.5
Store_1	2019-01-10 12:30:00	3	1	55.0
		0.3333	55.0	18.3333
Store_1	2019-01-10 13:00:00	4	2	31.9
		0.5	15.95	7.975
Store_1	2019-01-10 13:30:00	2	2	168.3
		1.0	84.15	84.15
Store_1	2019-01-10 14:00:00	1	1	59.4
		1.0	59.4	59.4
Store_1	2019-01-10 14:30:00	2	2	39.9899
		1.0	19.9949	19.9949
Store_1	2019-01-10 15:00:00	0	0	0.0
		0.0	0.0	0.0
Store_1	2019-01-10 15:30:00	4	6	317.1
		1.5	52.85	79.275
Store_1	2019-01-10 16:00:00	1	1	88.0
		1.0	88.0	88.0
Store_1	2019-01-10 16:30:00	3	1	66.0
		0.3333	66.0	22.0
Store_1	2019-01-10 17:00:00	3	0	72.0
		0.0	0.0	24.0
Store_1	2019-01-10 17:30:00	5	4	476.0
		0.8	119.0	95.2
Store_1	2019-01-10 18:00:00	3	3	54.7
		1.0	18.2333	18.2333
Store_1	2019-01-10 18:30:00	2	0	9.0
		0.0	0.0	4.5
Store_1	2019-01-10 19:00:00	2	2	56.9899
		1.0	28.4949	28.4949
Store_1	2019-01-10 19:30:00	2	2	89.2
		1.0	44.6	44.6
Store_1	2019-01-10 20:00:00	2	2	157.0
		1.0	78.5	78.5
Store_1	2019-01-10 20:30:00	5	3	166.4
		0.6	55.4666	33.28
Store_1	2019-01-10 21:00:00	3	1	130.0
		0.3333	130.0	43.3333
Store_1	2019-01-10 21:30:00	3	3	142.0
		1.0	47.3333	47.3333
Store_1	2019-01-10 22:00:00	3	1	27.3
		0.3333	27.3	9.1
Store_1	2019-01-10 22:30:00	0	0	0.0

		0.0	0.0	0.0
Store_1	2019-01-10 23:00:00	1	1	84.0
		1.0	84.0	84.0
Store_1	2019-01-10 23:30:00	2	3	190.0
		1.5	63.3333	95.0

Table A.9: Prepared shifts example.

location	shift_id	shift_order	shift_duration	task_duration	start_time	end_time	task_type
Store_1	30298	1	0 days 09:00:00	0 days 03:00:00	2019-01-09 15:00:00	2019-01-09 18:00:00	SALES
Store_1	30298	2	0 days 09:00:00	0 days 01:00:00	2019-01-09 18:00:00	2019-01-09 19:00:00	BREAK
Store_1	30298	3	0 days 09:00:00	0 days 05:00:00	2019-01-09 19:00:00	2019-01-10 00:00:00	SALES
Store_1	30299	1	0 days 09:00:00	0 days 06:00:00	2019-01-09 10:00:00	2019-01-09 16:00:00	SALES
Store_1	30299	2	0 days 09:00:00	0 days 01:00:00	2019-01-09 16:00:00	2019-01-09 17:00:00	BREAK
Store_1	30299	3	0 days 09:00:00	0 days 02:00:00	2019-01-09 17:00:00	2019-01-09 19:00:00	SALES
Store_1	30300	1	0 days 09:00:00	0 days 05:00:00	2019-01-09 12:00:00	2019-01-09 17:00:00	SALES
Store_1	30300	2	0 days 09:00:00	0 days 01:00:00	2019-01-09 17:00:00	2019-01-09 18:00:00	BREAK
Store_1	30300	3	0 days 09:00:00	0 days 03:00:00	2019-01-09 18:00:00	2019-01-09 21:00:00	SALES
Store_1	30301	1	0 days 09:00:00	0 days 04:00:00	2019-01-09 15:00:00	2019-01-09 19:00:00	SALES
Store_1	30301	2	0 days 09:00:00	0 days 01:00:00	2019-01-09 19:00:00	2019-01-09 20:00:00	BREAK
Store_1	30301	3	0 days 09:00:00	0 days 04:00:00	2019-01-09 20:00:00	2019-01-10 00:00:00	SALES
Store_1	30302	1	0 days 05:00:00	0 days 05:00:00	2019-01-09 19:00:00	2019-01-10 00:00:00	SALES
Store_1	30303	1	0 days 09:00:00	0 days 04:00:00	2019-01-10 10:00:00	2019-01-10 14:00:00	SALES
Store_1	30303	2	0 days 09:00:00	0 days 01:00:00	2019-01-10 14:00:00	2019-01-10 15:00:00	BREAK
Store_1	30303	3	0 days 09:00:00	0 days 04:00:00	2019-01-10 15:00:00	2019-01-10 19:00:00	SALES
Store_1	30304	1	0 days 09:00:00	0 days 02:00:00	2019-01-10 15:00:00	2019-01-10 17:00:00	SALES
Store_1	30304	2	0 days 09:00:00	0 days 01:00:00	2019-01-10 17:00:00	2019-01-10 18:00:00	BREAK
Store_1	30304	3	0 days 09:00:00	0 days 06:00:00	2019-01-10 18:00:00	2019-01-11 00:00:00	SALES
Store_1	30305	1	0 days 14:00:00	0 days 14:00:00	2019-01-10 10:00:00	2019-01-11 00:00:00	OTHERS
Store_1	30306	1	0 days 09:00:00	0 days 05:00:00	2019-01-10 10:00:00	2019-01-10 15:00:00	SALES
Store_1	30306	2	0 days 09:00:00	0 days 01:00:00	2019-01-10 15:00:00	2019-01-10 16:00:00	BREAK
Store_1	30306	3	0 days 09:00:00	0 days 03:00:00	2019-01-10 16:00:00	2019-01-10 19:00:00	SALES
Store_1	30307	1	0 days 09:00:00	0 days 04:00:00	2019-01-10 12:00:00	2019-01-10 16:00:00	SALES
Store_1	30307	2	0 days 09:00:00	0 days 01:00:00	2019-01-10 16:00:00	2019-01-10 17:00:00	BREAK
Store_1	30307	3	0 days 09:00:00	0 days 04:00:00	2019-01-10 17:00:00	2019-01-10 21:00:00	SALES
Store_1	30308	1	0 days 14:00:00	0 days 14:00:00	2019-01-10 10:00:00	2019-01-11 00:00:00	OTHERS
Store_1	30309	1	0 days 09:00:00	0 days 03:00:00	2019-01-10 15:00:00	2019-01-10 18:00:00	SALES
Store_1	30309	2	0 days 09:00:00	0 days 01:00:00	2019-01-10 18:00:00	2019-01-10 19:00:00	BREAK
Store_1	30309	3	0 days 09:00:00	0 days 05:00:00	2019-01-10 19:00:00	2019-01-11 00:00:00	SALES
Store_1	30310	1	0 days 05:00:00	0 days 05:00:00	2019-01-10 19:00:00	2019-01-11 00:00:00	SALES
Store_1	30311	1	0 days 05:00:00	0 days 05:00:00	2019-01-10 10:00:00	2019-01-10 15:00:00	SALES

Table A.10: Prepared open hours example.

location	datetime	close_hours	max_employees	open_hours	weekday
Store_1	2019-01-01	2018-12-31 10:00:00	0	2018-12-31 10:00:00	1
Store_1	2019-01-02	2019-01-02 23:30:00	0	2019-01-02 09:00:00	2
Store_1	2019-01-03	2019-01-04 00:00:00	11	2019-01-03 09:00:00	3
Store_1	2019-01-04	2019-01-05 00:00:00	11	2019-01-04 09:00:00	4
Store_1	2019-01-05	2019-01-06 00:00:00	11	2019-01-05 09:00:00	5
Store_1	2019-01-06	2019-01-07 00:00:00	11	2019-01-06 09:00:00	6

Store_1	2019-01-07	2019-01-08 00:00:00	7	2019-01-07 09:00:00	0
Store_1	2019-01-08	2019-01-09 00:00:00	7	2019-01-08 09:00:00	1
Store_1	2019-01-09	2019-01-10 00:00:00	7	2019-01-09 09:00:00	2
Store_1	2019-01-10	2019-01-11 00:00:00	9	2019-01-10 09:00:00	3
Store_1	2019-01-11	2019-01-12 00:00:00	7	2019-01-11 09:00:00	4
Store_1	2019-01-12	2019-01-13 00:00:00	10	2019-01-12 09:00:00	5
Store_1	2019-01-13	2019-01-14 00:00:00	10	2019-01-13 09:00:00	6
Store_1	2019-01-14	2019-01-15 00:00:00	8	2019-01-14 09:00:00	0
Store_1	2019-01-15	2019-01-16 00:00:00	7	2019-01-15 09:00:00	1
Store_1	2019-01-16	2019-01-17 00:00:00	7	2019-01-16 09:00:00	2
Store_1	2019-01-17	2019-01-18 00:00:00	7	2019-01-17 10:00:00	3
Store_1	2019-01-18	2019-01-19 00:00:00	8	2019-01-18 10:00:00	4
Store_1	2019-01-19	2019-01-20 00:00:00	10	2019-01-19 10:00:00	5
Store_1	2019-01-20	2019-01-21 00:00:00	10	2019-01-20 10:00:00	6
Store_1	2019-01-21	2019-01-22 00:00:00	8	2019-01-21 10:00:00	0
Store_1	2019-01-22	2019-01-23 00:00:00	7	2019-01-22 10:00:00	1
Store_1	2019-01-23	2019-01-24 00:00:00	7	2019-01-23 10:00:00	2
Store_1	2019-01-24	2019-01-25 00:00:00	6	2019-01-24 10:00:00	3
Store_1	2019-01-25	2019-01-26 00:00:00	7	2019-01-25 10:00:00	4
Store_1	2019-01-26	2019-01-27 00:00:00	9	2019-01-26 10:00:00	5
Store_1	2019-01-27	2019-01-28 00:00:00	9	2019-01-27 10:00:00	6
Store_1	2019-01-28	2019-01-29 00:00:00	6	2019-01-28 10:00:00	0
Store_1	2019-01-29	2019-01-30 00:00:00	7	2019-01-29 10:00:00	1
Store_1	2019-01-30	2019-01-31 00:00:00	4	2019-01-30 10:00:00	2
Store_1	2019-01-31	2019-02-01 00:00:00	3	2019-01-31 10:00:00	3
Store_1	2019-02-01	2019-02-02 00:00:00	4	2019-02-01 10:00:00	4
Store_1	2019-02-02	2019-02-03 00:00:00	6	2019-02-02 10:00:00	5
Store_1	2019-02-03	2019-02-04 00:00:00	6	2019-02-03 10:00:00	6
Store_1	2019-02-04	2019-02-05 00:00:00	6	2019-02-04 10:00:00	0
Store_1	2019-02-05	2019-02-06 00:00:00	5	2019-02-05 10:00:00	1
Store_1	2019-02-06	2019-02-07 00:00:00	7	2019-02-06 10:00:00	2
Store_1	2019-02-07	2019-02-08 00:00:00	8	2019-02-07 10:00:00	3
Store_1	2019-02-08	2019-02-09 00:00:00	8	2019-02-08 10:00:00	4
Store_1	2019-02-09	2019-02-10 00:00:00	10	2019-02-09 10:00:00	5
Store_1	2019-02-10	2019-02-11 00:00:00	9	2019-02-10 10:00:00	6
Store_1	2019-02-11	2019-02-12 00:00:00	5	2019-02-11 10:00:00	0
Store_1	2019-02-12	2019-02-13 00:00:00	8	2019-02-12 10:00:00	1
Store_1	2019-02-13	2019-02-14 00:00:00	5	2019-02-13 10:00:00	2
Store_1	2019-02-14	2019-02-15 00:00:00	6	2019-02-14 10:00:00	3
Store_1	2019-02-15	2019-02-16 00:00:00	6	2019-02-15 10:00:00	4
Store_1	2019-02-16	2019-02-17 00:00:00	8	2019-02-16 10:00:00	5
Store_1	2019-02-17	2019-02-18 00:00:00	8	2019-02-17 10:00:00	6
Store_1	2019-02-18	2019-02-19 00:00:00	7	2019-02-18 10:00:00	0
Store_1	2019-02-19	2019-02-20 00:00:00	6	2019-02-19 10:00:00	1
Store_1	2019-02-20	2019-02-21 00:00:00	7	2019-02-20 10:00:00	2
Store_1	2019-02-21	2019-02-22 00:00:00	6	2019-02-21 10:00:00	3
Store_1	2019-02-22	2019-02-23 00:00:00	8	2019-02-22 10:00:00	4
Store_1	2019-02-23	2019-02-24 00:00:00	9	2019-02-23 10:00:00	5
Store_1	2019-02-24	2019-02-25 00:00:00	10	2019-02-24 10:00:00	6
Store_1	2019-02-25	2019-02-26 00:00:00	7	2019-02-25 10:00:00	0
Store_1	2019-02-26	2019-02-27 00:00:00	6	2019-02-26 10:00:00	1
Store_1	2019-02-27	2019-02-28 00:00:00	6	2019-02-27 10:00:00	2
Store_1	2019-02-28	2019-03-01 00:00:00	7	2019-02-28 10:00:00	3

A.2 Data Analysis

A.2.1 Events

Table A.11: Raw events dataset analysis.

No. Rows	No. Duplicates
324	0

Table A.12: Analysis of attributes in the raw events dataset.

Attributes	Missing	Unique	Range
location	0	4	[Store_1, Store_4]
start	0	82	[2016-01-01, 2019-03-19]
end	0	82	[2016-01-01, 2019-03-19]
type	0	4	-

Table A.13: Event type attribute values.

Type
PROMOTION-LOCAL
CALENDAR-PUBLIC
EVENT-PUBLIC
CALENDAR-LOCAL

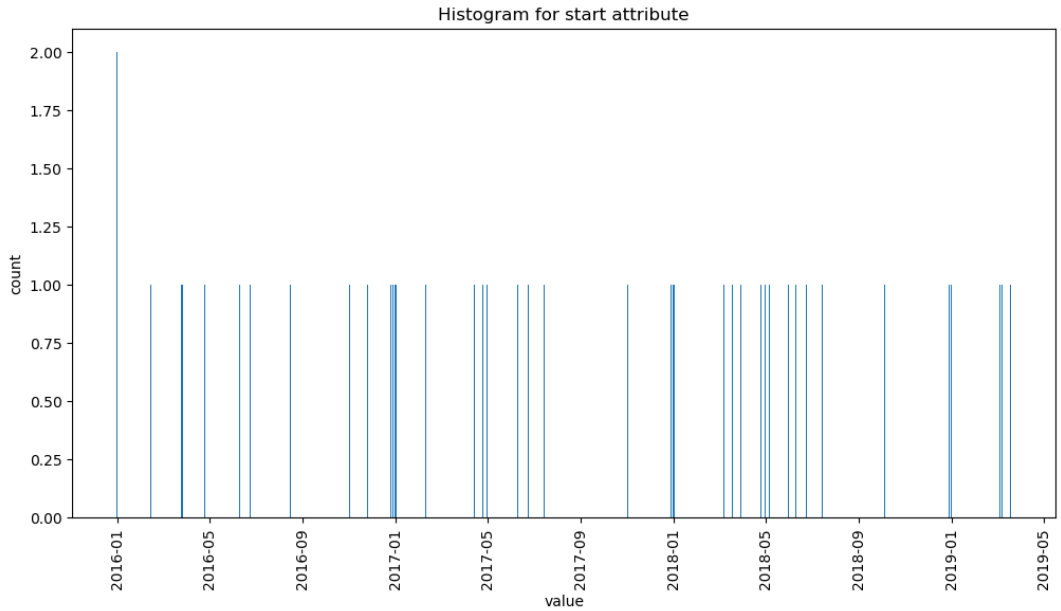


Figure A.1: Histogram for the *start* attribute values of the raw events dataset in *Store_1*.

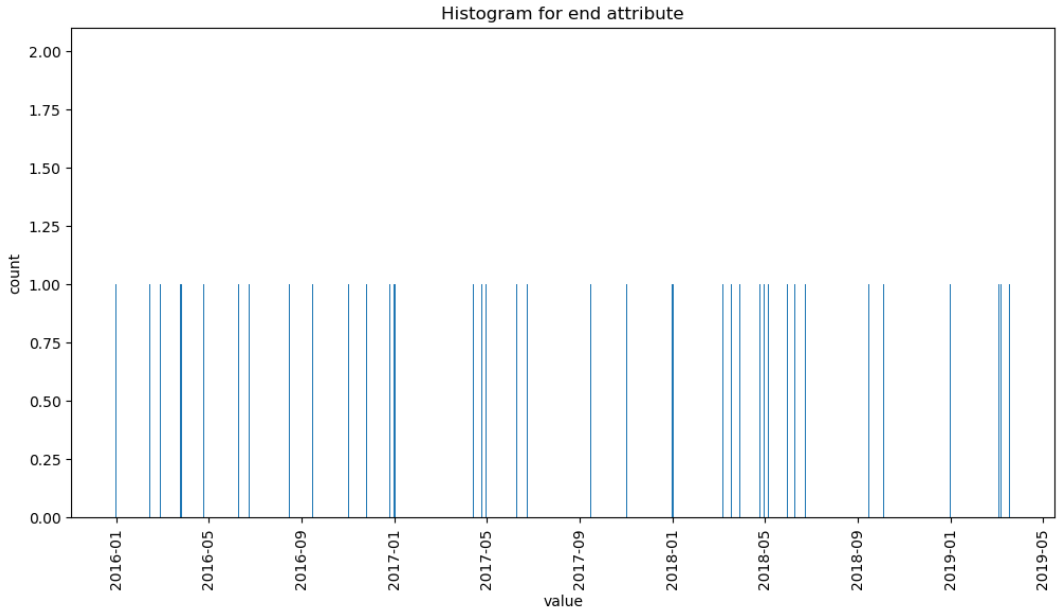


Figure A.2: Histogram for the *end* attribute values of the raw events datase in *Store_1*.

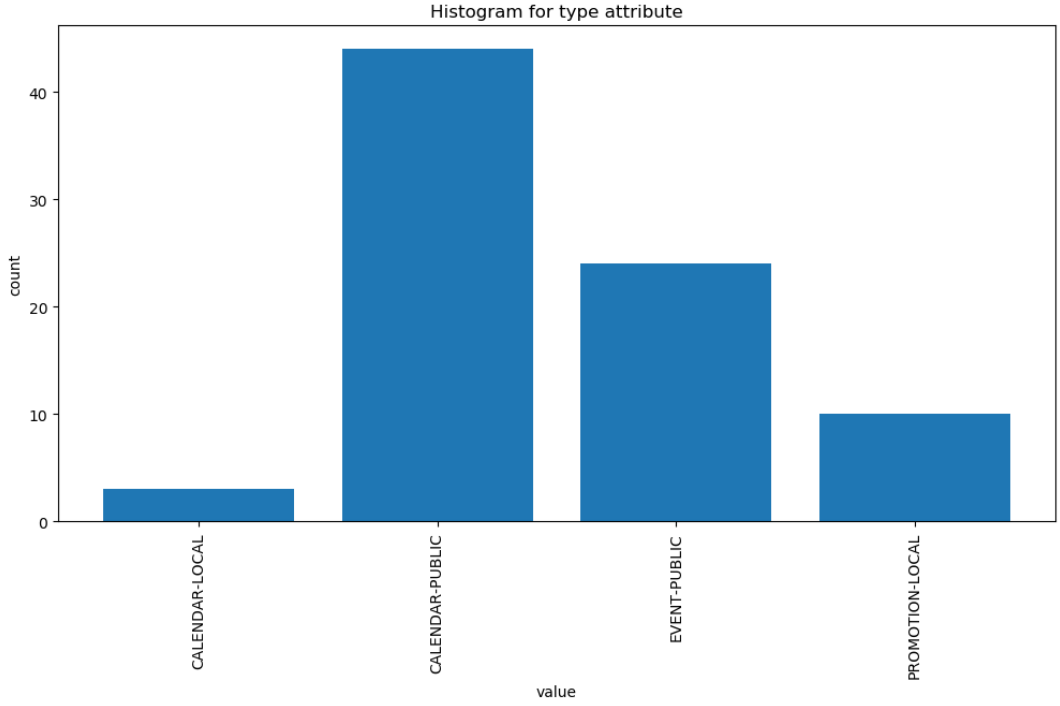


Figure A.3: Histogram for the *type* attribute values of the raw events dataset in *Store_1*.

A.2.2 Footfalls

Table A.14: Raw footfalls dataset analysis.

No. Rows	No. Duplicates
60239	0

Table A.15: Analysis of attributes in the raw footfalls dataset.

Attributes	Missing	Unique	Range
location	0	4	[Store_1, Store_4]
date	0	1165	[2016-01-02, 2019-03-17]
time	0	17	[0, 900000]
footfall in	0	8996	[-653584065.3, 207136.0009]
footfall out	0	1	[0, 0]

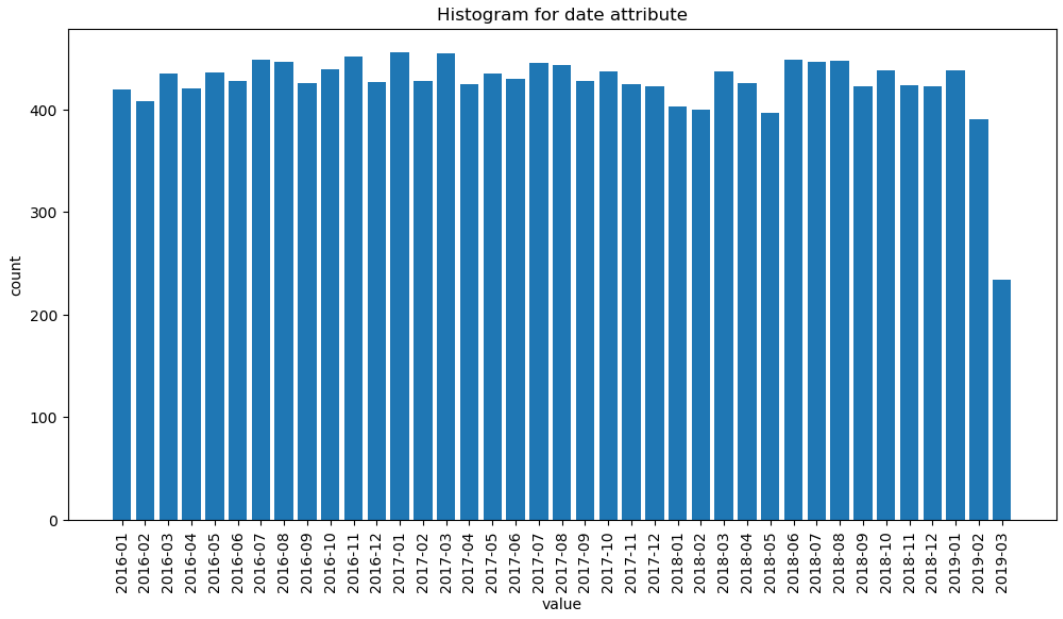


Figure A.4: Histogram for the *date* attribute values of the raw footfalls dataset in *Store_1*.

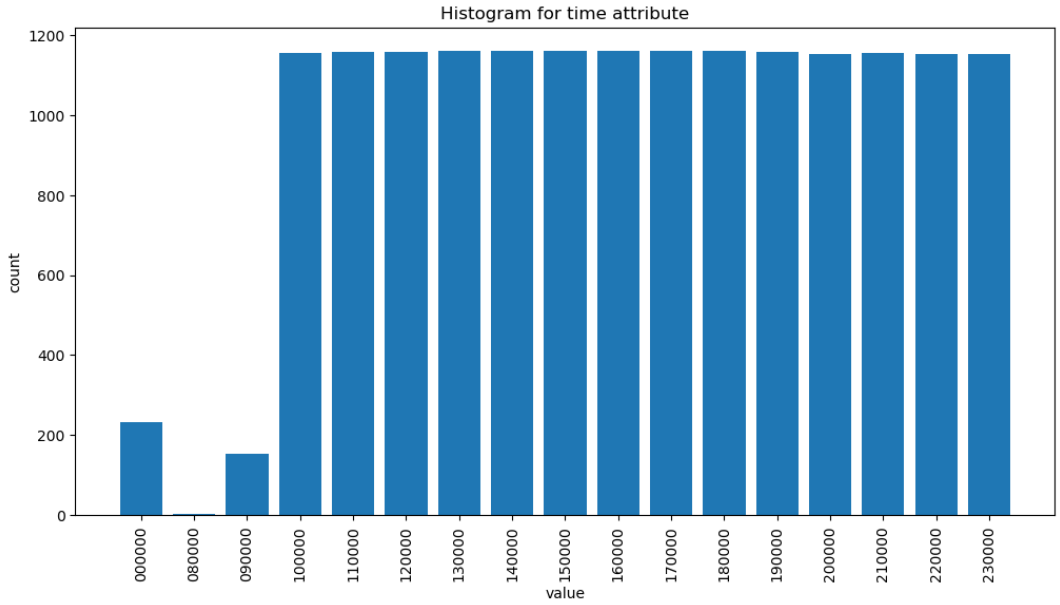


Figure A.5: Histogram for the *time* attribute values of the raw footfalls dataset in *Store_1*.

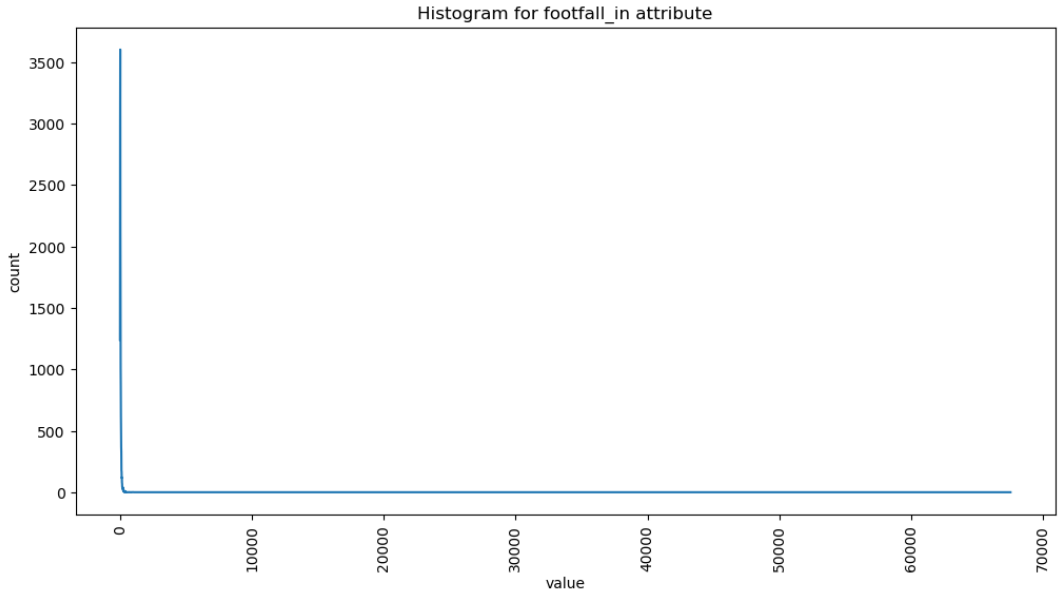


Figure A.6: Histogram for the *footfall in* attribute values of the raw footfalls dataset in *Store_1*.

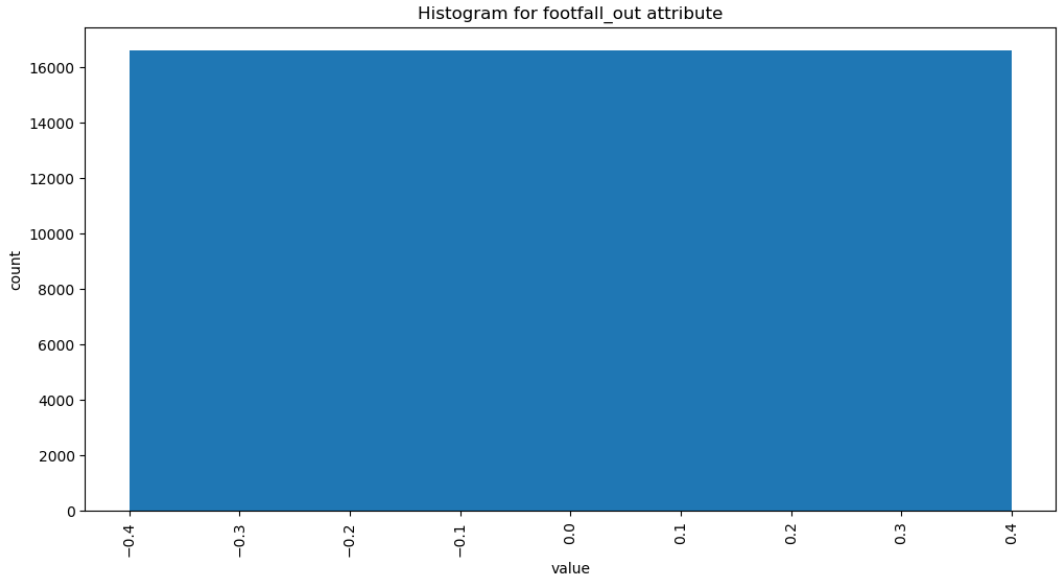


Figure A.7: Histogram for the *footfall out* attribute values of the raw footfalls dataset in *Store_1*.

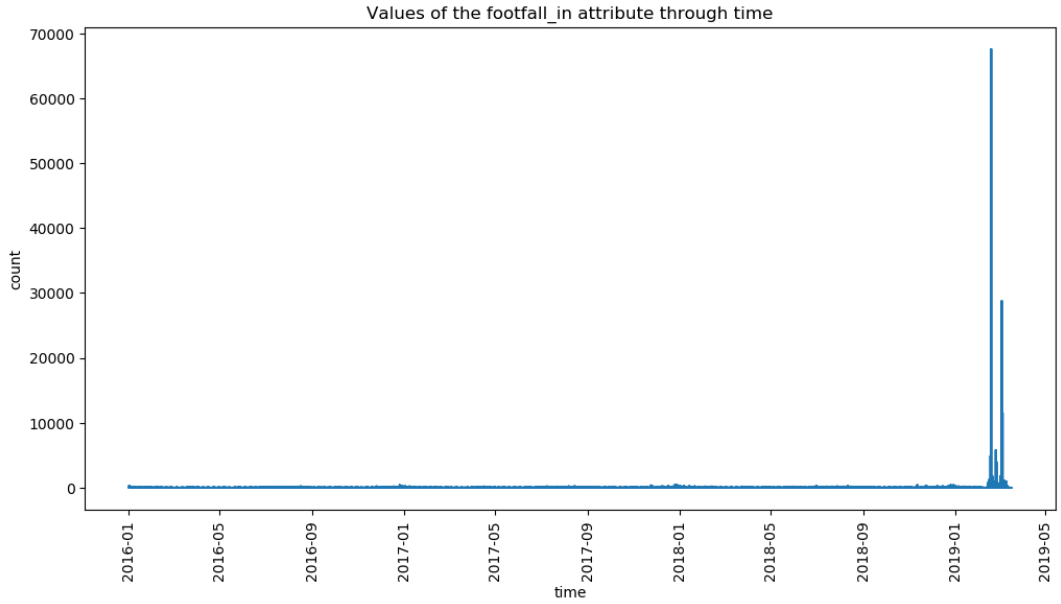


Figure A.8: Values of the *footfall in*, per day, through time for *Store_1*.

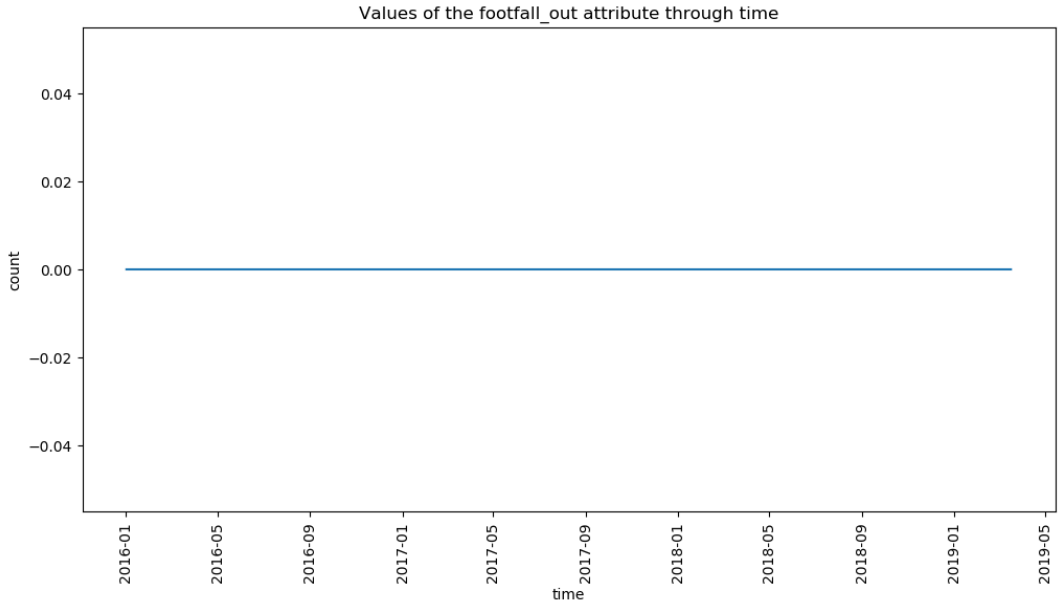


Figure A.9: Values of the *footfall out*, per day, through time for *Store_1*.

A.2.3 Schedules

Table A.16: Raw schedules dataset analysis.

No. Rows	No. Duplicates
170574	0

Table A.17: Analysis of attributes in the raw schedules dataset.

Attributes	Missing	Unique	Range
location	0	4	[Store_1, Store_4]
employee code	0	185	[Employee_1, Employee_99]
date	0	4280	[2006-09-18, 2019-05-05]
start	0	38	[00:00:00, 23:30:00]
end	0	43	[00:00:00, 23:30:00]
shift type	0	3	-

Table A.18: Schedule shift type attribute values.

Type
SALES
OTHER
BREAK

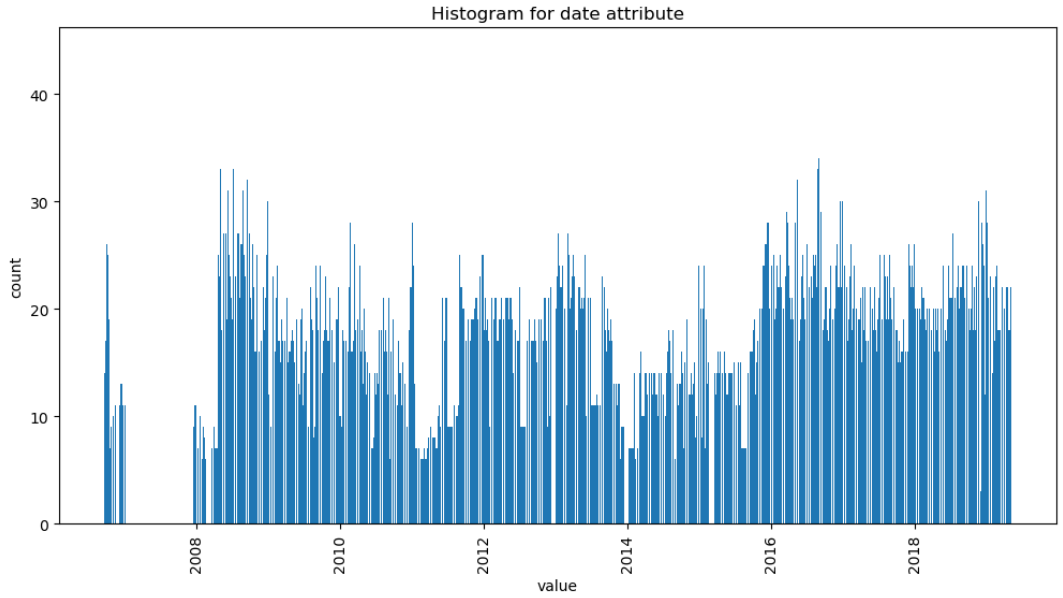


Figure A.10: Histogram for the *date* attribute values of the raw schedules dataset in *Store_1*.

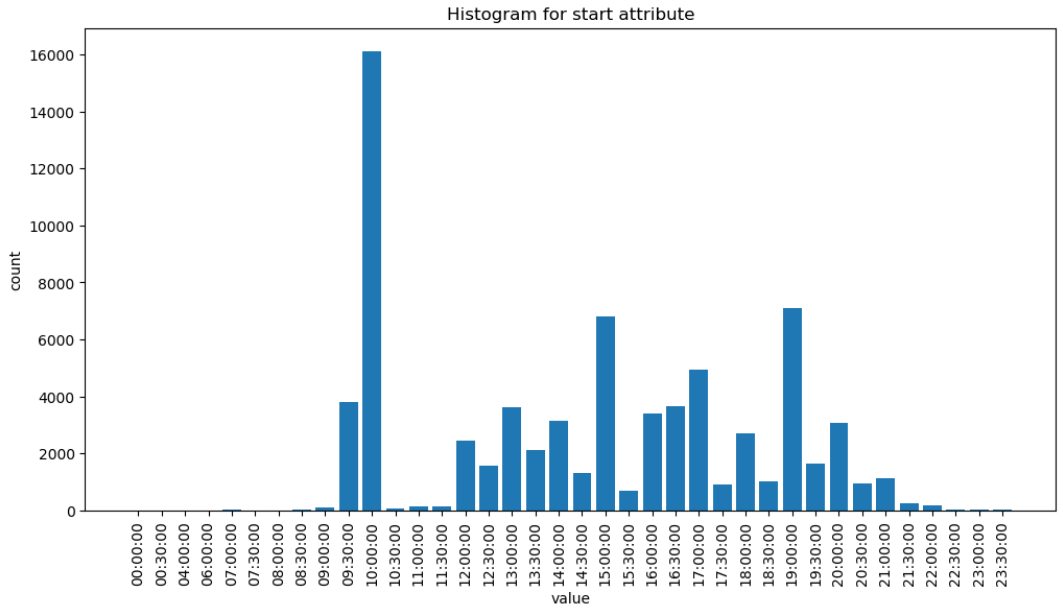


Figure A.11: Histogram for the *start* attribute values of the raw schedules dataset in *Store_1*.

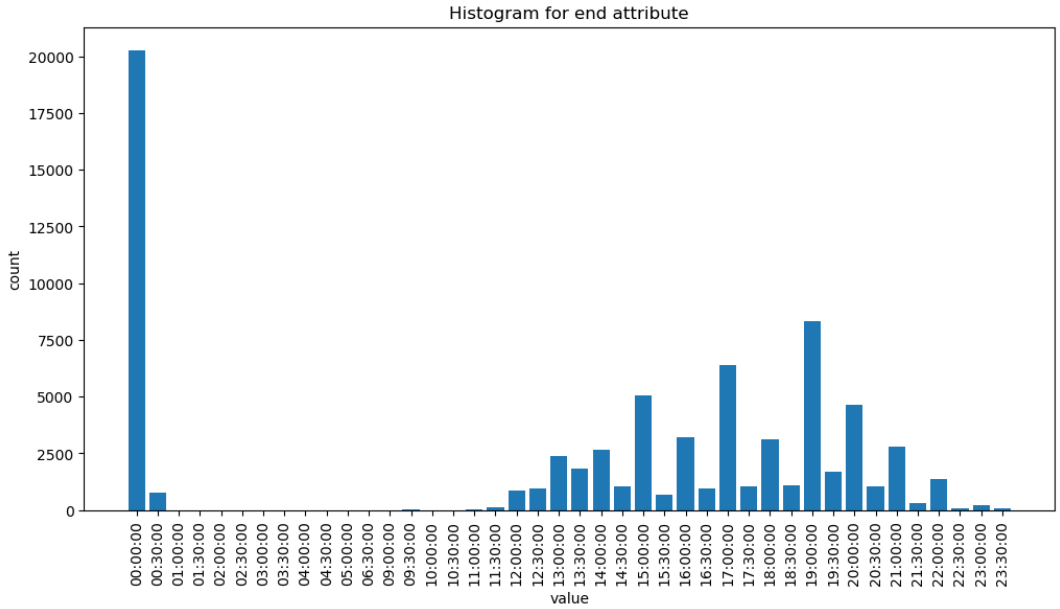


Figure A.12: Histogram for the *end* attribute values of the raw schedules dataset in *Store_1*.

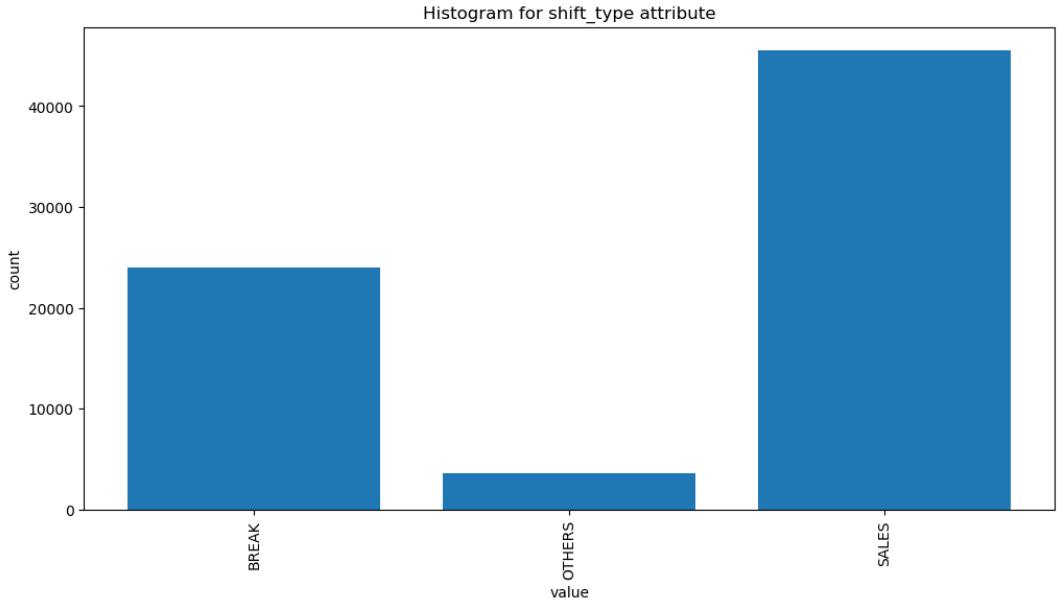


Figure A.13: Histogram for the *shift type* attribute values of the raw schedules dataset in *Store_1*.

A.2.4 Transactions

Table A.19: Raw transactions dataset analysis.

No. Rows	No. Duplicates
378067	0

Table A.20: Analysis of attributes in the raw transactions dataset.

Attributes	Missing	Unique	Range
location	0	4	[Store_1, Store_4]
product	0	30456	[Product_1, Product_9999]
ticket_code	0	99999	[ticket_1, ticket_99999]
date	0	1582	[2014-01-02, 2019-03-18]
hour	0	49165	[0, 95847]
employee code	0	265	[Employee_1, Employee_98]
qty	0	15	[-6, 14]
net sales	0	708	[-456,700]

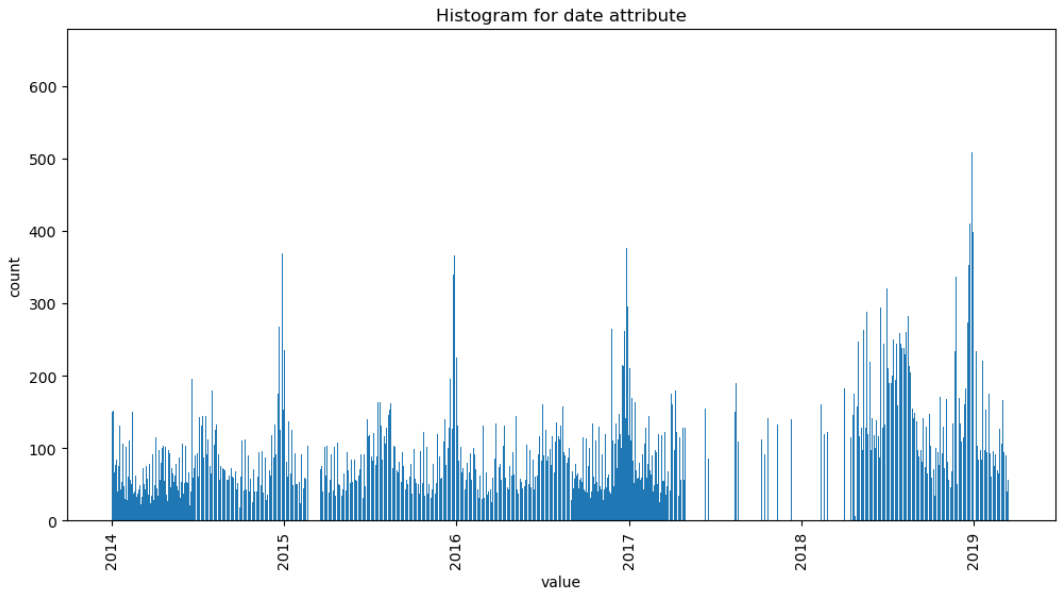


Figure A.14: Histogram for the *date* attribute values of the raw transactions dataset in *Store_1*.

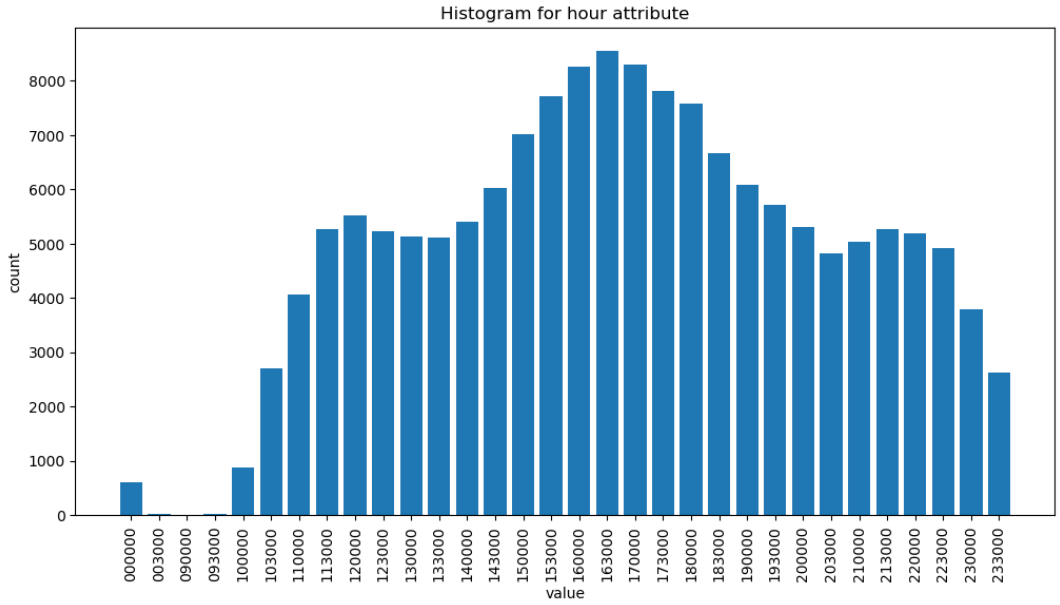


Figure A.15: Histogram for the *hour* attribute values of the raw transactions dataset in *Store_1*. Each bin represents a 3000 value range.

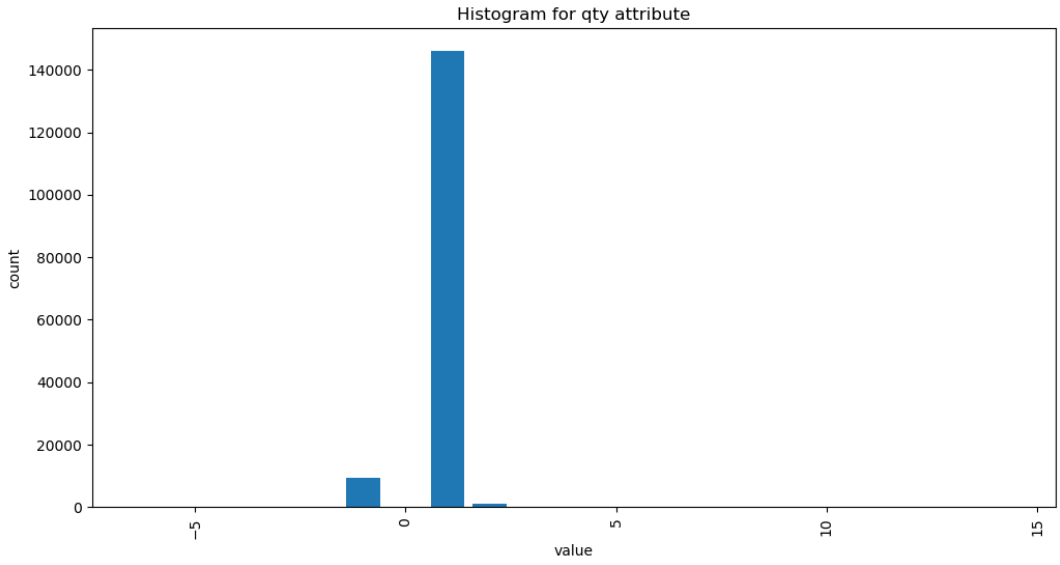


Figure A.16: Histogram for the *qty* attribute values, of the raw transactions dataset in *Store_1*.

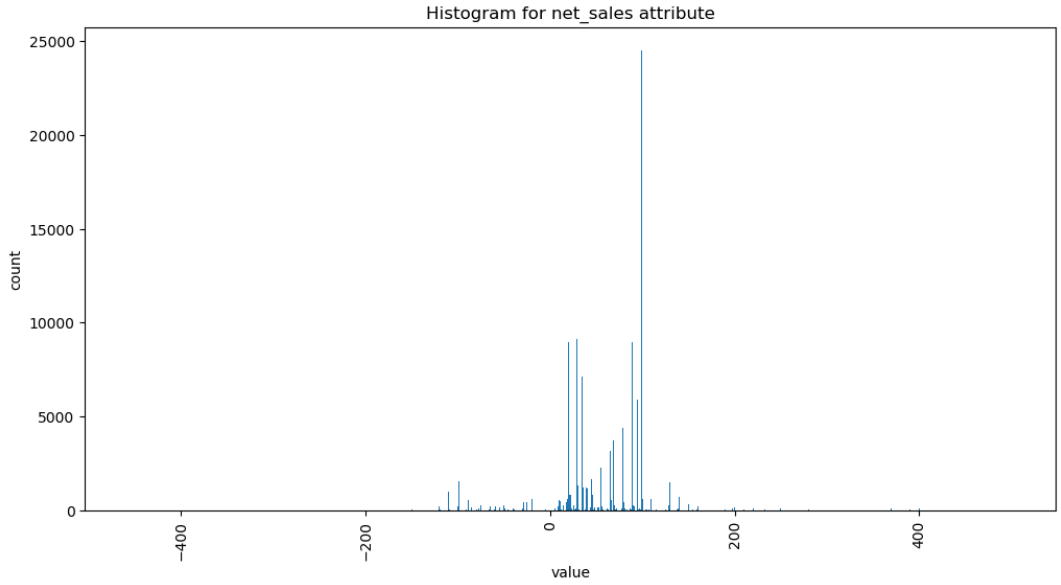


Figure A.17: Histogram for the *net sales* attribute values of the raw transactions dataset in *Store_1*.

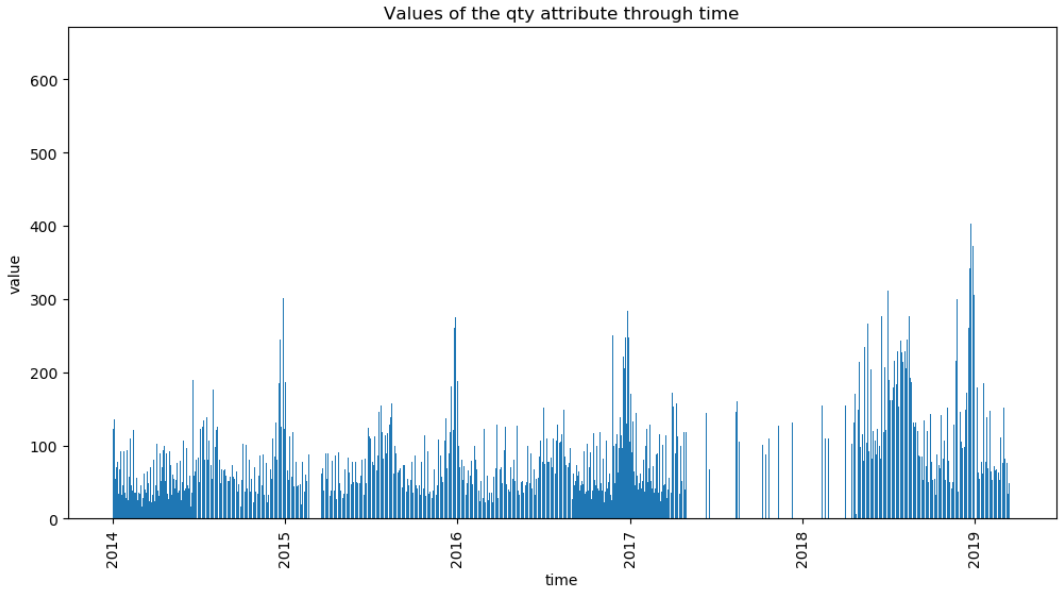


Figure A.18: Values of the *qty*, per day, through time for *Store_1*.

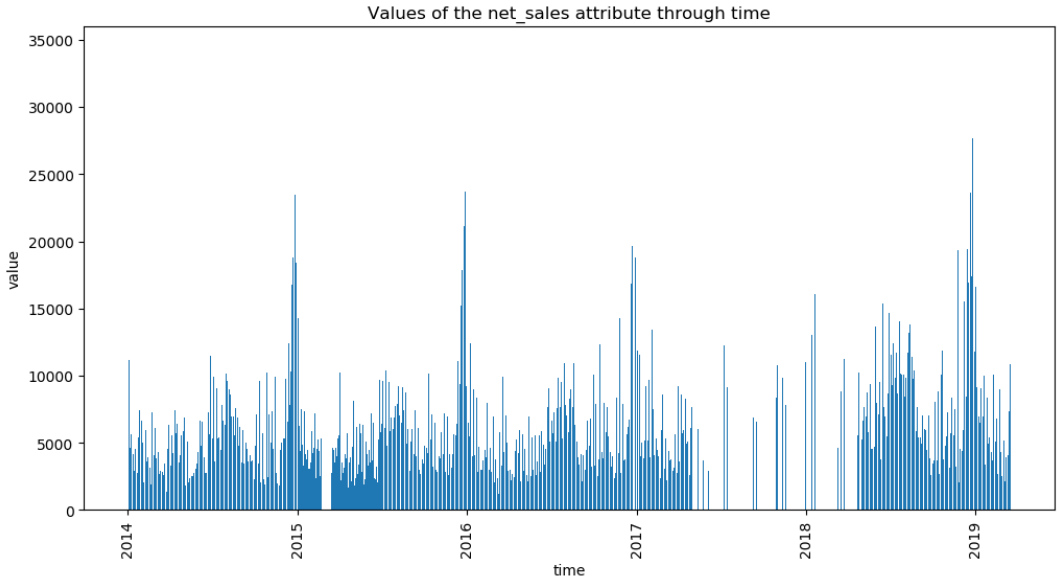


Figure A.19: Values of the *net sales*, per day, through time for *Store_1*.

A.2.5 Shifts

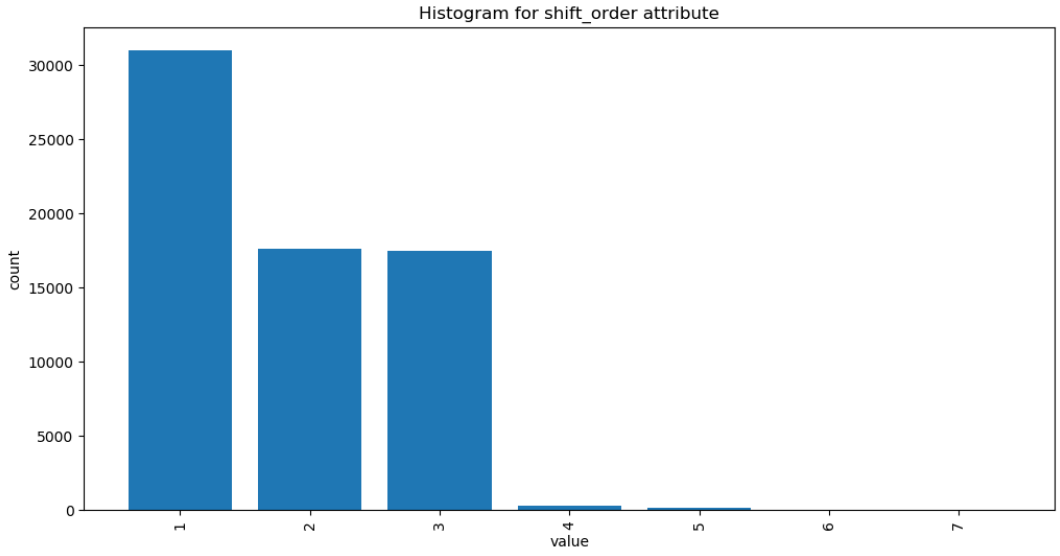


Figure A.20: Histogram for the *shift_order* attribute values of the shifts dataset in *Store_1*.

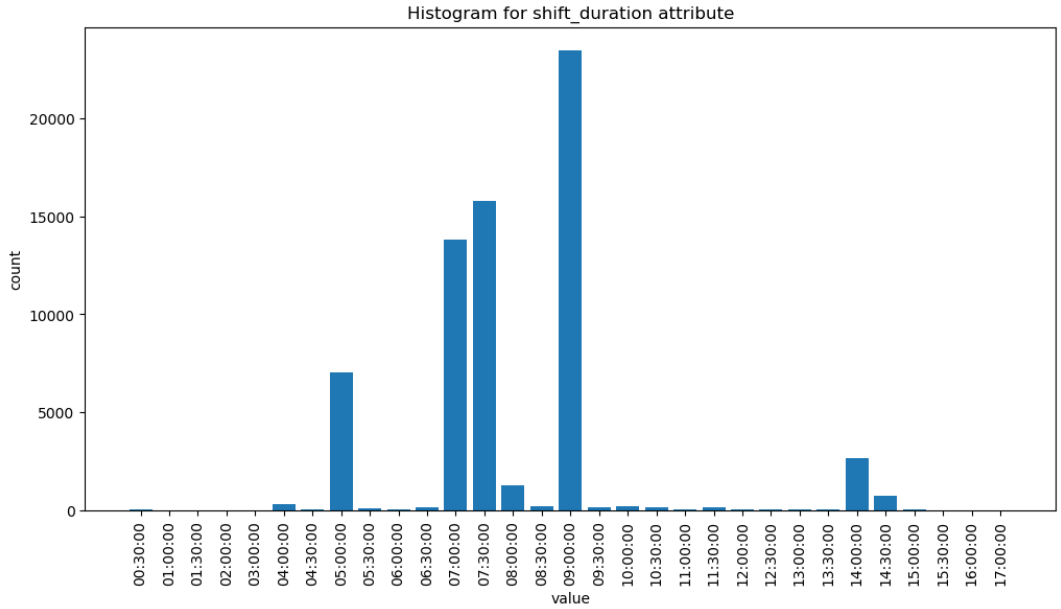


Figure A.21: Histogram for the *shift_duration* attribute values of the shifts dataset in *Store_1*.

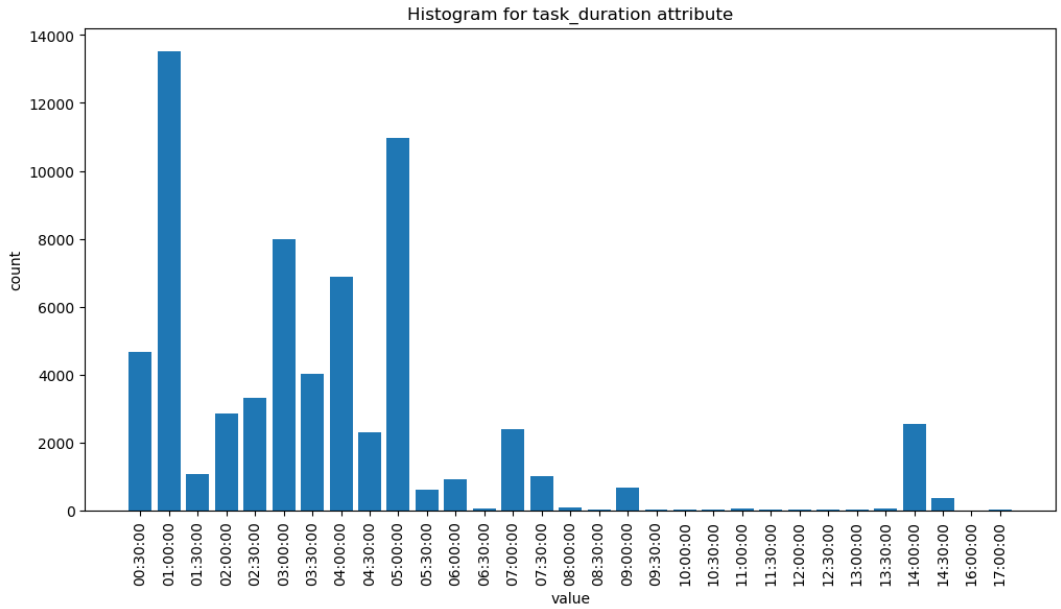


Figure A.22: Histogram for the *task_duration* attribute values of the shifts dataset in *Store_1*.

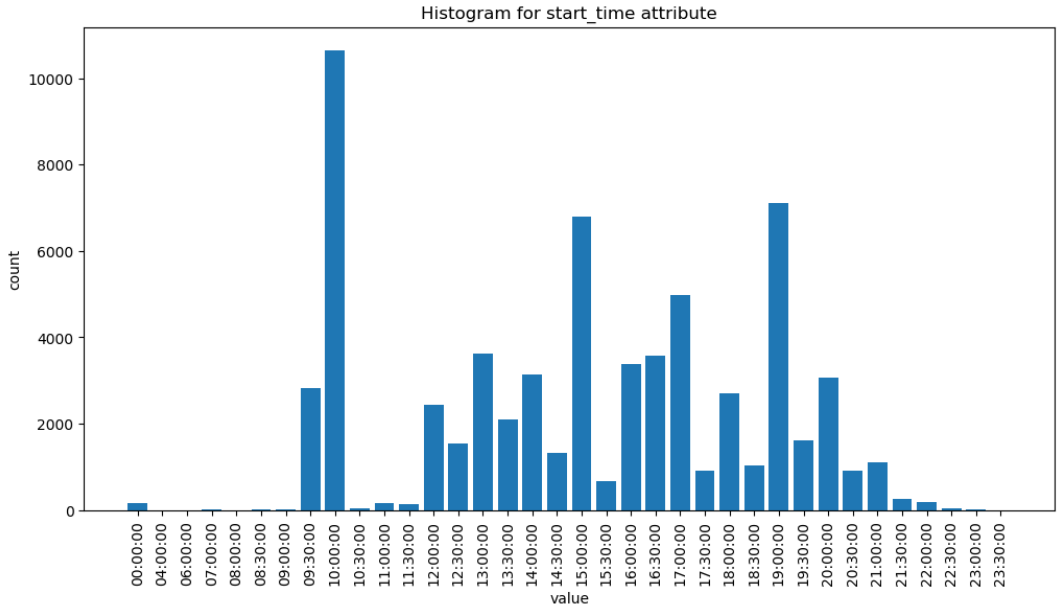


Figure A.23: Histogram for the *start_time* attribute values of the shifts dataset in *Store_1*.

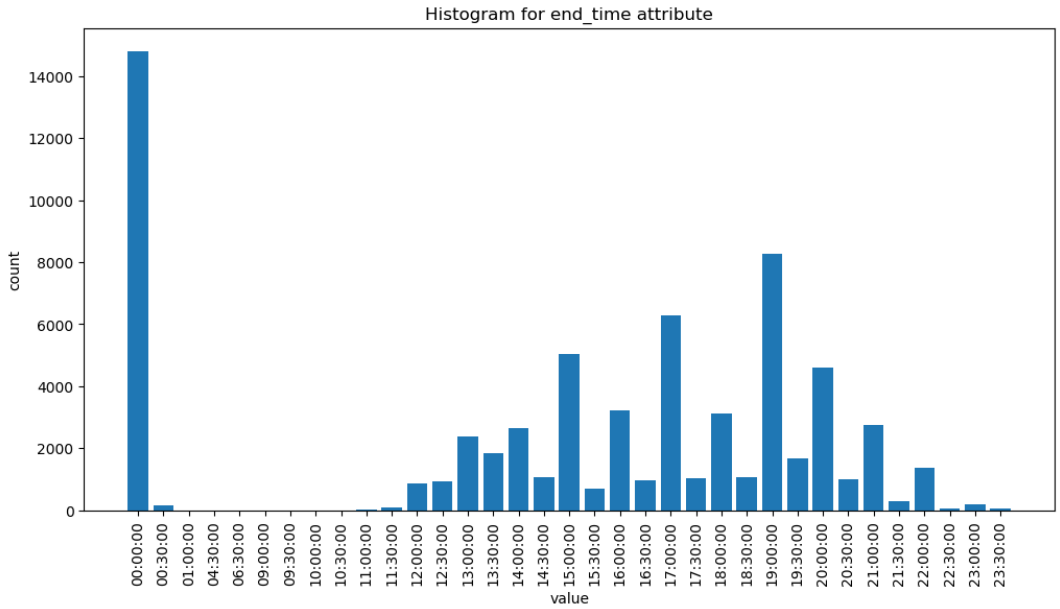


Figure A.24: Histogram for the *end_time* attribute values of the shifts dataset in *Store_1*.

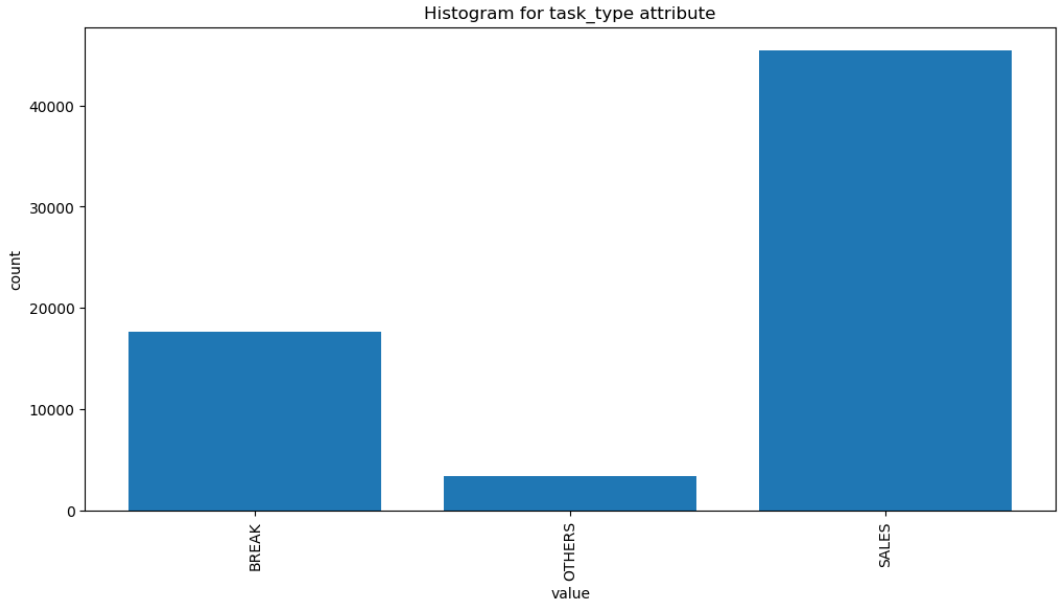


Figure A.25: Histogram for the *task_type* attribute values of the shifts dataset in *Store_1*.

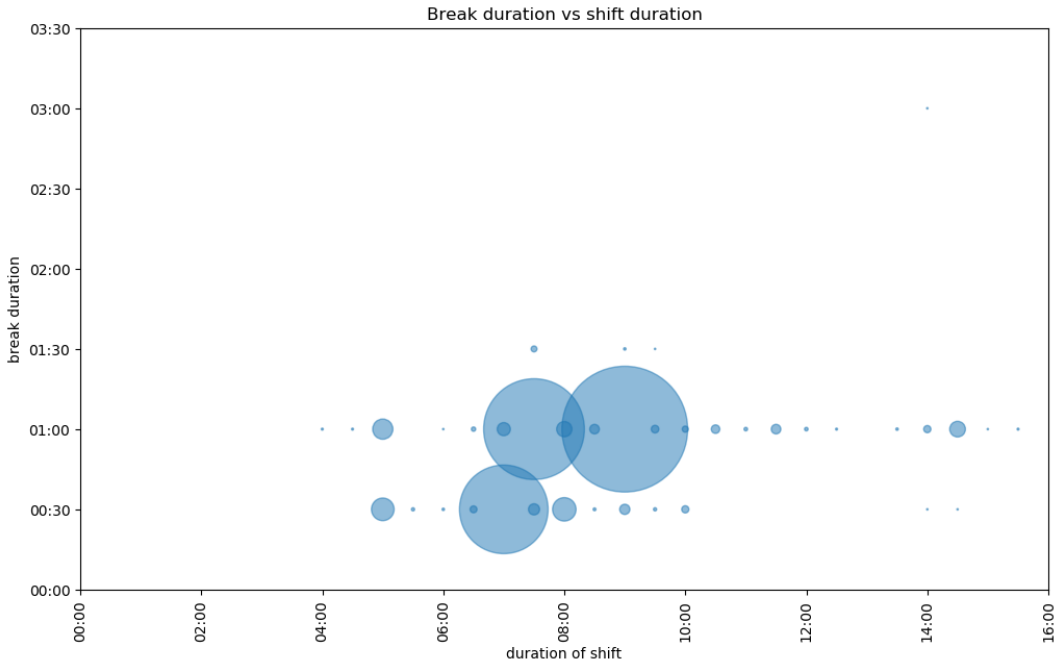


Figure A.26: Comparison between the duration of shifts that have breaks and the break task duration.

A.2.6 Open Hours

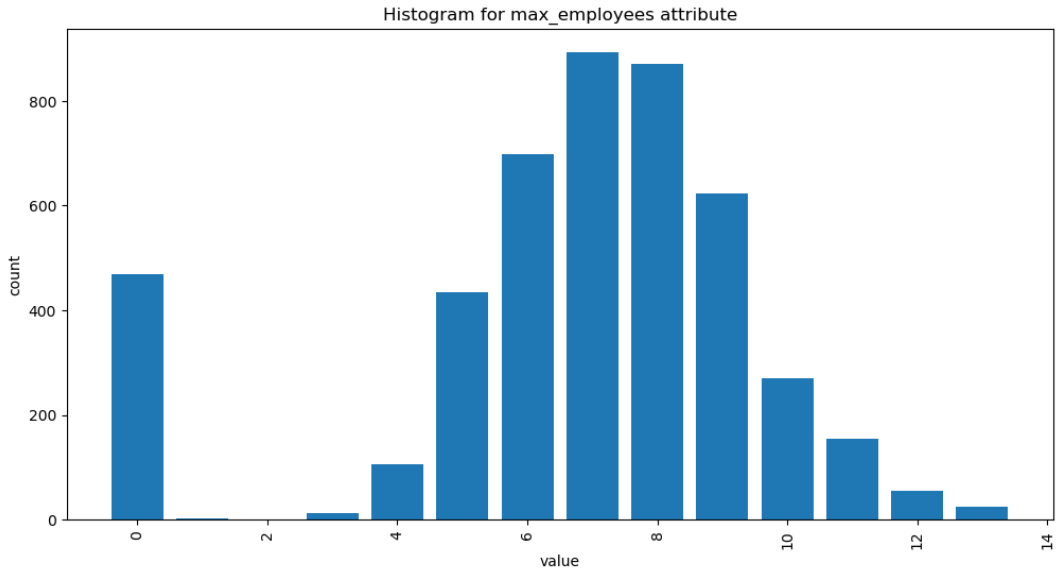


Figure A.27: Histogram for the *max_employees* attribute values of the open hours dataset in *Store_1*.

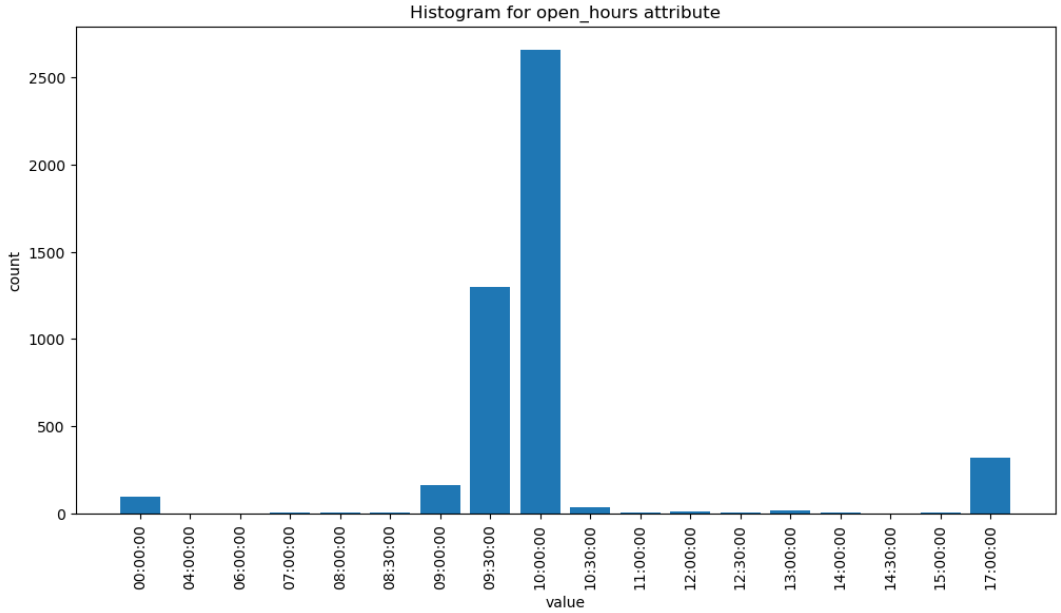


Figure A.28: Histogram for the *open_hours* attribute values of the open hours dataset in *Store_1*.

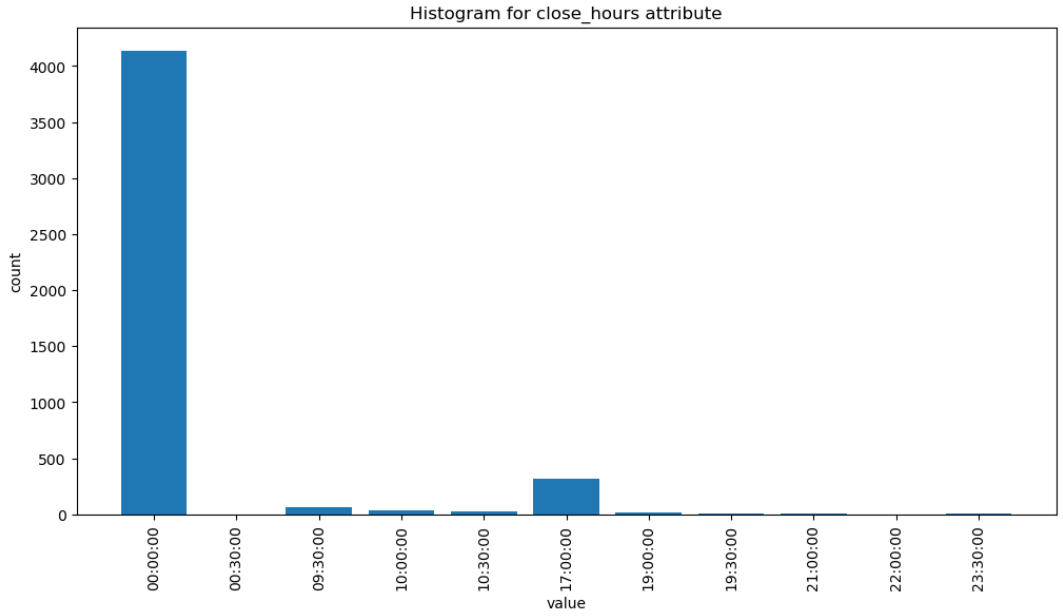


Figure A.29: Histogram for the *close_hours* attribute values of the open hours dataset in *Store_1*.

A.3 Data Cleaning

Table A.21: Problems identified in the data.

Code	Dataset	Problem
E1	Events	The attributes *start* and *end* are represented by integers. Due to this, records between [00:00:00,09:59:59] are represented by only one digit and the others by 6 digits.
F1	Footfall	The attributes *date* and *time* are in the integer format. For this reason, the same situation described in problem E1 can happen to the start and *time* attributes.
F2	Footfall	Negative values in the *footfall in* attribute. It's impossible for the number of entries in the store to be less than zero.
F3	Footfall	Footfall is recorded for each hour, however, there are no records for some time intervals. For example, between 01:00h and 8:00h.
F4	Footfall	The attribute *footfall in* is not an integer.
F5	Footfall	Anomaly in the values of the *footfall in* attribute after February in 2019. This was caused by an error in the system where, for several day, the entries count was not being updated. When the system got fixed, the entries count of the past days, where all inserted for the same time. This resulted in large values of *footfall in*, in comparison with the other values through the years.
F6	Footfall	The attribute *footfall out* is always zero.

S1	Schedules	The attributes *date*, *start* and *end* are in the string format.
S2	Schedules	Employees with overlapping shifts.
S3	Schedules	Break shifts that do not start by the end of a previous shift or don't end at the start of another.
T1	Transactions	The *date* and *hour* attributes are in the integer format. Due to this, the same situation described in problem E1 can happen to the *hour* attribute.
T2	Transactions	Lack of information between the start of year 2017 until March 2018, in comparison with the past years.

Table A.22: Cleaning solution for the problems identified in the data and step-by-step explanation.

Code	**Solution**	**Step-by-step**
E1	Convert to a 6 digits representation and then to the integer format.	For each attribute: 1. Convert the values to string. 2. Fill the necessary zeros at left, so that the string has length equal to 6. 3. Convert to the date format.
F1	Convert to the correct format.	Steps: 1. Apply the same solution described in the problem E1 for the *time* attribute. 2. Create a new column *datetime* with the joined information from *date* and *time*. 3. Discard columns *date* and *time*.
F2	Update the *footfall in* to zero when it's a negative value.	Steps: 1. Identify records where *footfall in* < 0. 2. Update *footfall in* $= 0$.

F3	Insert new records in the dataset for the missing time intervals of each day with zero *footfall in* and *footfall out*.	Steps: 1. Calculate minimum time unit between records. 2. Search for missing records. 3. Create new record, if missing, with *footfall in* = 0 and *footfall out* = 0.
F4	Without knowing the source of this error or how the number of entries is calculated, we decided not to handle it. Moreover, the error can be either $x - \lfloor x \rfloor$ or $\lceil x \rceil - x$ but is always less than 1. This can be a better approach than assuming that the real value is always the *floor* or *ceiling* which, if it's incorrect, the error would be 1. For instance, if the value of *footfall in* is 2.5 but the real value is 2, then, for this case, *ceiling* would produce an error of $\lceil 2.5 \rceil - 2 = 1$ and *floor* of $\lfloor 2.5 \rfloor - 2 = 0$.	
F5	Change the *footfall in* outliers with the average value.	Steps: 1. Identify outliers. 2. Set the *footfall in* attribute with the average for the records in step 1.
F6	There is no solution for this case because we can't try to estimate the *footfall out* value since there is no historical data with this information.	

S1	Convert to the correct format.	Steps: 1. Join the information of *date* and *start* in the *start* column. 2. Convert *start* to the date format. 3. Join the information of *date* and *end* in the *end* column. 4. Convert *end* to the date format. 5. Discard the *date* attribute.
S2	Adjust the overlapping shift's boundaries so that they will be side by side instead of overlapping.	For each group of shifts for each employee and each day: 1. Sort shifts by their ascending *start* and descending *end*. 2. Create an array with length equal to $24 * 60$. Each element corresponds to one minute interval of the day. 3. Iterate each shift and map the shift's *shift type* value to the corresponding array elements. For example, for a sales shift between [10:00h,12:00h], set each element of the array between [10*60-1,12*60-1] with the value "sales". 4. Extract the shifts from the array by looking for consecutive *shift type* values. For example, for the array ["A","A","A","B","C","C"], we will extract the shifts: "A" between [00:00h,00:02h], "B" between [00:03h,00:04h] and "C" between [00:05h,00:06h].

S3	Remove the shift if it does not have, in the same day, a non-break shift before or after the shift. Otherwise, sets its *start* with the *end* of the previous shift and *end* with the *start* of the next shift.	Steps: 1. Identify records of shift type *BREAK* that do not have another shift, for the same employee, that ends at the start of the shift and another that starts at the end. 2. Remove the shift if there is no shifts before or after the shift for the same day and same employee. Otherwise, set the *start* attribute with the *end* value of the previous shift and the *end* attribute with the *start* value of the next shift.
T1	Convert to the correct format.	Steps: 1. Apply the same solution described in the problem E1 for the *hour* attribute. 2. Create a new column *datetime* with the joined information from *date* and *hour*. 3. Discard columns *date* and *hour*.
T2	Information is discarded until March 2018.	

A.4 Data Preparation

A.4.0.1 Events

In this dataset, the preparations steps require:

1. One hot encoding in the *type* attribute. For this, four new binary attributes are created, corresponding to the types of events (table A.13), with value one if they are occur, or zero otherwise.

2. Compact the information into time intervals. Each row will have information about the events overlapping with that time interval.

3. Discard the *start*, *end* and *type* attributes.

A summary of the final attributes of this dataset, after the first data preparation, can be consulted in table A.23. Table A.24 shows some record examples after this process. Here, we can see that in 2019-01-10 between [10:00:00h,12:30:00h[there is a local promotion. For more examples, consult the table A.5.

Table A.23: Events dataset attributes after the first data preparation.

Attributes	Description
location	Store id.
datetime	Start date of the associated time interval.
event_calendar-local	Binary attribute corresponding to the presence or not of the event *calendal-local*.
event_calendar-public	Binary attribute corresponding to the presence or not of the event *calendal-public*.
event_event-public	Binary attribute corresponding to the presence or not of the event *event-public*.
event_promotion-local	Binary attribute corresponding to the presence or not of the event *promotion-local*.

Table A.24: Example of records in the events dataset after the first data preparation.

location	datetime	event_calendar-local	event_calendar-public	event_event-public	event_promotion-local
Store_1	2019-01-10 10:00:00	0	0	0	1
Store_1	2019-01-10 10:30:00	0	0	0	1
Store_1	2019-01-10 11:00:00	0	0	0	1
Store_1	2019-01-10 11:30:00	0	0	0	1
Store_1	2019-01-10 12:00:00	0	0	0	1

A.4.0.2 Footfall

In this dataset, the preparations steps require:

1. Compact the information into time intervals. The *footfall in* and *footfall out* attributes are associated to an one hour interval. To surpass this, and since a 30 minutes interval is being used, they are divided equally between each compacted interval.

2. Attributes *date* and *time* are discarded since this information is already available in the *datetime* attribute.

The final attributes of this dataset, after the first data prepararion process, can be consulted in table A.25. In table A.26 some examples of records are presented and we can notice that the *footfall in* between [10:00:00,10:30:00] and[11:00:00,11:30:00] was previously divided. For more record examples consult the table A.6.

Table A.25: Footfall dataset attributes after the first data preparation.

Attributes	Description
location	Store id.
datetime	Start date of the associated time interval.
footfall_in	Number of store entries.
footfall_out	Number of exists in store.

Table A.26: Example of records in the footfall dataset after the first data preparation.

location	datetime	footfall_in	footfall_out
Store_1	2019-01-10 10:00:00	2.625	0.0
Store_1	2019-01-10 10:30:00	2.625	0.0
Store_1	2019-01-10 11:00:00	12.1295	0.0
Store_1	2019-01-10 11:30:00	12.1295	0.0
Store_1	2019-01-10 12:00:00	13.6216	0.0

A.4.0.3 Schedules

In this dataset, the preparation steps are:

1. One hot encoding on the *shift type* attribute. For this, three new attributes are created, corresponding to the different tasks of a shift (table A.18). In this step, the value of each attribute is either 1, if the current employee is allocated to that task, or zero otherwise.

2. Aggregate the information into time intervals. The attributes created in the previous step are now the sum of employees assigned to each particular task, between the associated time interval.

3. Create a new attribute *total_employees* which is the sum of all employees allocated to the different tasks, for that time interval.

4. The attributes *employee code*, *date*, *start*, *end* and *shift type* are discarded.

Table A.27 presents a summary of the final attributes of this dataset after the first data preparation. In the record examples of table A.28, we can see that the number of employees on sales between [12:00:00,12:30:00[increased along with the *total_employees*. For more record examples consult the table A.7.

Table A.27: Schedules dataset attributes after the first data preparation.

Attributes	Description
location	Store id.
datetime	Start date of the associated time interval.
total_employees	Number of working employees.
n_at_sales	Number of employees allocated to 'SALES'.
n_at_others	Number of employees allocated to 'OTHERS'.
n_at_break	Number of employees allocated to 'BREAK'.

Table A.28: Example of records in the schedules dataset after the first data preparation.

location	datetime	total_employees	n_at_sales	n_at_others	n_at_break
Store_1	2019-01-10 10:00:00	5.0	3.0	2.0	0.0
Store_1	2019-01-10 10:30:00	5.0	3.0	2.0	0.0
Store_1	2019-01-10 11:00:00	5.0	3.0	2.0	0.0
Store_1	2019-01-10 11:30:00	5.0	3.0	2.0	0.0
Store_1	2019-01-10 12:00:00	6.0	4.0	2.0	0.0

A.4.0.4 Transactions

In this dataset, the preparations steps are:

1. Compact the information into time intervals and create three new attributes that records: the number of transactions, products and total profit.

2. From the newly created attributes in the previous step, create three more by calculating the average number of products per each transaction, average price per product and average sales per transaction.

3. The attributes *product*, *ticket_code*, *date*, *hour*, *employee_code*, *qty* and *net_sales* are discarded.

A summary is presented in table A.29 for the final attributes of this dataset after the first data preparation process. In table A.30 we can access to some example of records after this process is conducted. Here, we can see that, at 10:00:00 and 10:30:00 the number of transactions are 3 but the number of products is 1. This is due to the fact that two products were sold but one was returned, result in one product transacted at the end. For more data record examples consult table A.8.

Table A.29: Transactions dataset attributes after the first data preparation.

Attributes	Description
location	Store id.
datetime	Start date of the associated time interval.
n_transactions	Number of transactions (different *ticket code* values).
n_products	The sum of products transacted.
total_sales	Total sales.
avg_products_per_t	Average number of products transacted per each transaction.
avg_price_per_product	Average price per each product.
avg_sales_per_t	Average sales profit per each transaction.

Table A.30: Example of records in the transactions dataset after the first data preparation.

location	datetime	n_transactions	n_products	total_sales
		avg_products_per_t	avg_price_per_product	avg_sales_per_t
Store_1	2019-01-10 10:00:00	0	0	0.0
		0.0	0.0	0.0
Store_1	2019-01-10 10:30:00	3	1	86.0
		0.3333	86.0	28.6666
Store_1	2019-01-10 11:00:00	3	1	76.0
		0.3333	76.0	25.3333
Store_1	2019-01-10 11:30:00	2	9	585.6999
		4.5	65.07777	292.8499
Store_1	2019-01-10 12:00:00	2	2	179.0
		1.0	89.5	89.5

A.4.0.5 Shifts

In this dataset, the preparations steps are:

1. Gather the shifts by day and by employee in the schedules dataset.
2. For each set of employee shifts:
   (a) Create a new *shift_id* (increasing).
   (b) Calculate the total duration of the employee's shifts.
   (c) For each shift (tasks):
      i. Create a new record with increasing *shift_order*, taking into account the order of the tasks, with information about the *location*, *shift_id*, *shift_order*, *shift_duration*, *task_duration*, *start_time*, *end_time* and *task*.

Table A.31 contains the attributes presented in this dataset along with the description for each one. Some record examples can be consulted in table A.32, for more, consult table A.9.

Table A.31: Shifts dataset attributes.

Attributes	Description
location	Store id.
shift_id	Shift id.
shift_order	Order of task associated to the shift.
shift_duration	Total duration of the shift.
task_duration	Duration of the task.
start_time	Start time of the task.
end_time	End time of the task.
task_type	Type of task.

Table A.32: Example of records in the shifts dataset.

location	shift_id	shift_order	shift_duration	task_duration	start_time	end_time	task_type
Store_1	30298	1	0 days 09:00:00	0 days 03:00:00	2019-01-09 15:00:00	2019-01-09 18:00:00	SALES
Store_1	30298	2	0 days 09:00:00	0 days 01:00:00	2019-01-09 18:00:00	2019-01-09 19:00:00	BREAK
Store_1	30298	3	0 days 09:00:00	0 days 05:00:00	2019-01-09 19:00:00	2019-01-10 00:00:00	SALES
Store_1	30299	1	0 days 09:00:00	0 days 06:00:00	2019-01-09 10:00:00	2019-01-09 16:00:00	SALES
Store_1	30299	2	0 days 09:00:00	0 days 01:00:00	2019-01-09 16:00:00	2019-01-09 17:00:00	BREAK
Store_1	30299	3	0 days 09:00:00	0 days 02:00:00	2019-01-09 17:00:00	2019-01-09 19:00:00	SALES

A.4.0.6 Open Hours

In this dataset, the preparations steps are:

1. Calculate the number of maximum working employees in each day by accessing the schedules dataset.

2. Calculate the earliest (open hours) and the later (close hours) time between a footfall and a transaction for each day.

3. Create a new record with information about the *location*, *datetime*, *close_hours*, *open_hours*, *max_employees* and *weekday*.

Table A.33 summarizes the attributes of this dataset with a brief description and table A.34 provides some examples of records. For more records consult table A.10.

Table A.33: Open hours dataset attributes.

Attributes	**Description**
location	Store id.
datetime	Start date of the associated time interval.
close_hours	Close hours of the store.
max_employees	Maximum employees used.
open_hours	Open hours of the store.
weekday	Number of the weekday between [0,6] starting on Sunday.

Table A.34: Example of records in the open hours dataset.

location	datetime	close_hours	max_employees	open_hours	weekday
Store_1	2019-01-01	2018-12-31 10:00:00	0	2018-12-31 10:00:00	1
Store_1	2019-01-02	2019-01-02 23:30:00	0	2019-01-02 09:00:00	2
Store_1	2019-01-03	2019-01-04 00:00:00	11	2019-01-03 09:00:00	3
Store_1	2019-01-04	2019-01-05 00:00:00	11	2019-01-04 09:00:00	4

Data

Appendix B

Implementation

B.1 Forecasting

Table B.1: Input attributes for the M1 and M2 forecasting models.

Time Attributes	Target-Oriented Attributes	Others Atributes
coord_x_day_of_month	target_dm1	event_calendar-local
coord_x_day_of_season	target_dm1_tm1	event_calendar-public
coord_x_day_of_week	target_dm1_tm2	event_event-public
coord_x_day_of_year	target_dm1_tm3	event_promotion-local
coord_x_min_of_day	target_dm1_tm4	
coord_y_day_of_month	target_dm1_tm5	
coord_y_day_of_season	target_mm1	
coord_y_day_of_week	target_mm1_tm1	
coord_y_day_of_year	target_mm1_tm2	
coord_y_min_of_day	target_mm1_tm3	
	target_mm1_tm4	
	target_mm1_tm5	
	target_sm1	
	target_sm1_tm1	
	target_sm1_tm2	
	target_sm1_tm3	
	target_sm1_tm4	
	target_sm1_tm5	
	target_tm1	
	target_tm2	
	target_tm3	
	target_tm4	
	target_tm5	
	target_wm1	
	target_wm1_tm1	
	target_wm1_tm2	
	target_wm1_tm3	
	target_wm1_tm4	
	target_wm1_tm5	
	target_ym1	
	target_ym1_tm1	
	target_ym1_tm2	
	target_ym1_tm3	
	target_ym1_tm4	

	target_ym1_tm5	

Table B.2: Input attributes for the M3, M4, M5 and M6 forecasting models.

Time Attributes	**Target-Oriented Attributes**	**Others Atributes**
coord_x_day_of_month	target_dm1	event_calendar-local
coord_x_day_of_season	target_dm1_tm1	event_calendar-public
coord_x_day_of_week	target_dm1_tm2	event_event-public
coord_x_day_of_year	target_dm1_tm3	event_promotion-local
coord_x_min_of_day	target_dm1_tm4	total_employees
coord_y_day_of_month	target_dm1_tm5	footfall_in
coord_y_day_of_season	target_mm1	footfall_out
coord_y_day_of_week	target_mm1_tm1	n_at_break
coord_y_day_of_year	target_mm1_tm2	n_at_sales
coord_y_min_of_day	target_mm1_tm3	n_at_others
	target_mm1_tm4	
	target_mm1_tm5	
	target_sm1	
	target_sm1_tm1	
	target_sm1_tm2	
	target_sm1_tm3	
	target_sm1_tm4	
	target_sm1_tm5	
	target_tm1	
	target_tm2	
	target_tm3	
	target_tm4	
	target_tm5	
	target_wm1	
	target_wm1_tm1	
	target_wm1_tm2	
	target_wm1_tm3	
	target_wm1_tm4	
	target_wm1_tm5	
	target_ym1	
	target_ym1_tm1	
	target_ym1_tm2	
	target_ym1_tm3	
	target_ym1_tm4	
	target_ym1_tm5	

B.2 Optimization

B.2.1 Initialization Example

Table B.3: Constraints for an initialization example.

Optimization Constraints	**Value**
Number of planning days	1 day
Day slots	16 slots

Shift Constraints	**Value**
Tasks	[SALES, OTHERS, BREAK]
Max. shift duration	8 slots
Max. shift duration without break	4
Min. shift duration	2 slots
Break/shift duration	1/4 slots

Store Constraints	**Value**
Number of employees	5
Cost of 1 worker per slot	4
Opening hours	Day 0: [3,15]

In this section, an example is presented for the process described in section 4.2.3, which consists in generating a valid initial solution. The constraints of interest for this type of process can be visualized in table B.3, as long as their values for this example. To ease the demonstration of this type of process, only one planning day with 16 time slots is considered.

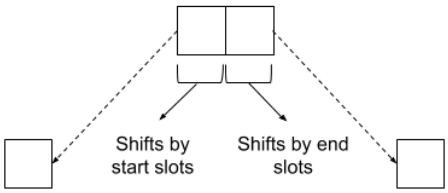
Before shifts are created to form a solution, the representation structures are initialized. Figures B.1, B.2 and B.3 show the content of the allocation, availability and shift catalog structures at iteration zero.

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
Sales	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Others	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Breaks	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Total	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure B.1: Day 0 allocation structure at iteration 0.

S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
0	0	0	5	5	5	5	5	5	5	5	5	5	5	5	5

Figure B.2: Day 0 availability structure at iteration 0.



Shift	Day	Start slot	Last slot	Break start slot	Break last slot

Figure B.3: Day 0 shifts catalog structure at iteration 0.

The process took 4 iterations to generate a valid solution and no extra shifts were created. Below, information about each iteration is presented. This is organized as the following:

1. Table containing information about the current gaps and the random gap chosen for the iteration.

2. Calculations to determine the shift's characteristics. Shift duration, start slot and break start slot are chosen randomly according to the valid range of values.

3. Table containing information about the shift created.

4. Allocation, availability and shift catalog structures after the shift creation. Changes in the structure are highlighted in red.

Iteration: 1

Table B.4: Gaps information at iteration 1.

Day	No. Iteration	Gaps List	Random Gap	Gap Duration	Type
0	1	[[3,15]]	[3,15]	13	Multiple Shift

$$
\begin{aligned}
shift_{duration} &\in [2,8] \\
shift_{duration} &= 6 \\
shift_{start_slot} &\in [3, 15+1-6] \\
&\in [3,10] \\
shift_{start_slot} &= 8 \\
shift_{last_slot} &= 8+6-1 \\
&= 13 \\
break_{duration} &= \lfloor 6 * \frac{1}{4} \rfloor \\
&= 1 \\
break_{start_slot} &\in [\lfloor 6 * \frac{1}{3} \rceil + 8, 6 - \lfloor 6 * \frac{1}{3} \rceil - 1 + 8] \\
&\in [10,11] \\
break_{start_slot} &= 10 \\
break_{last_slot} &= 10+1-1 \\
&= 10
\end{aligned}
\tag{B.1}
$$

Table B.5: Shift created at iteration 1.

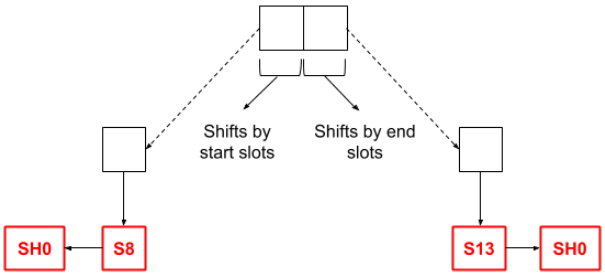
Shift	Start Slot	Last Slot	Break Start Slot	Break Last Slot
SH0	8	13	10	10

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
Sales	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	0
Others	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Breaks	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
Total	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0

Figure B.4: Day 0 allocation structure at iteration 1.

S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
0	0	0	5	5	5	5	5	4	4	4	4	4	4	5	5

Figure B.5: Day 0 availability structure in iteration 1.



Shift	Day	Start slot	Last slot	Break start slot	Break last slot
SH0	0	8	13	10	10

Figure B.6: Day 0 shifts catalog structure at iteration 1.

Iteration: 2

Table B.6: Gaps information at iteration 2.

Day	No. Iteration	Gaps List	Random Gap	Gap Duration	Type
0	2	[[3,7],[10,10],[14,15]]	[10,10]	1	Single Shift

$$
\begin{aligned}
shift_{duration} &\in [\max(1,2), \min(\max(1,8),13)] \\
&\in [2,8] \\
shift_{duration} &= 8 \\
shift_{start_slot} &\in [16-8, 16+1-8] \\
&\in [3,8] \\
shift_{start_slot} &= 3 \\
shift_{last_slot} &= 3+8-1 \\
&= 10 \\
break_{duration} &= \lfloor 8 * \frac{1}{4} \rfloor \\
&= 2 \\
break_{start_slot} &\in [\lfloor 8 * \frac{1}{3} \rceil + 3, 8 - \lfloor 8 * \frac{1}{3} \rceil - 2 + 3] \\
&\in [6,6] \\
break_{start_slot} &= 6 \\
break_{last_slot} &= 6+2-1 \\
&= 7
\end{aligned}
\tag{B.2}
$$

Table B.7: Shift created at iteration 2.

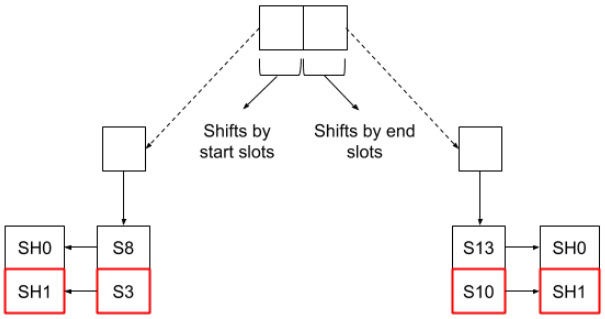
Shift	Start Slot	Last Slot	Break Start Slot	Break Last Slot
SH1	3	10	6	7

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
Sales	0	0	0	1	1	1	0	0	1	2	1	1	1	1	0	0
Others	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
Breaks	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0
Total	0	0	0	1	1	1	1	1	2	2	2	1	1	1	0	0

Figure B.7: Day 0 allocation structure at iteration 2.

S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
0	0	0	5	5	5	5	5	4	4	4	4	4	4	5	5

Figure B.8: Day 0 availability structure at iteration 2



Shift	Day	Start slot	Last slot	Break start slot	Break last slot
SH0	0	8	13	10	10
SH1	0	3	10	6	7

Figure B.9: Day 0 shifts catalog structure at iteration 2.

Iteration: 3

Table B.8: Gaps information at iteration 3.

Day	No. Iteration	Gaps List	Random Gap	Gap Duration	Type
0	3	[[6,7],[14,15]]	[14,15]	2	Single Shift

$$
\begin{aligned}
shift_{duration} &\in [\max(2,2), \min(\max(2,8),13)] \\
&\in [2,8] \\
shift_{duration} &= 3 \\
shift_{start_slot} &\in [16-3, 15+1-3] \\
&\in [13,13] \\
shift_{start_slot} &= 13 \\
shift_{last_slot} &= 13+3-1 \\
&= 15 \\
break_{duration} &= \lfloor 3 * \frac{1}{4} \rfloor \\
&= 0
\end{aligned}
\tag{B.3}
$$

Table B.9: Shift created at iteration 3.

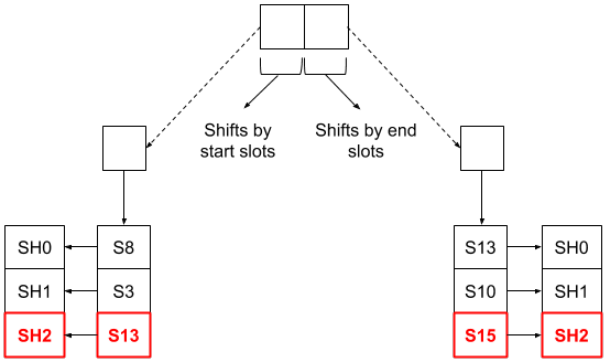
Shift	Start Slot	Last Slot	Break Start Slot	Break Last Slot
SH2	13	15	-	-

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
Sales	0	0	0	1	1	1	0	0	1	2	1	1	1	1	1	1
Others	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0
Breaks	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0
Total	0	0	0	1	1	1	1	1	2	2	2	1	1	2	1	1

Figure B.10: Day 0 allocation structure in iteration 3.

S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
0	0	0	4	4	4	4	4	3	3	3	4	4	3	4	4

Figure B.11: Day 0 availability structure at iteration 3.



Shift	Day	Start slot	Last slot	Break start slot	Break last slot
SH0	0	8	13	10	10
SH1	0	3	10	6	7
SH2	0	13	15	-	-

Figure B.12: Day 0 shifts catalog structure at iteration 3.

Iteration: 4

Table B.10: Gaps information at iteration 4.

Day	No. Iteration	Gaps List	Random Gap	Gap Duration	Type
0	4	[[6,7]]	[6,7]	2	Single Shift

$$
\begin{aligned}
shift_{duration} &\in [\max(2,2), \min(\max(2,8),13)] \\
&\in [2,8] \\
shift_{duration} &= 5 \\
shift_{start_slot} &\in [6, 7+1-5] \\
&\in [3,6] \\
shift_{start_slot} &= 5 \\
shift_{last_slot} &= 5+5-1 \\
&= 10 \\
break_{duration} &= \lfloor 5 * \frac{1}{4} \rfloor \\
&= 1 \\
break_{start_slot} &\in [\lfloor 5 * \frac{1}{3} \rceil + 6, 5 - \lfloor 5 * \frac{1}{3} \rceil - 2 + 6] \\
&\in [7,8] \\
break_{start_slot} &= 8 \\
break_{last_slot} &= 8+1-1 \\
&= 8
\end{aligned}
\tag{B.4}
$$

Table B.11: Shift created at iteration 4.

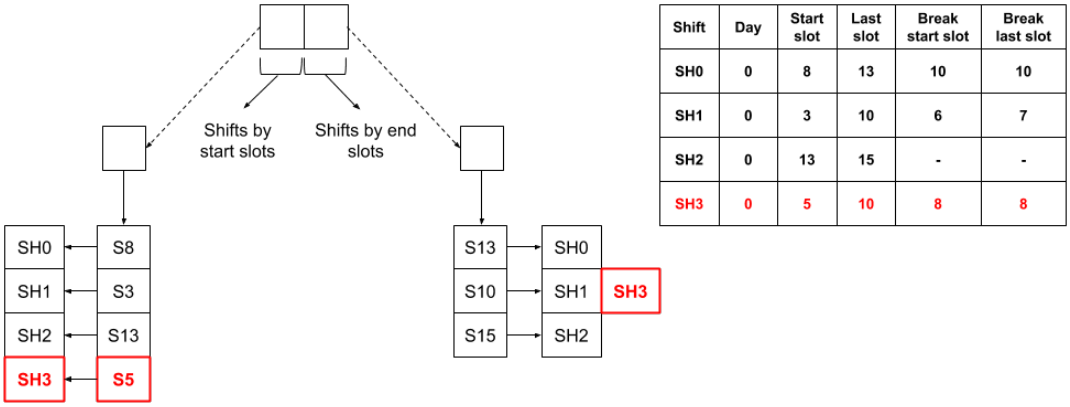
Shift	Start Slot	Last Slot	Break Start Slot	Break Last Slot
SH3	5	10	8	8

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
Sales	0	0	0	1	1	2	1	1	1	2	1	1	1	1	1	1
Others	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0
Breaks	0	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0
Total	0	0	0	1	1	2	2	2	3	3	3	1	1	2	1	1

Figure B.13: Day 0 allocation structure at iteration 4.

S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
0	0	0	4	4	3	3	3	2	2	2	4	4	3	4	4

Figure B.14: Day 0 availability structure at iteration 4.



Shift	Day	Start slot	Last slot	Break start slot	Break last slot
SH0	0	8	13	10	10
SH1	0	3	10	6	7
SH2	0	13	15	-	-
SH3	0	5	10	8	8

Figure B.15: Day 0 shifts catalog structure at iteration 4.

B.2.2 Crossover Example

In this section, an example will be given for the crossover between two individuals, parent 1 and parent 2 respectively. The constraints related to this example and their values can be consulted in

table B.12. To ease the demonstration of this type of process, only one planning day with 16 time slots is considered.

Table B.12: Constraints for a crossover example.

Optimization Constraints	**Value**
Number of planning days	1 day
Day slots	16 slots

Shift Constraints	**Value**
Tasks	[SALES, OTHERS, BREAK]
Max. shift duration	8 slots
Max. shift duration without break	4
Min. shift duration	3 slots
Break/shift duration	1/4 slots

Store Constraints	**Value**
Number of employees	5
Cost of 1 worker per slot	4
Opening hours	Day 0: [0,15]

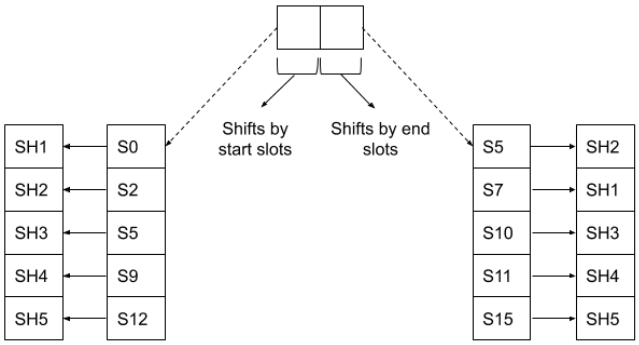
Figures B.16, B.17 and B.18 show the content of the representation structure for the parent 1. Figure B.19 shows how the shifts are distributed through the planning day.

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
Sales	1	1	1	1	1	2	2	1	1	1	2	1	1	1	1	1
Others	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0
Breaks	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0
Total	1	1	2	2	2	3	2	2	1	2	2	1	1	1	1	1

Figure B.16: Parent 1 allocation structure.

S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
4	4	3	3	3	2	3	3	4	3	3	4	4	4	4	4

Figure B.17: Parent 1 availability structure.



Shift	Day	Start slot	Last slot	Break start slot	Break last slot
SH1	0	0	7	2	3
SH2	0	2	5	-	-
SH3	0	5	10	7	7
SH4	0	9	11	-	-
SH5	0	12	15	-	-

Figure B.18: Parent 1 shift catalog structure.

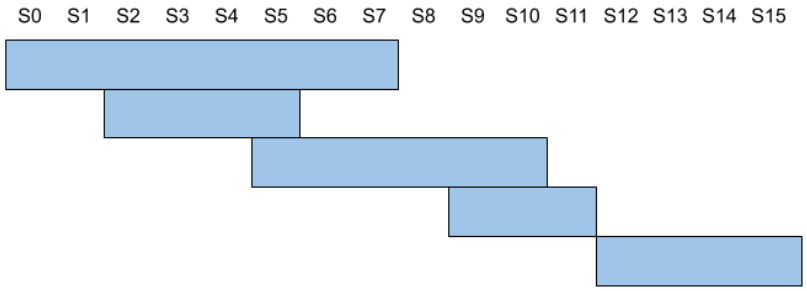


Figure B.19: Parent 1 shift visualization throughout the day.

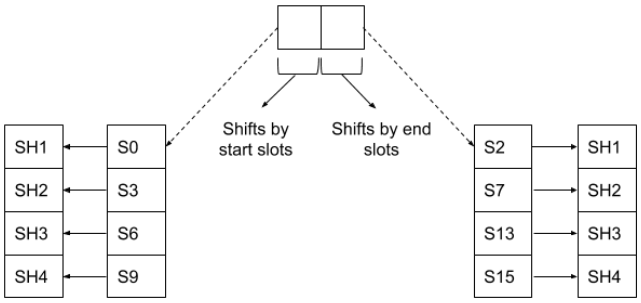
Figures B.20, B.21 and B.22 show the representation structures for parent 2. Figure B.23 shows how the shifts of this parent are distributed through the planning day time slots.

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
Sales	1	1	1	1	1	1	1	2	1	1	1	1	1	2	1	1
Others	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
Breaks	0	0	0	0	0	0	1	0	0	0	1	1	1	0	0	0
Total	1	1	1	1	1	1	2	2	1	2	2	2	2	2	1	1

Figure B.20: Parent 2 allocation structure.

S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
4	4	4	4	4	4	3	3	4	3	3	3	3	3	4	4

Figure B.21: Parent 2 availability structure.



Shift	Day	Start slot	Last slot	Break start slot	Break last slot
SH1	0	0	2	-	-
SH2	0	3	7	6	6
SH3	0	6	13	10	11
SH4	0	9	15	12	12

Figure B.22: Parent 2 shift catalog structure.

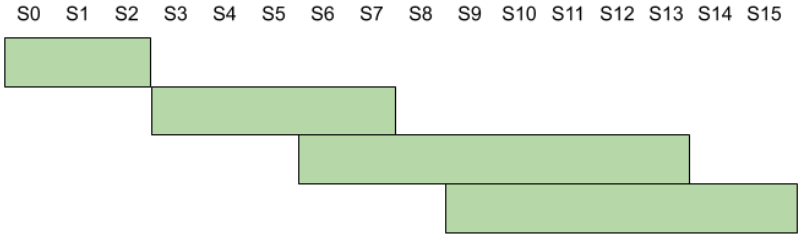


Figure B.23: Parent 2 shift visualization throughout the day.

To start the crossover process, slot 9 was randomly chosen has the slicing slot. This means that the information will be swapped between the 9^{th} and 10^{th} slot.

Table B.13: Slicing slot.

Slicing slot	9

Figure B.24 represents how shifts are split and classified after the information is sliced in parent 1, according to section 4.2.6. Table B.14 shows the characteristics of the new shifts and other details for future reference. Parent 1 is now composed by a total of 7 shifts, 3 fixed and 4 overlay. The same kind of information can be found in figure B.25 and table B.15 for parent 2. After the slice, this is composed by a total of 5 shifts, 2 fixed and 4 overlay.

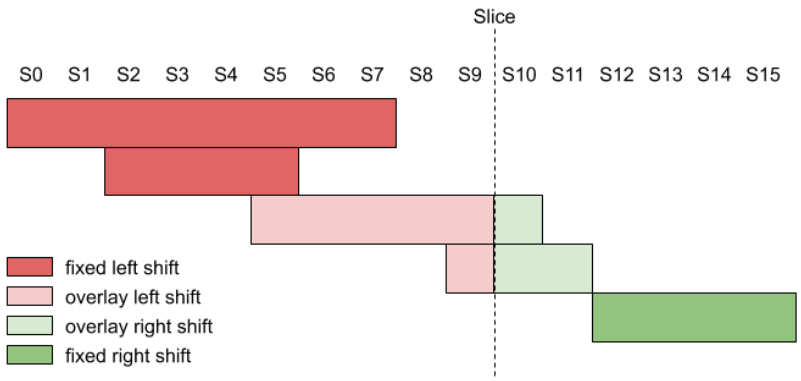


Figure B.24: Shift representation for parent 1 after the information is sliced.

Table B.14: Shift classification for parent 1 after the information is sliced.

Shift	Previous	Type	Start Slot	Last Slot	Break Start Slot	Break Last Slot
P1LF1	SH1	Left Fixed	0	7	2	3
P1LF2	SH2	Left Fixed	2	5	-	-
P1LO1	SH3	Left Overlay	5	9	-	-
P1LO2	SH4	Left Overlay	9	9	-	-
P1RF1	SH5	Right Fixed	12	15	-	-
P1RO1	SH3	Right Overlay	10	10	-	-
P1RO2	SH4	Right Overlay	10	11	-	-

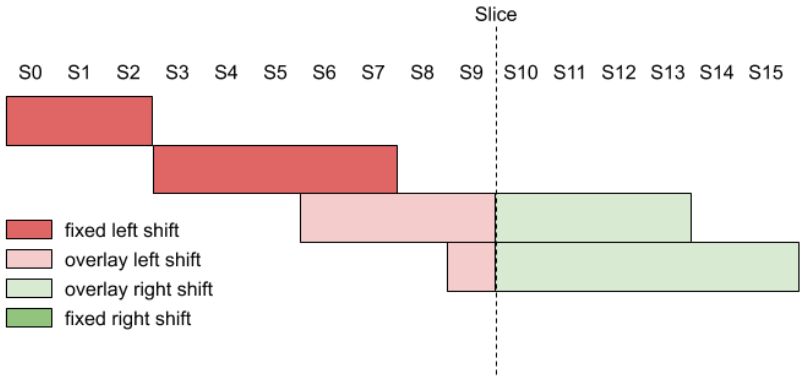


Figure B.25: Shift representation for parent 2 after the information is sliced.

Table B.15: Shift classification for parent 2 after the information is sliced.

Shift	Previous	Type	Start Slot	Last Slot	Break Start Slot	Break Last Slot
P2LF1	SH1	Left Fixed	0	2	-	-
P2LF2	SH2	Left Fixed	3	7	6	6
P2LO1	SH3	Left Overlay	6	9	-	-
P2LO2	SH4	Left Overlay	9	9	-	-
P2RO1	SH3	Right Overlay	10	13	-	-
P2RO2	SH4	Right Overlay	10	15	-	-

With the shifts defined after the slice, the children are the results of swapping the parent's information after the slicing slot. Child 1 is the result of the left shifts of parent 1 with the right ones from parent 2 (figure B.26) and child 2, the left shifts of parent 2 with the right ones from parent 1 (figure B.27).

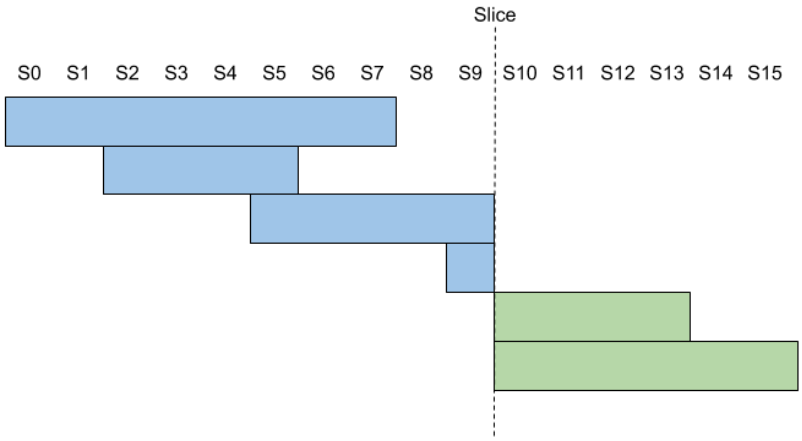


Figure B.26: Child 1 before the join of overlay shifts.

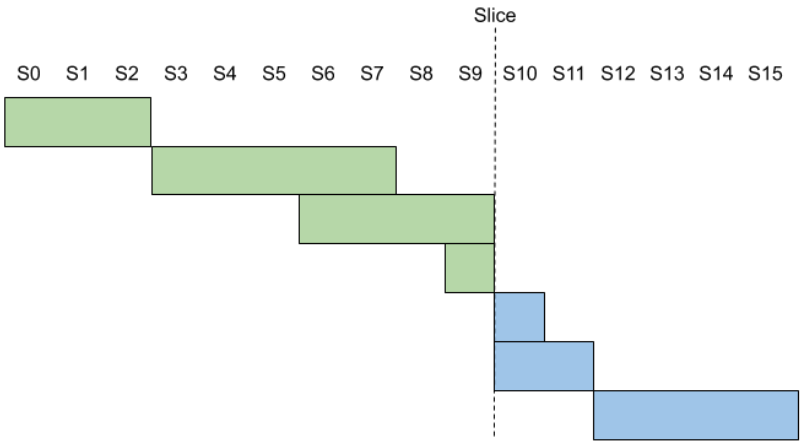


Figure B.27: Child 2 before the join of overlay shifts.

Now that we have the information of each child, the shift overlay joining process starts. In child 1, we will try to join the shifts P1LO1 and P1L02 with P2RO1 and P2RO2. And in child 2, shifts P2LO1 and P2LO2 with P1RO1 and P1RO2. For this, the mandatory and possible extends are calculating and the compatible shifts are annotated. Then, a priority order is established, according to table 4.11, and the shifts are joined by priority. If the two children are valid after the join of the overlay shifts, then the swap is successful and the representation structures are updates.

Child 1

Table B.16 shows the shift's priority, for child 1, in the join process. In table B.17 we can consult which shifts were paired with who, at each iteration. resulting in the pairs in table B.18.

Table B.16: Priority of the overlay shift of child 1 of the join process

Shift	Duration	Mandatory Extend	Possible Extend	Compatible Shifts	Priority
P1LO1	5	0	3	∅	4
P1LO2	1	2	7	[P2RO1,P2RO2]	1
P2RO1	4	0	4	[P1LO2]	3
P2RO2	6	0	2	[P1LO2]	2

Table B.17: Join process iterations for child 1.

Iteration	Selected Shift	Selected Compatible Shift	Remaining	State
1	P1LO2	P2RO2	[P1LO1,P2RO1]	Continue
2	P2RO1	∅	[P1LO1,P2RO1]	End

Table B.18: Shifts after the join process for child 1.

Shifts
P1LO2 + P2RO2
P1LO1
P2RO1

Table B.19: Shifts of child 1 without break insertion for the joined shifts.

Shift	Old Name	Start Slot	Last Slot	Break Start Slot	Break Last Slot
SH1	P1LF1	0	7	2	3
SH2	P1LF2	2	5	-	-
SH3	P1LO2 + P2RO2	9	15	-	-
SH4	P1LO1	5	9	-	-
SH5	P2RO1	10	13	-	-

Child 2

Table B.16 shows the shift's priority, for child 2, in the join process. In table B.17 we can consult which shifts were paired with who, at each iteration. resulting in the pairs in table B.18.

Table B.20: Priority of the overlay shift of child 2 of the join process

Shift	Duration	Mandatory Extend	Possible Extend	Compatible Shifts	Priority
P2LO1	4	0	4	[P1RO1,P1RO2]	4
P2LO2	1	2	7	[P1RO2]	1
P1RO1	1	2	7	[P2LO1]	2
P1RO2	2	1	6	[P2LO1,P2LO2]	3

Table B.21: Join process iterations for child 2.

Iteration	Selected Shift	Selected Compatible Shift	Remaining	State
1	P2LO2	P1RO2	[P2LO1,P1RO1]	Continue
2	P1RO1	P2LO1	$\emptyset$	End

Table B.22: Shifts after the join process for child 2.

Shifts
P2LO2 + P1RO2
P1RO1 + P2LO1

Table B.23: Shifts of child 2 without break insertion for the joined shifts.

Shift	Old Name	Start Slot	Last Slot	Break Start Slot	Break Last Slot
SH1	P2LF1	0	2	-	-
SH2	P2LF2	3	7	6	6
SH3	P2LO2 + P1RO2	9	11	-	-
SH4	P1RO1 + P2LO1	6	10	-	-
SH5	P1RF1	12	15	-	-

Tables B.19 and B.23 shows the characteristics of the shifts after the join process. Both children are valid since all constraints are fulfilled (section 4.2.1), so, we now can update the breaks for the joined shifts and update the representation structures. The shifts that require breaks are SH2 and SH3 in child 1 and SH3 in child 2. Below, the steps for the insertion of the breaks are presented.

In child 1, the break for shift SH2 can range between [11,13], which are all available for break allocation (calculations B.5). The random break start slot chosen was slot 12.

$$
\begin{aligned}
break_{duration} &= \lfloor \frac{1}{4} * 7 \rfloor \\
&= 1 \\
shift_{break_start_slot} &\in [\lfloor 7 * \frac{1}{3} \rceil + 9, 7 - \lfloor 7 * \frac{1}{3} \rceil - 1 + 9] \\
&\in [11, 13] \\
feasible_{slots} &= [[11, 13]] \rightarrow break \\
shift_{break_start_slot} &= random(11, 13) \\
&= 12 \\
shift_{break_last_slot} &= 12 + 1 - 1 \\
&= 12
\end{aligned}
\tag{B.5}
$$

The break for shift SH3, for the child 1, can range between [7,7], which are also available for break allocation (calculations B.6). Therefore, 7 was the chosen slot for the break start slot.

$$\begin{aligned} break_{duration} &= \lfloor \frac{1}{4} * 5 \rfloor \\ &= 1 \\ shift_{break_start_slot} &\in [\lfloor 5 * \frac{1}{3} \rceil + 5, 5 - \lfloor 5 * \frac{1}{3} \rceil - 1 + 5] \\ &\in [7,7] \\ feasible_{slots} &= [[7,7]] \rightarrow break \\ shift_{break_start_slot} &= random(7,7) \\ &= 7 \\ shift_{break_last_slot} &= 7 + 1 - 1 \\ &= 7 \end{aligned} \tag{B.6}$$

In child 2, the shift SH3 has [8,8] has the possible slot range for the break starting slot (calculations B.7. However, none of these are feasible since no other shifts guarantee coverage on sales. Thus, no break is allocated.

$$\begin{aligned} break_{duration} &= \lfloor \frac{1}{4} * 5 \rfloor \\ &= 1 \\ shift_{break_start_slot} &\in [\lfloor 5 * \frac{1}{3} \rceil + 6, 5 - \lfloor 5 * \frac{1}{3} \rceil - 1 + 6] \\ &\in [8,8] \\ feasible_{slots} &= \emptyset \rightarrow NoBreak \end{aligned} \tag{B.7}$$

Tables B.24 and B.25 show the shifts characteristics, for child 1 and child 2, after the feasible and necessary breaks are inserted.

Table B.24: Final shifts of child 1.

Shift	Start Slot	Last Slot	Break Start Slot	Break Last Slot
SH1	0	7	2	3
SH2	2	5	-	-
SH3	9	15	12	12
SH4	5	9	7	7
SH5	10	13	-	-

Table B.25: Final shifts of child 2.

Shift	Start Slot	Last Slot	Break Start Slot	Break Last Slot
SH1	0	2	-	-
SH2	3	7	6	6
SH3	9	11	-	-
SH4	6	10	-	-
SH5	12	15	-	-

After the representations are updated, the content of the allocation, availability and shift catalog, for child 1, are represented by figures B.28, B.29 and B.30. The blue cells are related to parent's 1 information and the green cells, related to parent 2. The same kind of information is also available for child 2 in figures B.32, B.33 and B.34. For a better understanding of how shifts spawn through the planning day, figures B.31 and B.35 can be consulted for parent 1 and parent, respectively.

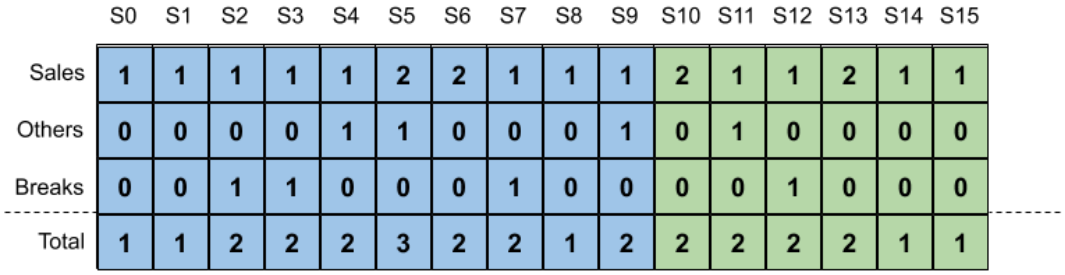


Figure B.28: Child 1 allocation structure.

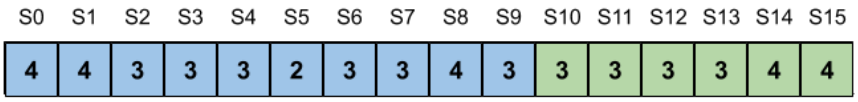


Figure B.29: Child 1 availability structure.

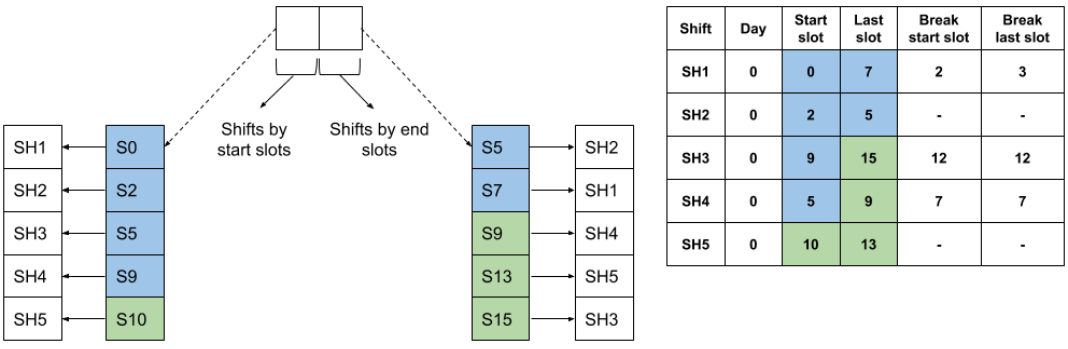


Figure B.30: Child 1 shift catalog structure.

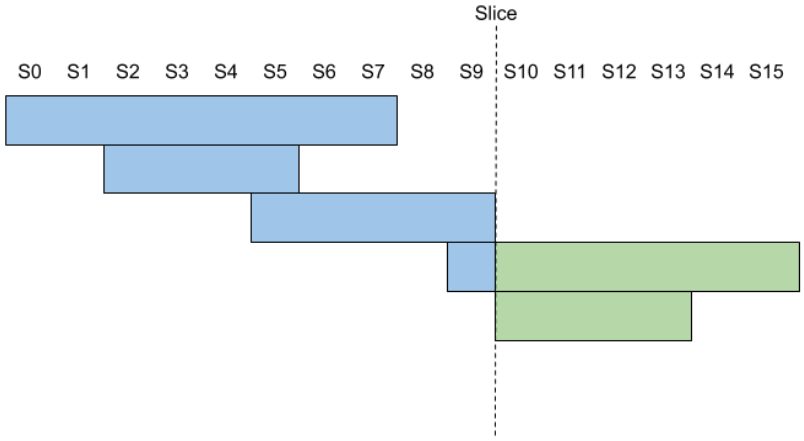


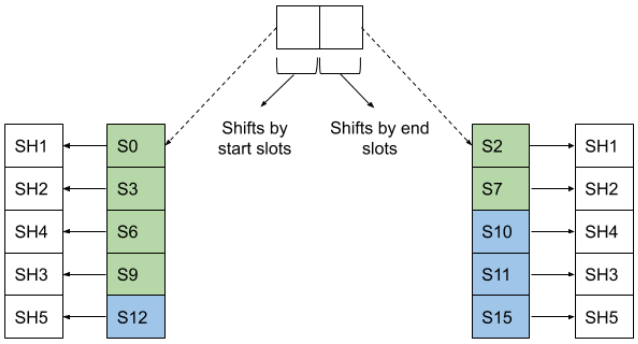
Figure B.31: Child 1 shifts visualization throughout the day.

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
Sales	1	1	1	1	1	1	1	2	1	1	2	1	1	1	1	1
Others	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
Breaks	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
Total	1	1	1	1	1	1	2	2	1	2	2	1	1	1	1	1

Figure B.32: Child 2 allocation structure.

S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
4	4	4	4	4	4	3	3	4	3	3	4	4	4	4	4

Figure B.33: Child 2 availability structure.



Shift	Day	Start slot	Last slot	Break start slot	Break last slot
SH1	0	0	2	-	-
SH2	0	3	7	6	6
SH3	0	9	11	-	-
SH4	0	6	10	-	-
SH5	0	12	15	-	-

Figure B.34: Child 2 shift catalog structure.

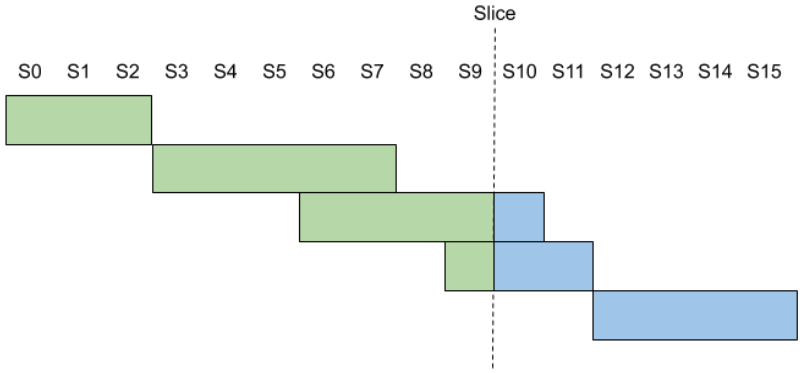


Figure B.35: Child 2 shifts visualization throughout the day.

B.2.3 Mutation Example

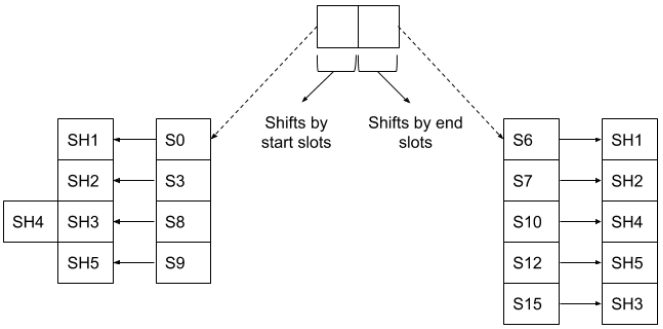
In this section, several examples will be given for the application of different types of mutation (section 4.2.7). The constraints related to this example and their values, are the same for the crossover example and can be consulted in table B.12. The same individual will be used as an example of the application of each mutation. Figures B.36, B.37 and B.38 show the content of the representation structures for this individual.

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
Sales	1	1	1	1	2	1	1	1	2	2	1	1	1	1	1	1
Others	0	0	0	0	0	0	1	0	0	1	2	0	0	0	0	0
Breaks	0	0	0	1	0	1	0	0	0	0	0	1	1	0	0	0
Total	1	1	1	2	2	2	2	1	2	3	3	2	2	1	1	1

Figure B.36: Individual allocation structure.

S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
4	4	4	3	3	3	3	4	3	2	2	3	3	4	4	4

Figure B.37: Individual availability structure.



Shift	Day	Start slot	Last slot	Break start slot	Break last slot
SH1	0	0	6	3	3
SH2	0	3	7	5	5
SH3	0	8	15	11	12
SH4	0	8	10	-	-
SH5	0	9	12	-	-

Figure B.38: Individual shift catalog structure.

To understand which mutation types are valid, for the different shifts, and in which slots, table B.26 summarizes this information. In table B.27 the valid slots for the reallocate mutation are presented.

Table B.26: Valid mutation slots and their candidates for the different mutation types.

	SH1	**SH2**	**SH3**	**SH4**	**SH5**
Reduce Start	-	3	8	8	9
Reduce Last	6	-	-	10	-
Extend Start	-	2	-	7	8
Extend Last	7	8	-	11	13
Break	2-4	-	10-13	-	-
Join	-	7-8	-	7-8	-
Remove	-	-	-	8-10	-

Table B.27: Valid slots for the reallocate mutation.

Reallocate	4,6,8,9,10

In the subsections below, examples are given for each mutation type, one per subsection. Each example is organized as the following:

- A table containing the chosen mutation slot and the candidates for the mutation, according to tables B.26 and B.27.
- Calculations for the shift's characteristics and break adjustment.
- Figures showing the content of the representation structures of the individual, after the mutation. Changes in the structures are highlighted in red.

B.2.3.1 Mutation: Reduce Start

Table B.28: Characteristics of the *reduce start* mutation.

Mutation Slot	Candidates	Selected Candidate	Total Duration
8	SH3,SH4	SH3	8

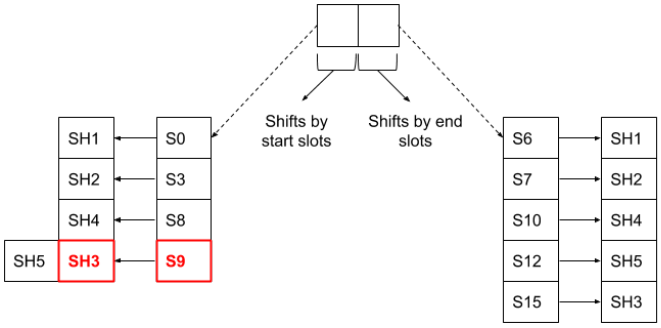
$$
\begin{aligned}
shift_{start_slot} &= 8+1 \\
&= 9 \\
shift_{duration} &= 15-9+1 \\
&= 7 \\
shift_{break_duration} &= \lfloor 7 * \frac{1}{4} \rfloor \\
&= 1 \rightarrow Reducebreak \\
slots_to_reduce &= 2-1 \\
&= 1 \\
shift_{break_start_slot} - 1 &\rightarrow Valid \\
distance_break_start &= 11-9 \\
&= 2 \rightarrow Closest \rightarrow RemoveSlot \\
shift_{break_last_slot} + 1 &\rightarrow Valid \\
distance_break_last &= 15-12 \\
&= 1
\end{aligned}
\tag{B.8}
$$

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
Sales	1	1	1	1	2	1	1	1	1	2	1	2	1	1	1	1
Others	0	0	0	0	0	0	1	0	0	1	2	0	0	0	0	0
Breaks	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0
Total	1	1	1	2	2	2	2	1	1	3	3	2	2	1	1	1

Figure B.39: Individual allocation structure after the *reduce start* mutation on slot 8.

S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
4	4	4	3	3	3	3	4	4	2	2	3	3	4	4	4

Figure B.40: Individual availability structure after the *reduce start* mutation on slot 8.



Shift	Day	Start slot	Last slot	Break start slot	Break last slot
SH1	0	0	6	3	3
SH2	0	3	7	5	5
SH3	0	9	15	12	12
SH4	0	8	10	-	-
SH5	0	9	12	-	-

Figure B.41: Individual shift catalog structure after the *reduce start* mutation on slot 8.

B.2.3.2 Mutation: Reduce Last

Table B.29: Characteristics of the *reduce last* mutation.

Mutation Slot	Candidates	Selected Candidate	Total Duration
6	SH1	SH1	7

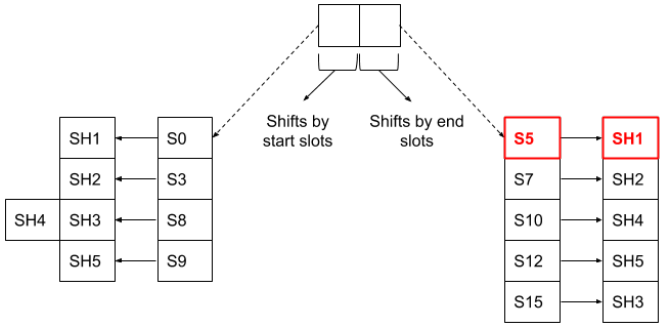
$$
\begin{aligned}
shift_{last_slot} &= 6-1 \\
&= 5 \\
shift_{duration} &= 5+1 \\
&= 6 \\
shift_{break_duration} &= \lfloor 6 * \frac{1}{4} \rfloor \\
&= 1 \rightarrow NoAdjustment
\end{aligned}
\tag{B.9}
$$

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
Sales	1	1	1	1	2	1	1	1	2	2	1	1	1	1	1	1
Others	0	0	0	0	0	0	**0**	0	0	1	2	0	0	0	0	0
Breaks	0	0	0	1	0	1	0	0	0	0	0	1	1	0	0	0
Total	1	1	1	2	2	2	**1**	1	2	3	3	2	2	1	1	1

Figure B.42: Individual allocation structure after the *reduce last* mutation on slot 6.

S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
4	4	4	3	3	3	**4**	4	3	2	2	3	3	4	4	4

Figure B.43: Individual availability structure after the *reduce last* mutation on slot 6.



Shift	Day	Start slot	Last slot	Break start slot	Break last slot
SH1	0	0	5	3	3
SH2	0	3	7	5	5
SH3	0	8	15	11	12
SH4	0	8	10	-	-
SH5	0	9	12	-	-

Figure B.44: Individual shift catalog structure after the *reduce last* mutation on slot 6.

B.2.3.3 Mutation: Extend Start

Table B.30: Characteristics of the *extend start* mutation.

Mutation Slot	Candidates	Selected Candidate	Total Duration
8	SH5	SH5	4

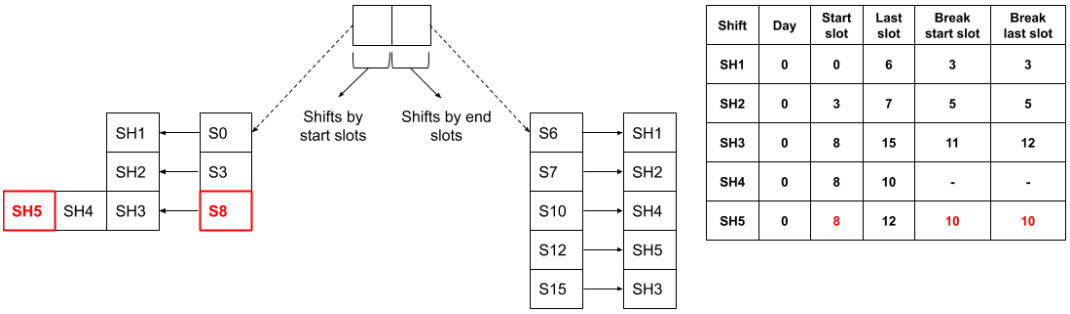
$$
\begin{aligned}
shift_{start_slot} &= 9-1 \\
&= 8 \\
shift_{duration} &= 12-8+1 \\
&= 5 \\
shift_{break_duration} &= \lfloor 5 * \frac{1}{4} \rfloor \\
&= 1 \rightarrow Addbreak \\
shift_{start_slot} &\in [\lfloor 5 * \frac{1}{3} \rceil + 8, 5 - \lfloor 5 * \frac{1}{3} \rceil - 1 + 8] \\
&\in [9,11] \\
feasible_{slots} &= [[9,10]] \\
shift_{break_start_slot} &= random(9,10) \\
&= 10 \\
shift_{break_last_slot} &= 10+1-1 \\
&= 10
\end{aligned}
\tag{B.10}
$$

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
Sales	1	1	1	1	2	1	1	1	2	2	1	1	1	1	1	1
Others	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0
Breaks	0	0	0	1	0	1	0	0	0	0	1	1	1	0	0	0
Total	1	1	1	2	2	2	2	1	3	3	3	2	2	1	1	1

Figure B.45: Individual allocation structure after the *extend start* mutation on slot 8.

S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
4	4	4	3	3	3	3	4	2	2	2	3	3	4	4	4

Figure B.46: Individual availability structure after the *extend start* mutation on slot 8.



Shift	Day	Start slot	Last slot	Break start slot	Break last slot
SH1	0	0	6	3	3
SH2	0	3	7	5	5
SH3	0	8	15	11	12
SH4	0	8	10	-	-
SH5	0	8	12	10	10

Figure B.47: Individual shift catalog structure after the *extend start* mutation on slot 8.

B.2.3.4 Mutation: Extend Last

Table B.31: Characteristics of the *extend last* mutation.

Mutation Slot	Candidates	Selected Candidate	Total Duration
7	SH1	SH1	7

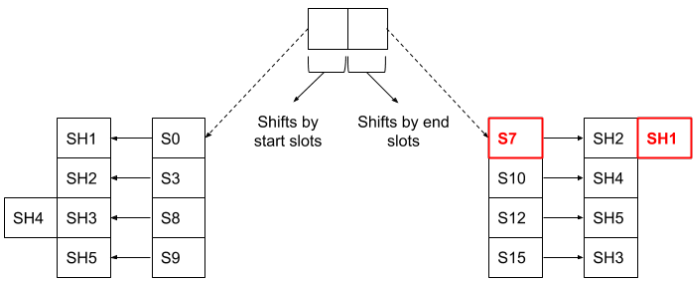
$$
\begin{aligned}
shift_{last_slot} &= 6+1 \\
&= 7 \\
shift_{duration} &= 7-0+1 \\
&= 8 \\
shift_{break_duration} &= \lfloor 8 * \frac{1}{4} \rfloor \\
&= 2 \rightarrow Extendbreak \\
slots_to_extend &= 2-1 \\
&= 1 \\
shift_{break_start_slot} - 1 &\rightarrow Invalid \\
shift_{break_last_slot} + 1 &\rightarrow Valid \\
distance_break_last &= 7-3 \\
&= 4 \rightarrow Farthest \rightarrow ExtendSlot
\end{aligned}
\tag{B.11}
$$

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
Sales	1	1	1	1	**1**	1	1	**2**	2	2	1	1	1	1	1	1
Others	0	0	0	0	0	0	1	0	0	1	2	0	0	0	0	0
Breaks	0	0	0	1	**1**	1	0	0	0	0	0	1	1	0	0	0
Total	1	1	1	2	2	2	2	**2**	2	3	3	2	2	1	1	1

Figure B.48: Individual allocation structure after the *extend last* mutation on slot 7.

S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
4	4	4	3	3	3	3	**3**	3	2	2	3	3	4	4	4

Figure B.49: Individual availability structure after the *extend last* mutation on slot 7.



Shift	Day	Start slot	Last slot	Break start slot	Break last slot
SH1	0	0	7	3	4
SH2	0	3	7	5	5
SH3	0	8	15	11	12
SH4	0	8	10	-	-
SH5	0	9	12	-	-

Figure B.50: Individual shift catalog structure after the *extend last* mutation on slot 7.

B.2.3.5 Mutation: Break

Table B.32: Characteristics of the *break* mutation.

Mutation Slot	Candidates	Selected Candidate	Total Duration
11	SH3	SH3	8

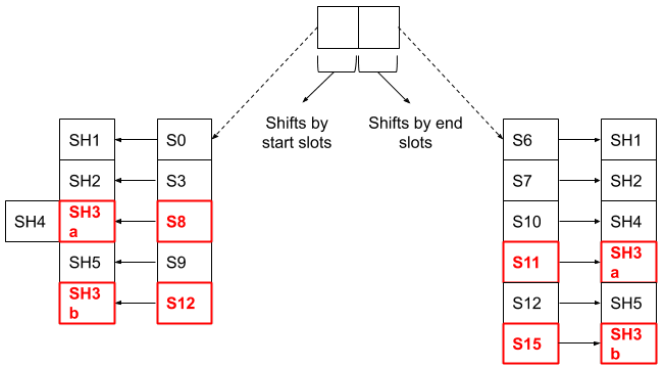
$$
\begin{aligned}
break &\in [10-11, 11-12] \rightarrow AllValid \\
break &= 11-12 \\
shift_{1,start_slot} &= 8 \\
shift_{1,last_slot} &= 11 \\
shift_{1,duration} &= 11-8+1 \\
&= 4 \\
shift_{1,break_duration} &= 0 \rightarrow NoBreak \\
shift_{2,start_slot} &= 12 \\
shift_{2,last_slot} &= 15 \\
shift_{2,duration} &= 15-12+1 \\
&= 4 \\
shift_{2,break_duration} &= 0 \rightarrow NoBreak
\end{aligned}
\tag{B.12}
$$

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
Sales	1	1	1	1	2	1	1	1	2	2	1	1	**2**	1	1	1
Others	0	0	0	0	0	0	1	0	0	1	2	**1**	0	0	0	0
Breaks	0	0	0	1	0	1	0	0	0	0	0	**0**	**0**	0	0	0
Total	1	1	1	2	2	2	2	1	2	3	3	2	2	1	1	1

Figure B.51: Individual allocation structure after the *break* mutation on slot 11.

S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
4	4	4	3	3	3	3	4	3	2	2	3	3	4	4	4

Figure B.52: Individual availability structure after the *break* mutation on slot 11.



Shift	Day	Start slot	Last slot	Break start slot	Break last slot
SH1	0	0	6	3	3
SH2	0	3	7	5	5
SH3a	0	8	11	-	-
SH3b	0	12	15	-	-
SH4	0	8	10	-	-
SH5	0	9	12	-	-

Figure B.53: Individual shift catalog structure after the *break* mutation on slot 11.

B.2.3.6 Mutation: Join

Table B.33: Characteristics of the *join* mutation.

Mutation Slot	Candidates	Selected Candidate	Total Duration
7	SH2-SH4	SH2-SH4	5+3

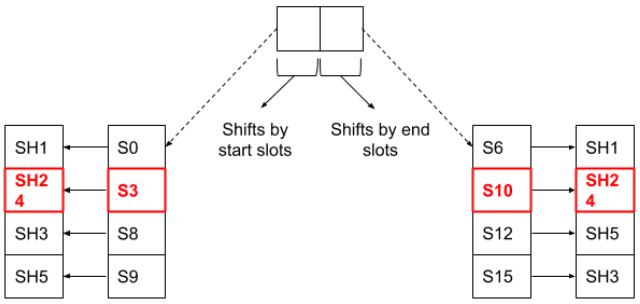
$$
\begin{aligned}
shift_{start_slot} &= 3 \\
shift_{last_slot} &= 10 \\
shift_{duration} &= 10-3+1 \\
&= 8 \\
shift_{break_duration} &= \lfloor 8 * \frac{1}{4} \rfloor \\
&= 2 \rightarrow Addbreak \\
shift_{break_start_slot} &\in [\lfloor 8 * \frac{1}{3} \rceil + 3, 8 - \lfloor 8 * \frac{1}{3} \rceil] - 2 + 3] \\
&\in [6,6] \\
feasible_{slots} &= [[6,6]] \rightarrow shift_{break_duration} = 1 \\
shift_{break_start_slot} &= random(6,6) \\
&= 6 \\
shift_{break_last_slot} &= 6+1-1 \\
&= 6
\end{aligned}
\tag{B.13}
$$

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
Sales	1	1	1	1	2	1	1	1	2	2	1	1	1	1	1	1
Others	0	0	0	0	0	**1**	**0**	0	0	1	2	0	0	0	0	0
Breaks	0	0	0	1	0	**0**	**1**	0	0	0	0	1	1	0	0	0
Total	1	1	1	2	2	2	2	1	2	3	3	2	2	1	1	1

Figure B.54: Individual allocation structure after the *join* mutation on slot 7.

S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
4	4	4	3	3	3	3	4	3	2	2	3	3	4	4	4

Figure B.55: Individual availability structure after the *join* mutation on slot 7.



Shift	Day	Start slot	Last slot	Break start slot	Break last slot
SH1	0	0	6	3	3
SH24	0	3	10	6	6
SH3	0	8	15	11	12
SH5	0	9	12	-	-

Figure B.56: Individual shift catalog structure after the *join* mutation on slot 7.

B.2.3.7 Mutation: Remove

Table B.34: Characteristics of the *remove* mutation.

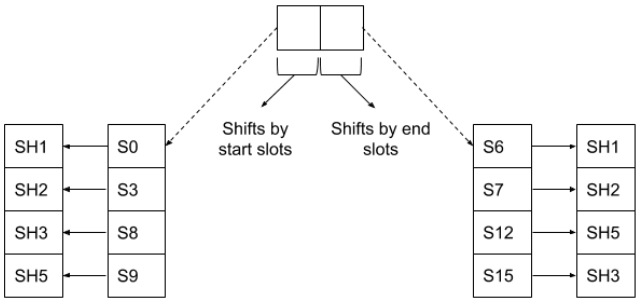
Mutation Slot	Candidates	Selected Candidate	Total Duration
10	SH4	SH4	3

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
Sales	1	1	1	1	2	1	1	1	**1**	2	1	1	1	1	1	1
Others	0	0	0	0	0	0	1	0	0	**0**	**1**	0	0	0	0	0
Breaks	0	0	0	1	0	1	0	0	0	0	0	1	1	0	0	0
Total	1	1	1	2	2	2	2	1	**1**	**2**	**2**	2	2	1	1	1

Figure B.57: Individual allocation structure after the *remove* mutation on slot 10.

S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
4	4	4	3	3	3	3	4	4	3	3	3	3	4	4	4

Figure B.58: Individual availability structure after the *remove* mutation on slot 10.



Shift	Day	Start slot	Last slot	Break start slot	Break last slot
SH1	0	0	6	3	3
SH2	0	3	7	5	5
SH3	0	8	15	11	12
SH5	0	9	12	-	-

Figure B.59: Individual shift catalog structure after the *remove* mutation on slot 10.

B.2.3.8 Mutation: Reallocate

Table B.35: Characteristics of the *reallocate* mutation.

Mutation Slot	Candidates	Selected Candidate	Random Allocation
10	OTHERS	OTHERS	2

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
Sales	1	1	1	1	2	1	1	1	2	2	3	1	1	1	1	1
Others	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0
Breaks	0	0	0	1	0	1	0	0	0	0	0	1	1	0	0	0
Total	1	1	1	2	2	2	2	1	2	3	3	2	2	1	1	1

Figure B.60: Individual allocation structure after the *reallocate* mutation on slot 10.

Appendix C

Results

C.1 Forecasting

C.1.1 Experiment F1

Table C.1: Importance of the input attributes in the M1 and M2 forecasting models.

Attribute	M1 Importance	M2 Importance
coord_x_day_of_month	0.001563	0.0
coord_x_day_of_season	0.001615	0.0
coord_x_day_of_week	0.001079	0.0
coord_x_day_of_year	0.001995	0.0
coord_x_min_of_day	0.004893	0.0
coord_y_day_of_month	0.001355	0.0
coord_y_day_of_season	0.002026	0.0
coord_y_day_of_week	0.000503	0.0
coord_y_day_of_year	0.003248	0.0
coord_y_min_of_day	0.035984	0.0
event_calendar-local	0.000025	0.0
event_calendar-public	0.000230	0.0
event_event-public	0.000095	0.0
event_promotion-local	0.000144	0.0
target_dm1	0.016031	0.0
target_dm1_tm1	0.001548	0.0
target_dm1_tm2	0.002046	0.0
target_dm1_tm3	0.001094	0.0
target_dm1_tm4	0.001507	0.0
target_dm1_tm5	0.001131	0.0
target_mm1	0.001845	0.0
target_mm1_tm1	0.000887	0.0
target_mm1_tm2	0.001433	0.0
target_mm1_tm3	0.000871	0.0
target_mm1_tm4	0.001268	0.0
target_mm1_tm5	0.000982	0.0
target_sm1	0.002820	0.0
target_sm1_tm1	0.000834	0.0
target_sm1_tm2	0.001430	0.0
target_sm1_tm3	0.000968	0.0
target_sm1_tm4	0.001313	0.0
target_sm1_tm5	0.001043	0.0

target_tm1	0.866396	0.0
target_tm2	0.007535	0.0
target_tm3	0.003641	0.0
target_tm4	0.003274	0.0
target_tm5	0.002399	0.0
target_wm1	0.006289	0.0
target_wm1_tm1	0.002133	0.0
target_wm1_tm2	0.002041	0.0
target_wm1_tm3	0.001103	0.0
target_wm1_tm4	0.001306	0.0
target_wm1_tm5	0.001249	0.0
target_ym1	0.003692	0.0
target_ym1_tm1	0.000884	0.0
target_ym1_tm2	0.001465	0.0
target_ym1_tm3	0.000857	0.0
target_ym1_tm4	0.001070	0.0
target_ym1_tm5	0.000861	0.0

Table C.2: Importance of the input attributes in the M2, M4, M5 and M6 forecasting models.

Attribute	M3 Importance	M4 Importance	M5 Importance	M6 Importance
coord_x_day_of_month	0.007601	0.037381	0.027026	0.009509
coord_x_day_of_season	0.006794	0.011729	0.012089	0.007517
coord_x_day_of_week	0.006296	0.012320	0.009618	0.006414
coord_x_day_of_year	0.019173	0.010554	0.008709	0.015715
coord_x_min_of_day	0.006281	0.010265	0.009059	0.005431
coord_y_day_of_month	0.006269	0.009195	0.008257	0.006004
coord_y_day_of_season	0.011543	0.013684	0.014248	0.014860
coord_y_day_of_week	0.002550	0.010430	0.009427	0.002391
coord_y_day_of_year	0.011015	0.009715	0.009819	0.009614
coord_y_min_of_day	0.013256	0.008737	0.009275	0.013149
employees_on_store	0.002446	0.008766	0.007543	0.002549
event_calendar-local	0.000021	0.008919	0.006816	0.000060
event_calendar-public	0.000447	0.014217	0.048002	0.000521
event_event-public	0.000747	0.008759	0.008222	0.000112
event_promotion-local	0.000538	0.007106	0.009211	0.000678
footfall_in	0.579904	0.006793	0.007208	0.483238
footfall_out	0.000000	0.005196	0.007714	0.000000
n_at_break	0.002124	0.006409	0.006992	0.002496
n_at_others	0.001032	0.438816	0.438510	0.001504
n_at_sales	0.003834	0.012711	0.017995	0.004124
total_employees	0.015749	0.012436	0.012965	0.005818
target_dm1	0.006849	0.013809	0.011553	0.025690
target_dm1_tm1	0.007576	0.013447	0.009981	0.017009
target_dm1_tm2	0.008184	0.038013	0.030110	0.009371
target_dm1_tm3	0.005483	0.011495	0.011561	0.010318
target_dm1_tm4	0.005076	0.010135	0.013683	0.010683
target_dm1_tm5	0.006974	0.010713	0.009079	0.008131
target_mm1	0.005527	0.009243	0.008470	0.010446
target_mm1_tm1	0.005511	0.008140	0.008179	0.010067
target_mm1_tm2	0.004765	0.000000	0.000000	0.009596
target_mm1_tm3	0.004874	0.000000	0.000000	0.007965
target_mm1_tm4	0.004759	0.000000	0.000000	0.007498
target_mm1_tm5	0.005815	0.000000	0.000000	0.006873
target_sm1	0.005499	0.000000	0.000000	0.008495

target_sm1_tm1	0.004506	0.000000	0.000000	0.006449
target_sm1_tm2	0.004164	0.012407	0.010507	0.006802
target_sm1_tm3	0.003631	0.012578	0.012773	0.005372
target_sm1_tm4	0.003756	0.006480	0.006582	0.006942
target_sm1_tm5	0.125297	0.025027	0.022182	0.006781
target_tm1	0.019619	0.012058	0.008357	0.108197
target_tm2	0.007705	0.010874	0.009122	0.027858
target_tm3	0.008206	0.015028	0.011066	0.014104
target_tm4	0.006800	0.004988	0.004153	0.013464
target_tm5	0.007125	0.021195	0.019739	0.010080
target_wm1	0.006510	0.043859	0.038491	0.012275
target_wm1_tm1	0.006756	0.003665	0.003236	0.010037
target_wm1_tm2	0.006345	0.000188	0.000008	0.010963
target_wm1_tm3	0.005418	0.000510	0.000359	0.009600
target_wm1_tm4	0.005077	0.000126	0.000264	0.008670
target_wm1_tm5	0.000000	0.000736	0.000599	0.008558
target_ym1	0.000000	0.037192	0.039977	0.000000
target_ym1_tm1	0.000000	0.000000	0.000000	0.000000
target_ym1_tm2	0.000000	0.001945	0.001006	0.000000
target_ym1_tm3	0.000000	0.002275	0.002188	0.000000
target_ym1_tm4	0.000000	0.004277	0.008447	0.000000
target_ym1_tm5	0.004574	0.005459	0.009621	0.000000

C.1.2 Experiment F3

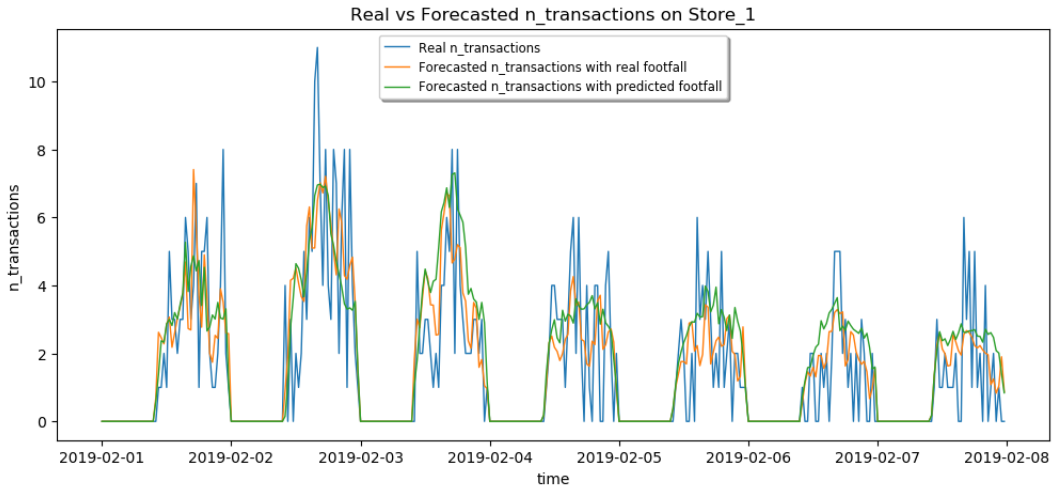


Figure C.1: Real versus forecasted number of transactions using accurate and forecasted footfall.

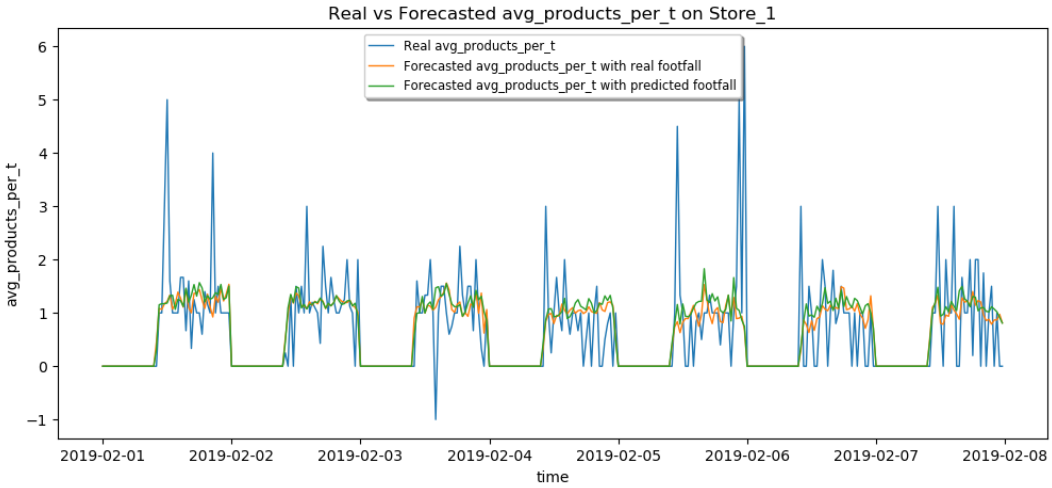


Figure C.2: Real versus forecasted average products per transaction using accurate and forecasted footfall.

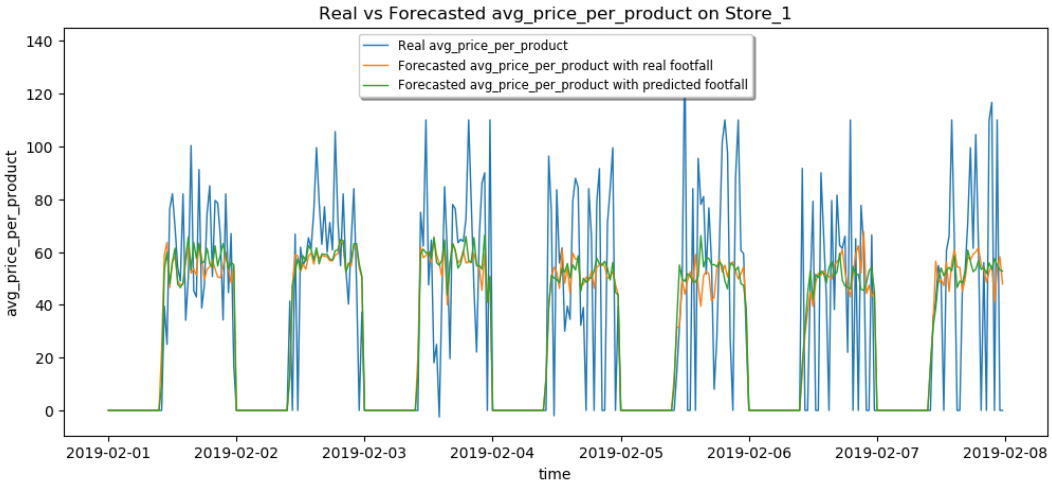


Figure C.3: Real versus forecasted average price per product using accurate and forecasted footfall.

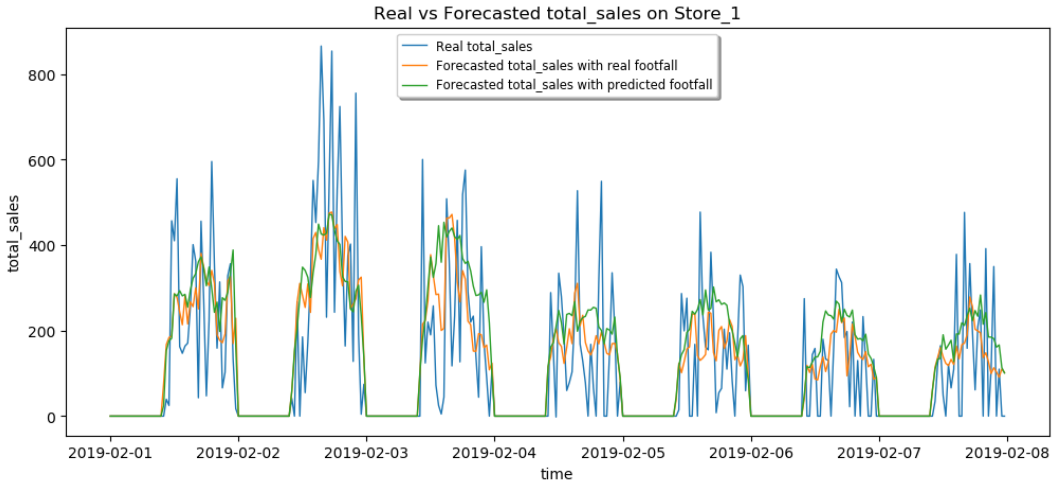


Figure C.4: Real versus forecasted direct sales using accurate and forecasted footfall.

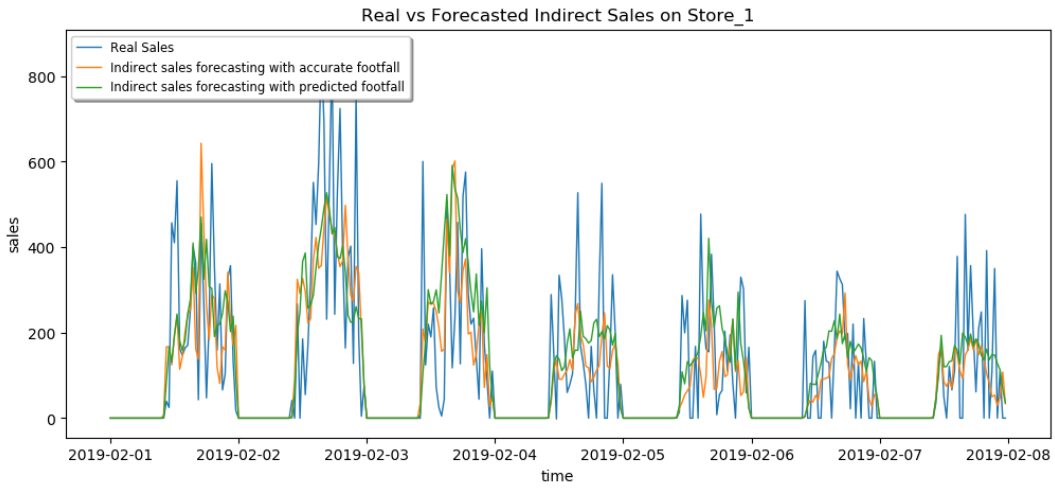


Figure C.5: Real versus forecasted indirect sales using accurate and forecasted footfall.

C.1.3 Experiment F4

Table C.3: Evaluation metrics values for the forecasted number of transactions with the increasing error in the footfall in.

Error	MSE	RMSE	MAE	MDAE	EVS	MER	R2
0.0	1.7977	0.8988	0.8232	0.32	0.6057	4.9	0.6021
0.1	1.7974	0.8987	0.8289	0.29	0.6052	4.4386	0.6021
0.2	1.8087	0.9044	0.8238	0.2855	0.6013	5.26	0.5996
0.3	1.8542	0.9271	0.8173	0.29	0.5906	6.6	0.5896
0.4	1.922	0.961	0.824	0.28	0.5764	5.56	0.5746
0.5	2.2344	1.1172	0.8763	0.36	0.5054	6.91	0.5054
0.6	2.6305	1.3152	0.9512	0.37	0.4177	7.23	0.4177
0.7	2.6832	1.3416	0.948	0.3255	0.4082	7.03	0.4061
0.8	2.965	1.4825	1.0017	0.3455	0.3438	8.33	0.3437
0.9	3.1863	1.5932	1.0453	0.3153	0.3043	7.38	0.2947

1.0	2.9427	1.4714	1.0037	0.38	0.3747	7.61	0.3486
1.1	3.1805	1.5903	1.0294	0.365	0.3122	7.74	0.296
1.2	3.7297	1.8648	1.099	0.35	0.1946	8.17	0.1744
1.3	4.1868	2.0934	1.1841	0.43	0.1123	8.46	0.0732
1.4	3.989	1.9945	1.2189	0.475	0.1736	8.32	0.117
1.5	4.3318	2.1659	1.2013	0.405	0.0922	8.67	0.0411
1.6	3.8789	1.9395	1.1533	0.45	0.2055	8.31	0.1414
1.7	4.2522	2.1261	1.2183	0.335	0.1489	8.72	0.0587
1.8	4.954	2.477	1.2834	0.345	-0.0096	9.62	-0.0966
1.9	5.4205	2.7103	1.353	0.45	-0.0666	10.11	-0.1999
2.0	5.0424	2.5212	1.2706	0.3	-0.0215	8.64	-0.1162
2.1	5.3139	2.657	1.3659	0.5	-0.0642	7.92	-0.1763
2.2	5.0086	2.5043	1.2995	0.34	-0.0401	9.3	-0.1087
2.3	6.0853	3.0427	1.4042	0.355	-0.1995	10.67	-0.347
2.4	5.925	2.9625	1.3978	0.415	-0.1731	11.39	-0.3116
2.5	6.0975	3.0488	1.4649	0.52	-0.2361	10.02	-0.3497
2.6	6.3817	3.1909	1.518	0.64	-0.2411	10.08	-0.4127
2.7	6.4983	3.2491	1.5043	0.485	-0.2897	9.34	-0.4385
2.8	6.5125	3.2563	1.5435	0.6	-0.1946	10.64	-0.4416
2.9	7.5015	3.7508	1.6399	0.6	-0.4306	11.03	-0.6605
3.0	7.5715	3.7857	1.6734	0.75	-0.4795	9.5	-0.676

Table C.4: Evaluation metrics values for the forecasted average products per transaction with the increasing error in the footfall in.

Error	MSE	RMSE	MAE	MDAE	EVS	MER	R2
0.0	0.5102	0.2551	0.3467	0.0767	0.3468	5.3245	0.3467
0.1	0.5095	0.2548	0.3476	0.0742	0.3476	5.3245	0.3476
0.2	0.5169	0.2585	0.3448	0.0931	0.3381	5.3195	0.3381
0.3	0.512	0.256	0.3515	0.0685	0.3445	5.3011	0.3444
0.4	0.5188	0.2594	0.3523	0.0902	0.3356	5.312	0.3356
0.5	0.5246	0.2623	0.3608	0.118	0.3289	5.2945	0.3283
0.6	0.5209	0.2605	0.3621	0.0988	0.3352	5.2095	0.3329
0.7	0.5369	0.2685	0.359	0.0996	0.3137	5.292	0.3125
0.8	0.5248	0.2624	0.3632	0.1127	0.3313	5.3153	0.328
0.9	0.5435	0.2718	0.3683	0.0878	0.3058	5.1563	0.304
1.0	0.5237	0.2618	0.3582	0.0978	0.3307	5.287	0.3295
1.1	0.5237	0.2619	0.363	0.0936	0.3338	5.2013	0.3294
1.2	0.5333	0.2667	0.3625	0.1049	0.3189	5.3175	0.3171
1.3	0.5459	0.2729	0.3772	0.136	0.303	5.3401	0.301
1.4	0.5571	0.2785	0.3776	0.1219	0.2875	5.3013	0.2867
1.5	0.5196	0.2598	0.3541	0.0816	0.3382	5.2446	0.3347
1.6	0.5506	0.2753	0.3772	0.095	0.297	5.1586	0.2949
1.7	0.5351	0.2676	0.3657	0.1047	0.316	5.272	0.3148
1.8	0.5648	0.2824	0.3756	0.1149	0.2799	5.2938	0.2768
1.9	0.5166	0.2583	0.3588	0.0809	0.3403	5.3153	0.3385
2.0	0.5817	0.2909	0.3815	0.1109	0.257	5.3186	0.2551
2.1	0.5304	0.2652	0.3667	0.0953	0.3227	5.223	0.3209
2.2	0.5463	0.2731	0.3724	0.1263	0.304	5.2568	0.3005
2.3	0.5541	0.277	0.3639	0.0866	0.2931	5.3263	0.2905
2.4	0.5556	0.2778	0.3737	0.1135	0.2916	5.332	0.2886
2.5	0.5514	0.2757	0.3841	0.1118	0.2971	5.1578	0.294
2.6	0.5326	0.2663	0.3739	0.1012	0.3205	5.2195	0.318
2.7	0.5431	0.2715	0.3696	0.0858	0.3066	5.2651	0.3046

2.8	0.5061	0.2531	0.3616	0.1176	0.3527	5.2971	0.3519
2.9	0.5439	0.2719	0.3691	0.1285	0.3045	5.2963	0.3036
3.0	0.5264	0.2632	0.376	0.1498	0.3275	5.3403	0.326

Table C.5: Evaluation metrics values for the forecasted average price per product with the increasing error in the footfall in.

Error	MSE	RMSE	MAE	MDAE	EVS	MER	R2
0.0	630.6505	315.3252	15.4555	4.8687	0.5392	93.9348	0.5385
0.1	635.9868	317.9934	15.4921	5.1144	0.5354	94.5231	0.5346
0.2	639.3339	319.667	15.624	5.1877	0.5333	87.3071	0.5322
0.3	648.2875	324.1437	15.7177	6.8842	0.5268	97.2527	0.5256
0.4	650.2421	325.1211	15.7741	6.0759	0.5252	92.1395	0.5242
0.5	645.3678	322.6839	15.8903	6.8986	0.5291	99.5544	0.5278
0.6	678.8216	339.4108	16.2066	6.6765	0.5051	94.5809	0.5033
0.7	632.0806	316.0403	15.7285	6.5445	0.539	95.6145	0.5375
0.8	679.9847	339.9924	16.3987	6.6082	0.5036	94.5331	0.5025
0.9	671.7026	335.8513	15.9993	5.5408	0.5094	90.5681	0.5085
1.0	663.6117	331.8059	15.9871	5.6371	0.5148	100.6162	0.5144
1.1	683.7309	341.8654	16.2163	6.2329	0.5008	94.4818	0.4997
1.2	648.0615	324.0308	15.8638	6.521	0.5261	93.875	0.5258
1.3	662.0512	331.0256	16.0353	6.3548	0.5161	91.5563	0.5156
1.4	671.1053	335.5527	16.0095	6.0282	0.5093	97.8976	0.5089
1.5	664.2463	332.1232	16.2884	7.3635	0.515	85.1974	0.514
1.6	686.7904	343.3952	16.31	6.0671	0.4977	92.1138	0.4975
1.7	681.1948	340.5974	15.9733	5.9079	0.5018	93.2134	0.5016
1.8	666.7494	333.3747	16.0892	7.4902	0.5126	84.3774	0.5121
1.9	645.6973	322.8487	15.7455	6.2602	0.5279	95.1502	0.5275
2.0	715.7447	357.8724	16.384	5.2974	0.4774	98.2939	0.4763
2.1	635.7334	317.8667	15.6366	5.9148	0.5349	83.9812	0.5348
2.2	652.0028	326.0014	15.8265	5.8339	0.5236	93.8959	0.5229
2.3	640.1039	320.052	15.7535	6.9925	0.5321	92.5835	0.5316
2.4	651.2868	325.6434	15.7338	5.9832	0.5241	83.124	0.5234
2.5	687.5586	343.7793	16.4997	6.8416	0.4971	91.7907	0.4969
2.6	668.2383	334.1191	15.9322	5.8634	0.5117	94.1433	0.511
2.7	700.4942	350.2471	16.4512	6.0697	0.4882	96.2388	0.4874
2.8	641.8414	320.9207	15.6576	6.2017	0.5306	94.4737	0.5304
2.9	669.6798	334.8399	16.186	6.7249	0.5103	93.3566	0.51
3.0	661.7852	330.8926	15.9806	6.2323	0.5164	93.458	0.5158

Table C.6: Evaluation metrics values for the direct forecasted sales with the increasing error in the footfall in.

Error	MSE	RMSE	MAE	MDAE	EVS	MER	R2
0.0	13626.5075	6813.2538	71.3033	29.8749	0.5368	498.7829	0.5351
0.1	14180.0425	7090.0213	73.977	29.9513	0.5184	495.5042	0.5162
0.2	13643.4632	6821.7316	72.7832	32.705	0.5361	421.1681	0.5346
0.3	14801.4888	7400.7444	73.8921	23.2294	0.497	558.2509	0.495
0.4	17041.2406	8520.6203	80.2351	29.6801	0.422	568.0326	0.4186
0.5	15238.9054	7619.4527	76.1487	31.1143	0.4891	561.8748	0.4801
0.6	16946.5727	8473.2864	79.6219	37.4765	0.4246	630.5087	0.4219

0.7	19393.0836	9696.5418	82.8663	27.8808	0.3459	608.2724	0.3384
0.8	19816.6172	9908.3086	83.0542	31.2115	0.328	639.4205	0.324
0.9	20184.644	10092.322	85.371	36.4912	0.3189	642.1655	0.3114
1.0	19615.1472	9807.5736	85.5704	34.7993	0.3587	532.4806	0.3308
1.1	20225.2879	10112.6439	84.6956	31.1612	0.3237	580.2926	0.31
1.2	23608.8209	11804.4104	91.8198	34.3791	0.2088	628.2811	0.1946
1.3	29085.159	14542.5795	100.3581	39.3693	0.0483	644.6321	0.0078
1.4	28707.5785	14353.7893	99.3244	31.8822	0.0879	794.1102	0.0206
1.5	27406.4793	13703.2397	100.6669	37.1293	0.1338	831.0761	0.065
1.6	27019.0129	13509.5065	97.4696	40.7816	0.1102	810.6634	0.0782
1.7	28625.5544	14312.7772	101.3616	42.1644	0.1008	798.7828	0.0234
1.8	35899.2011	17949.6006	112.0351	33.4457	-0.0684	743.4114	-0.2247
1.9	30401.2416	15200.6208	103.2504	43.6503	0.0705	794.8474	-0.0371
2.0	32112.4972	16056.2486	110.5249	61.4067	0.0398	680.263	-0.0955
2.1	32823.2074	16411.6037	109.8837	44.7784	0.0235	847.2354	-0.1198
2.2	33195.7257	16597.8629	110.2065	36.5727	-0.0204	798.2391	-0.1325
2.3	41445.3003	20722.6501	123.1686	55.5508	-0.2115	826.3316	-0.4139
2.4	35778.4735	17889.2367	112.2943	39.043	-0.0448	780.9676	-0.2206
2.5	32940.6725	16470.3362	110.3778	48.0841	-0.0225	660.9549	-0.1238
2.6	42272.8283	21136.4142	122.1381	56.5657	-0.2304	809.2777	-0.4421
2.7	37978.0347	18989.0173	116.9869	42.755	-0.1192	773.3539	-0.2956
2.8	39818.9539	19909.477	117.504	39.8247	-0.2278	859.7321	-0.3584
2.9	39871.5083	19935.7542	116.8307	32.5065	-0.1892	808.2625	-0.3602
3.0	49343.0843	24671.5422	132.7084	45.7032	-0.3898	954.4285	-0.6833

Table C.7: Evaluation metrics values for the indirect forecasted sales with the increasing error in the footfall in.

Error	MSE	RMSE	MAE	MDAE	EVS	MER	R2
0.0	14705.7691	7352.8845	70.3948	25.6049	0.5034	508.4782	0.4983
0.1	15121.3307	7560.6654	71.71	23.6875	0.4883	505.3511	0.4841
0.2	15577.1516	7788.5758	72.8935	19.0131	0.4769	447.4709	0.4686
0.3	16300.8907	8150.4454	74.0395	29.3858	0.4505	541.667	0.4439
0.4	18734.7607	9367.3804	81.0788	32.4685	0.3686	539.5415	0.3609
0.5	16968.6189	8484.3094	76.6199	26.1788	0.4218	622.5394	0.4211
0.6	19463.7583	9731.8792	80.9088	31.4239	0.3464	695.7367	0.336
0.7	21281.9532	10640.9766	84.0094	30.2767	0.2811	710.1283	0.274
0.8	24126.996	12063.498	88.5827	35.7041	0.187	763.1097	0.1769
0.9	25016.552	12508.276	89.2211	30.3263	0.1532	772.7726	0.1466
1.0	22050.2001	11025.1001	86.6874	33.5128	0.2478	707.2194	0.2478
1.1	22282.3358	11141.1679	86.771	37.058	0.243	571.4145	0.2398
1.2	27404.908	13702.454	96.1069	33.9172	0.0669	669.283	0.0651
1.3	30333.6148	15166.8074	100.3542	38.8177	-0.0348	741.1931	-0.0348
1.4	30260.3161	15130.1581	102.8181	43.6046	-0.0276	780.0044	-0.0323
1.5	30478.8002	15239.4001	101.5708	37.0864	-0.0324	794.0568	-0.0398
1.6	30290.0328	15145.0164	102.65	34.8187	-0.0333	734.5696	-0.0333
1.7	27576.4525	13788.2263	97.8588	32.7739	0.0661	737.1575	0.0592
1.8	35375.0585	17687.5293	110.1713	38.5836	-0.1676	746.011	-0.2068
1.9	33795.0409	16897.5205	106.3017	40.9043	-0.1222	665.8203	-0.1529
2.0	35170.7678	17585.3839	113.5801	47.535	-0.1443	665.8853	-0.1998
2.1	37298.9684	18649.4842	116.1684	43.3403	-0.2181	840.4648	-0.2725
2.2	40544.2543	20272.1272	120.7832	46.3132	-0.3327	720.4141	-0.3832
2.3	45984.6679	22992.334	128.2	51.0634	-0.4813	845.8932	-0.5688
2.4	35162.8404	17581.4202	108.5804	38.599	-0.1424	728.4193	-0.1996

2.5	38319.6784	19159.8392	112.8108	40.2782	-0.2834	798.2208	-0.3073
2.6	43287.3421	21643.6711	120.3708	38.5407	-0.4091	818.3652	-0.4767
2.7	43531.1431	21765.5716	121.2952	46.1025	-0.4089	789.3601	-0.4851
2.8	43475.6666	21737.8333	122.8436	41.5174	-0.4444	840.834	-0.4832
2.9	45081.3094	22540.6547	123.2699	40.4263	-0.4771	784.3477	-0.5379
3.0	52797.2841	26398.6421	136.2183	39.8885	-0.6443	871.8883	-0.8012

C.2 Optimization

C.2.1 Optimization Runs

C.2.1.1 Run: R1a

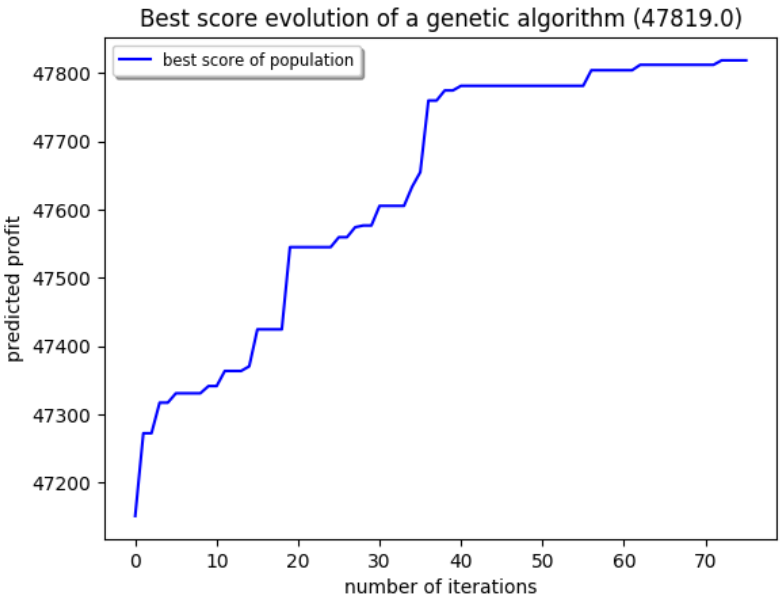


Figure C.6: Evolution of the best individual score through iterations in run *R*1*a*.

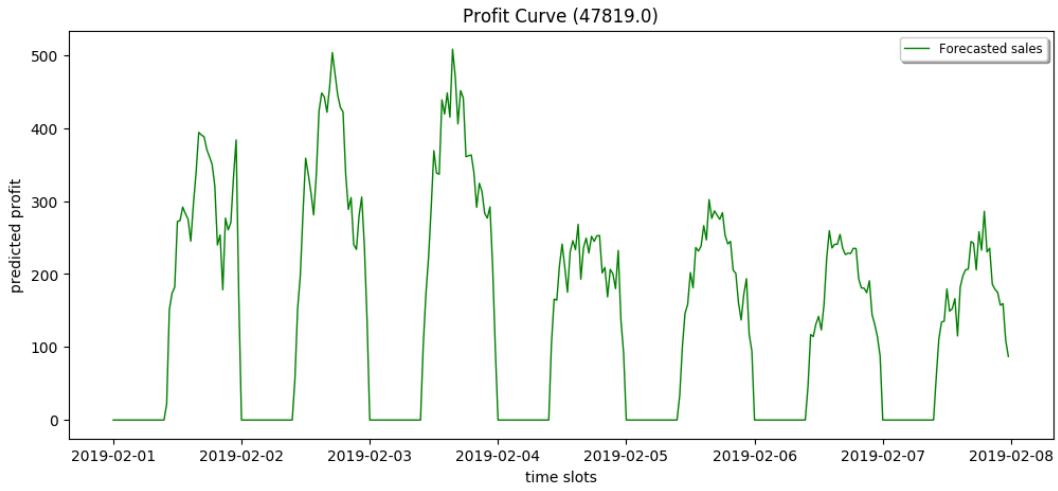


Figure C.7: Forecasted sales curve for the prescribed solution in run *R*1*a*.

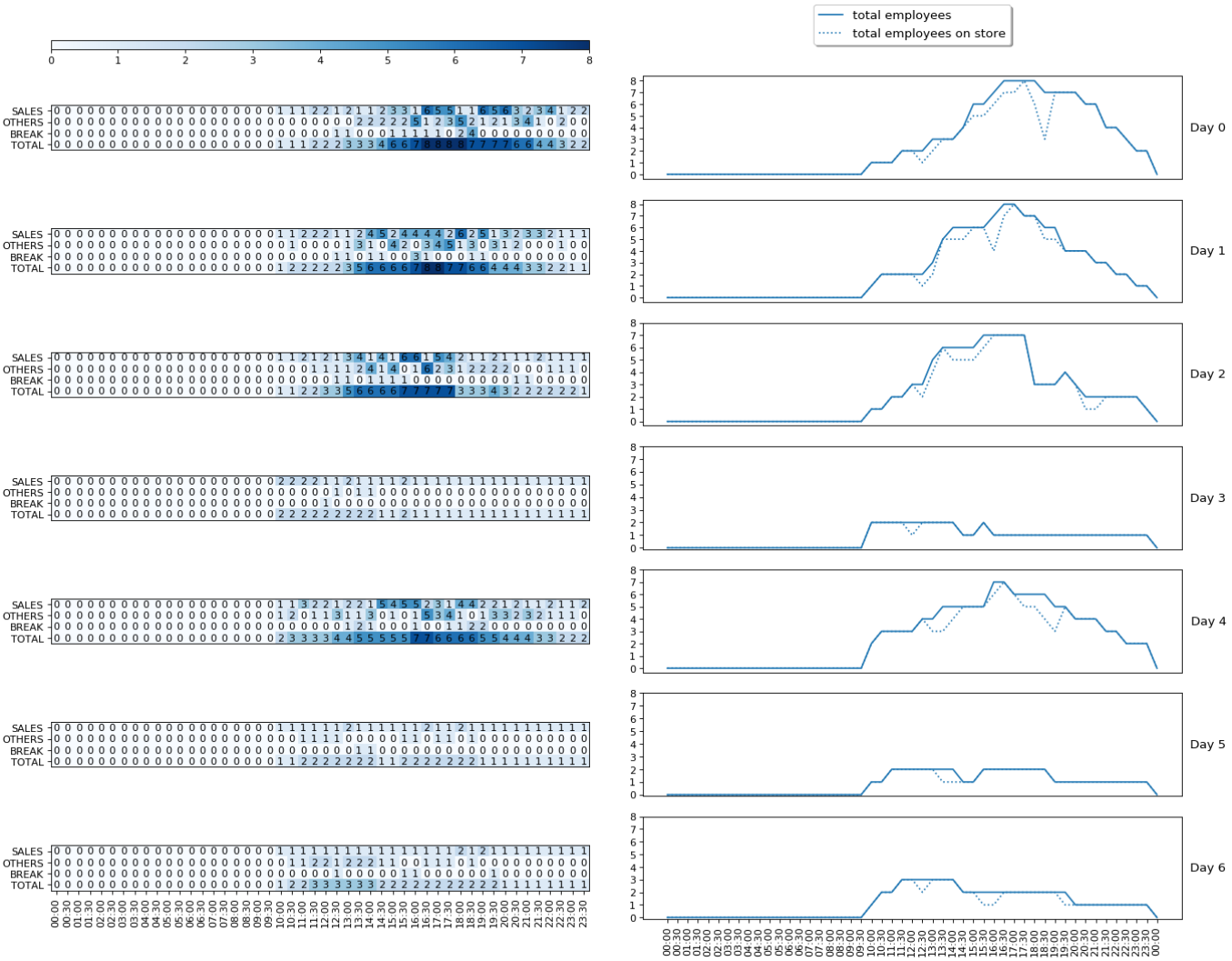


Figure C.8: Allocation for the prescribed solution in run *R*1*a*.

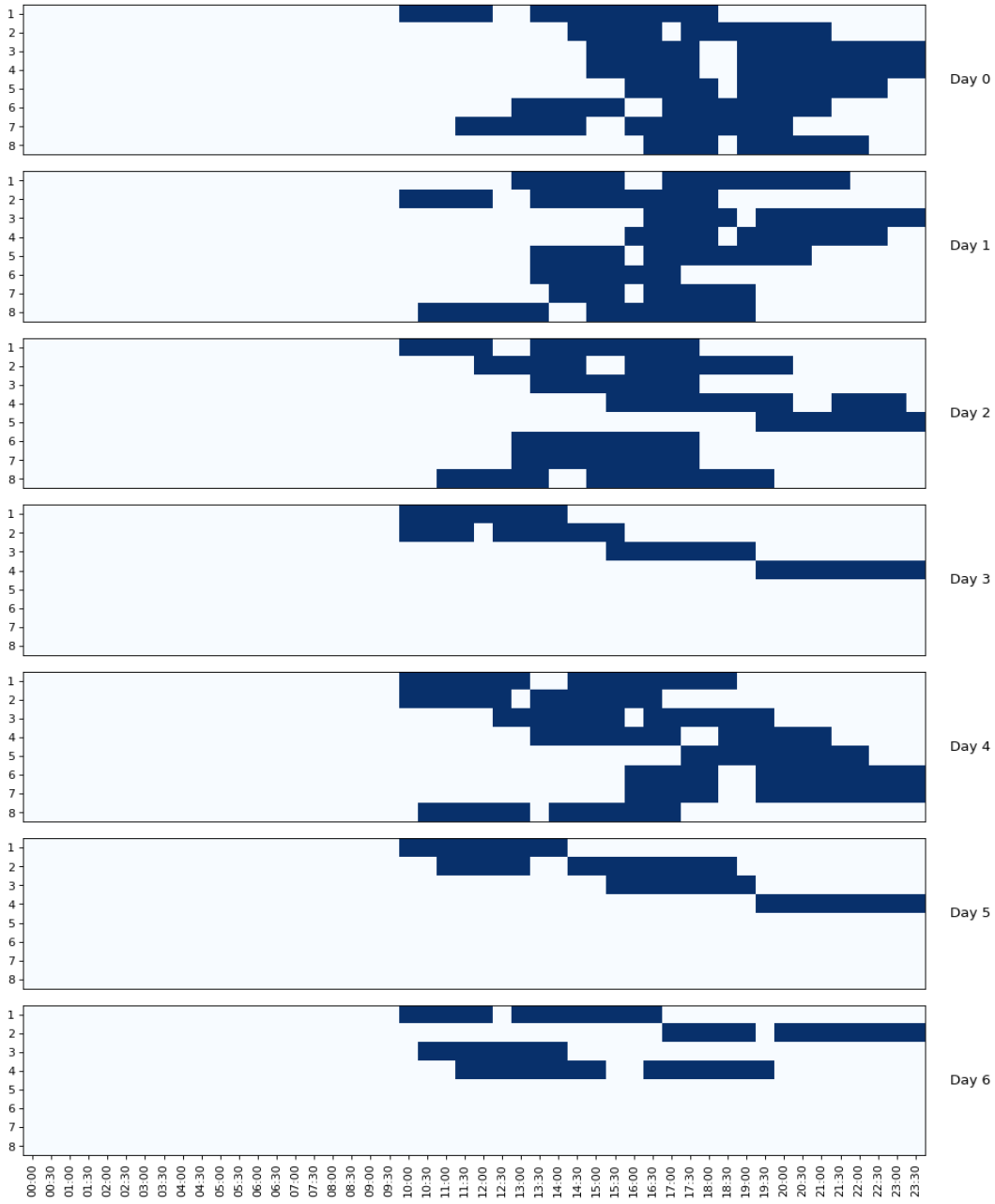


Figure C.9: Shifts boundaries for the prescribed solution in run $R1a$.

C.2.1.2 Run: R1b

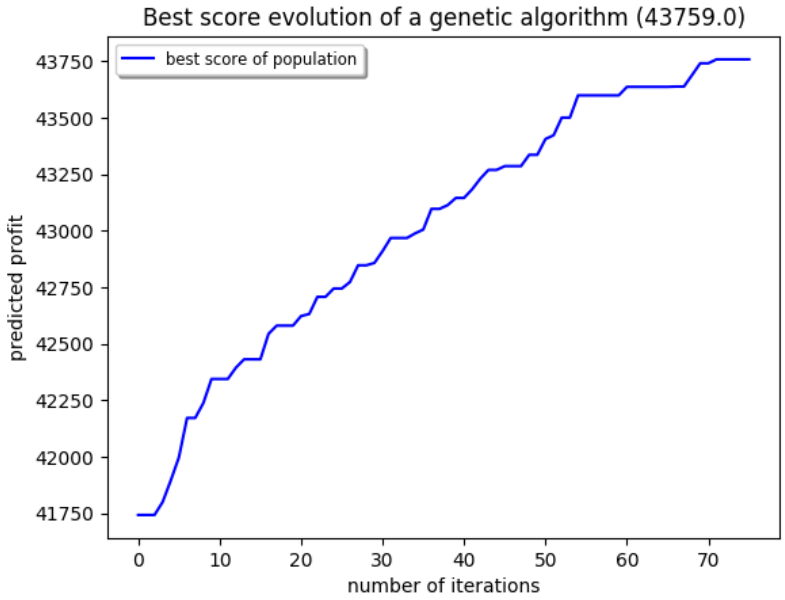


Figure C.10: Evolution of the best individual score through iterations in run *R*1*b*.

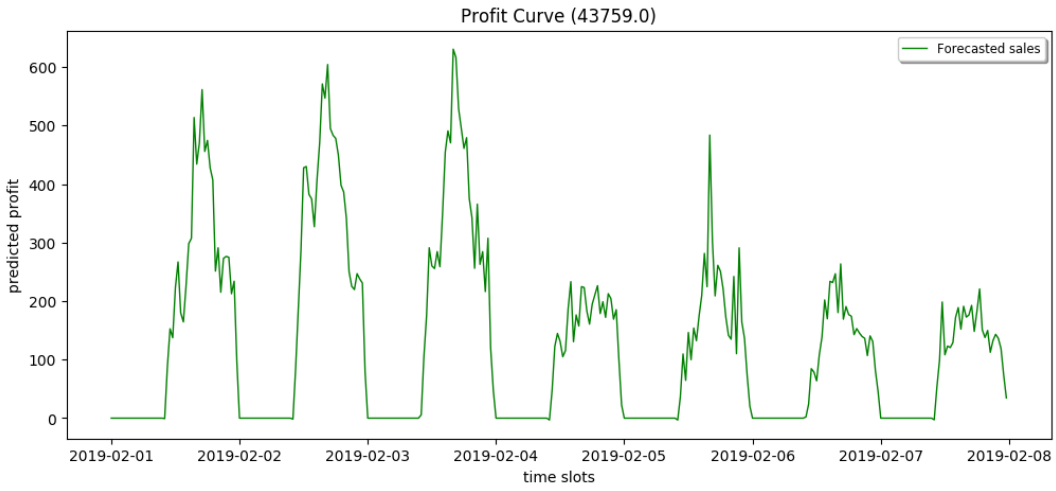


Figure C.11: Forecasted sales curve for the prescribed solution in run *R*1*b*.

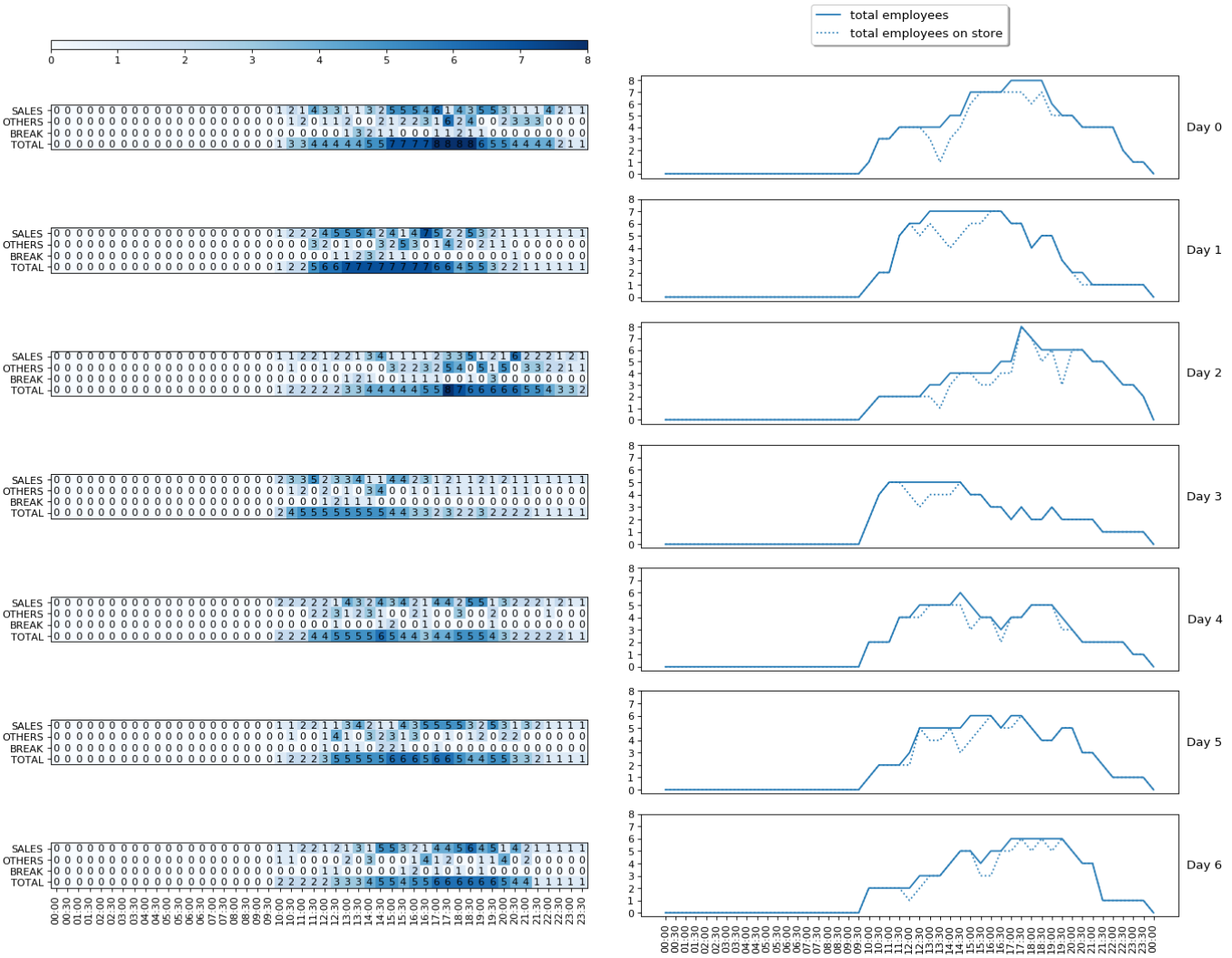


Figure C.12: Allocation for the prescribed solution in run *R1b*.

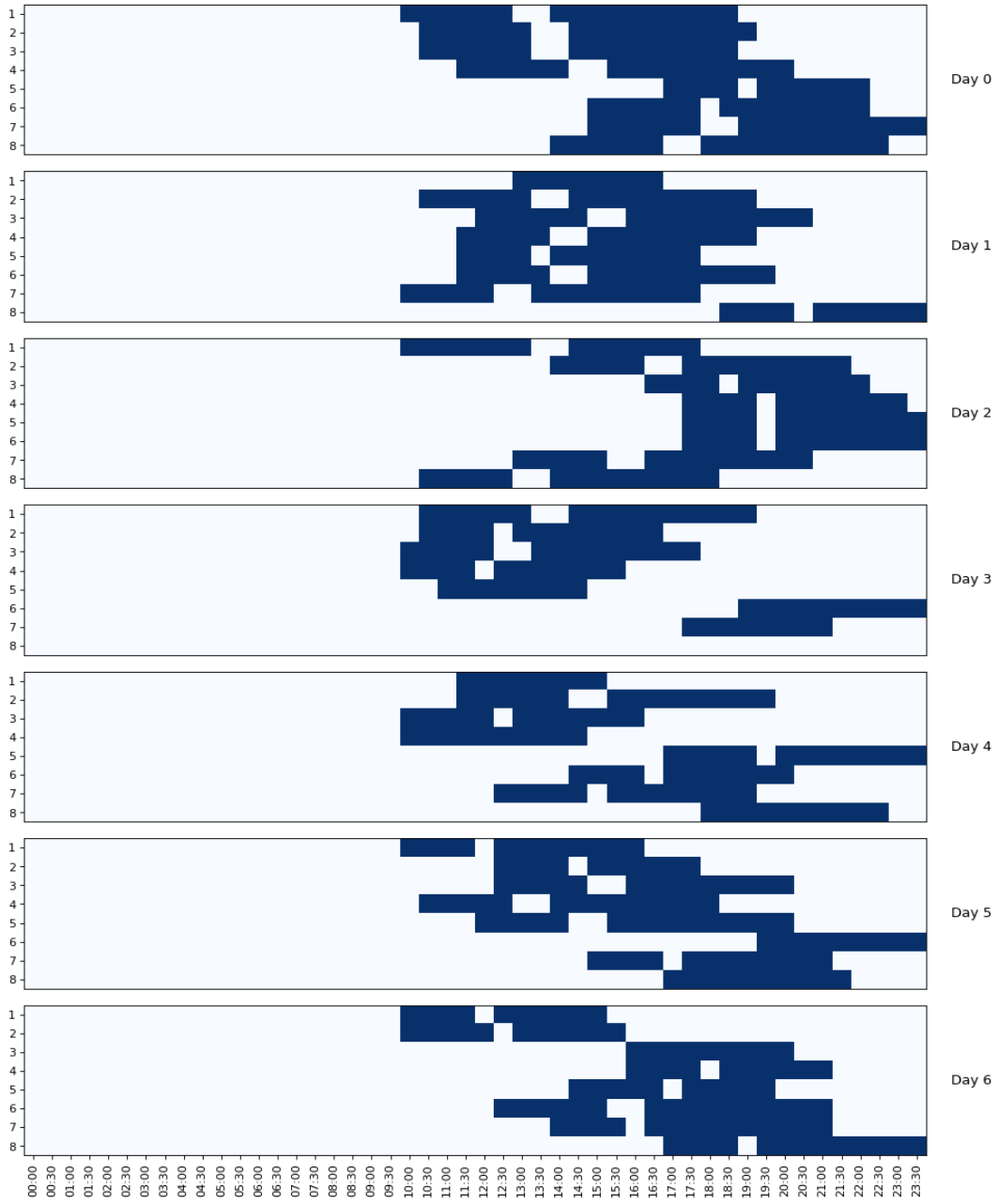


Figure C.13: Shifts boundaries for the prescribed solution in run *R*1*b*.

C.2.1.3 Run: R2a

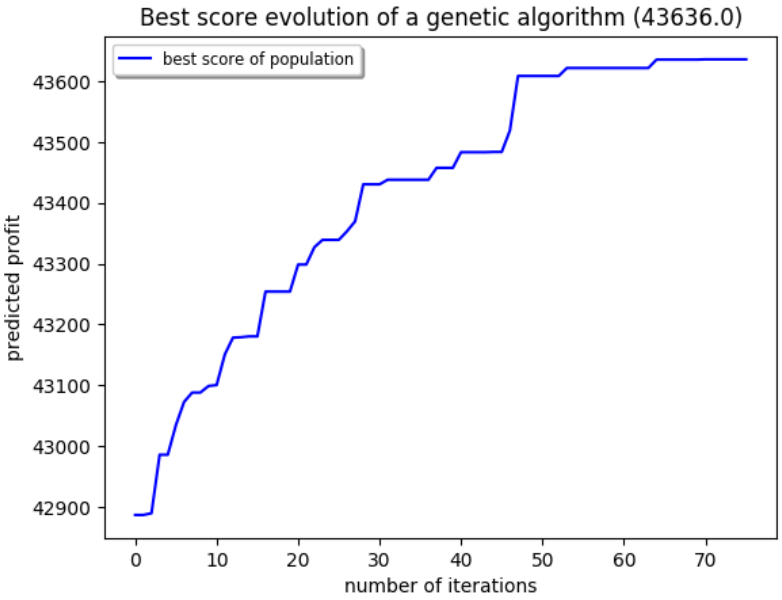


Figure C.14: Evolution of the best individual score through iterations in run *R*2*a*.

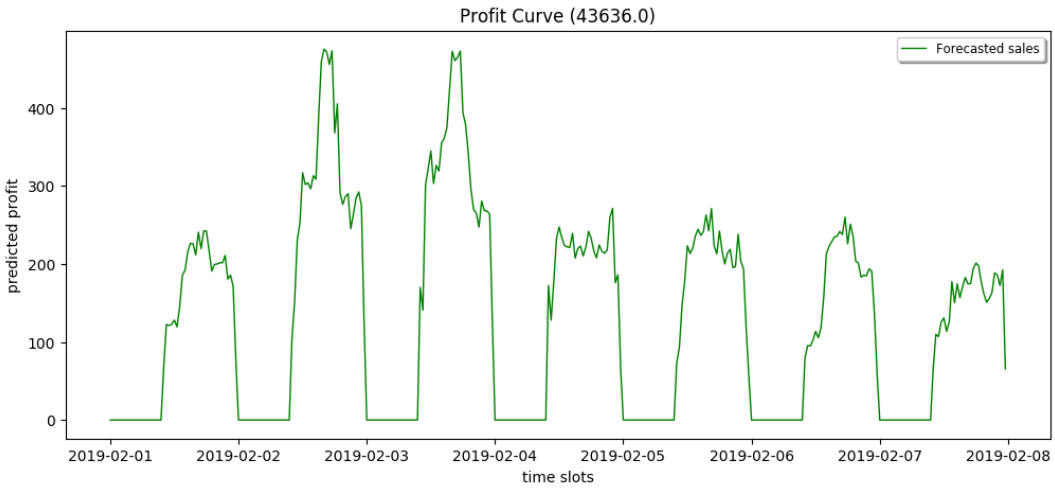


Figure C.15: Forecasted sales curve for the prescribed solution in run *R*2*a*.

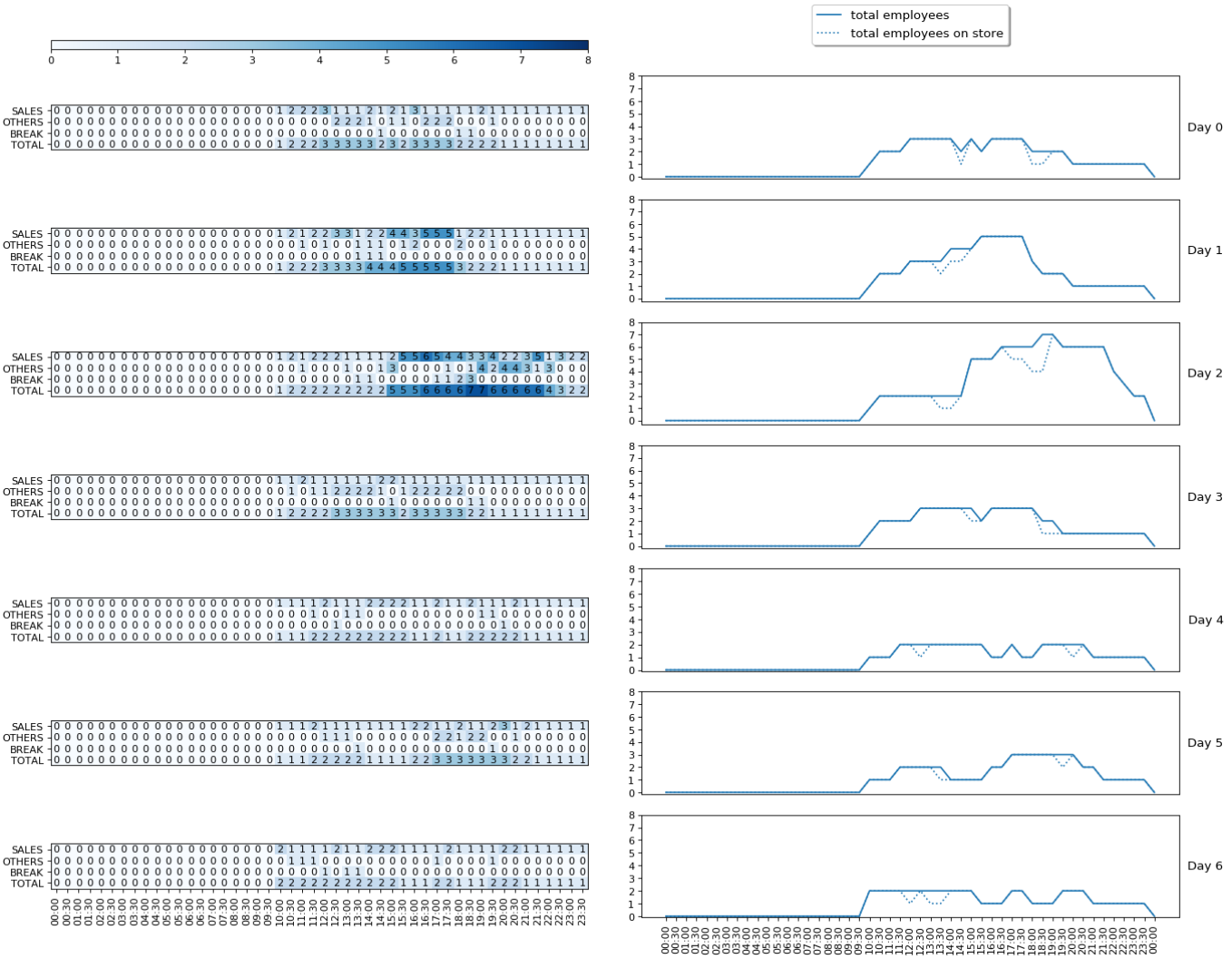


Figure C.16: Allocation for the prescribed solution in run *R2a*.

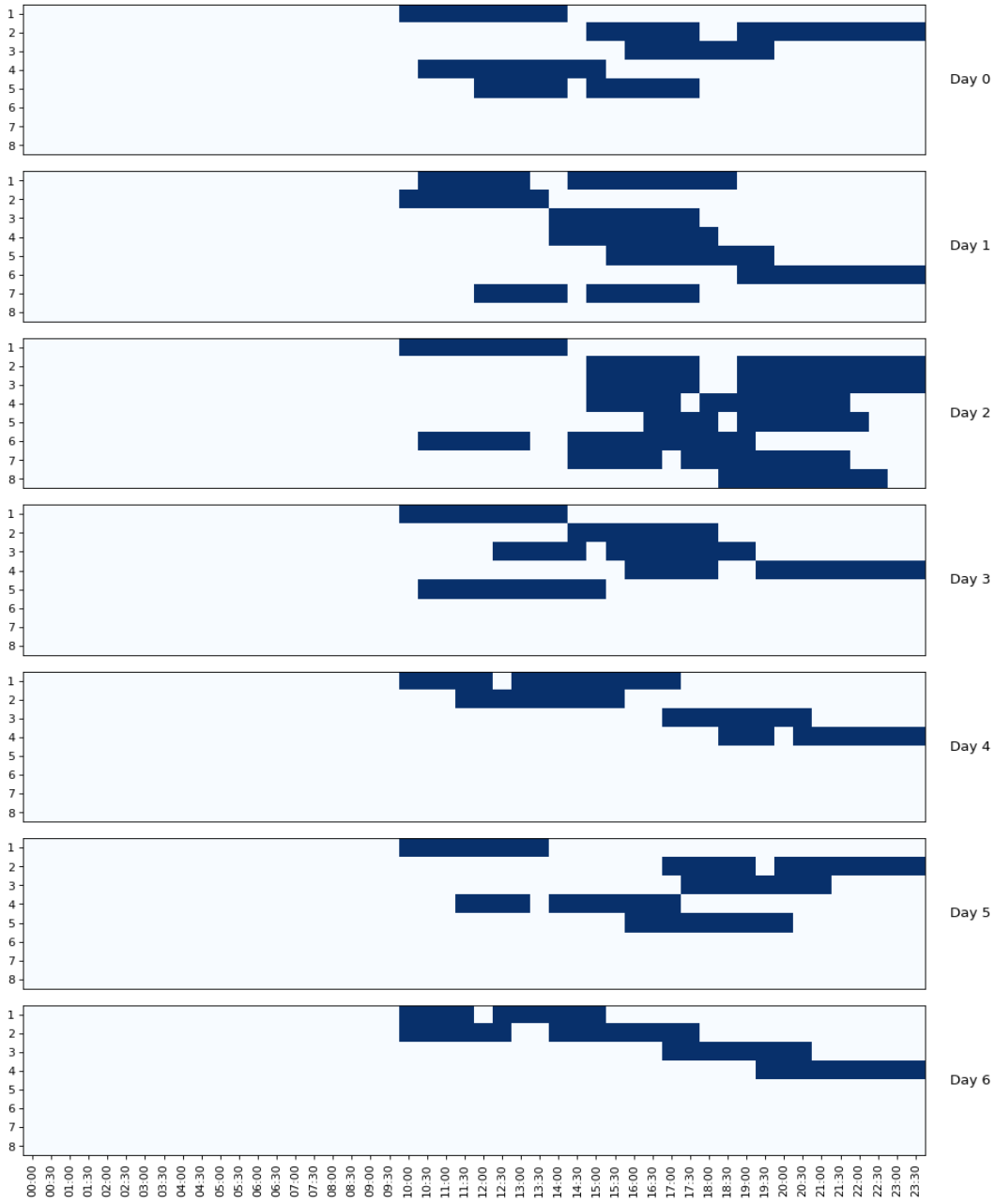


Figure C.17: Shifts boundaries for the prescribed solution in run *R*2*a*.

C.2.1.4 Run: R2b

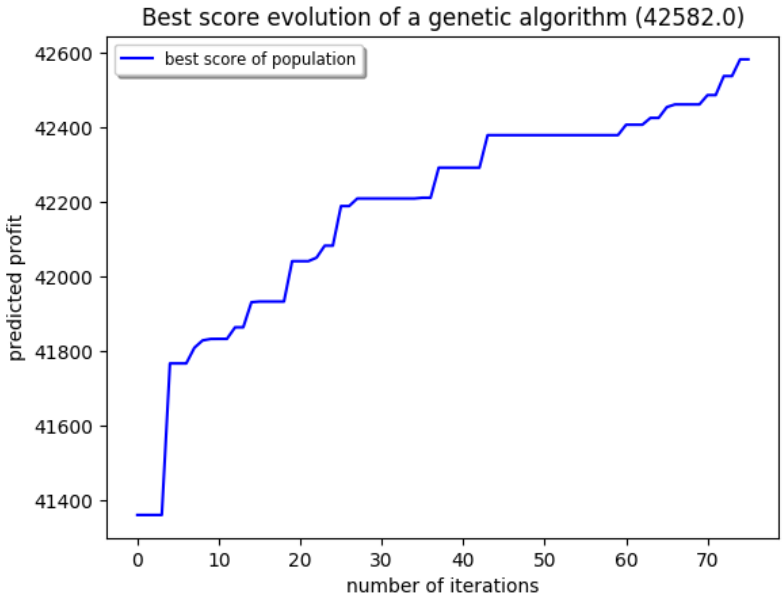


Figure C.18: Evolution of the best individual score through iterations in run *R2b*.

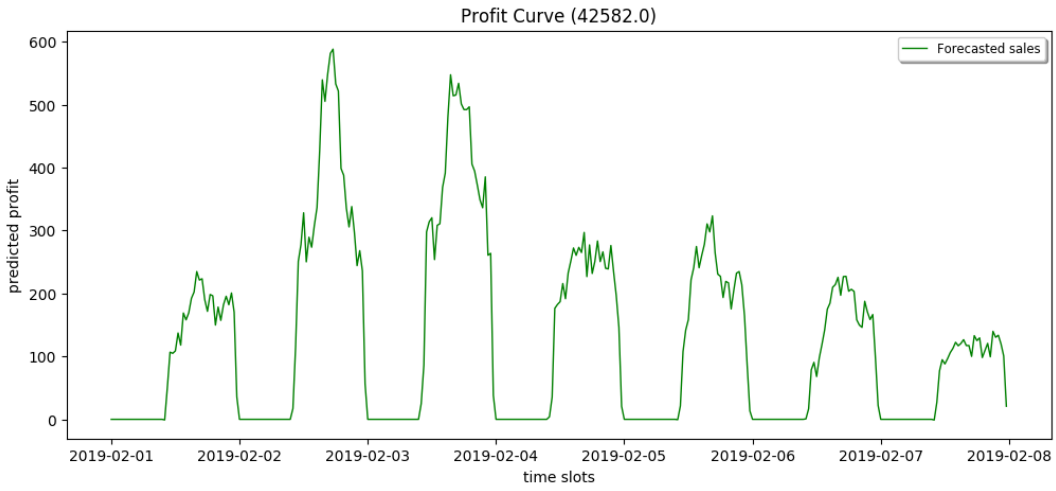


Figure C.19: Forecasted sales curve for the prescribed solution in run *R2b*.

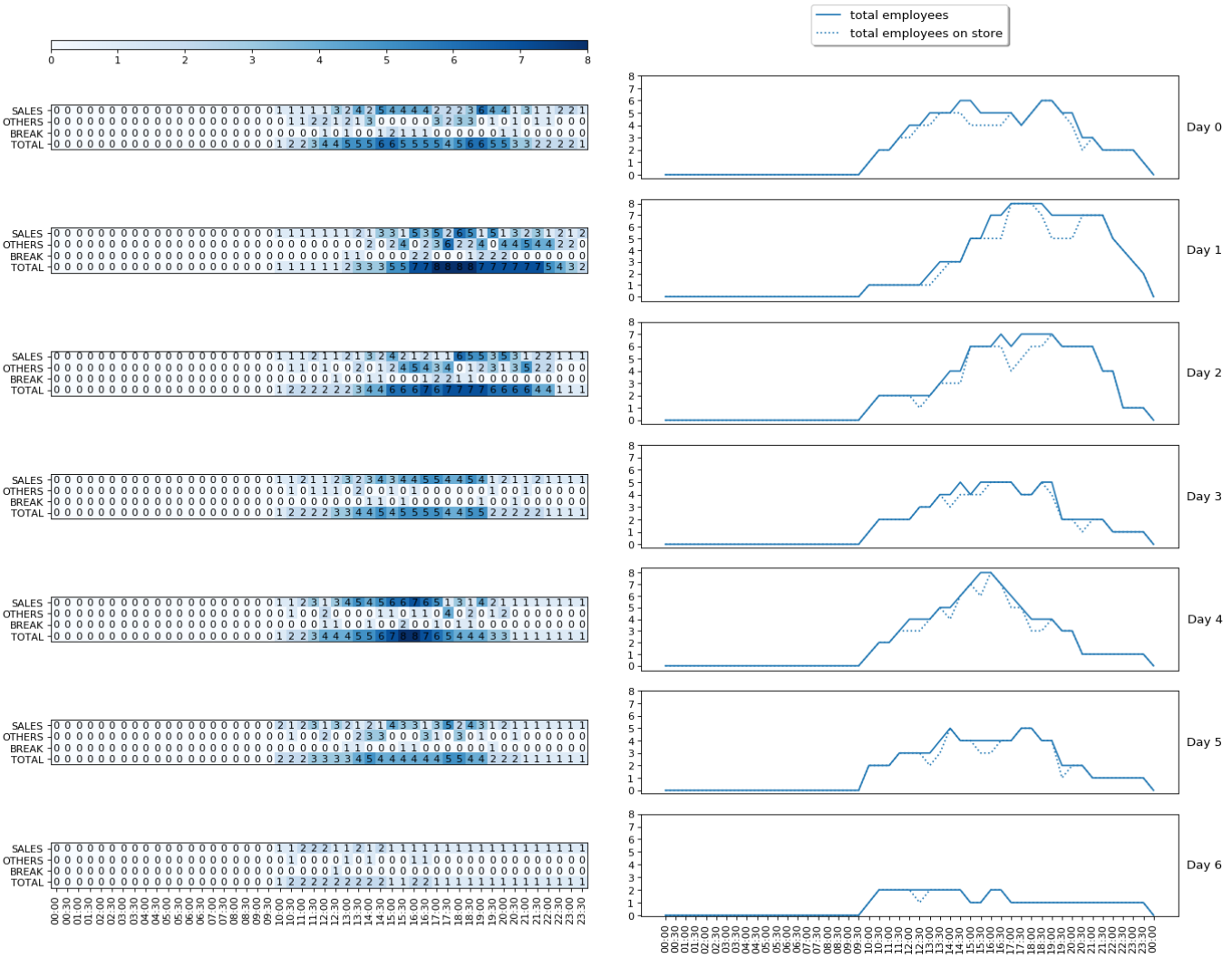


Figure C.20: Allocation for the prescribed solution in run *R2b*.

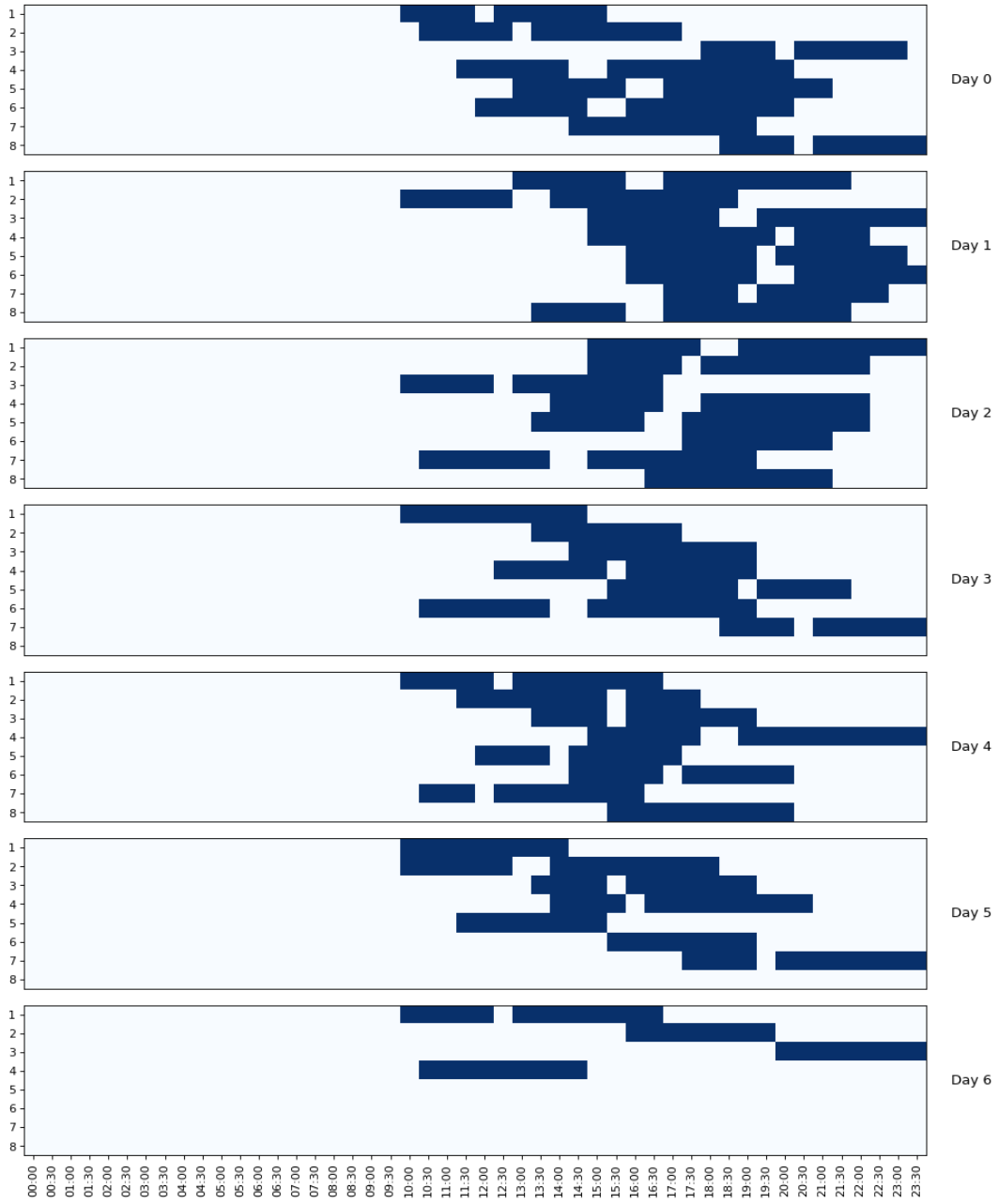


Figure C.21: Shifts boundaries for the prescribed solution in run *R2b*.

C.2.1.5 Run: R3a

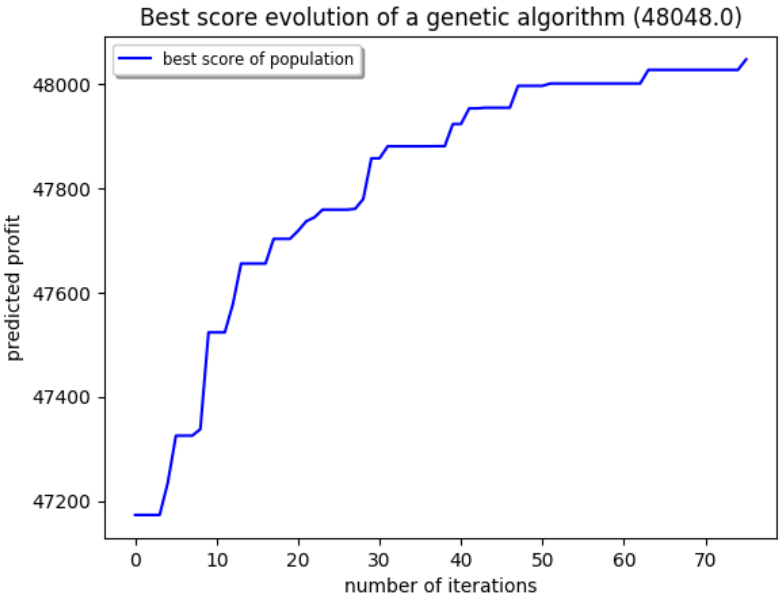


Figure C.22: Evolution of the best individual score through iterations in run *R*3*a*.

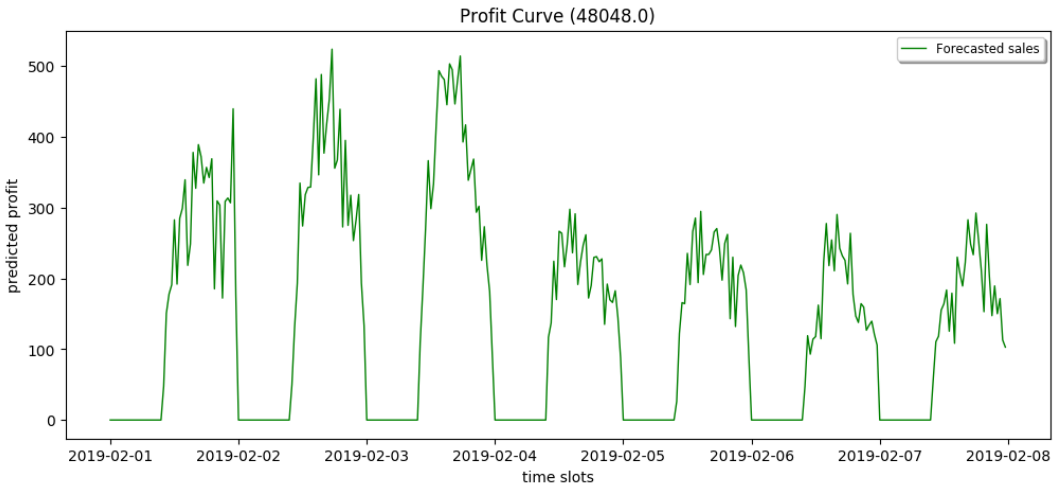


Figure C.23: Forecasted sales curve for the prescribed solution in run *R*3*a*.

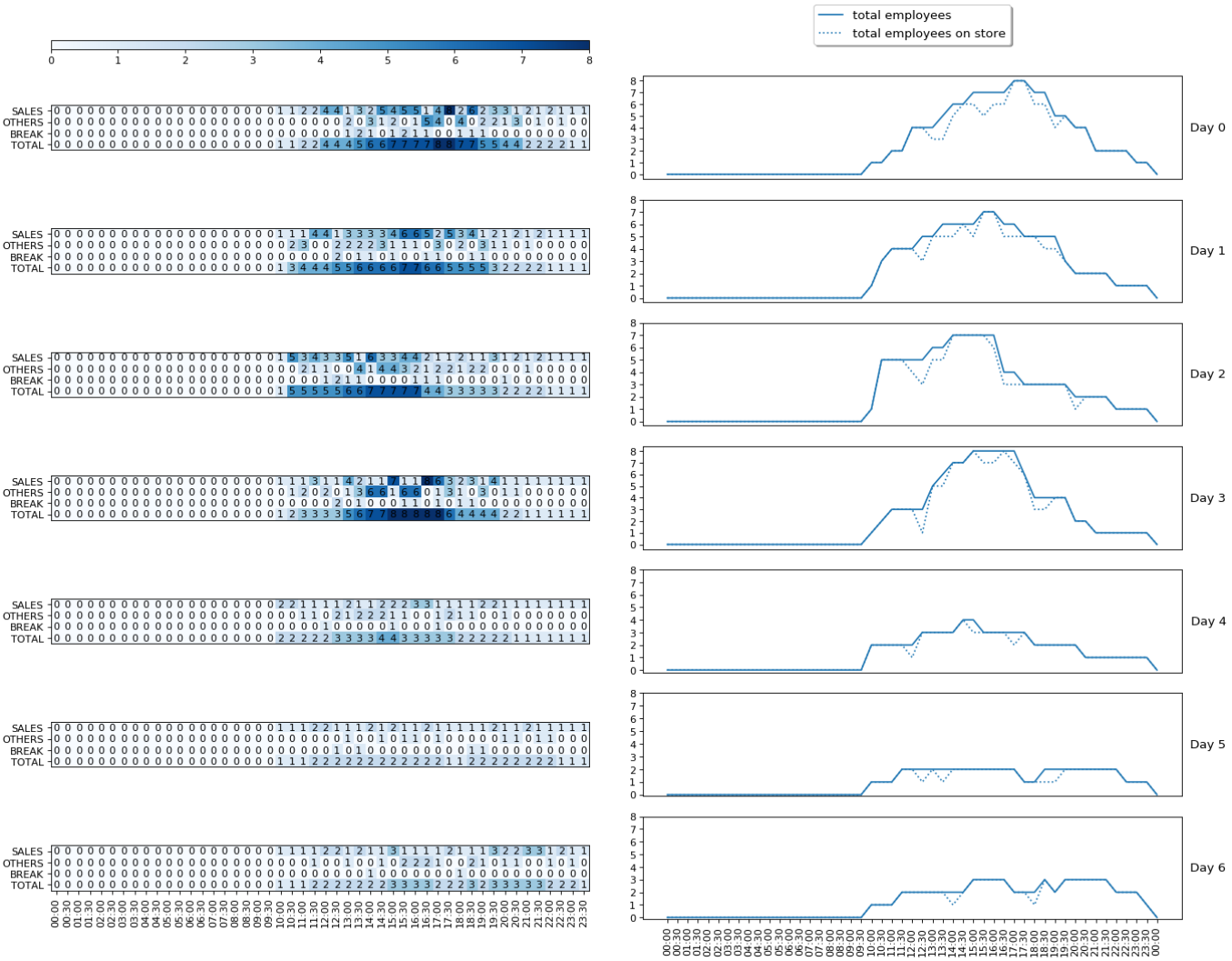


Figure C.24: Allocation for the prescribed solution in run *R*3*a*.

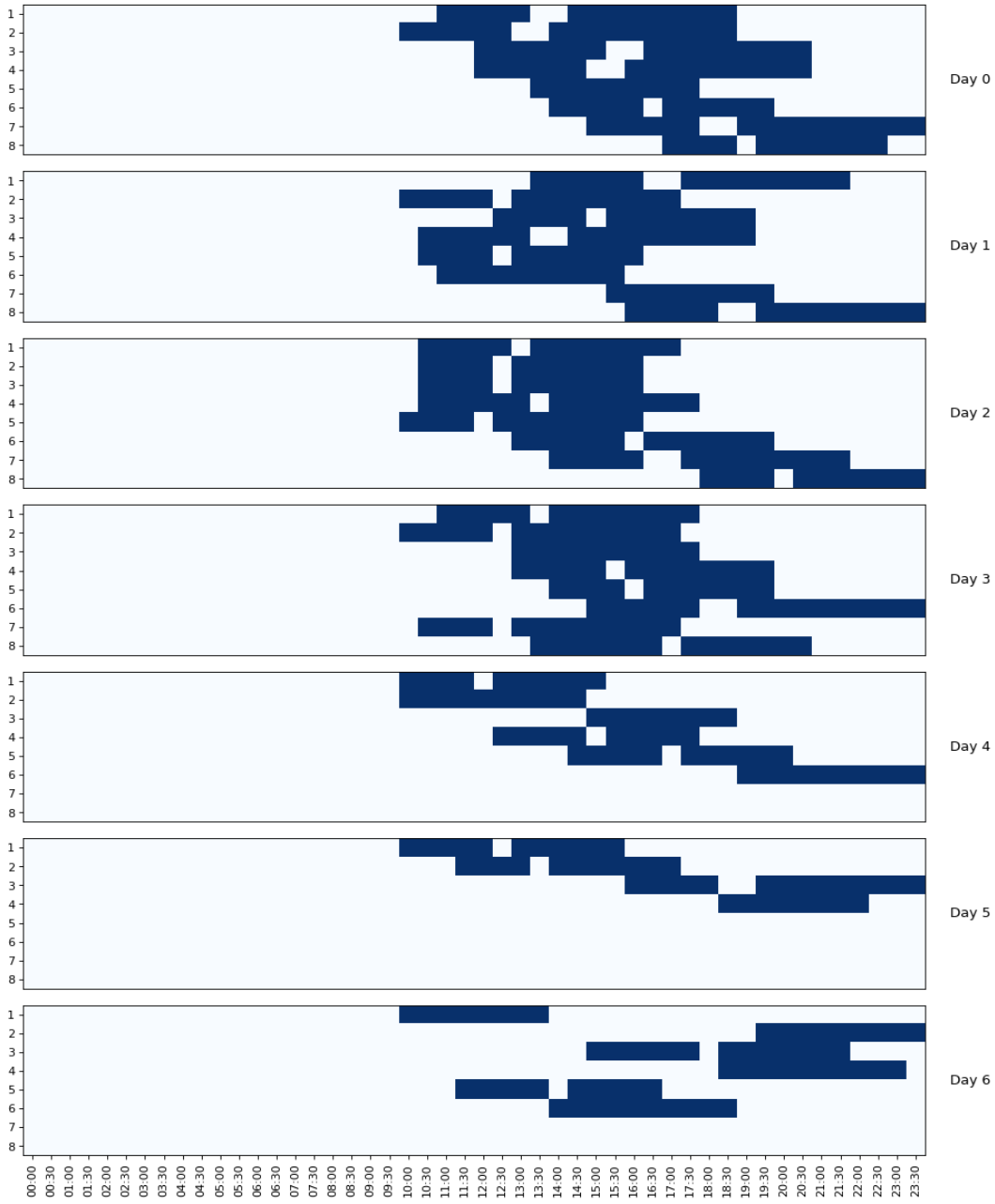


Figure C.25: Shifts boundaries for the prescribed solution in run *R*3*a*.

C.2.1.6 Run: R3b

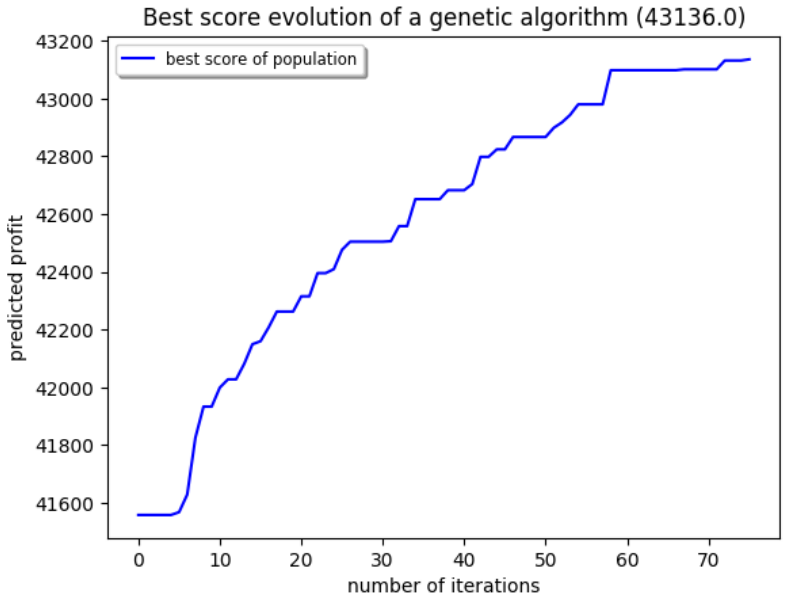


Figure C.26: Evolution of the best individual score through iterations in run *R3b*.

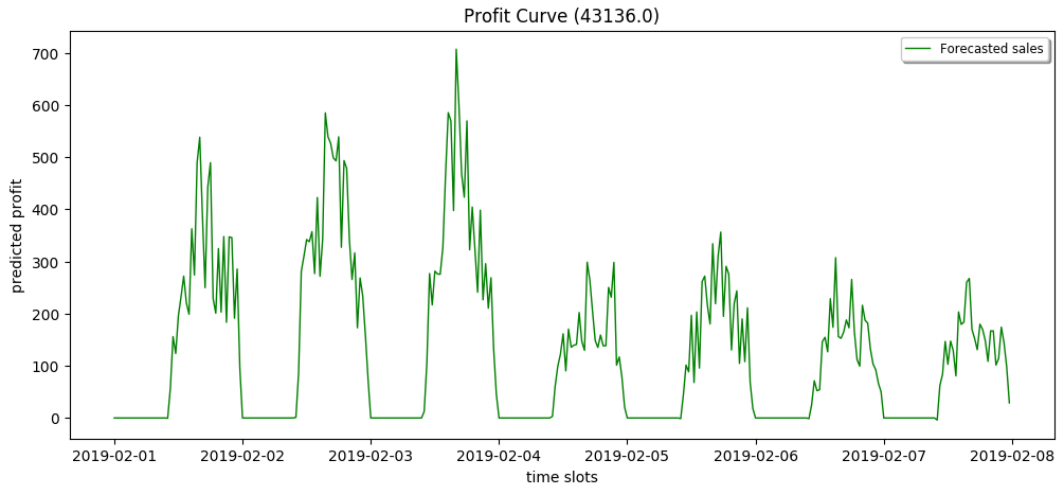


Figure C.27: Forecasted sales curve for the prescribed solution in run *R3b*.

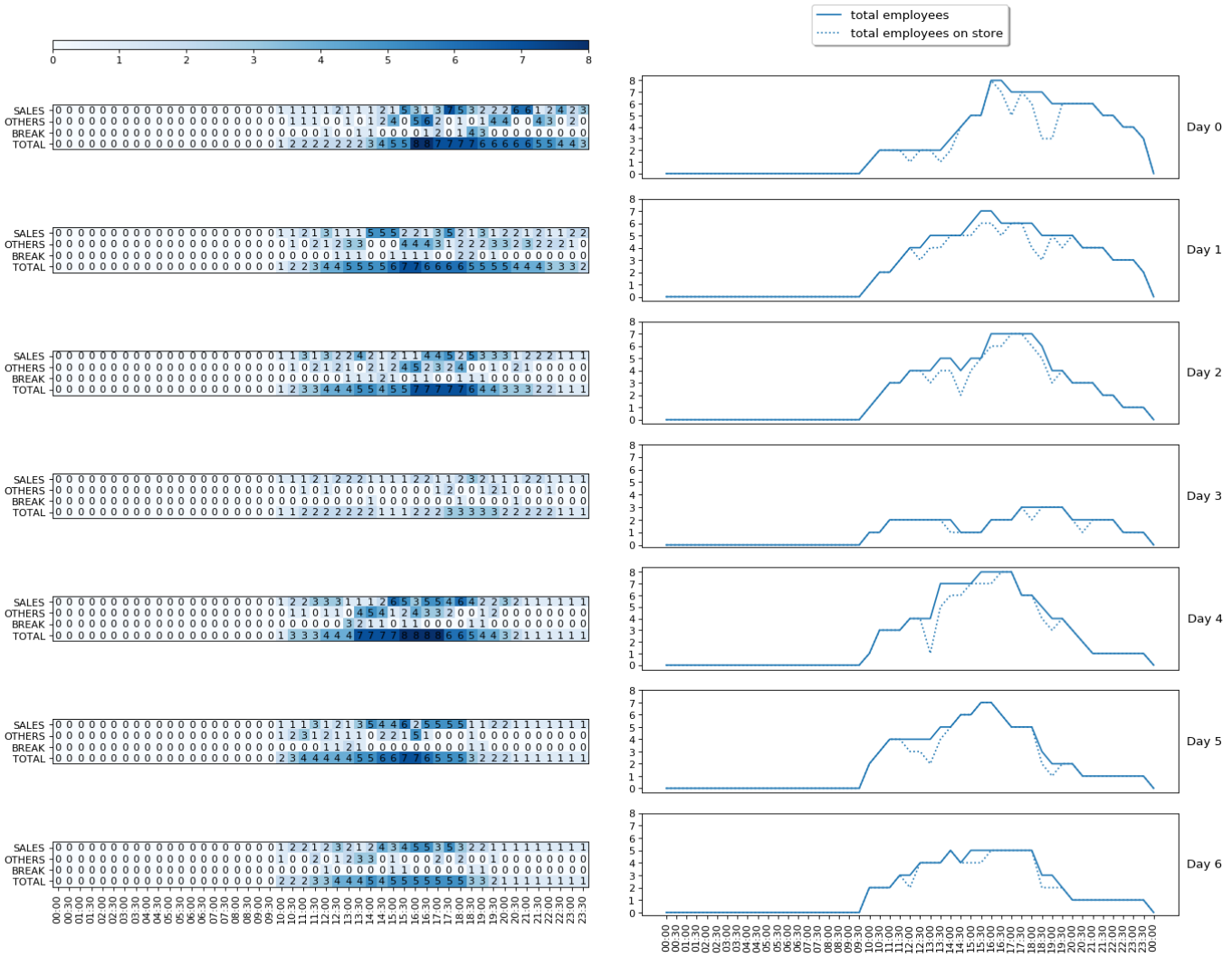


Figure C.28: Allocation for the prescribed solution in run *R3b*.

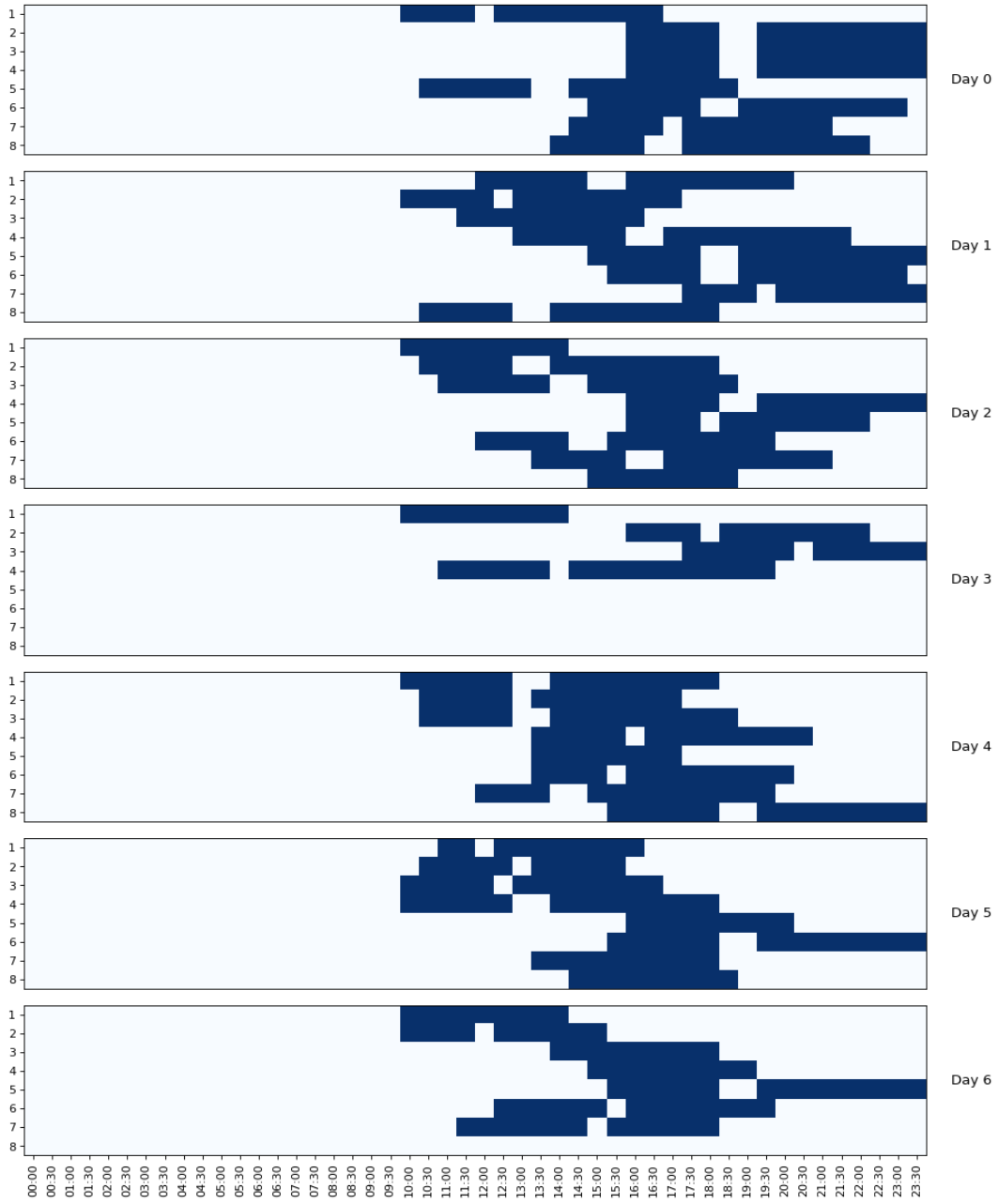


Figure C.29: Shifts boundaries for the prescribed solution in run *R3b*.

C.2.2 Experiment O1

Table C.8: Solutions descriptions in experiment $O1$.

Solution Name	Description
Prescribed Solution	Prescribed solution from run $R1a$ or $R1b$.
Manual Solution	Solution designed by the retailer for the same planning horizon as R1a and R1b.

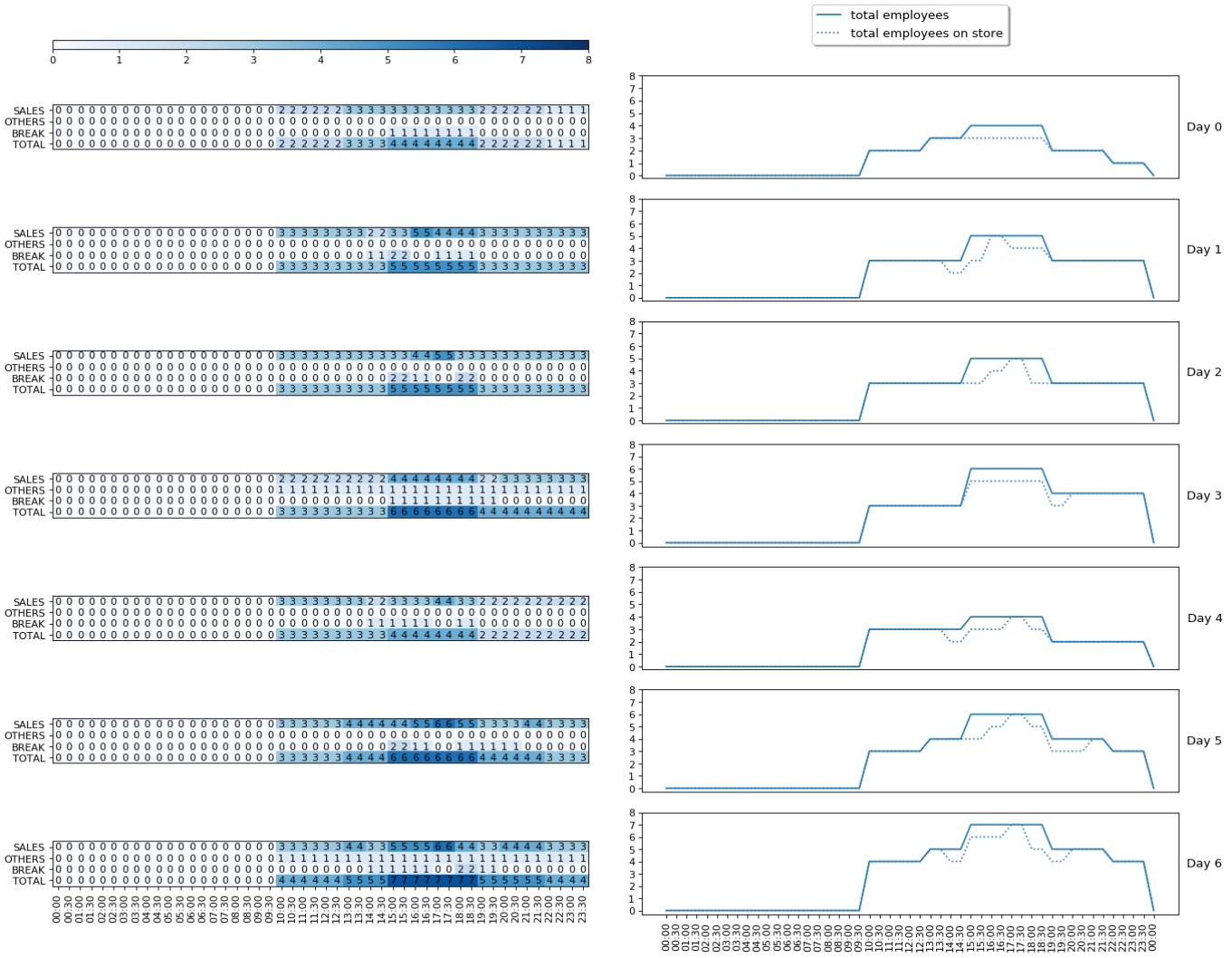


Figure C.30: Allocation for the solution solution in experiment $O1$

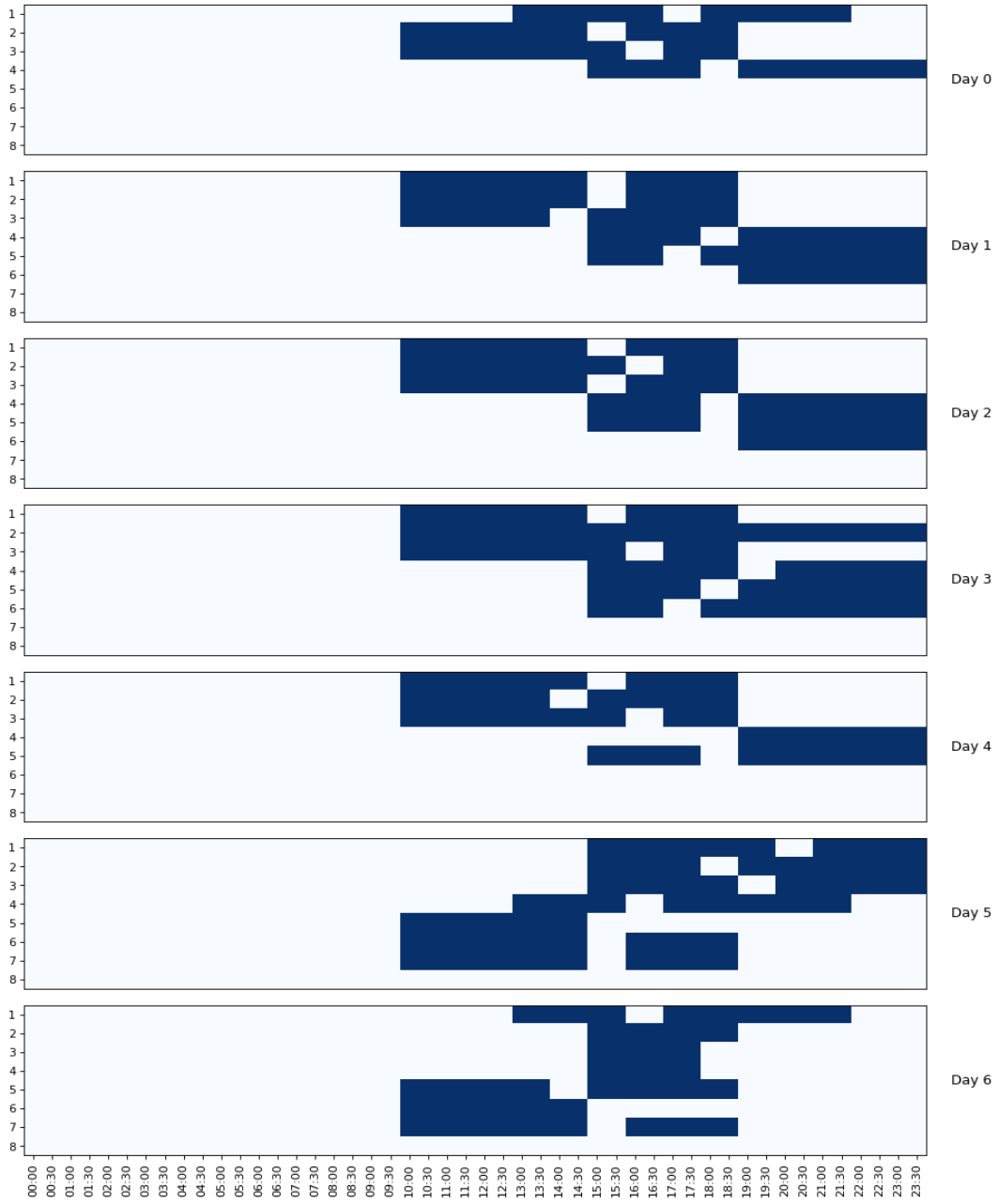


Figure C.31: Shifts boundaries for the manual solution in experiment $O1$.

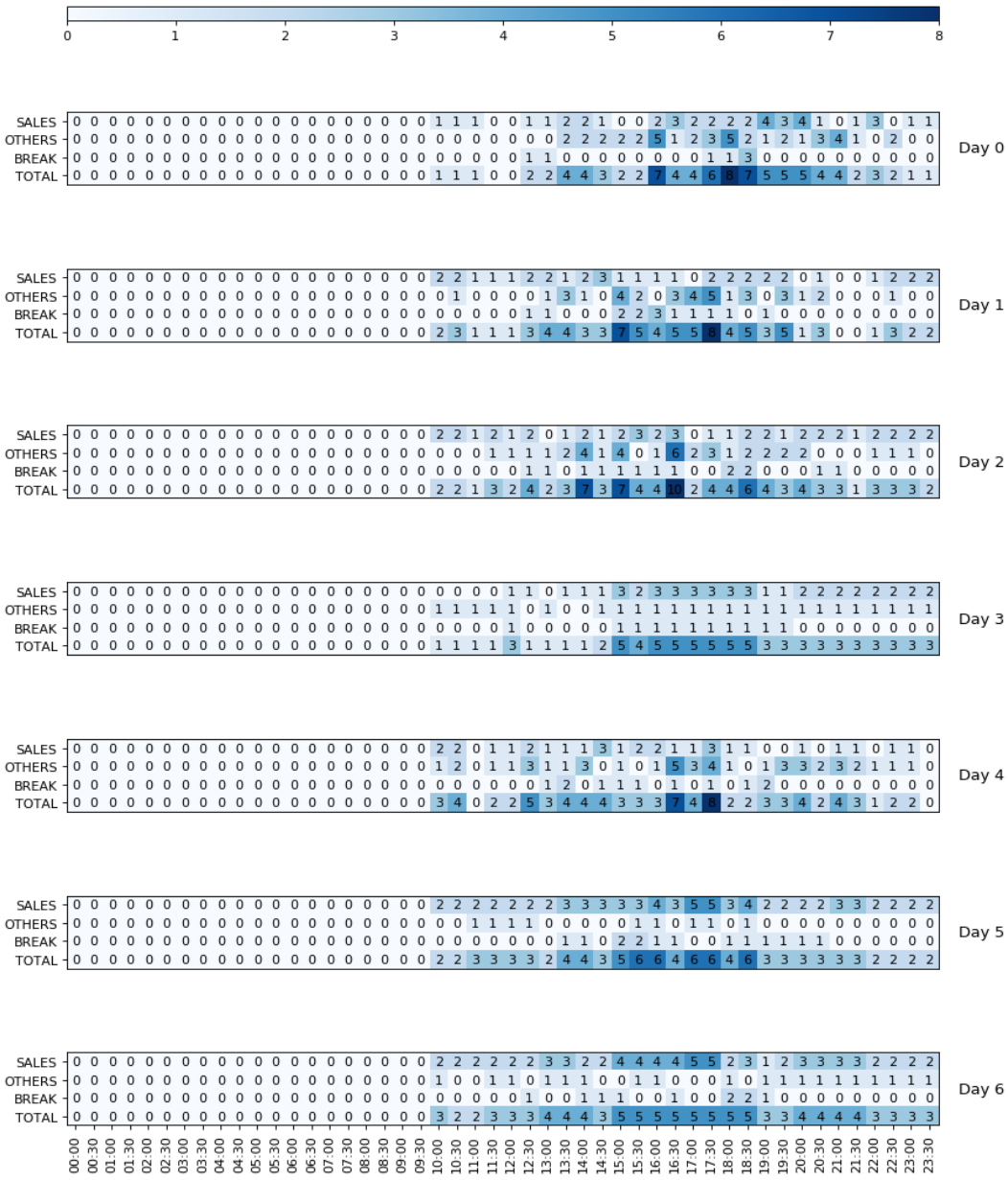


Figure C.32: Allocation distances for the schedules in experiment *O*1*a*

.

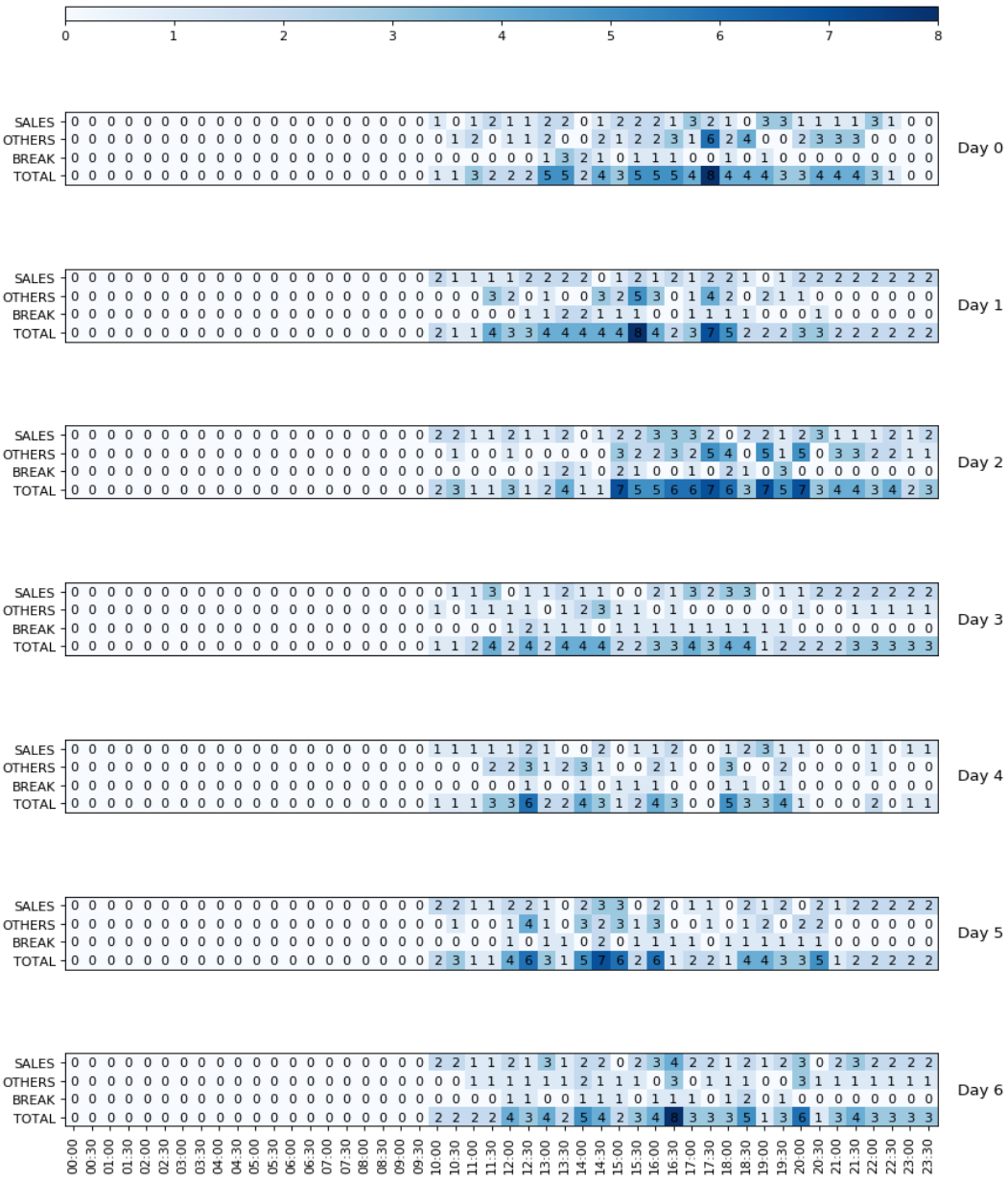


Figure C.33: Allocation distances for the schedules in experiment *O1b*

.

Table C.9: Distance and similarity between the prescribed and manual solutions in experiment $O1$.

Metric	Sub-Experiment	Level:0	Level:1	Level:2
Unit Distance	O1a	433	568	649
Unit Distance	O1b	327	500	591
Euclidean Distance	O1a	36.92	37.26	38.74
Euclidean Distance	O1b	27.95	33.41	35.17
Unit Similarity	O1a	68.44%	58.60%	52.70%
Unit Similarity	O1b	76.17%	63.56%	56.92%
Euclidean Similarity	O1a	62.33%	61.98%	60.47%
Euclidean Similarity	O1b	71.48%	65.91%	64.11%

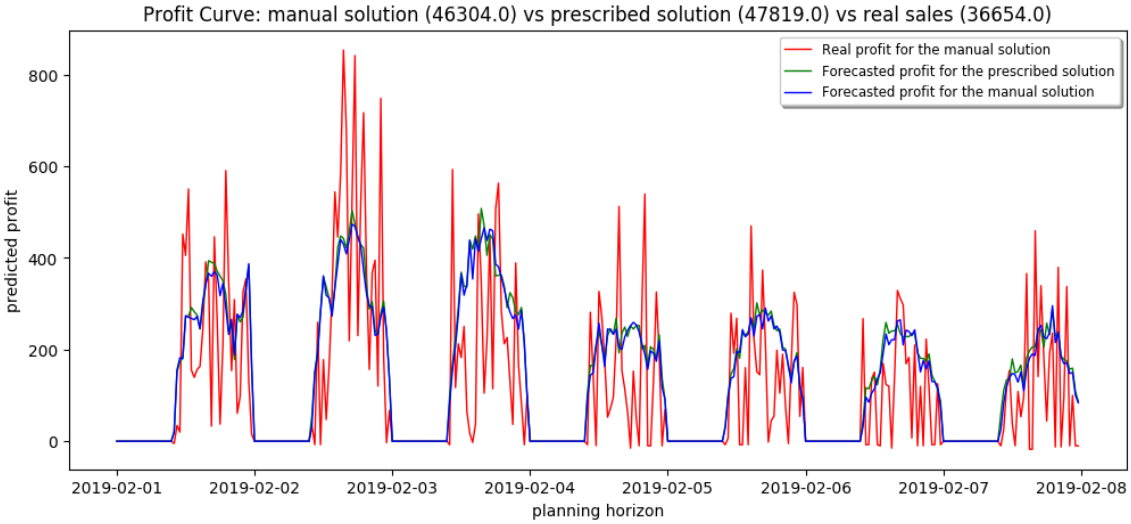


Figure C.34: Forecasted sales for the prescribed and manual solution in experiment $O1a$, with the real sales observed for the manual solution.

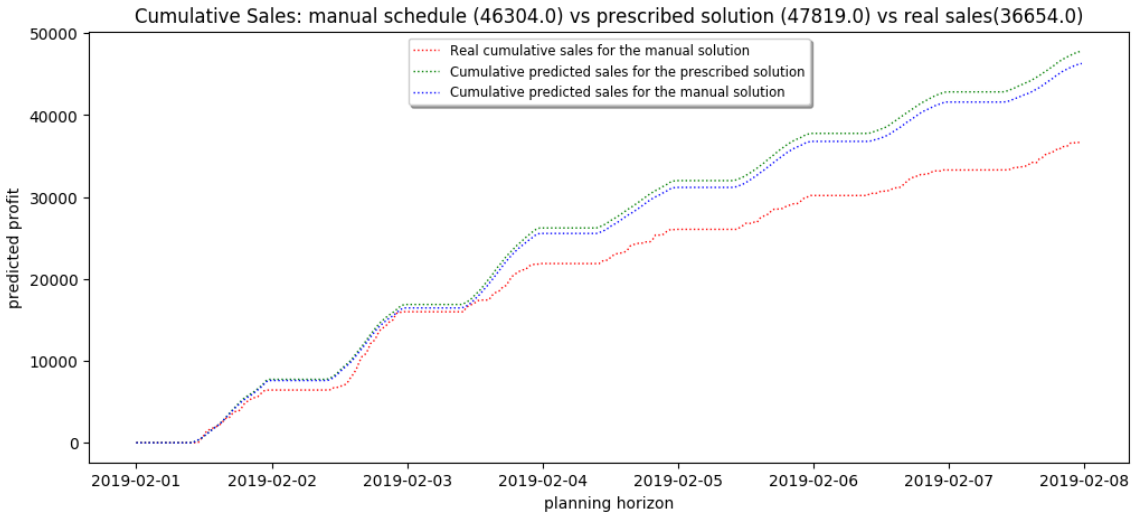


Figure C.35: Cumulative forecasted sales for the prescribed and manual solution in experiment $O1a$, with the real sales observed for the manual solution.

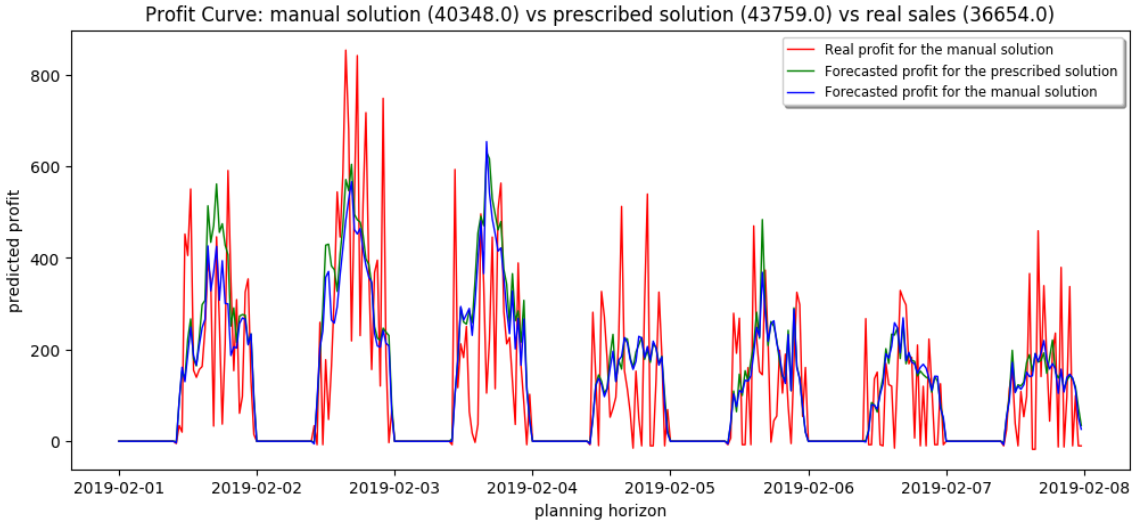


Figure C.36: Forecasted sales for the prescribed and manual solution in experiment $O1b$, with the real sales observed for the manual solution.

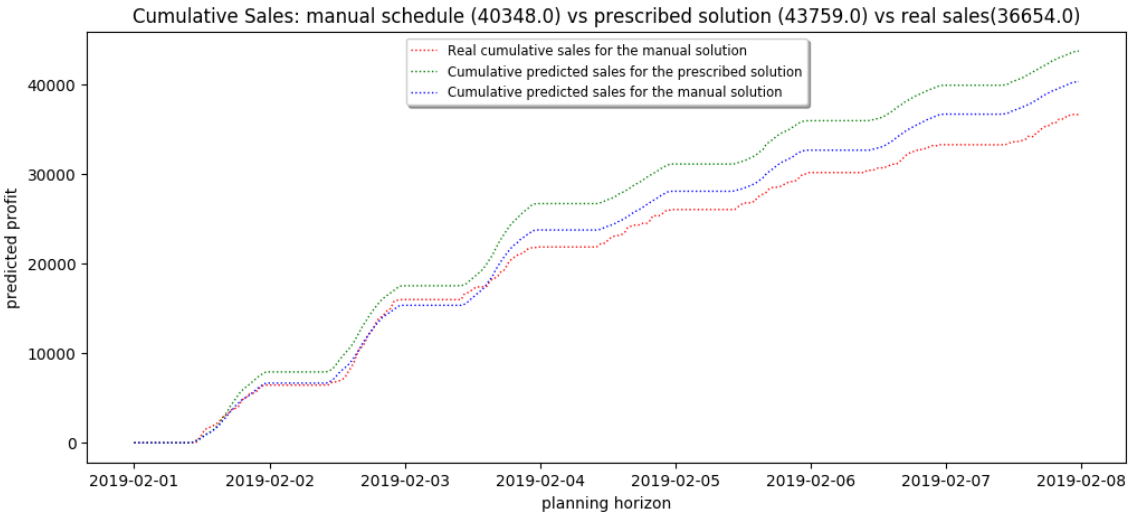


Figure C.37: Cumulative forecasted sales for the prescribed and manual solution in experiment $O1b$, with the real sales observed for the manual solution.

C.2.3 Experiment O2

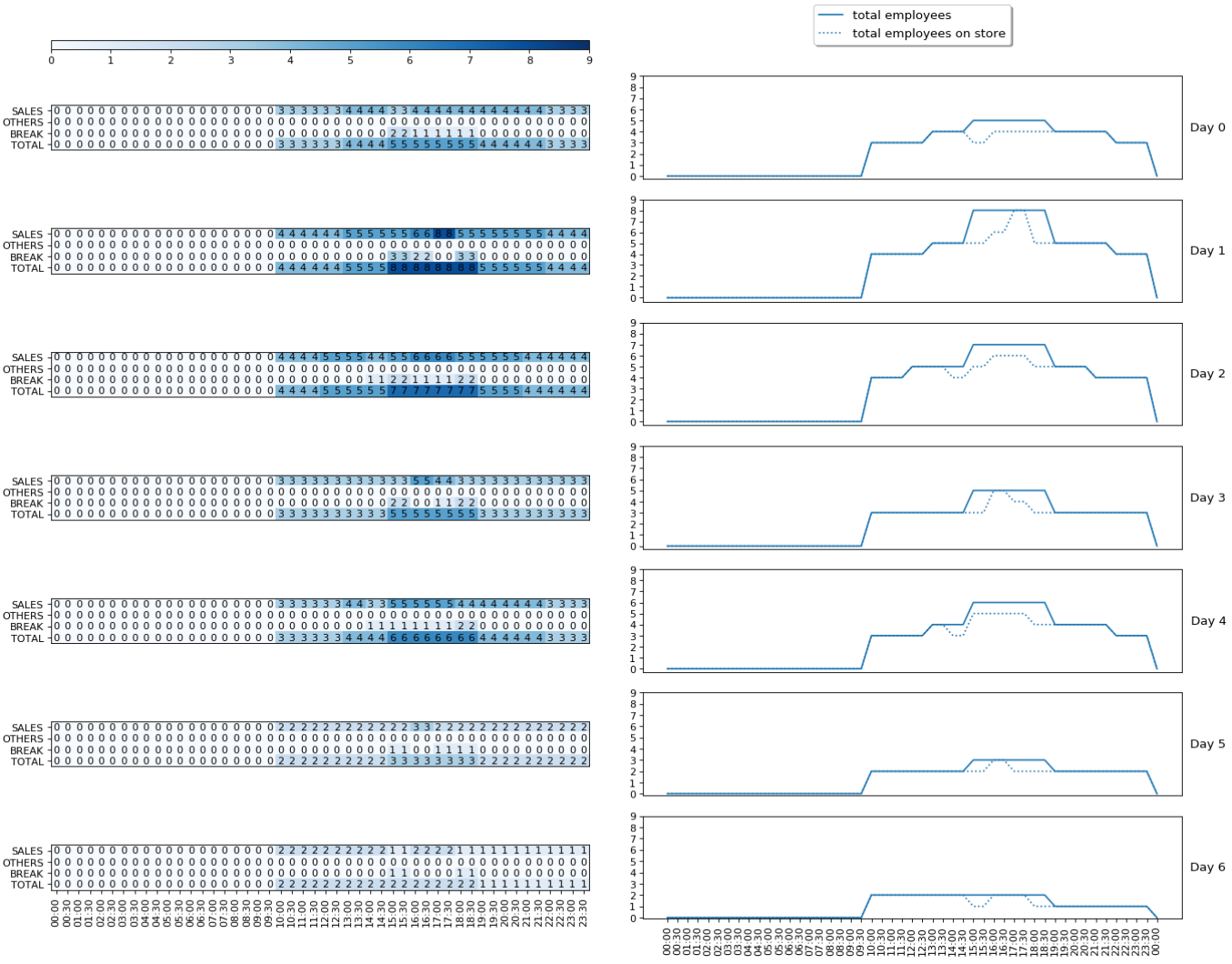


Figure C.38: Allocation for the first week in advance in the prescribed solution of run *R*2.

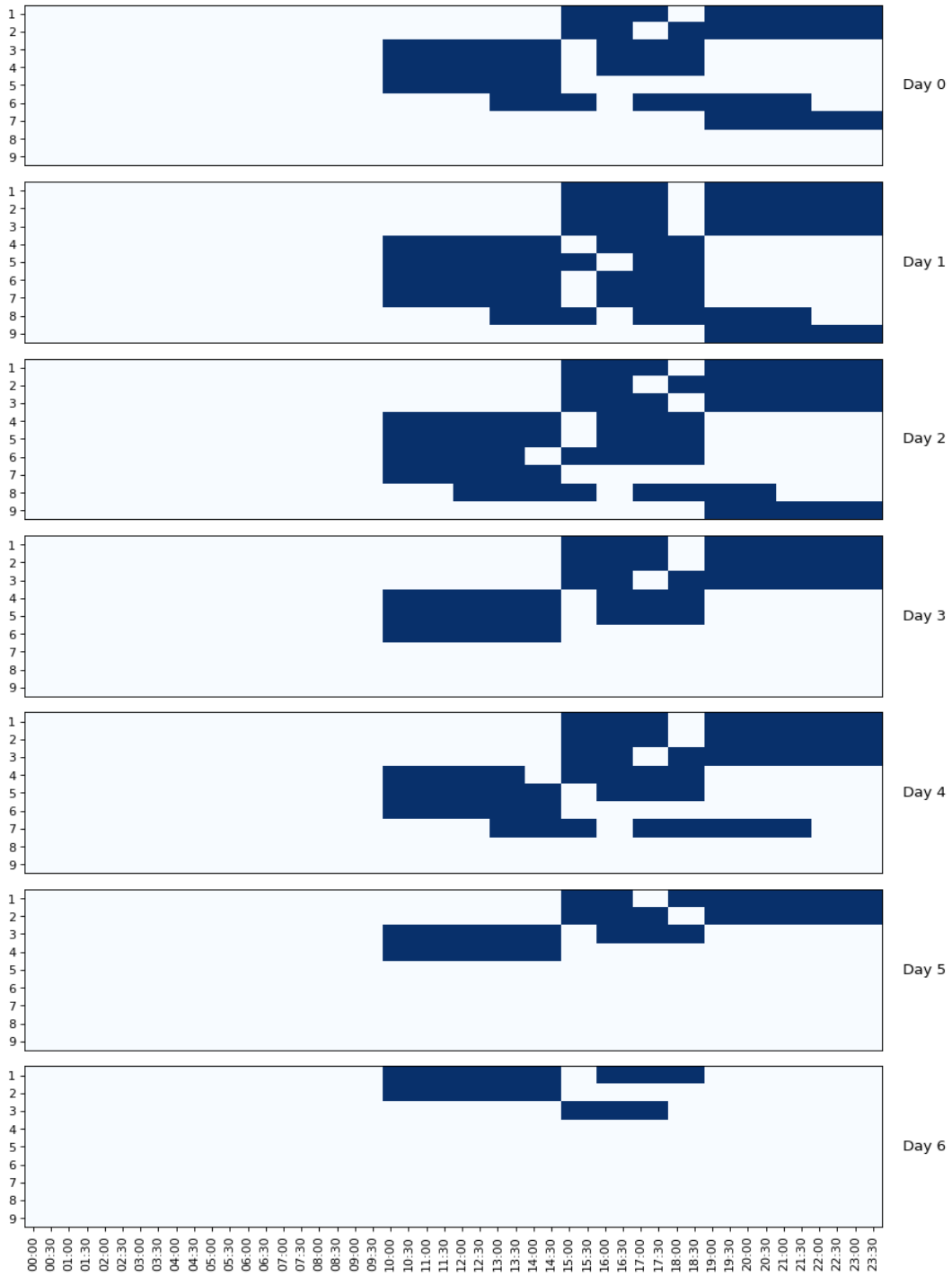


Figure C.39: Shifts boundaries for the first week in advance in the prescribed solution for run *R*2.

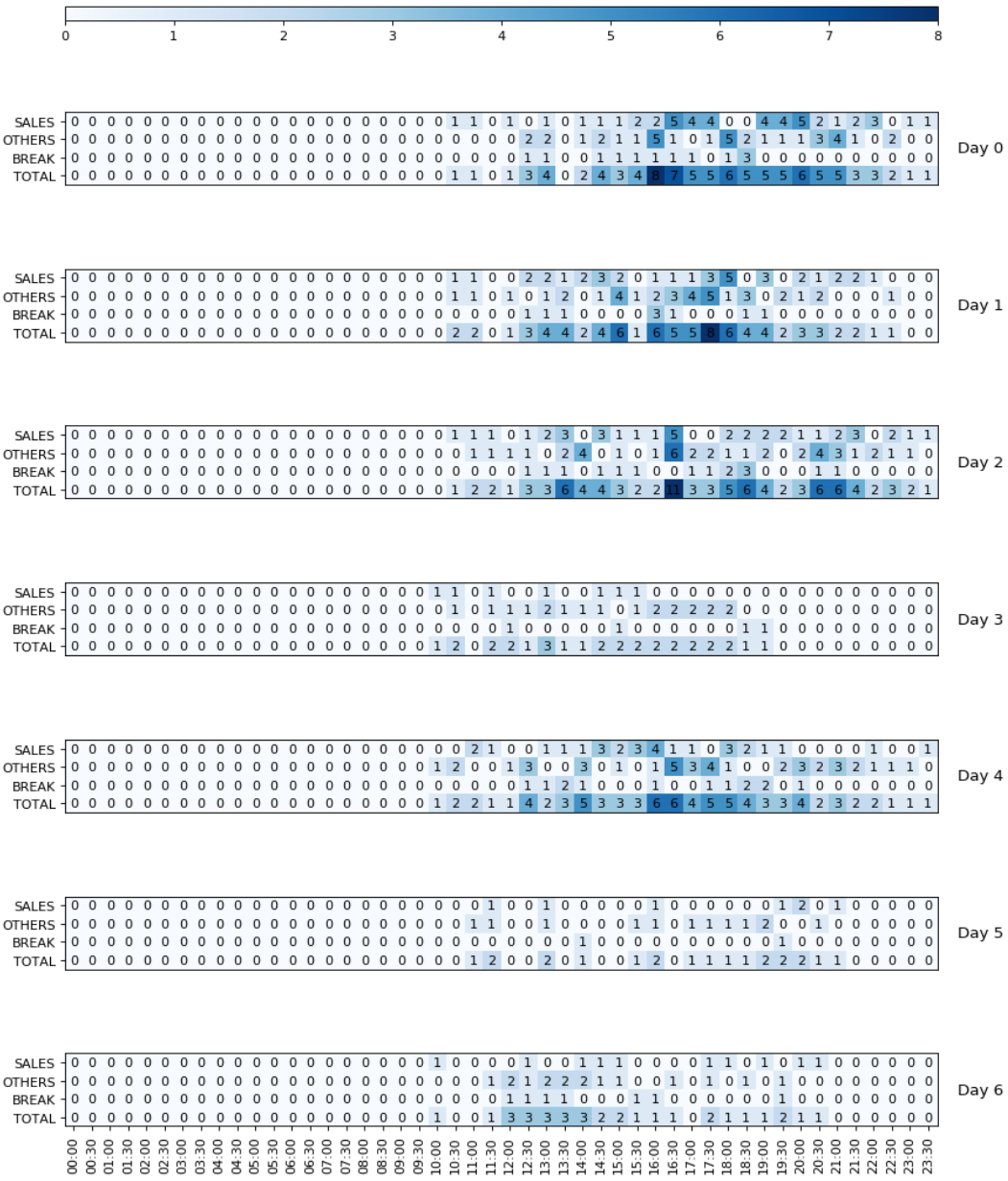


Figure C.40: Allocation distances for the schedules in experiment *O2a*

.

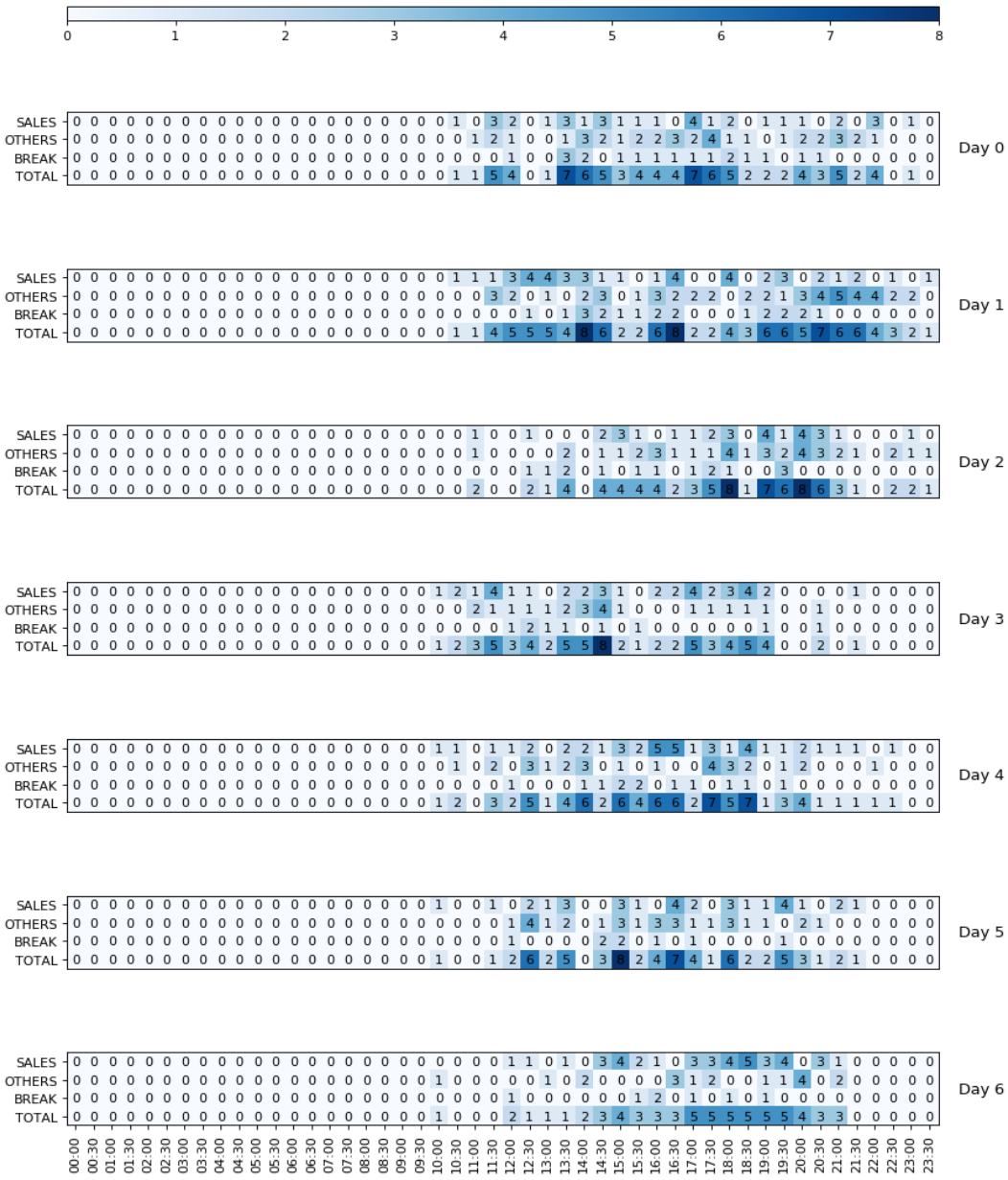


Figure C.41: Allocation distances for the schedules in experiment *O2b*

.

Table C.10: Allocation distances between the prescribed solutions in experiment *O*2.

Metric	Sub-Experiment	Level:0	Level:1	Level:2
Unit Distance	O2a	311	375	437
Unit Distance	O2b	293	476	565
Euclidean Distance	O2a	31.38	30.25	31.67
Euclidean Distance	O2b	30.02	35.13	37.05
Unit Similarity	O2a	77.33%	72.67%	68.15%
Unit Similarity	O2b	78.64%	65.31%	58.82%
Euclidean Similarity	O2a	67.97%	69.13%	67.68%
Euclidean Similarity	O2b	69.37%	64.15%	62.19%

Table C.11: Solutions descriptions in experiment *O*2.

Solution Name	Description
Solution 1	Prescribed solution in run *R*2*a* or *R*2*b*, with one week in advance.
Solution 1a	Solution 1 evaluated with one week in advance.
Solution 1b	Solution 1 evaluated with no time in advance.
Solution 2	Prescribed solution from run *R*1*a* or *R*1*b*

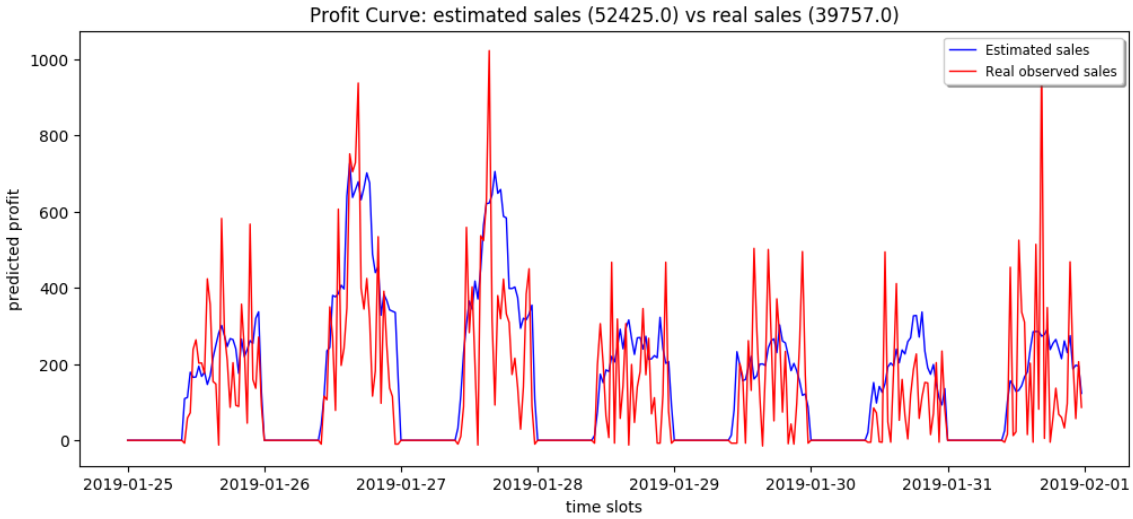


Figure C.42: Forecasted sales for the first week in advance of the prescribed solution in run *R*2*a*, and the real sales observed in *R*1*a*, in experiment *O*2*a*.

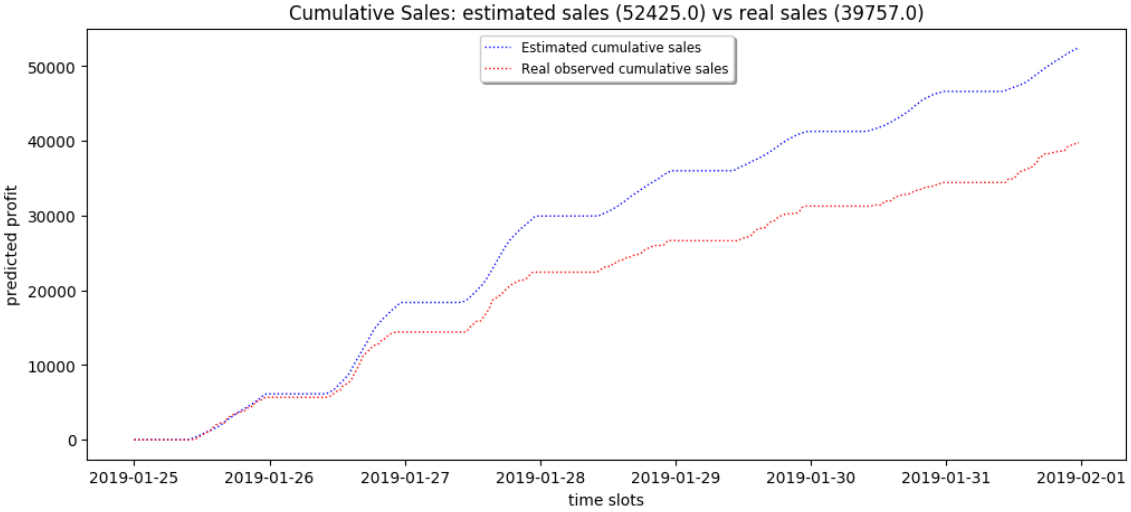


Figure C.43: Cumulative forecasted sales for the first week in advance of the prescribed solution in run *R*2*a*, and the real sales observed in *R*1*a*, in experiment *O*2*a*.

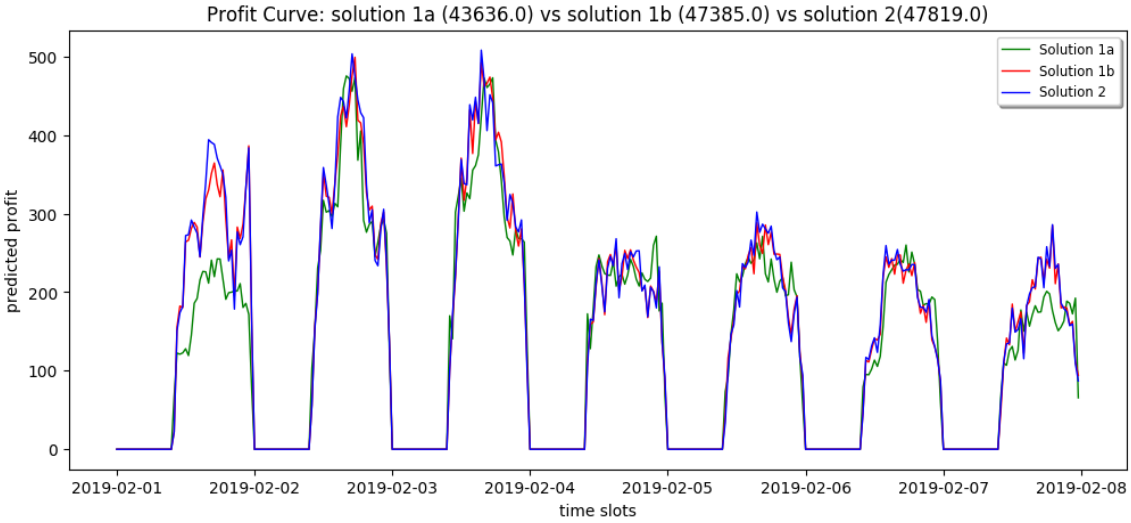


Figure C.44: Forecasted sales for the prescribed solution in run *R*2*a* and the solution in *R*1*a*, in experiment *O*2*a*.

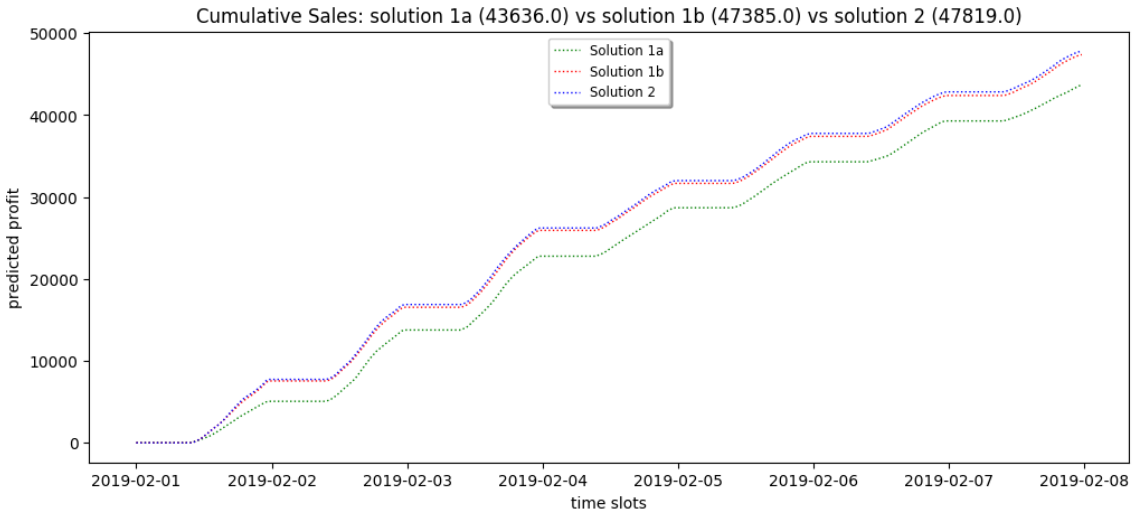


Figure C.45: Cumulative forecasted sales for the prescribed solution in run *R*2*a* and the solution in *R*1*a*, in experiment *O*2*a*.

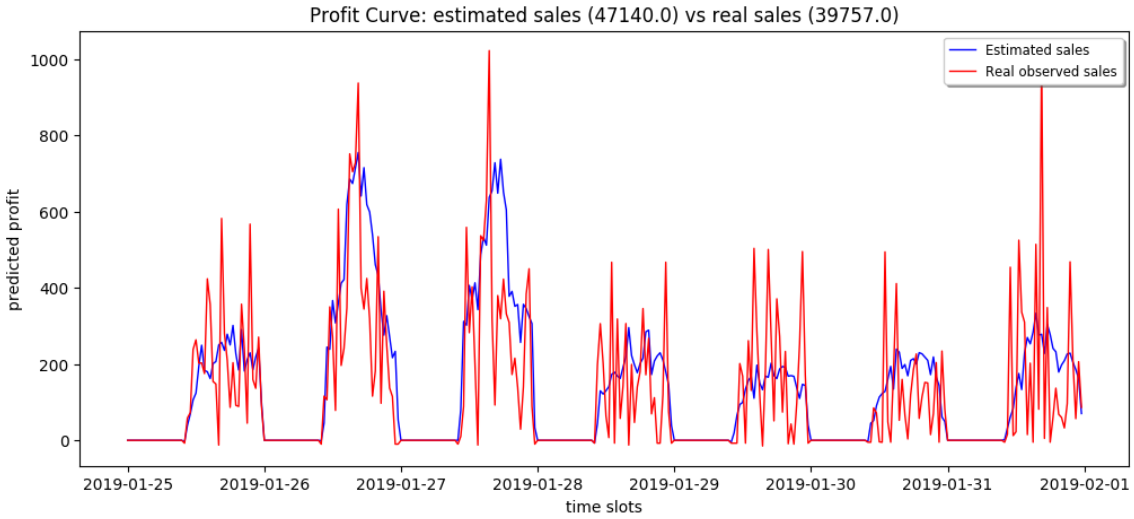


Figure C.46: Forecasted sales for the first week in advance, of the prescribed solution in run *R*2*b*, and the real sales observed in *R*1*b*, in experiment *O*2*b*.

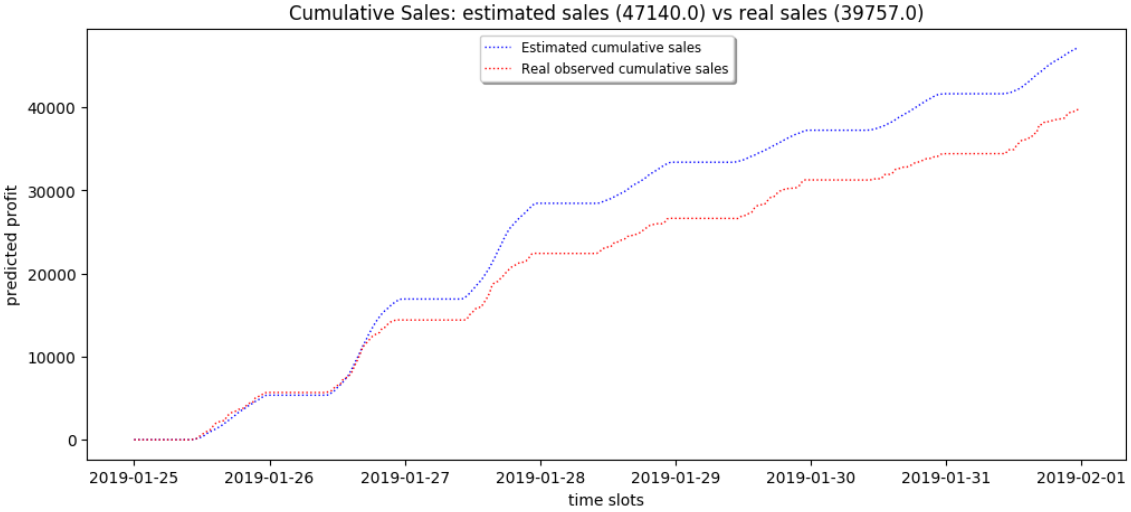


Figure C.47: Cumulative forecasted sales for the first week in advance, of the prescribed solution in run *R*2*b*, and the real sales observed in *R*1*b*, in experiment *O*2*b*.

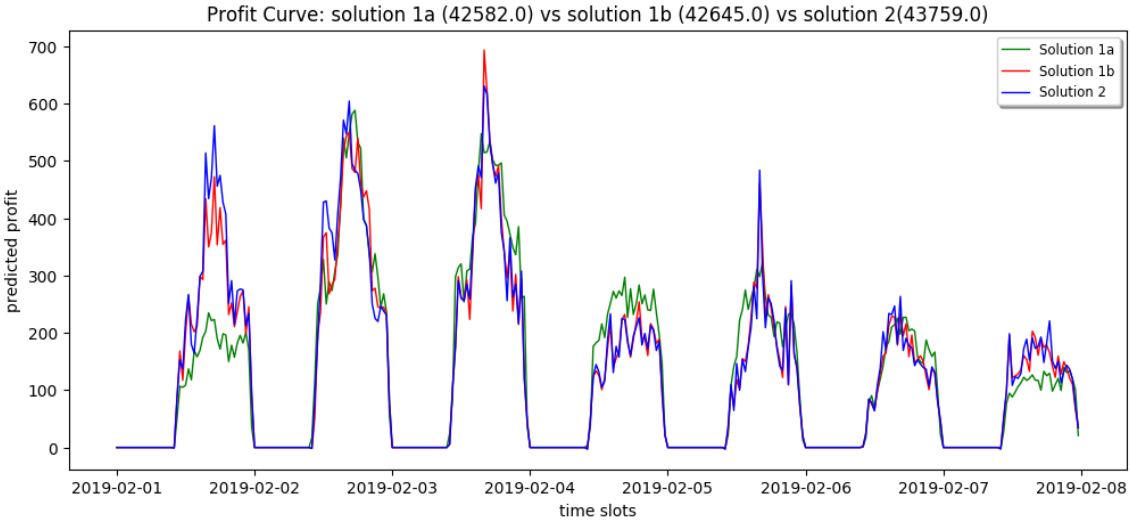


Figure C.48: Forecasted sales for the prescribed solution in run *R*2*b* and solution in *R*1*b*, in experiment *O*2*b*.

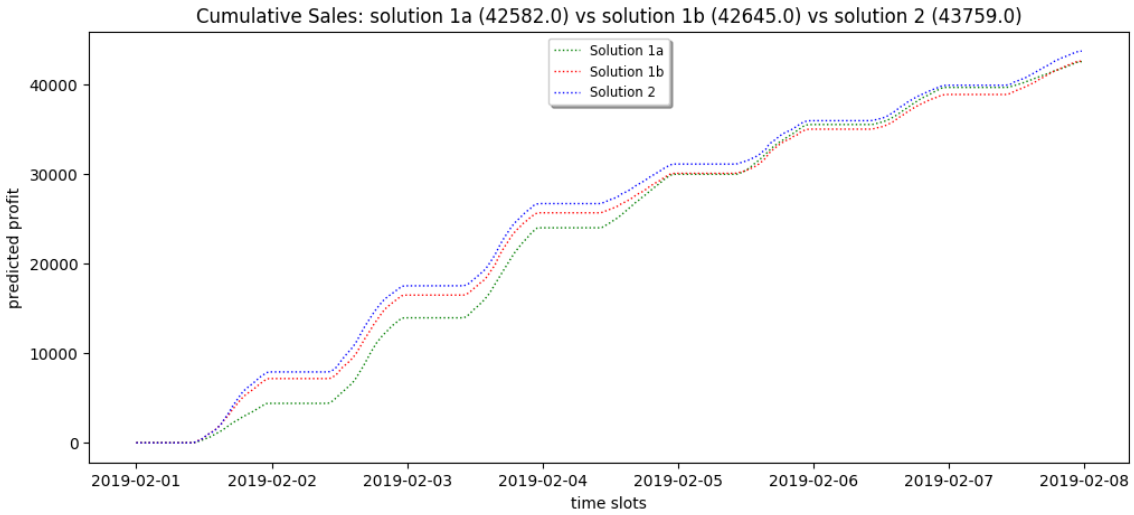


Figure C.49: Cumulative forecasted sales for the prescribed solution in run *R*2*b* and solution in *R*1*b*, in experiment *O*2*b*.

C.2.4 Experiment O3

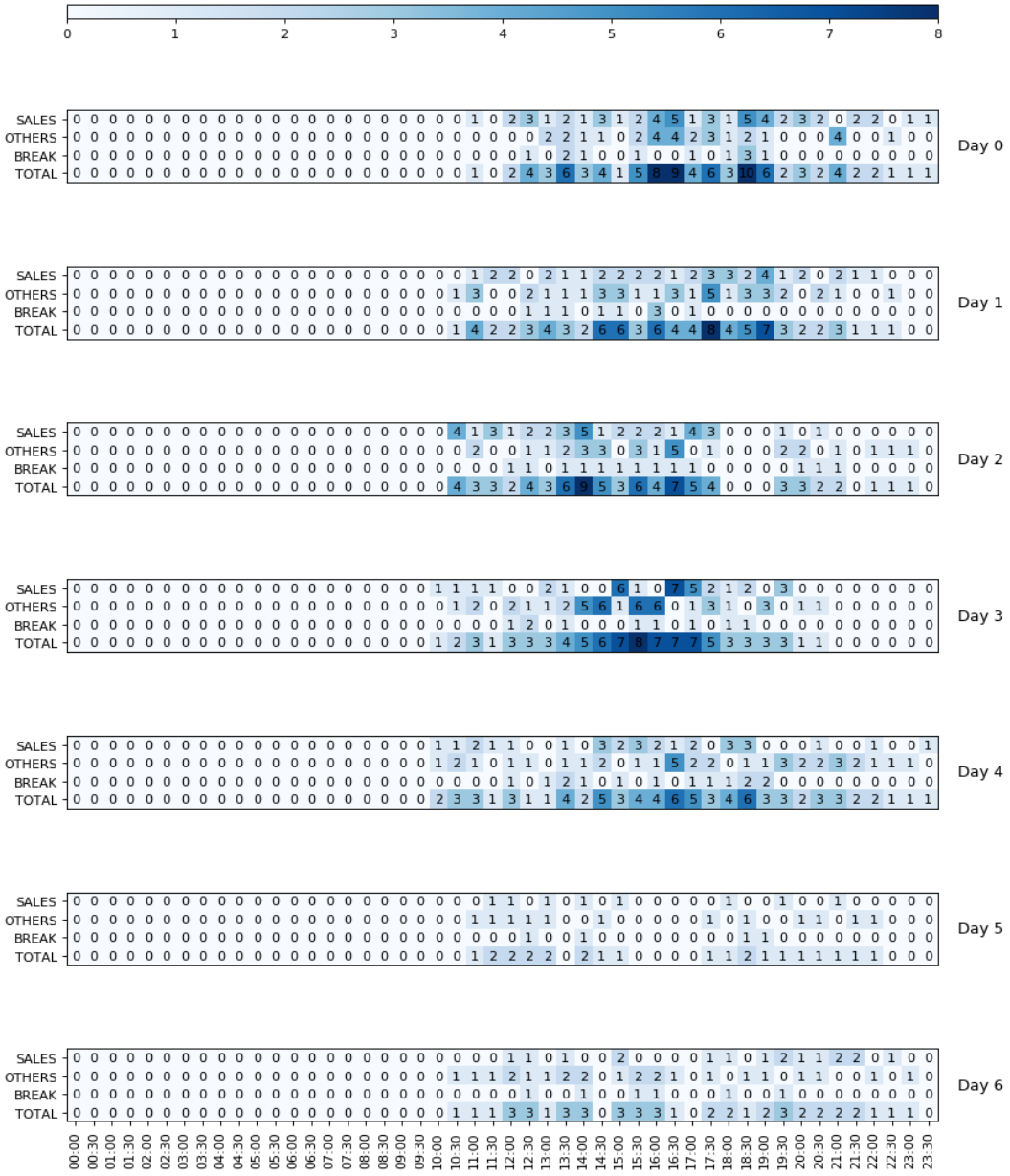


Figure C.50: Allocation distances for the schedules in experiment *O3a*

.



Figure C.51: Allocation distances for the schedules in experiment *O3b*

.

Table C.12: Allocation distances between the prescribed solutions in experiment *O*3.

Metric	Sub-Experiment	Level:0	Level:1	Level:2
Unit Distance	O2a	273	433	499
Unit Distance	O2b	260	490	570
Euclidean Distance	O2a	28.41	33.78	35.06
Euclidean Distance	O2b	24.21	35.52	37.28
Unit Similarity	O2a	80.10%	68.44%	63.63%
Unit Similarity	O2b	81.05%	64.29%	58.45%
Euclidean Similarity	O2a	71.01%	65.53%	64.23%
Euclidean Similarity	O2b	75.30%	63.75%	61.96%

Table C.13: Solutions descriptions for experiment *O*3.

Solution Name	Description
Solution 1	Prescribed solution from run *R*1*a* or *R*1*b*.
Solution 2	Prescribed solution from run *R*3*a* or *R*3*b* evaluated with additional error in footfall in.
Solution 2a	Solution 2 evaluated with an additional 20% error in the forecasted footfall in attribute.
Solution 2b	Solution 2 evaluated with no additional error in the forecasted footfall in attribute.

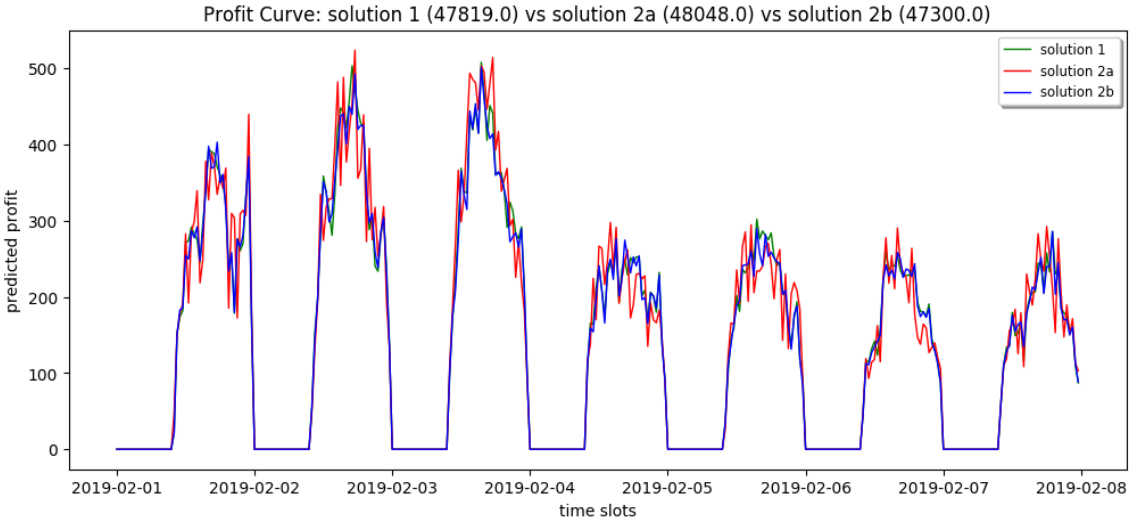


Figure C.52: Forecasted sales for the prescribed solution in run *R*1*a* and solution in *R*3*a*, when evaluated with and without any additional error in footfall_in.

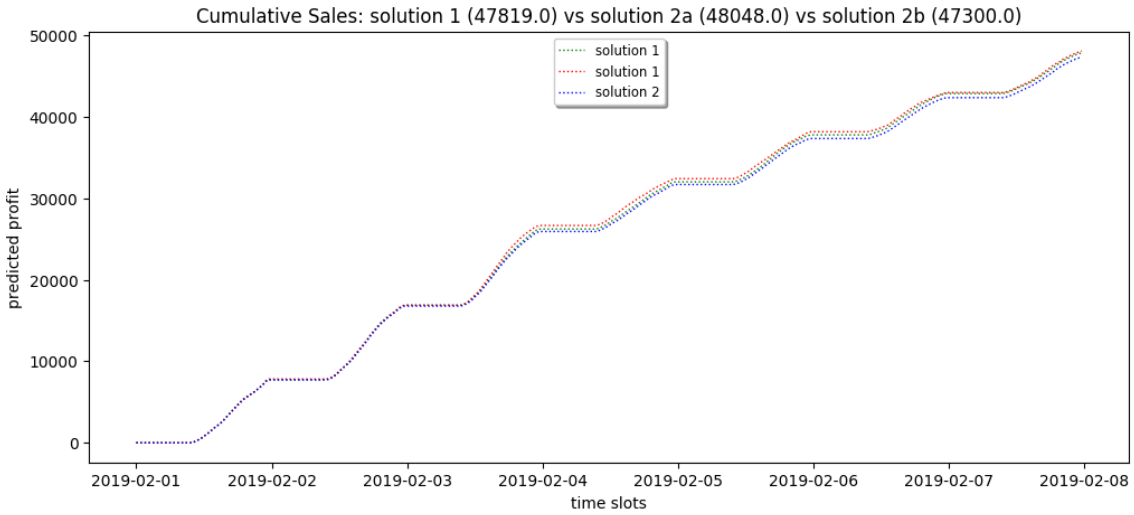


Figure C.53: Cumulative forecasted sales for the prescribed solution in run *R*1*a* and solution in *R*3*a*, when evaluated with and without any additional error in footfall_in.

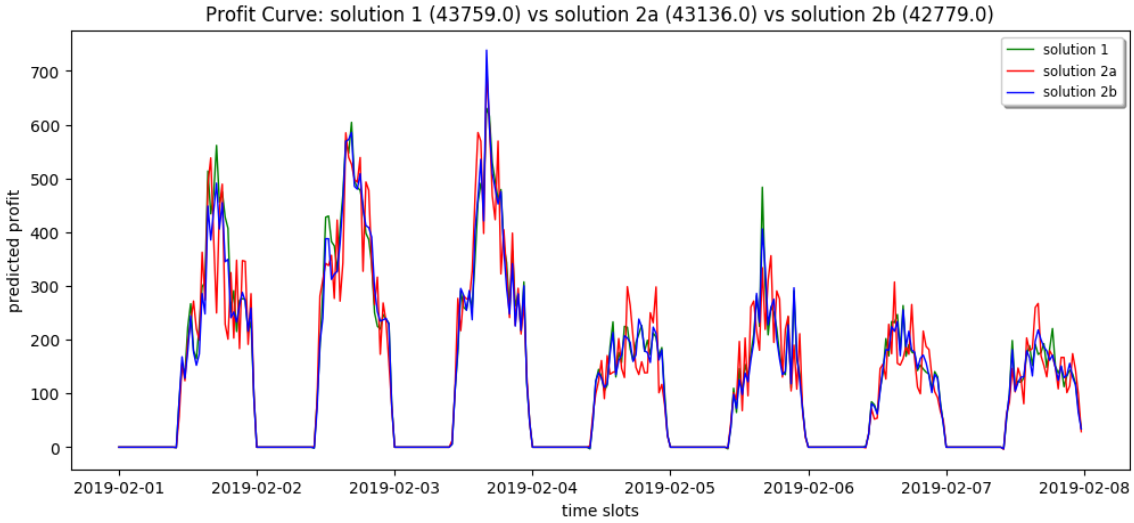


Figure C.54: Forecasted sales for the prescribed solution in run *R*1*b* and solution in *R*3*b*, when evaluated with and without any additional error in footfall_in.

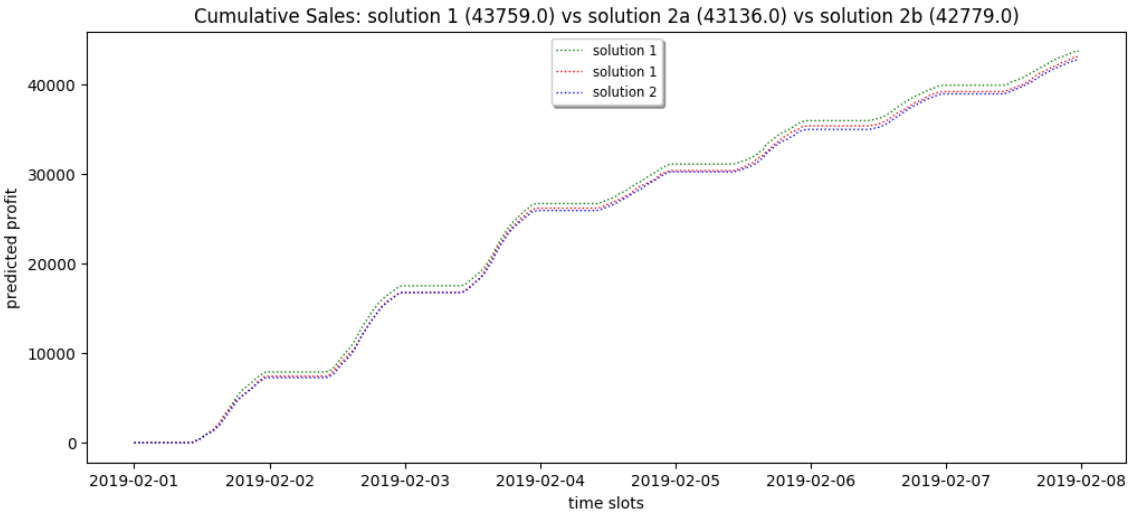


Figure C.55: Cumulative forecasted sales for the prescribed solution in run *R*1*b* and solution in *R*3*b*, when evaluated with and without any additional error in footfall_in.

C.2.5 Experiment O4

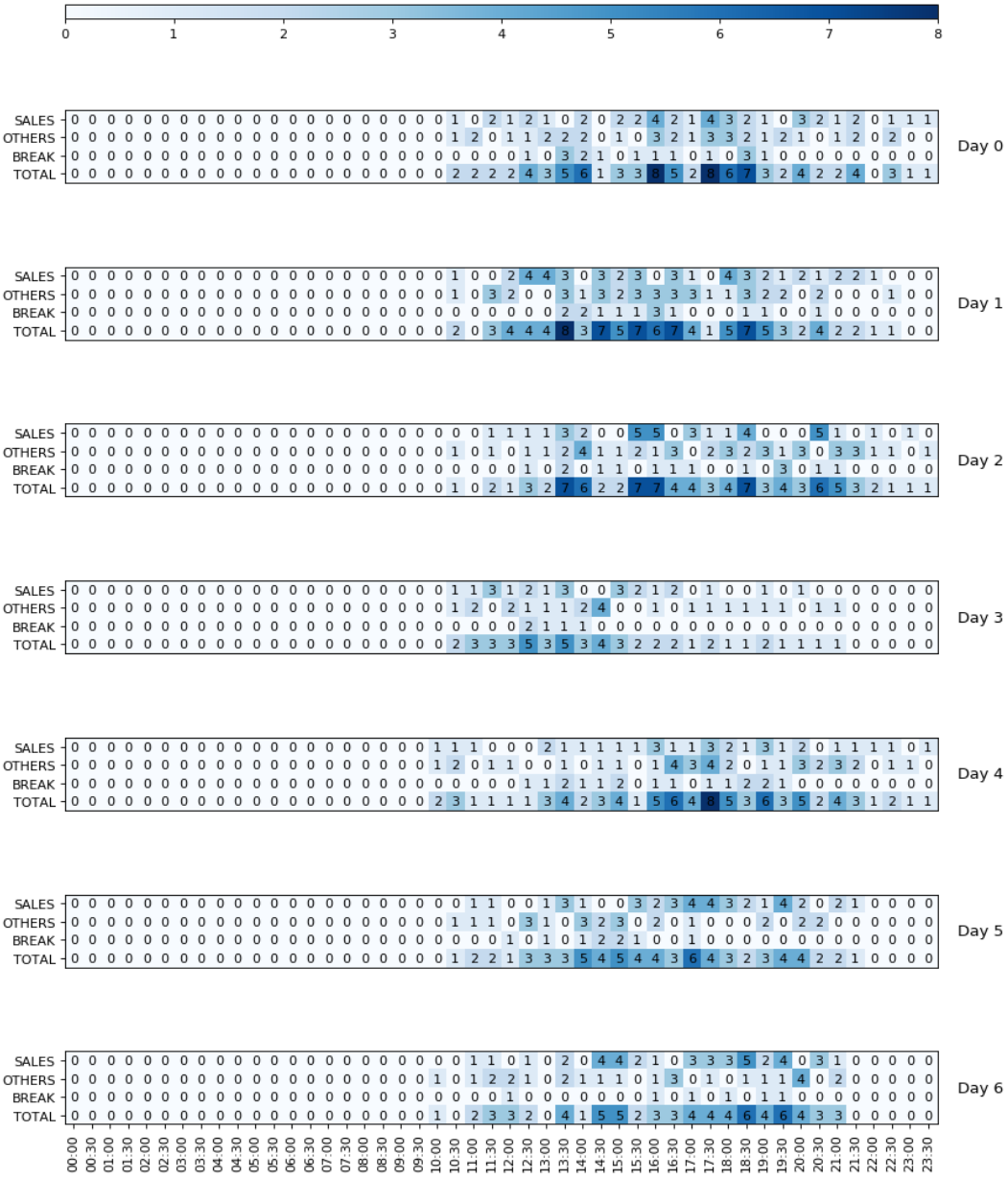


Figure C.56: Allocation distances for the schedules in experiment *O*4

.

Table C.14: Allocation distances between the prescribed solutions in experiment $O4$.

Metric	Level:0	Level:1	Level:2
Unit Distance	320	478	558
Euclidean Distance	29.33	34.21	36.00
Unit Similarity	76.68%	65.16%	59.33%
Euclidean Similarity	70.08%	65.10%	63.27%

Table C.15: Solutions descriptions for experiment $O4$.

Solution Name	Description
Solution 1	Prescribed solution from run $R1a$ through a direct sales forecasting architecture.
Solution 2	Prescribed solution from run $R1b$ through an indirect sales forecasting architecture.

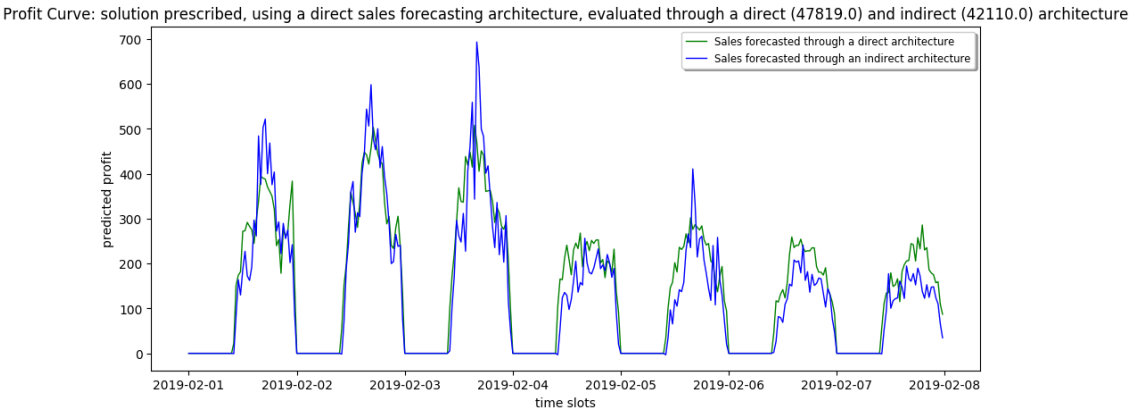


Figure C.57: Forecated sales for the prescribing solution in run $R1a$ using a direct and indirect sales forecasting architecture.

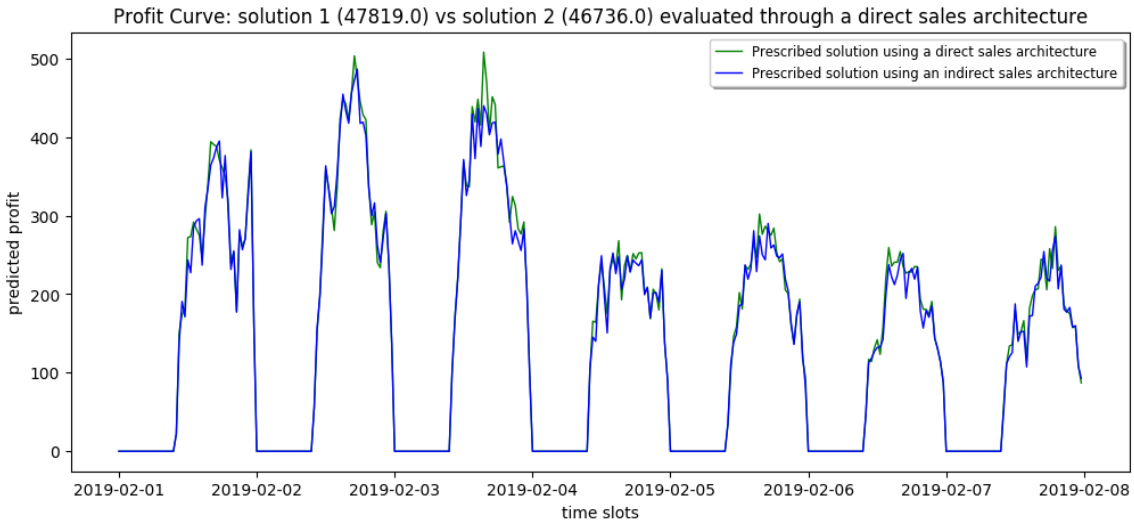


Figure C.58: Forecasted sales for the prescribed solution in run $R1a$ and solution in $R1b$, when evaluated through a direct sales forecasting architecture.

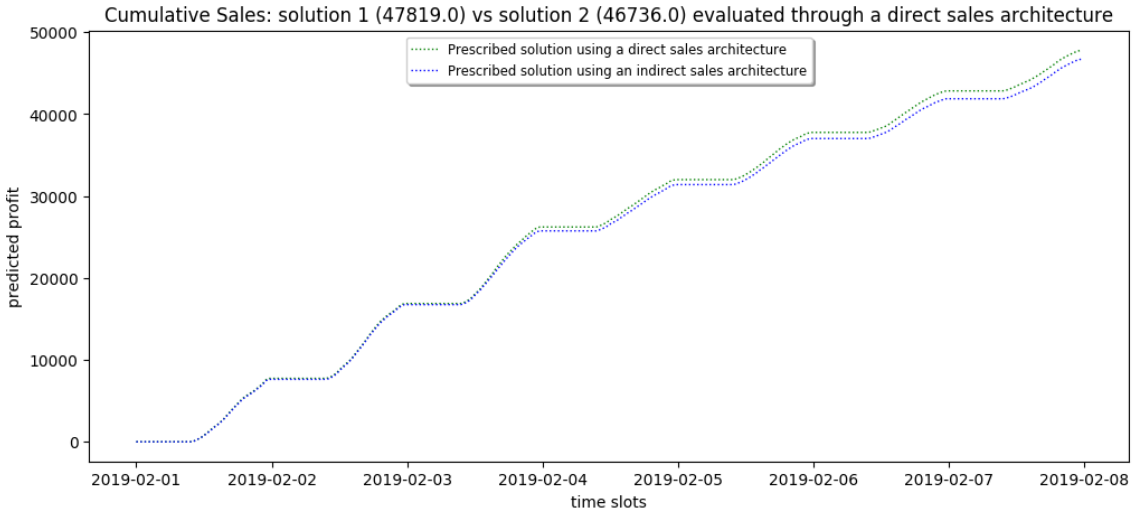


Figure C.59: Cumulative forecasted sales for the prescribed solution in run $R1a$ and solution in $R1b$, when evaluated through a direct sales forecasting architecture.

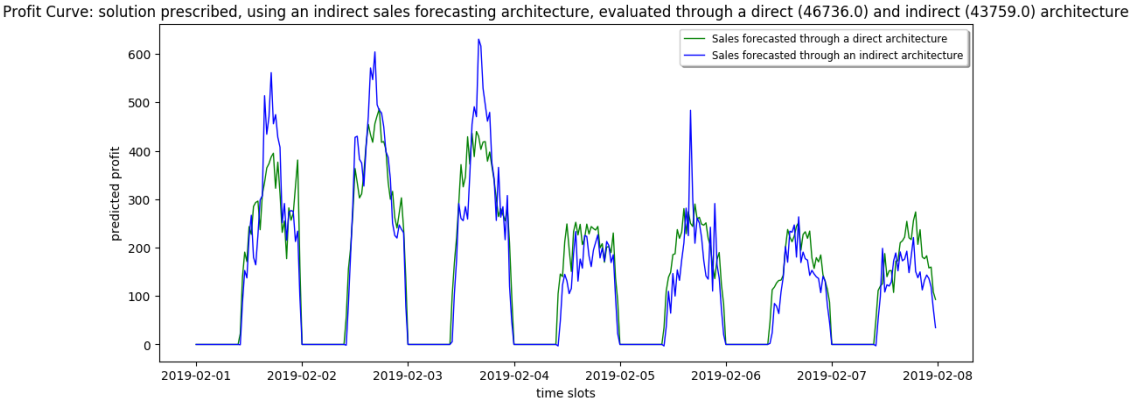


Figure C.60: Forecasted sales for the prescribed solution in run *R*1*b* using a direct and indirect sales forecasting architecture.

.

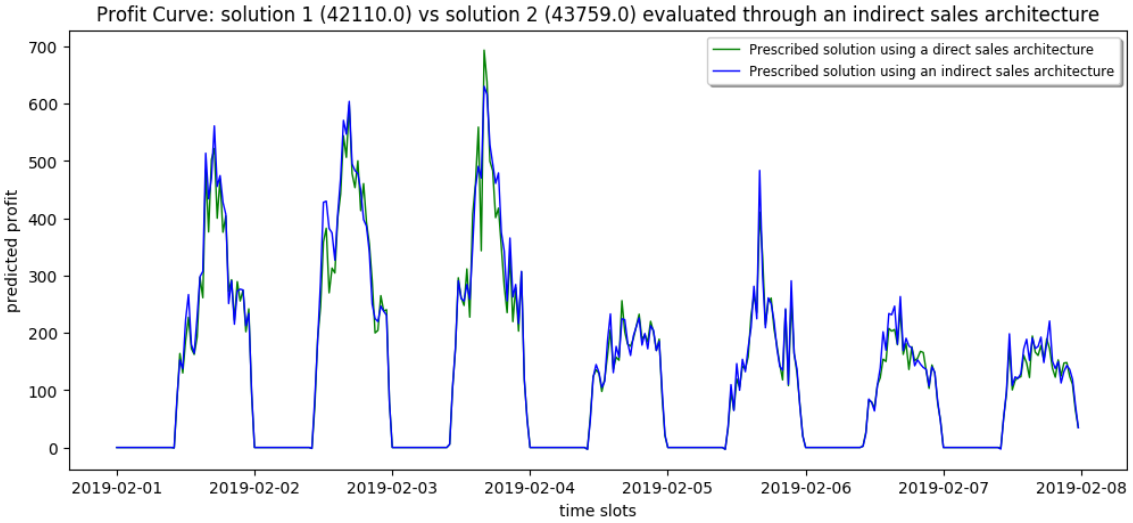


Figure C.61: Forecasted sales for the prescribed solution in run *R*1*a* and solution in *R*1*b*, when evaluated through an indirect sales forecasting architecture.

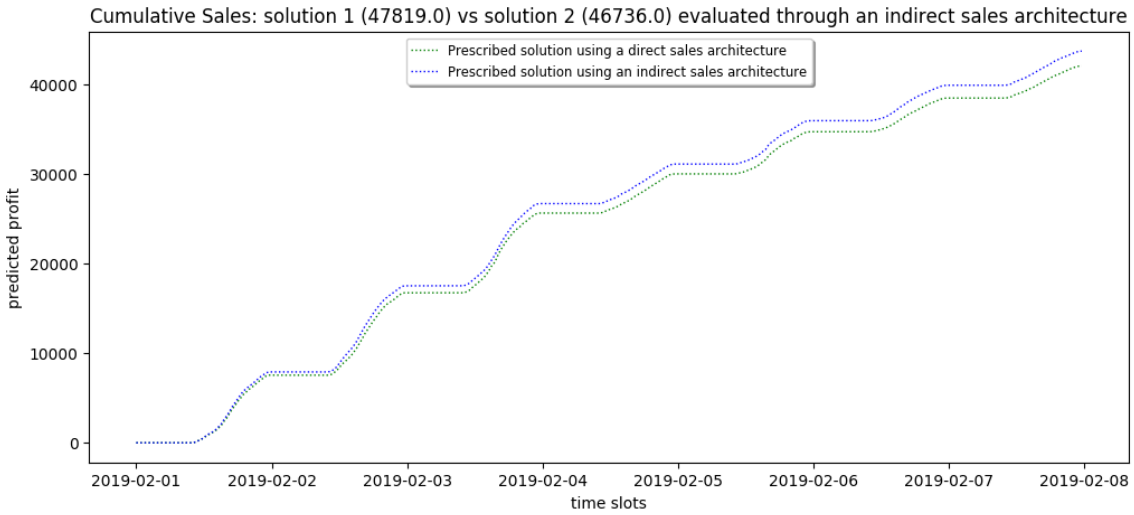


Figure C.62: Cumulative forecasted sales for the prescribed solution in run $R1a$ and solution in $R1b$, when evaluated through an indirect sales forecasting architecture.