# The University of Southampton

**2023/24**

## Faculty of Social Sciences

## Southampton Business School

## MSc Dissertation

# Exact and Heuristic Methods for Box Allocation Optimisation in Meal Kit Delivery

ERGO number: 94359

Student number: 35436506

**Presented for MSc. Logistics and Supply Chain Analytics**

This project is entirely the original work of the student registration number 35436506. I declare that this dissertation is my own work and that where material is obtained from published or unpublished works, this has been fully acknowledged in the references. This may include material from my own work from a research proposal that has been previously submitted for assessment for this program.

Word Count: 16,387 words

# Abstract

The rapid expansion of meal kit delivery services has intensified the pressure on companies in this sector to optimize their supply chain operations. This dissertation focuses on the critical problem of allocating the recipe boxes to suitable factories to minimize the day-to-day variations in the number of recipes prepared at each factory, measured by the Weighted mean absolute percentage error (WMAPE). By minimizing this error, companies can have more stable operations across factories, improving efficiency and reducing waste.

Although research on box allocation problem (BAP) is limited, this problem shares similarities with two main optimization problems including Bin packing problem (BPP) and Capacitated vehicle routing problem (CVRP), offering ideas for its solution approaches. However, the meal kit delivery context introduces unique features and constraints that require a specialized approach. This study aims to bridge the research gap by proposing some algorithms effective for BAP.

A mixed-integer linear programming (MILP) model is first developed to capture the characteristics of BAP. Based on this model, three methods: Branch and Bound (B&B) in CBC solver, Iterative targeted pairwise swap (ITPS), and Tabu search (TS) are applied to find the optimal order allocation that minimizes WMAPE site.

The effectiveness of the proposed methods is evaluated by comparing WMAPE site (which considers errors in each factory) with WMAPE global (no factories are involved), as global error is the theoretical lower bound of site error. Comprehensive computational experiments are conducted across a wide range of datasets to assess the scalability and robustness of the algorithms. The results indicate that while two heuristics specifically designed for BAP, particularly TS, are highly effective, B&B algorithm in CBC solver consistently outperforms them in both solving time and optimality. This highlights its potential for real business applications.

**Keywords:** Box allocation problem, Meal kit delivery, Branch and Bound, CBC solver, Heuristics, Iterative targeted pairwise swap, Tabu search

# Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Dr. Alain Zemkoho, for his invaluable guidance and unwavering support throughout my dissertation. His deep understanding and experience in optimization helped guide my research greatly. I am truly grateful for the hours he dedicated to discussing ideas, providing constructive feedback, and helping me identify the challenges of this project. His mentorship has not only enriched my academic experience but also inspired me to strive for excellence in the future.

I would like to extend my sincere gratitude to the project sponsor, Mr. Loïc Genest, for providing me with an incredible opportunity to work on this exciting and impactful project. I am deeply grateful for the time and effort he dedicated to guiding me through the intricacies of the problem and patiently explaining each step. His hands-on approach and direct guidance in troubleshooting and refining my code have been crucial to the success of this project. I truly appreciate his trust in my abilities and the resources he made available to support my work.

I am also deeply grateful to the academic staff and faculty members at the University of Southampton, particularly Dr. Carlos Lamas Fernandez and Dr. Toni Martinez Sykora, for their guidance on optimization techniques, which is crucial for this dissertation.

Finally, I would like to express my heartfelt gratitude to my family and friends for their persistent love and understanding throughout this challenging but rewarding process. Their belief in my abilities and encouragement has been a driving force behind the completion of this project.

# Contents

# List of Figures

# List of Tables

# Abbreviations

| Abbreviation | Full name |
|---|---|
| ACO | Ant Colony Optimization |
| B&B | Branch and Bound |
| B&C | Branch and Cut |
| B&P | Branch and Price |
| BAP | Box Allocation Problem |
| BF | Best Fit |
| BFD | Best Fit Decreasing |
| BPP | Bin Packing Problem |
| CBC | COIN-OR Branch and Cut |
| CLP | COIN-OR Linear Programming |
| COP | Combinatorial Optimization Problem |
| CVRP | Capacitated Vehicle Routing Problem |
| FF | First Fit |
| FFD | First Fit Decreasing |
| FP | Feasibility Pump |
| GA | Genetic Algorithm |
| ITPS | Iterative Targeted Pairwise Swap |
| LB | Lower Bound |
| LD | Lead Day |
| LNS | Large Neighborhood Search |
| LP | Linear Programming |
| LS | Local Search |
| ILP | Integer Linear Programming |
| MILP | Mixed-Integer Linear Programming |
| SA | Simulated Annealing |
| TS | Tabu Search |
| TSP | Traveling Salesman Problem |
| UB | Upper Bound |
| VRP | Vehicle Routing Problem |
| WMAPE | Weighted Mean Absolute Percentage Error |

# Chapter 1

# Introduction

This chapter is organized into four sections. Section 1.1 provides the background information and motivation for this research. Section 1.2 outlines the research objectives and their significance. Section 1.3 offers a brief overview of the optimization methods for the targeted problem. Section 1.4 introduces the content of the subsequent chapters.

## 1.1 Background and motivation

The meal kit delivery market in the United Kingdom has experienced rapid growth, driven by the increasing demand for convenient, healthy, and personalized meal solutions. With projected revenue of $1.4bn in 2024, competition has intensified as many new players enter the market alongside established companies like Hello Fresh, Gousto, Mindful Chef, etc. (Statista, 2024). The success of these companies relies heavily on efficient supply chain management to ensure timely delivery, ingredient freshness, and profitability.

A critical challenge for meal kit providers is the daily allocation of boxes (customer orders) to factories because inefficiencies in this process can lead to reduced customer satisfaction and increased costs from higher food waste. However, the box allocation problem (BAP) is complicated by various constraints, such as factory capacities and recipe eligibility, and requires continous optimization.

As there has been no research on optimizing BAP, this dissertation aims to bridge the gap by tackling this problem of a leading meal-kit company whose mission is to reduce food waste and promote sustainable practices. This company offers a diverse weekly menu catering to various dietary preferences and skill levels. Customers select their preferred meals and receive pre-measured ingredients with step-by-step recipe cards delivered to their doorstep, as illustrated in Figure 1.1


Figure 1.1. An exemplary box with ingredients and recipe cards (Woop, no date)

## 1.2 Research objectives and significance

The research question of this dissertation is: *Could exact and heuristic optimization methods be applied to minimize recipe differences between two allocation solutions?*

With this question in mind, the study aims to enhance the efficiency of box allocation process in meal kit delivery by advanced techniques. The specific objectives are:

- To develop a mathematical model that captures the key features of BAP, including constraints, objective, and temporal aspects. This model will serve as a foundation for designing and implementing optimization algorithms.
- To propose and implement both exact and heuristic methods across various experiments to identify the most efficient approach for industrial use.

This study is necessary for several reasons:

- By developing new approaches for BAP, this research aims to advance current practices in meal kit delivery.

- The proposed methods are expected to reduce costs by minimizing food waste and enhance the company's competitiveness in the market.
- The insights from this study may have broader applications, extending to other resource allocation problems.

## 1.3 Methodology

This research applies the following steps to address BAP:

1. Generate orders for two consecutive days, adhering to the characteristics of real-world data.
2. For exact method (Branch and Bound), CBC solver in PuLP library is used to directly find the optimal allocation for current day's orders, minimizing WMAPE site.
3. For heuristic methods, current orders are firstly allocated to factories using a greedy heuristic, and the initial WMAPE site is calculated. Then, Iterative targeted pairwise swap (ITPS) and Tabu search (TS) are applied to optimize the initial allocation through order swaps. The final solutions of all methods must satisfy both eligibility and factory capacity constraints.
4. The effectiveness of heuristics is evaluated by comparing the final WMAPE site with the initial WMAPE site to quantify the percentage of improvement. Additionally, the optimality of results provided by all methods is assessed by comparing WMAPE site against WMAPE global, which is the theoretical lower bound of WMAPE site.
5. Steps 1 to 4 are repeated with increasing order quantities to assess the scalability of all methods. This evaluation will consider both the optimality of results and computational efficiency. The performance of the three methods will be illustrated through comparative graphs, offering a comprehensive analysis.

Once the best method is found, it will be applied to optimize the full planning period in scenarios with and without capacity and order changes. This helps to evaluate the chosen method's effectiveness in high-frequency and varying decision-making processes.

Through the above steps, this dissertation aims to identify a practical approach to BAP, thereby contributing to the optimization of meal kit delivery operations.

## 1.4 Dissertation structure

The remainder of this dissertation is structured as follows:

- Chapter 2 conducts a comprehensive literature review, exploring related optimization problems and their solution techniques. It also provides an in-depth examination of the most promising and widely used optimization methods, establishing a strong foundation for the methodology chapter.
- Chapter 3 explains BAP in detail, including its characteristics, purpose, and mathematical formulation.

- Chapter 4 clarifies the methodology, focusing on how each algorithm assigns boxes to factories to minimize the WMAPE site.
- Chapter 5 details the experimental design, presents the results, and discusses the managerial implications thoroughly.
- Chapter 6 concludes the dissertation by summarizing key findings and contributions, acknowledging limitations, and outlining promising directions for future research.

# Chapter 2

# Literature Review

BAP in meal kit delivery shares similarities with some classic combinatorial optimization problems (COPs) while introducing unique constraints and objectives. This chapter will first explore the two most relevant problems which are Bin packing problem (BPP) and Capacitated vehicle routing problem (CVRP). The second section will examine three popular methods that show potential for solving BAP.

## 2.1 Related optimization problems

### 2.1.1 Bin packing problem (BPP)

Bin packing problem is a well-known optimization problem involving packing a set of items with varying sizes into the minimum number of bins, each with a fixed capacity. It can be formulated as follows (Johnson et al., 1974):

Given a set of $n$ items with weights $w_1, w_2, \ldots, w_n$ and a finite number of bins, each with a fixed capacity $C$, find a partition of items into a minimum number of bins such that the total weight of items in each bin does not exceed $C$.

$$\text{Minimize } \sum_{j=1}^{m} y_j$$

subject to:

$$\sum_{i=1}^{n} w_i x_{ij} \leq C y_j, \quad \forall j = 1, \ldots, m$$

$$\sum_{j=1}^{m} x_{ij} = 1, \quad \forall i = 1, \ldots, n$$

$$x_{ij} \in \{0, 1\}, \quad \forall i = 1, \ldots, n, \forall j = 1, \ldots, m$$

$$y_j \in \{0, 1\}, \quad \forall j = 1, \ldots, m$$

where $x_{ij}$ is a binary variable indicating whether item $i$ is assigned to bin $j$, and $y_j$ is a binary variable indicating whether bin $j$ is used.

BPP is closely related to BAP, as both involve assigning items (boxes) to bins (factories) while adhering to capacity constraints and minimizing an objective function. This similarity suggests that optimization methods used for BPP may also be applicable to BAP.

Research on BPP has progressed significantly over the years, addressing various aspects of this problem. Early exact methods focused on B&B. Martello and Toth (1990) developed reduction procedures and lower bounds for BPP, which influenced subsequent research. Scholl et al. (1997) introduced BISON, a hybrid algorithm combining B&B with TS, demonstrating the potential of integrating exact and metaheuristic methods. Valério de Carvalho (1999) proposed an arc flow formulation solved by column generation and B&B, which outperformed previous approaches by solving larger instances and effectively handling problems with many item sizes. Delorme et al. (2016) provided a comprehensive review and experimental evaluation of various exact and approximate methods for BPP, finding that branch-and-cut-and-price and pseudo-polynomial formulations, solved by advanced integer linear programming (ILP) solvers, perform best overall.

Regarding approximate algorithms, Johnson et al. (1974) provided foundational work on worst-case performance bounds for simple one-dimensional packing algorithms like First Fit (FF) and Best Fit (BF). Coffman et al. (1997) surveyed approximation algorithms for BPP, discussing extensions such as First Fit Decreasing (FFD) and Best Fit Decreasing (BFD), which achieve improved worst-case bounds.

Due to the computational challenges of exact methods for large instances, considerable focus has shifted to heuristics and metaheuristics. Alvim et al. (1999) proposed a local search (LS) procedure for BPP that progressively reduces the number of bins and redistributes items, achieving optimal solutions for most benchmark instances with faster times than other metaheuristics. Rao and Iyengar (1994) highlighted the potential of Simulated annealing (SA) for finding near-optimal solutions. Sonuc et al. (2017) combined First Fit Decreasing (FFD) with SA, showing this hybrid approach brings high-quality solutions. Tabu search (TS) has also proven effective. Lodi et al. (1999) developed a unified TS that excels across various BPP variants, employing advanced neighborhood structures and tabu mechanisms for efficient navigation. Alvim et al. (2004) developed a hybrid approach combining LB, initial solution generation, load redistribution, and TS for BPP, achieving state-of-the-art results and outperforming other methods on various classes of difficult problems. Additionally, Falkenauer (1996) demonstrated the effectiveness of a grouping Genetic algorithm (GA) specifically designed for BPP. Levine and Ducatelle (2004) proposed an Ant colony optimization (ACO) approach for BPP, demonstrating that pure ACO is competitive with evolutionary methods, and a hybrid ACO with LS outperforms the best evolutionary hybrids on some problems.

### 2.1.2 Capacitated vehicle routing problem (CVRP)

Capacitated vehicle routing problem is a fundamental optimization challenge in logistics and transportation. It is a variant of the Vehicle routing problem (VRP) introduced by Dantzig and Ramser in 1959. CVRP involves determining optimal delivery routes for a fleet of vehicles with limited capacity to serve customers with known demands. Its mathematical model is based on the formulation presented by Toth and Vigo (2002) and further discussed by Laporte (2009):

Let $G = (V, A)$ be a complete graph where $V = \{0, 1, \ldots, n\}$ is the vertex set and $A$ is the arc

set. Vertices $i = 1, \ldots, n$ correspond to the customers, while vertex 0 represents the depot.

**Parameters:**

- $c_{ij}$: travel cost from vertex $i$ to vertex $j$
- $d_i$: demand of customer $i$ (with $d_0 = 0$)
- $K$: number of vehicles
- $Q$: capacity of each vehicle

**Decision variables:**

- $x_{ij}^k = 1$ if vehicle $k$ travels directly from vertex $i$ to vertex $j$, 0 otherwise
- $y_i^k = 1$ if customer $i$ is served by vehicle $k$, 0 otherwise

**Objective function:** Minimize the total travel cost

$$\min \sum_{k=1}^{K} \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} x_{ij}^k$$

**Constraints:**

1. Each customer is visited exactly once:

$$\sum_{k=1}^{K} \sum_{j=0}^{n} x_{ij}^k = 1, \quad \forall i = 1, \ldots, n$$

2. Each route starts and ends at the depot:

$$\sum_{j=1}^{n} x_{0j}^k = 1, \quad \forall k = 1, \ldots, K$$

$$\sum_{i=1}^{n} x_{i0}^k = 1, \quad \forall k = 1, \ldots, K$$

3. Flow conservation for each customer:

$$\sum_{i=0}^{n} x_{ij}^k - \sum_{i=0}^{n} x_{ji}^k = 0, \quad \forall j = 0, \ldots, n, \forall k = 1, \ldots, K$$

4. Capacity constraints:

$$\sum_{i=1}^{n} d_i y_i^k \leq Q, \quad \forall k = 1, \ldots, K$$

5. Linking constraints between $x$ and $y$ variables:

$$y_i^k \leq \sum_{j=0}^{n} x_{ij}^k, \quad \forall i = 1, \ldots, n, \forall k = 1, \ldots, K$$

6. Binary constraints:

$$x_{ij}^k \in \{0, 1\}, \quad \forall i, j = 0, \ldots, n, \forall k = 1, \ldots, K$$

$$y_i^k \in \{0, 1\}, \quad \forall i = 1, \ldots, n, \forall k = 1, \ldots, K$$

CVRP shares several important similarities with BAP:

- Both problems involve managing limited capacity. In CVRP, vehicles have a maximum load they can carry. In BAP, factories have a maximum number of boxes they can produce.
- In CVRP, each vehicle's tour exemplified by Figure 2.1 can be compared to the allocation of boxes to a factory in BAP. The process of improving solutions in CVRP by swapping customers between tours is analogous to swapping boxes between factories in BAP.

Due to the above similarities, solution techniques that are effective for CVRP may also apply to BAP.

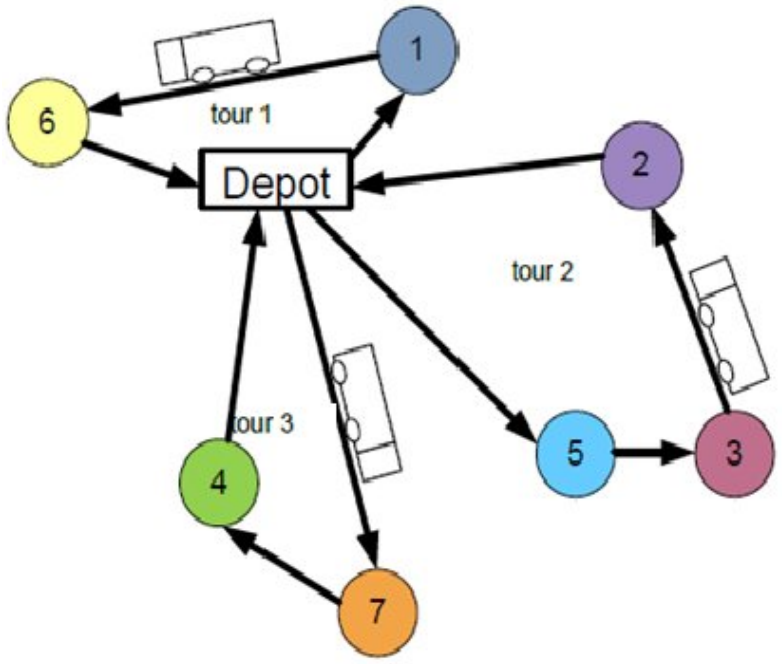



Figure 2.1. Solution of CVRP (Adiba et al., 2013)

Researchers have long developed both exact and heuristic methods to address CVRP due to its significance and diverse applications. Exact methods guarantee optimal solutions but can be computationally intensive for large instances. Laporte and Nobert (1987) pioneered one of the earliest

B&B algorithms for VRP. The set partitioning formulation, introduced by Balinski and Quandt (1964), has been the basis for many exact algorithms. Specifically, Fukasawa et al. (2006) introduced a branch-and-cut-and-price algorithm, combining column generation with cutting planes, which achieved optimal solutions for nearly all instances. This technique was further refined by Contardo and Martinelli (2014) and Pecin et al. (2017). Costa et al. (2019) enhanced it with advanced cutting planes and pricing algorithms, solving several previously unsolved benchmark instances.

Given the computational challenges of exact methods, heuristics and metaheuristics have been widely studied. Local search, particularly k-opt moves, has played a crucial role in solving CVRP. The concept of k-opt originated from Lin (1965) and his introduction of the 2-opt move for the Traveling salesman problem (TSP), which was later adapted for VRP. Toth and Vigo (2003) proposed a granular TS using 2-opt and 3-opt moves, efficiently solving large-scale CVRP instances. Metaheuristics have shown particular promise in tackling CVRP, with several approaches demonstrating excellent results. TS has been a standout method, with Gendreau et al. (1994) developing an influential adaptive memory procedure, later unified for several VRP variants by Cordeau et al. (1997). Osman (1993) provided a comparative study of TS and SA, showing that TS consistently outperformed SA in solution quality and robustness. GAs have also shown considerable success, as demonstrated by Baker and Ayechew (2003). Another notable metaheuristic is ACO, adapted for VRP by Bullnheimer et al. (1999) and further refined by Yu et al. (2009). Recently, Large neighborhood search (LNS), first introduced by Shaw (1998), has gained popularity because of its outstanding performance across various VRP variants, further enhanced by the adaptive LNS presented by Pisinger and Ropke (2007).

## 2.2 Potential optimization methods

Based on the optimization requirements of BAP and methods mentioned in the previous section, the three most potential approaches including one exact method (Branch and X), one improvement heuristics (Local search), and one metaheuristic (Tabu search) in Figure 2.2 are selected for further examination.

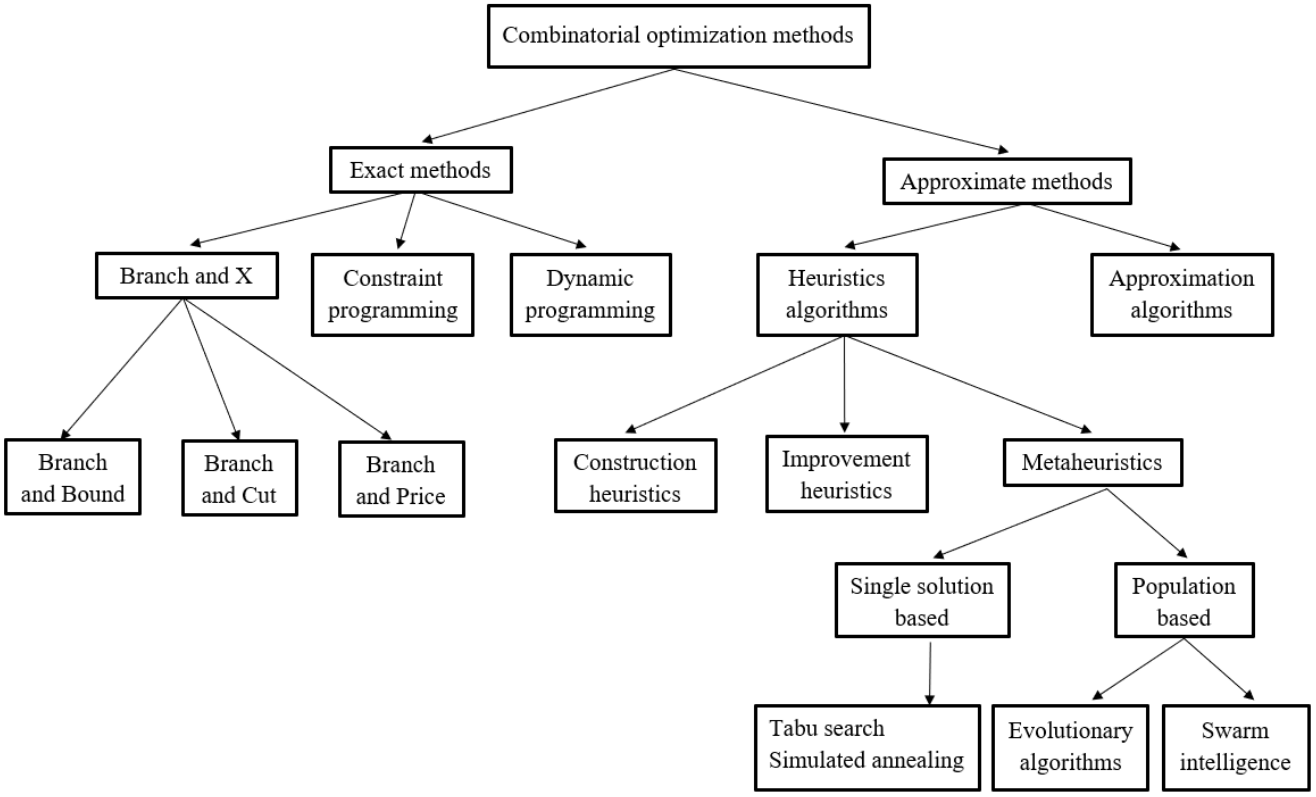



Figure 2.2. Taxonomy of combinatorial optimization methods (Tabli, 2009)

### 2.2.1 Branch and X

Branch and X algorithms, a family of exact methods, are fundamental in operations research. The most prominent method in this group is B&B, introduced by Land and Doig (1960), which forms the basis for many modern exact algorithms. B&B explores the solution space by constructing a search tree, where each node represents a partial solution. B&B employs two key strategies: branching and bounding. Branching involves dividing the problem into subproblems while bounding involves computing upper bound (UB) and lower bound (LB) for each subproblem to prune the search tree and eliminate suboptimal solutions (Clausen, 1999).

The B&B algorithm can be explained using the enumeration tree (upper tree in Figure 2.3). Each node represents a variable, and each branch corresponds to a variable's value. The tree displays four variables and all possible branches, with the bottom nodes representing all potential solutions. With four variables, there are 16 possible solutions, though some may be infeasible due to constraints, and others may be suboptimal. This structure enables B&B to prune many nodes and branches, resulting in a simplified version as the lower tree in Figure 2.3.

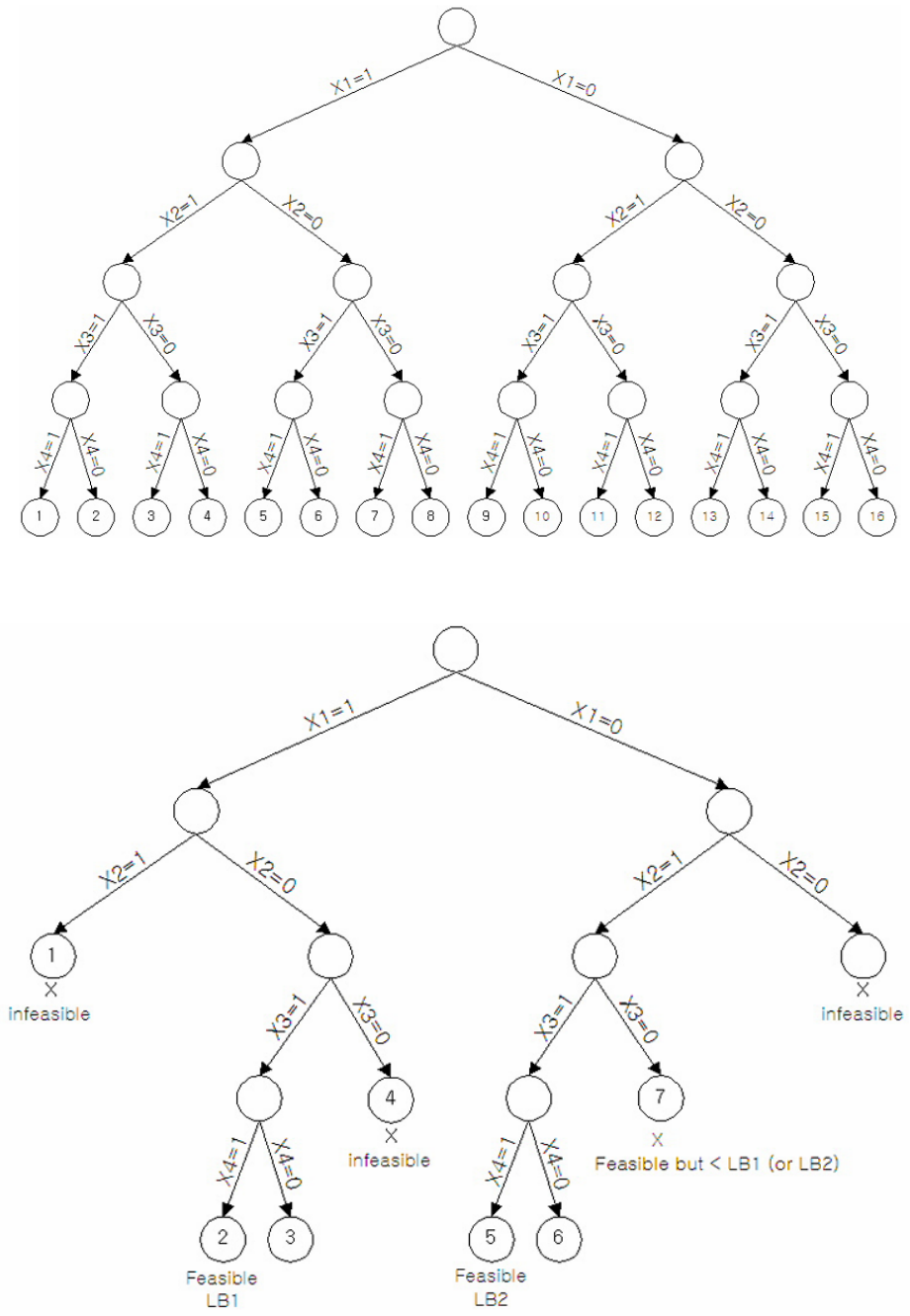



Figure 2.3. Enumeration and B&B trees (Rosenberger et al., 2005)

B&B, with its strong ability to reduce the search space, has been successfully applied to various problems. Regarding logistics, B&B has been used to solve TSP (Little et al., 1963) and VRP (Laporte et al., 1986), optimizing routes for salesman and delivery fleets respectively. It also tackled complex job shop scheduling problems (Lageweg et al., 1977), helping to maximize efficiency in production lines. The algorithm has proven effective for resource allocation challenges like the knapsack problem (Kolesar, 1967) and quadratic assignment problem (Hahn and Grant, 1998). B&B continues evolving, with researchers refining branching rules and bounding techniques to improve performance across various problems (Achterberg et al., 2005). Its ability to provide provably optimal solutions, combined with its adaptability, ensures B&B remains a prominent

method for combinatorial optimization.

Branch and Cut (B&C), an extension of B&B, incorporates cutting plane methods to tighten linear programming relaxations. Marchand et al. (2002) comprehensively reviewed cutting plane techniques in integer and mixed integer programming (MIP), highlighting their effectiveness in solving large-scale problems. The advantage of B&C lies in its ability to strengthen the formulation during the solution process, often leading to faster convergence and the ability to solve larger instances than pure B&B.

Branch and Price (B&P) is another variant that combines B&B with column generation. This approach is particularly successful in solving large-scale integer programming problems with complex structures. The key advantage of B&P is its ability to handle problems with a very large number of variables, where it would be impractical to enumerate all variables explicitly (Barnhart et al., 1998).

Several commercial and open-source solvers implementing these Branch and X algorithms have been developed, successfully addressing a wide range of real-world problems. Some of the most widely used include:

- CPLEX (Computational Programming LEXicon): A commercial solver of IBM that implements B&B and B&C for MIP problems.
- Gurobi: A commercial solver of Gurobi Optimization known for its high performance in solving MIP problems using B&B and B&C.
- CBC (COIN-OR Branch and Cut): An open-source MIP solver developed by Forrest et al. (2005) that uses B&B and B&C.
- SCIP (Solving Constraint Integer Programs): An open-source solver developed by Achterberg et al. (2009) that implements B&B, B&C, and B&P.

### 2.2.2 Local search (LS)

Local search is a fundamental approach in combinatorial optimization, operating by iteratively moving from one solution to another in the search space through local changes (Aarts and Lenstra, 2003). The effectiveness of LS has led to its widespread use and the development of various extensions. A prominent example of LS is the $k$-opt heuristic, particularly 2-opt. The 2-opt move, introduced by Croes (1958), involves swapping two edges in a tour for TSP. This concept was later generalized to $k$-opt moves by Lin (1965), leading to the powerful Lin-Kernighan heuristic (Lin and Kernighan, 1973). The $k$-opt heuristic, with its fundamental operation of swapping $k$ edges or elements, has proven to be a versatile and powerful tool in many COPs:

TSP is the original and most prominent application of $k$-opt. In TSP, this method operates by swapping $k$ edges in a tour to create a new, potentially shorter route. The continuing relevance of $k$-opt in TSP is evidenced by its crucial role in state-of-the-art algorithms. Helsgaun's work in 2000 shows how sophisticated implementations of $k$-opt can lead to remarkable improvements in solution quality, even for large-scale TSP instances. The success of $k$-opt in TSP has led to its adaptation for more complex routing scenarios, particularly VRP and its variants. Savelsbergh

(1992) pioneered this extension by introducing operators like 2-opt* for inter-route improvements in VRP. These operators effectively swap segments between different routes, optimizing not just individual routes but the overall routing solution. Building on this, Prins (2004) integrated 2-opt and other local search moves with GA for CVRP. This approach outperformed most published metaheuristics on classical benchmark instances.

The applications of $k$-opt extend beyond routing problems. Kernighan and Lin (1970) adapted k-opt principles for graph partitioning by developing a heuristic that sequentially identifies pairs of nodes to exchange between partitions, allowing temporary cost increases to escape local optima. Their algorithm builds larger node sets for swapping, quickly evaluates them with a gain function, and implements the best exchange for each pass, significantly surpassing previous methods. Potts and Van Wassenhove (1985) applied an adjacent job interchange rule to the single-machine total weighted tardiness problem, swapping the two most recently added jobs in a partial sequence. Though not a strict $k$-opt move, this approach effectively leveraged LS principles and served as a powerful pruning tool in their B&B algorithm.

The diverse applications highlight $k$-opt's effectiveness: by systematically exchanging elements such as edges, nodes, or other problem-specific entities, it adeptly navigates complex solution spaces and optimizes objectives.

### 2.2.3 Tabu search (TS)

Tabu search is a metaheuristic method that has significantly advanced the field of combinatorial optimization. Introduced by Glover (1986), TS builds upon LS techniques by incorporating adaptive memory structures to guide the search process more effectively. The core idea of TS is to overcome the limitations of simple LS by allowing moves to inferior solutions while avoiding cycling by using a tabu list. Moreover, Glover and Laguna (1997) provided a comprehensive framework for TS, detailing key components such as short-term and long-term memory structures, aspiration criteria, and strategies for intensification and diversification. These elements enable TS to balance between exploiting promising areas of the search space and exploring new regions, making it particularly effective for complex COPs.

The versatility of TS has led to its success across various problems. Gendreau et al. (1994) developed an influential adaptive memory procedure based on TS for VRP. This approach showed the power of combining TS with problem-specific knowledge, achieving high-quality solutions for various VRP instances. Golden et al. (1998) demonstrated that the most effective metaheuristic for VRP is TS, significantly outperforming other approaches such as SA, GA, and ACO. Barnes and Laguna (1993) applied TS to the weighted flowtime problem on parallel machines, using swap and insert moves with a tabu restriction on recently moved jobs. Their method improved efficiency by forbidding non-improving swaps after an initial period, focusing on insert moves when no improvements were found. This approach significantly outperformed existing B&B. Moreover, Hanafi and Freville (1998) developed a TS algorithm for the 0-1 multidimensional knapsack problem that combines strategic oscillation with surrogate constraints. Their method uses flexible memory structures to oscillate between feasible and infeasible regions, balancing intensification and diversification. This approach matched or improved the best-known solutions for large benchmarks, showing the effectiveness of combining oscillation with surrogate information in TS.

One of the key strengths of TS is its ability to integrate with other techniques, particularly LS. Nowicki and Smutnicki (1996) developed a highly effective TS for the job shop scheduling problem, using a specialized neighborhood structure based on critical path analysis. Their approach leveraged problem-specific knowledge to restrict the search to only those moves likely to improve the makespan, significantly reducing the neighborhood size and computational complexity. Díaz and Fernández (2001) developed a simple yet effective TS for the generalized assignment problem, integrating LS with strategic oscillation. Their approach, using adaptive penalty adjustments, produced high-quality solutions competitive with complex methods for small to medium-sized instances.

## 2.3 Summary

BAP in meal kit delivery closely relates to some classic problems including BPP and CVRP. The similarities between them indicate that optimization techniques developed for BPP and CVRP could be effective for our targeted problem. This literature review has explored various solving techniques, realizing the potential of three methods: B&B, LS, and TS for BAP. While there are other popular metaheuristics such as SA, GA, and LNS, the characteristics of BAP, particularly its strict recipe eligibility and full capacity constraints, suggest a cautious approach that involves small changes and focused search may be more suitable.

Regarding the advantages of three potential methods:

- B&B is an exact method that can provide optimal solutions, which is crucial in real business and serves as a benchmark for heuristic methods. Its widespread implementation in various solvers enhances its accessibility and ease of application.
- LS, particularly 2-opt, is highly effective for iteratively refining solutions. When applied to BAP, 2-opt moves can strategically swap boxes between factories, enabling efficient exploration of the solution space and constraint satisfaction.
- TS is a powerful metaheuristic for navigating complex solution spaces because of its balanced exploration and exploitation. This makes it well-suited for the intricacies of BAP. Additionally, integrating TS with LS can enhance solution quality by leveraging TS's global exploration ability alongside LS's targeted improvement strength.

# Chapter 3

# Box Allocation Problem in Meal Kit Delivery

## 3.1 Problem explanation

BAP in meal kit delivery involves assigning boxes (orders) with structure as Figure 3.1 to different factories, as illustrated in Figure 3.2. The goal is to minimize the recipe differences between two allocation decisions. For instance, if Factory 1 (F1) was assigned 100 units of Recipe 1 on the previous day, the company will aim to allocate the current day's orders so that F1 continues to produce 100 units of Recipe 1. They do this simultaneously for all recipe-factory combinations while respecting both capacity and eligibility constraints.

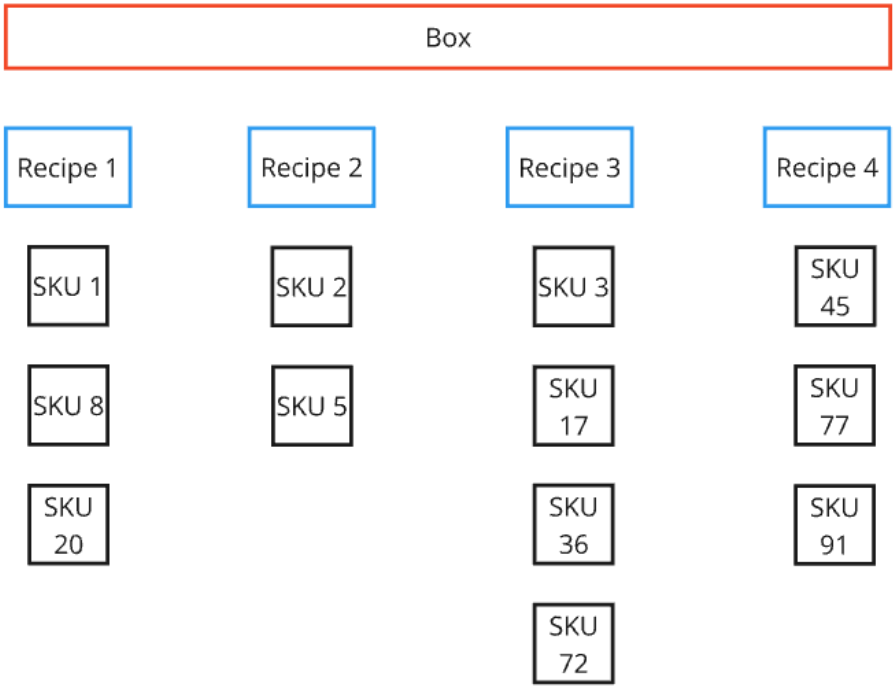



Figure 3.1. Box structure (SKU is ingredient)


Figure 3.2. Allocation process

Regarding the first constraint, each factory has a daily capacity, defined by the number of boxes it can produce. To ensure maximum resource utilization, the solution must fully meet this capacity as shown in Figure 3.3.

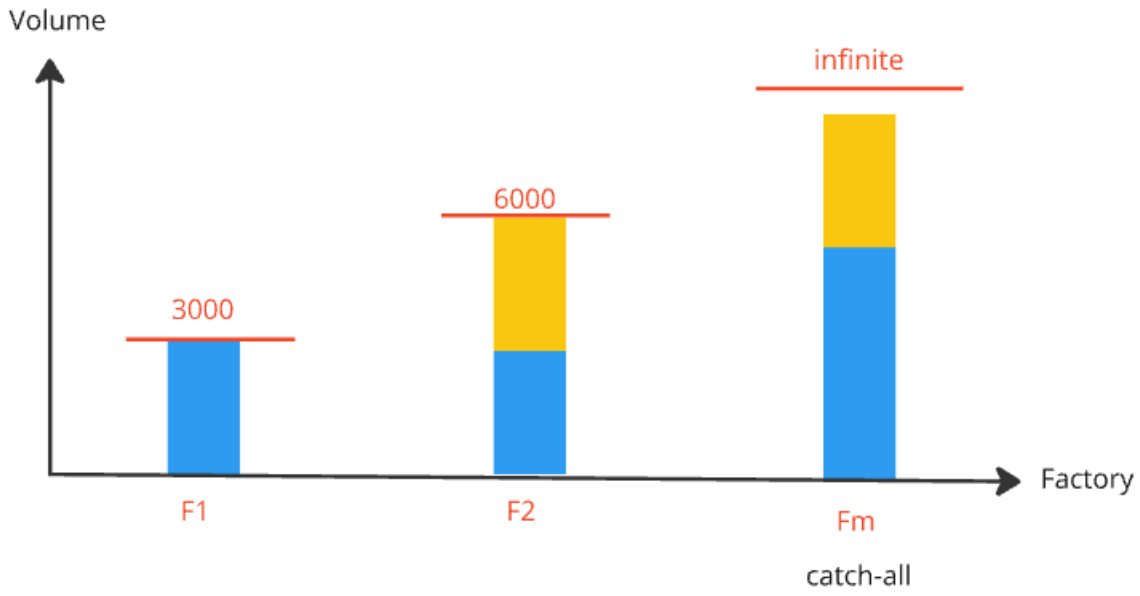



Figure 3.3. Feasible solution regarding the capacity constraint

Additionally, because not all factories are equipped to produce every recipe, the allocation decision must account for the eligibility constraint explained in Figure 3.4, ensuring that each box is only assigned to the factory capable of processing all of its recipes.

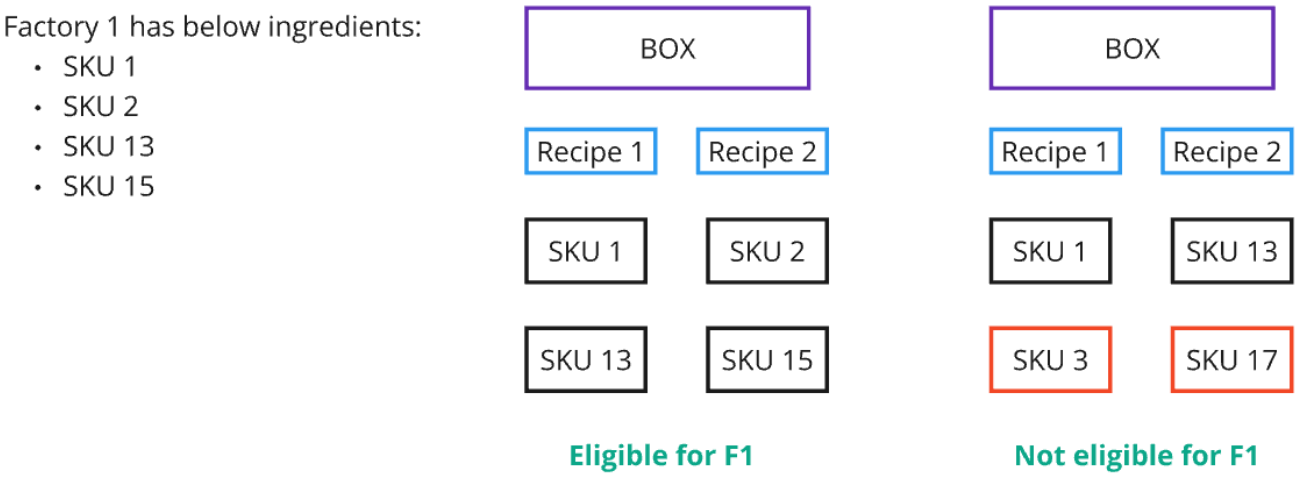



Figure 3.4. Eligibility constraint

Another key feature of BAP is its temporal nature, necessitating the daily, or even twice-daily, allocation of both simulated and real orders, as demonstrated in Figure 3.5. In this dissertation, a 15-day planning period is assumed, starting from lead day 18 (LD18) and ending on LD3.

The planning process is as follows:

- Initially, the company forecasts the total number of boxes expected to be ordered by customers for the upcoming period.
- Throughout the planning period, company monitors actual orders against the projection. When real orders are lower than the total forecasted quantity, the ordering platform is used to generate simulated orders to represent the anticipated future demand. The company then performs *soft allocation*, distributing both real and simulated orders across factories.
- As time progresses, the proportion of real orders in the total quantity increases, gradually replacing the simulated orders. Importantly, the allocation for each day includes all real orders from the previous day.
- On the final day (LD3), the allocation consists entirely of real orders. This stage, known as *hard allocation*, is fixed and no longer subject to changes. Then, company is assumed to have three days to produce the boxes and deliver them to customers.

This approach enables the company to maintain consistent daily allocations while accommodating the dynamic nature of incoming orders throughout the planning period. It facilitates a structured transition from forecast-based planning to actual order fulfillment.

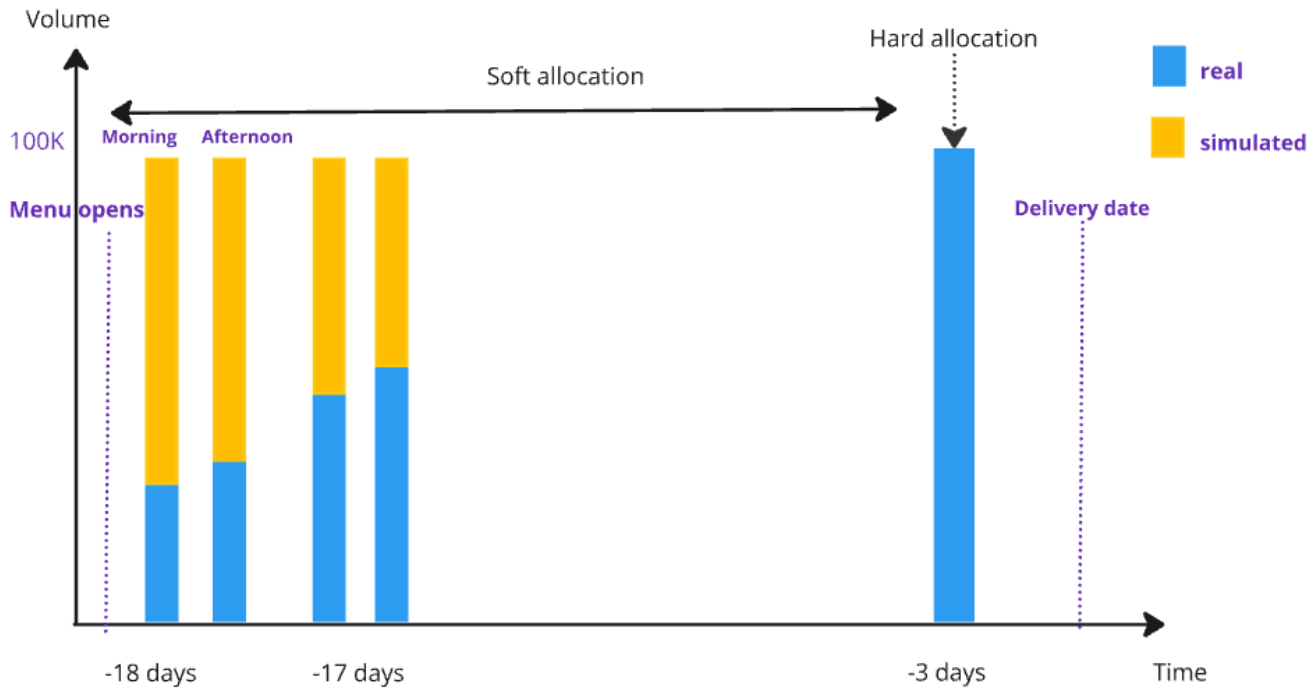



Figure 3.5. Temporal aspect of BAP

## 3.2 Mathematical formulation

Weighted mean absolute percentage error (WMAPE) is an advanced forecast accuracy measure used in business contexts like supply chain and production planning. It calculates errors using weights, effectively preventing infinite error by ensuring the sum of actual values in the denominator is never zero. Specifically, WMAPE sums absolute differences between actual and forecasted values, divided by the sum of actual values (GeeksforGeeks, 2021). A lower WMAPE indicates better accuracy. For instance, a WMAPE of 10% means forecasts deviate from actual values by an average of 10%.

Before delving into the mathematical model, it is crucial to distinguish between WMAPE site and WMAPE global as they serve different purposes:

- WMAPE site:
  - Considers each factory (site) separately
  - Sensitive to changes in allocation between factories
  - Useful for analyzing the stability of production plans at each factory
- WMAPE global:
  - Aggregates data across all factories
  - Not sensitive to changes in allocation between factories
  - Provides a broader view of overall recipe distribution changes

For the above characteristics, this dissertation focuses on minimizing WMAPE site which offers a detailed perspective that allows planners to assess the consistency of production at factory level.

The following notations are used in the formulation of BAP:

- $t$: Allocation day
- $N = \{1, 2, \ldots, n\}$: Set of recipes, indexed by $i$
- $S = \{1, 2, \ldots, m\}$: Set of factories, indexed by $j$
- $O = \{1, 2, \ldots, k\}$: Set of orders, indexed by $o$
- $a_{i,j,t}$: Number of times recipe $i$ is allocated to factory $j$ at day $t$
- $a_{i,j,t-1}$: Number of times recipe $i$ is allocated to factory $j$ at day $t-1$
- $q_{i,t}$: The quantity of recipe $i$ at day $t$
- $q_{i,t-1}$: The quantity of recipe $i$ at day $t-1$
- $x_{o,j,t}$: Binary decision variable indicating the allocation of order $o$ to factory $j$ at day $t$
- $C_{j,t}$: Capacity of factory $j$ at day $t$
- $E_{i,j,t}$: Binary parameter indicating the eligibility of recipe $i$ at factory $j$ at day $t$

BAP can be formulated as a mixed-integer linear programming (MILP) problem:

$$\text{Minimize } f(x) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} |a_{i,j,t} - a_{i,j,t-1}|}{\sum_{i=1}^{n} q_{i,t}} \tag{3.1}$$

subject to

$$\sum_{o=1}^{k} x_{o,j,t} = C_{j,t}, \quad \forall j \neq m, \forall t \tag{3.2}$$

$$x_{o,j,t} \leq \min_{i \in o} E_{i,j,t}, \quad \forall o, \forall j, \forall t \tag{3.3}$$

$$\sum_{j=1}^{m} x_{o,j,t} = 1, \quad \forall o, \forall t \tag{3.4}$$

$$x_{o,j,t} \in \{0, 1\}, \quad \forall o, \forall j, \forall t \tag{3.5}$$

The objective function (3.1) represents WMAPE site. It aims to minimize the differences in recipe counts between all consecutive days, weighted by the total number of all recipes on the current day. Specifically:

- The numerator adds up the absolute differences in the number of each recipe $i$ allocated to factory $j$ between the current day and the previous day.
- The denominator represents the total quantity of all recipes on the current day $t$.

While the whole WMAPE site is used to evaluate performance, the focus is primarily on its numerator because this part is influenced by how orders are allocated, while the denominator remains constant each day.

Constraint (3.2) ensures that the total number of orders allocated to each factory (except the catch-all factory $F_m$) matches the full capacity of that factory.

Constraint (3.3) ensures that an order $o$ can only be allocated to factory $j$ if all recipes in that order are eligible for factory $j$.

Constraint (3.4) ensures that each order $o$ is allocated to exactly one factory.

Constraints (3.5) specifies that allocation decisions are binary, meaning an order $o$ is either allocated to factory $j$ or not.

Table 3.1 illustrates how WMAPE site is calculated.

Table 3.1. WMAPE site calculation

| Recipe | Factory | Day -15 | Day -14 | Absolute site-recipe difference |
|---|---|---|---|---|
| 1 | F1 | 16 | 16 | 0 |
| 1 | F2 | 0 | 0 | 0 |
| 1 | F3 | 5 | 2 | 3 |
| 2 | F1 | 15 | 15 | 0 |
| 2 | F2 | 4 | 5 | 1 |
| 3 | F1 | 3 | 3 | 0 |
| 3 | F2 | 3 | 3 | 0 |
| 3 | F3 | 2 | 3 | 1 |
| 4 | F1 | 2 | 2 | 0 |
| 4 | F2 | 2 | 2 | 0 |
| 4 | F3 | 3 | 2 | 1 |
| 5 | F1 | 0 | 0 | 0 |
| 5 | F2 | 2 | 2 | 0 |
| 5 | F3 | 3 | 3 | 0 |
| 6 | F1 | 0 | 0 | 0 |
| 6 | F2 | 30 | 27 | 3 |
| 6 | F3 | 8 | 7 | 1 |
| 7 | F1 | 0 | 0 | 0 |
| 7 | F2 | 23 | 31 | 8 |
| 7 | F3 | 5 | 5 | 0 |
| 8 | F1 | 0 | 0 | 0 |
| 8 | F2 | 23 | 30 | 7 |
| 8 | F3 | 4 | 9 | 5 |
| 9 | F1 | 0 | 0 | 0 |
| 9 | F2 | 28 | 32 | 4 |
| 9 | F3 | 11 | 5 | 6 |
| 10 | F1 | 0 | 0 | 0 |
| 10 | F2 | 0 | 0 | 0 |
| 10 | F3 | 23 | 24 | 1 |
| **Sum** | | | **232** | **41** |
| **WMAPE site** | | | **41 / 232 = 0.177** | |

WMAPE global serves as a benchmark for evaluating the optimality of allocation solutions. This is achieved by comparing WMAPE site of solution with WMAPE global, which is calculated using Equation (3.8).

$$WMAPE_{\text{global}} = \frac{\sum_{i=1}^{n} |a_{i,t} - a_{i,t-1}|}{\sum_{i=1}^{n} q_{i,t}} \tag{3.8}$$

The following proofs explain why WMAPE global is the LB of WMAPE site:

Decomposing at the site level, equation (3.8) becomes:

$$WMAPE_{\text{global}} = \frac{\sum_{i=1}^{n} \left| \sum_{j=1}^{m} a_{i,j,t} - \sum_{j=1}^{m} a_{i,j,t-1} \right|}{\sum_{i=1}^{n} q_{i,t}} \tag{3.9}$$

Using the triangle inequality, we get:

$$WMAPE_{\text{global}} \leq \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} |a_{i,j,t} - a_{i,j,t-1}|}{\sum_{i=1}^{n} q_{i,t}} \tag{3.10}$$

which simplifies to:

$$WMAPE_{\text{global}} \leq WMAPE_{\text{site}} \tag{3.11}$$

To understand the transition from equation (3.9) to (3.10), consider the following analysis:

The triangle inequality states that for any real numbers $x$ and $y$:

$$|x + y| \leq |x| + |y|$$

Applying to this context, the decomposition of recipe difference at site level results in:

$$\left| \sum_{j=1}^{m} (a_{i,j,t} - a_{i,j,t-1}) \right| \leq \sum_{j=1}^{m} |a_{i,j,t} - a_{i,j,t-1}|$$

Summing these absolute differences across all recipes $i$, we get:

$$\sum_{i=1}^{n} \left| \sum_{j=1}^{m} (a_{i,j,t} - a_{i,j,t-1}) \right| \leq \sum_{i=1}^{n} \sum_{j=1}^{m} |a_{i,j,t} - a_{i,j,t-1}|$$

This inequality indicates that the absolute value of the sum of recipe differences across all sites is always less than or equal to the sum of absolute differences calculated individually at each site.

Dividing both sides by the total quantity of all recipes at day $t$ results in:

$$\frac{\sum_{i=1}^{n} \left| \sum_{j=1}^{m} (a_{i,j,t} - a_{i,j,t-1}) \right|}{\sum_{i=1}^{n} q_{i,t}} \leq \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} |a_{i,j,t} - a_{i,j,t-1}|}{\sum_{i=1}^{n} q_{i,t}}$$

This leads to:

$$WMAPE_{\text{global}} \leq WMAPE_{\text{site}}$$

Table 3.2 illustrates how WMAPE global is calculated. It can be seen that using the same order data, WMAPE global (0.142) is lower than WMAPE site in Table 3.1 (0.177).

Table 3.2. WMAPE global calculation

| **Recipe** | **Day -15** | **Day -14** | **Absolute recipe difference** |
|---|---|---|---|
| 1 | 21 | 18 | 3 |
| 2 | 19 | 20 | 1 |
| 3 | 8 | 9 | 1 |
| 4 | 7 | 6 | 1 |
| 5 | 9 | 9 | 0 |
| 6 | 38 | 34 | 4 |
| 7 | 28 | 36 | 8 |
| 8 | 27 | 39 | 12 |
| 9 | 39 | 37 | 2 |
| 10 | 23 | 24 | 1 |
| **Sum** | | **232** | **33** |
| **WMAPE global** | | **33 / 232 = 0.142** | |

**Proxy optimization**: The objective function described in (3.1) serves as an intermediate means to achieve the company's ultimate goal of minimizing area under the curve in Figure 3.6, which represents WMAPE site calculated based on the recipe differences between the hard allocation on LD3 and the soft allocations on previous days.

However, directly achieving this goal is not feasible due to uncontrollable forecast errors that affect allocation decisions before LD3. These errors can arise from various factors, including the randomness of simulated orders, modifications made by customers to existing real orders, and the arrival of new real orders.

To address this challenge, the company adopts a more manageable goal: minimizing errors between two consecutive days, as shown in the above mathematical model. This approach turns BAP into a *proxy optimization problem*, where a related, more tractable problem serves as a stand-in for achieving the ultimate goal (Zangl et al., 2006). In this case, minimizing the daily error acts as a proxy for minimizing the whole period error, making the optimization process more feasible while still working towards the broader objective of maintaining operational stability.

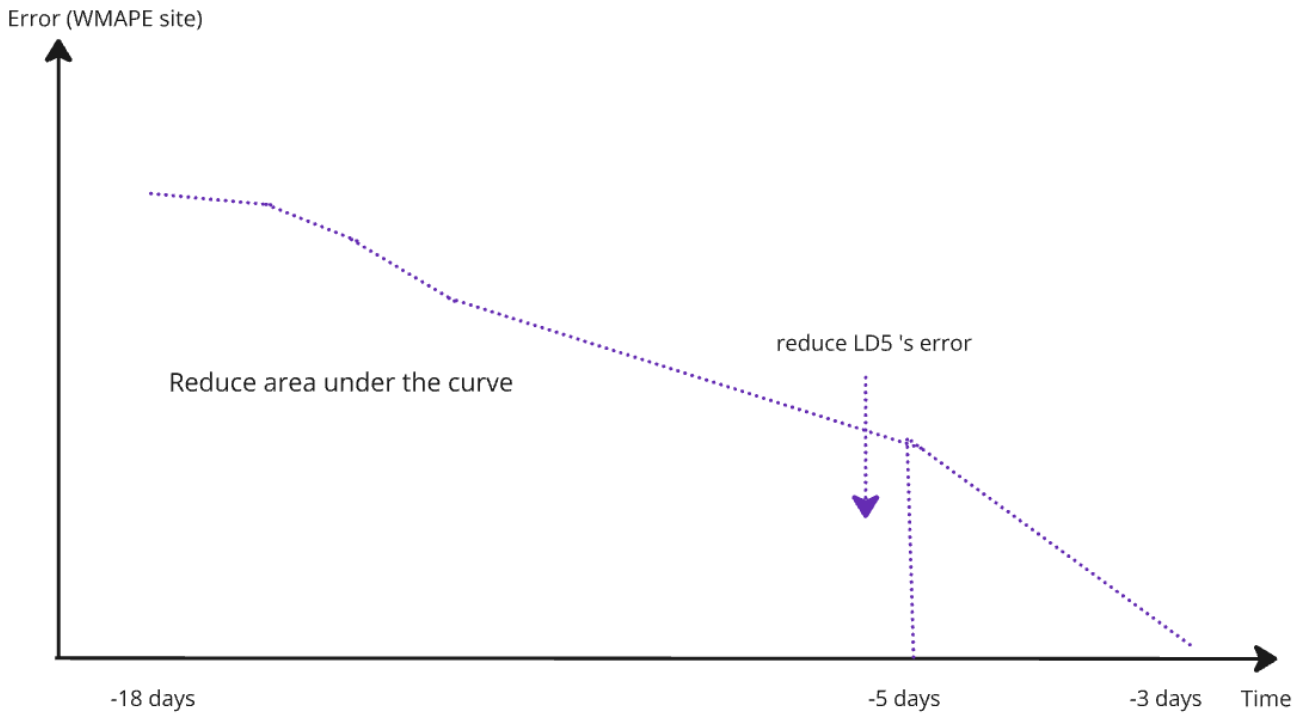



Figure 3.6. Objective of proxy optimization

# Chapter 4

# Methodology

This chapter covers three methods to solve BAP. It first describes the overall strategy, then explains B&B in CBC solver, and finally details two heuristics: Iterative targeted pairwise swap (ITPS) and Tabu search (TS).

## 4.1 Solving strategy

The company seeks an effective method to minimize recipe disparities between two allocation solutions, possibly through swapping boxes between factories. In this study, the 1:1 box swap in Figure 4.1 is used for heuristics because larger moves, such as 1:0 with 1:2 swaps, or swapping multiple orders at once, risk creating ineligible moves or failing to meet full factory capacity. To prevent constraint violations, a thorough feasibility check is required for complex swaps, which can significantly increase the time required for the optimization process. Thus, the 1:1 swap is preferred for its simplicity and efficiency in maintaining constraints.

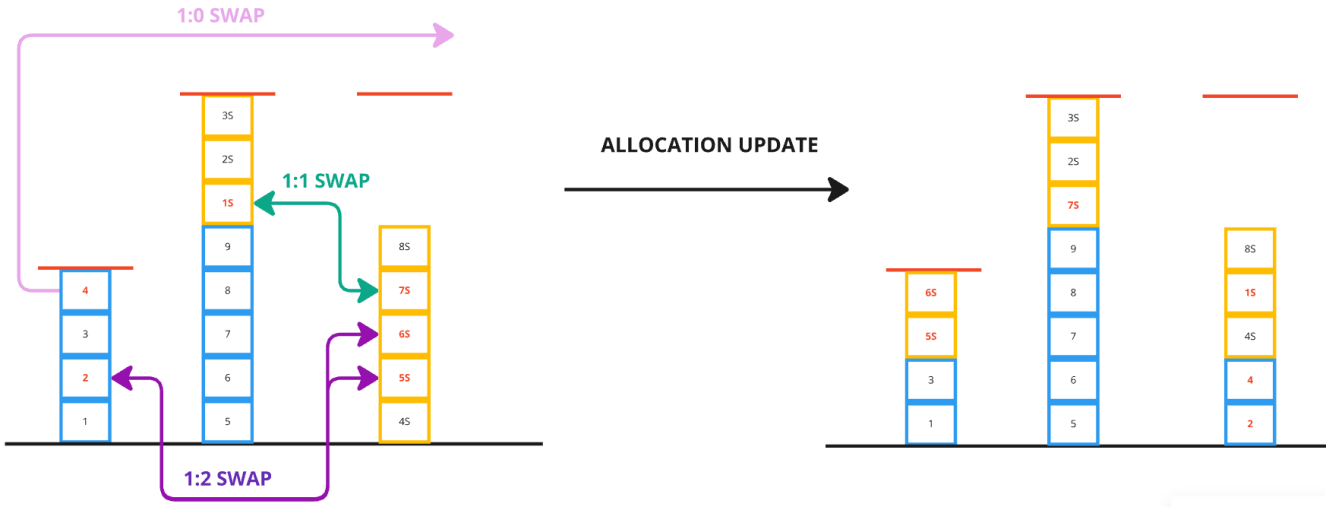



Figure 4.1. Exemplary types of box swaps

For a better understanding of the solving process, it is important to note there are four groups of factory eligibility for orders: F1-F3, F1-F2-F3, F2-F3, and F3 only.

The solving strategy is as follows:

- Regarding exact method: apply B&B directly on current orders to determine their optimal allocation that minimizes recipe difference with the previous day's allocation.
- Regarding heuristics, the process begins by constructing an initial feasible solution using a greedy algorithm, similar to the First Fit in BPP:
  - Orders are first allocated to F1, prioritizing those orders with fewer eligible factories (F1-F3 over F1-F2-F3) until fulfilling F1's capacity.
  - Allocate the remaining orders with eligible factories F1-F2-F3 and F2-F3 to F2 until its capacity is reached.
  - All unallocated orders are assigned to F3, a catch-all factory with no capacity or eligibility constraints.

  Then apply ITPS and TS to improve the initial solution.
- Evaluate the performance and scalability of each method by comparing the final WMAPE site against WMAPE global and comparing optimization times across different quantities. This analysis will help to identify the most effective method.
- After identifying the best approach, incorporate the temporal component of BAP to evaluate how well the chosen method optimizes allocation decisions over a 15-day planning period.

## 4.2 Branch and bound (B&B)

Before employing heuristics that potentially provide suboptimal solutions, B&B is applied to assess how well it can solve BAP. Due to its ability to deliver exact results, B&B has been implemented in many commercial and open-source solvers, such as CPLEX, Gurobi, CBC, and SCIP. Among various available solvers, CBC is chosen for this dissertation because it integrates seamlessly with PuLP library in Python, and is effective in solving complex MILP problems.

### 4.2.1 CBC solver

CBC (COIN-OR Branch and Cut) is an open-source MIP solver, developed as part of the ongoing Computational Infrastructure for Operations Research (COIN-OR) project, led by John Forrest. CBC is primarily designed as a callable library for developing customized B&C solvers. It integrates seamlessly with COIN-OR's linear programming (CLP) solver and the cut generation library (CGL) (Forrest and Lougee-Heimer, 2005). The user guide for this solver is available here.

Below is the general solving process of CBC:

1. Initial bound calculation: Start by relaxing the integrality constraints on the integer variables, treating them as continuous variables within their respective bounds (e.g., [0, 1] for binary variables). Solve this relaxed LP model using the CLP solver. The solution to this LP provides a LB on the MIP's objective function. If the solution to the LP has all integer values for the originally integer-constrained variables, this solution is optimal, and the process terminates.

2. Branch on non-integral variables: If any variables in the LP solution have non-integral values, select one of these variables for branching. Create two new subproblems (nodes) in the search tree: one where the selected variable is forced to its LB (e.g., 0) and another where it is forced to its upper bound (UB) (e.g., 1).

3. Heuristic application: Before or during the B&B process, heuristics like Feasibility Pump (FP) are applied to find MIP-feasible solutions quickly. These solutions provide UBs for the objective function, which can significantly improve the efficiency of the B&B process by pruning large portions of the search tree. DiveCoefficient and other rounding heuristics are used to further refine the solutions by diving into specific promising areas of the solution space and rounding fractional values in a way that often leads to feasible, near-optimal solutions.

4. Node selection and re-optimization: Select a node from the search tree to process. The strategy for choosing nodes can vary, including methods such as depth-first, breadth-first, or best-bound. For the selected node, solve the LP relaxation again, considering the additional constraints imposed by the branching decisions.

5. Prune the search tree by evaluating the node: If the LP relaxation is infeasible, prune the node. If the optimal value of the LP relaxation exceeds the current best UB, prune the node. If the solution is feasible for the MIP and provides a better objective value, update the best-known solution and UB. If LB equals UB, prune the node by optimality as it can no longer lead to a better solution.

6. Further branching and continued search: If the node cannot be pruned, select another non-integral variable to branch on and create two new subproblems (nodes). Add these new nodes to the search tree for further exploration. In some cases, CBC performs a mini B&B within a node to explore small, localized changes that could lead to improvements, particularly after heuristics like FP have provided a feasible solution.

7. Cut generation: At various stages, particularly at the root node, CBC applies cutting planes, such as Gomory cuts and Knapsack cuts to tighten the LP relaxation and exclude infeasible regions without excluding any feasible integer solutions. These cuts can improve the bounds and further prune the search tree.

8. Termination: The process continues until the search tree is exhausted or the optimal solution is found, with no further nodes to explore that can bring a better solution. The optimal solution is confirmed when the best-known feasible solution's objective matches the LB.

In reality, the full power of CBC solver is not required to solve BAP. Take order quantity of 100,000 (the company's highest demand) for example. The process is as follows:

1. Problem initialization and preprocessing: The model was loaded and initialized with 210,302 rows, 300,600 columns, and 1,359,124 elements. Preprocessing was applied, which fixed 110,000 variables and reduced the problem size. The continuous relaxation was solved, yielding an objective value of 4,646.

2. Heuristic application: Employ FP heuristic to quickly find initial integer-feasible solutions. Then apply the DiveCoefficient heuristic, which successfully found the optimal solution. For most order quantities under 100,000, FP alone can find the optimal solution immediately.

3. Resolving general integer variables: A limited B&B process was used to clear up the remaining general integer variables that were not fixed by the heuristics. This process focused on a reduced problem size, branching only on the unfixed general integer variables.

4. A mini B&B was conducted to enhance the solution. Although this process ultimately did not lead to an improved solution, it plays a crucial role in validating the quality of the solution obtained through heuristics.

5. Termination: The process concluded without further extensive branching, as the optimal solution was proven at the root node after the limited branching and mini B&B processes.

The description above details the full process for solving large-scale BAP. Although steps 3 and 4 may not be necessary for smaller instances, they become increasingly important as the problem's size and complexity grow.

Another key consideration is that CBC requires model to be linear. To ensure BAP's mathematical model aligns with the solver's requirements, the following analysis is conducted:

- The decision variable, which indicates whether an order is allocated to a factory, is binary, thus ensuring linearity.
- All constraints are linear combinations of the decision variable, satisfying the requirement.
- For the objective function, the model built in Python aims to minimize the numerator of WMAPE site (sum of recipe absolute differences) because the denominator (total quantity of all recipes) is constant within each day.

Although the sum of absolute differences may seem non-linear, the following technique is used to linearize the absolute value function, ensuring the whole model remains linear:

1. For each recipe-factory combination, two non-negative variables, $positive\_difference$ and $negative\_difference$, are introduced. These variables are constrained such that their values equal the actual difference between the counts for current and previous days. The lower bounds of both variables are set to 0.

2. The $positive\_difference$ and $negative\_difference$ are then summed across all recipe-factory combinations. This approach ensures BAP is a MILP problem suitable for CBC solver.

### 4.2.2 Heuristics in CBC

Because heuristics contribute greatly to the effectiveness of CBC, two heuristics used to solve BAP will be explained in more detail.

**Feasibility Pump**, introduced by Fischetti et al. (2005), is designed to quickly find feasible solutions for MIP problems. Its algorithm works as follows:

1. Start with the solution $x^*$ of the LP relaxation.

2. Round $x^*$ to the nearest integer feasible point $x'$.

3. Find the LP solution $x^*$ closest to $x'$ in terms of L1-norm.

4. Repeat steps 2 and 3 until a feasible solution is found or a stopping criterion is met.

The algorithm alternates between finding integer-feasible but possibly LP-infeasible solutions ($x'$) and LP-feasible but possibly integer-infeasible solutions ($x^*$). This *pumping* action often quickly leads to solutions that are both integer and LP feasible.

In the CBC output for order quantity of 100,000, multiple passes of FP progressively enhanced the solution quality. FP concluded with an objective value of 4,654, offering a strong initial feasible solution.

**Coefficient Diving** or DiveCoefficient in CBC is a diving heuristic introduced by Berthold (2006) and further enhanced by Paulus and Krause (2023). It works as follows:

1. It selects a fractional variable $x_j$ to bound (fix to an integer value) in each iteration by:

    - Examining the number of *locks* for each fractional variable. The *locks* indicate how many constraints could be violated if the variable is rounded in a certain direction.
    - Finding the variable(s) with the minimal positive number of locks.
    - If there are multiple variables with the same minimal number of locks, choosing the one with the smallest fractionality $f(\bar{x_j})$, where $f(\bar{x_j}) = |\bar{x_j} - \lfloor \bar{x_j} + 0.5 \rfloor|$.

2. The selected variable is then bounded (fixed) in the direction that achieves the minimal number of locks. This process is repeated, bounding one variable per iteration, until all integer variables are fixed or infeasibility is reached.

The goal is to make rounding decisions that are least likely to violate constraints (by choosing variables with few locks) and that are closest to integer values already (by using fractionality as a tie-breaker). This helps to find feasible integer solutions quickly.

In the CBC output for order quantity of 100,000, DiveCoefficient was particularly effective and found the optimal solution of 4,646 quickly, without requiring any B&B node exploration.

Overall, the FP and DiveCoefficient heuristics rapidly find high-quality feasible solutions by efficiently exploring different parts of the solution space, providing effective UBs, and reducing the need for extensive B&B exploration. Their complementary approaches, with FP offering broad exploration and DiveCoefficient providing focused descent, increase the likelihood of discovering optimal or near-optimal solutions early in the process. When combined with preprocessing and tight LP relaxation, these heuristics enable solving BAP to optimality without extensive tree exploration, significantly reducing computational effort.

## 4.3 Heuristic methods

To address large-scale instances of BAP, heuristics, and metaheuristics are considered, as these methods can deliver good solutions within a reasonable time. The following subsections provide a detailed explanation of two heuristics specifically developed for BAP.

### 4.3.1 Iterative targeted pairwise swap (ITPS)

Iterative targeted pairwise swap is inspired by local search heuristics, particularly the 2-opt algorithm commonly used for TSP and VRP. Like 2-opt, ITPS focuses on pairwise exchanges to improve the solution's quality. While 2-opt swaps adjacent elements in a tour, ITPS swaps two orders between two factories to reduce WMAPE site.

ITPS has some key characteristics:

- It is greedy, only accepting improvements and immediately reverting non-improvements.
- The process is highly targeted, using recipe differences to select the swap candidates (orders).
- The algorithm ensures the feasibility of the solution (respecting capacity constraint) by only swapping two orders each time.

Figure 4.2 presents the pseudo-code of ITPS.

Figure 4.3 visualizes the solving process with important steps as follows:

1. The process starts with the previous day's order allocation and the current day's greedy order allocation. Based on these inputs, the algorithm calculates the initial WMAPE site, which helps to evaluate how well the current allocation matches the target allocation.
2. **Main loop** (continues until reaching the maximum number of iterations):
   (a) For each factory, the algorithm checks how current recipe counts differ from the recipe counts of the previous day. This helps to identify which recipes have too high or too low quantity in each factory compared to the target.
   (b) Randomly shuffle factories to ensure an unbiased exploration of different factory combinations throughout the process.
   (c) **Factory pair loop:**
      i. For each pair of factories, the algorithm seeks orders that, if exchanged, might improve the recipe balance in both factories. This search considers the recipe differences identified earlier and ensures that any potential swap respects order eligibility for each factory.
      ii. If suitable orders for swapping are found, the algorithm exchanges these two orders between two factories.
      iii. After each swap, the algorithm recalculates WMAPE site. It then compares this new WMAPE to the previous best WMAPE to determine if the swap has led to an improvement.
      iv. If the swap improves WMAPE site, the algorithm updates its record of the best allocation. If the swap does not improve WMAPE site, the algorithm returns the orders to their original factories.
      v. The algorithm then considers the next pair of factories. This continues until all pairs have been examined.
3. The main loop ends when the maximum number of iterations is reached, and the algorithm exports the best allocation with its WMAPE site.

Overall, the iterative nature of ITPS, combined with its targeted yet randomized approach, allows for a thorough search of the solution space while maintaining focus on the most promising areas for improvement. This balance makes ITPS particularly well-suited for the complex nature of BAP.

```
target_allocation = allocation_t_minus_1    # Allocation of previous day's orders (Input)
current_allocation = allocation_t           # Greedy allocation of current day's orders (Input)
best_allocation = current_allocation
best_wmape = calculate_WMAPE(current_allocation, target_allocation)

While not reached max_iterations do
    recipe_diff = calculate_recipe_differences(current_allocation, target_allocation)
    factories = randomly_shuffle(factories)
    For each source_factory in factories do
        For each target_factory in factories do
            If source_factory ≠ target_factory then
                Find swap_candidates based on recipe_diff
                If swap_candidates is found then
                    Perform_swap(swap_candidates)
                    new_wmape = calculate_WMAPE(current_allocation, target_allocation)
                    If new_wmape < best_wmape then
                        best_wmape = new_wmape
                        best_allocation = current_allocation
                    Else
                        Revert_swap(swap_candidates)
                    End If
                End If
            End If
        End For
    End For
End While
Return best_allocation, best_wmape      # Output
```

Figure 4.2. Pseudo-code of ITPS

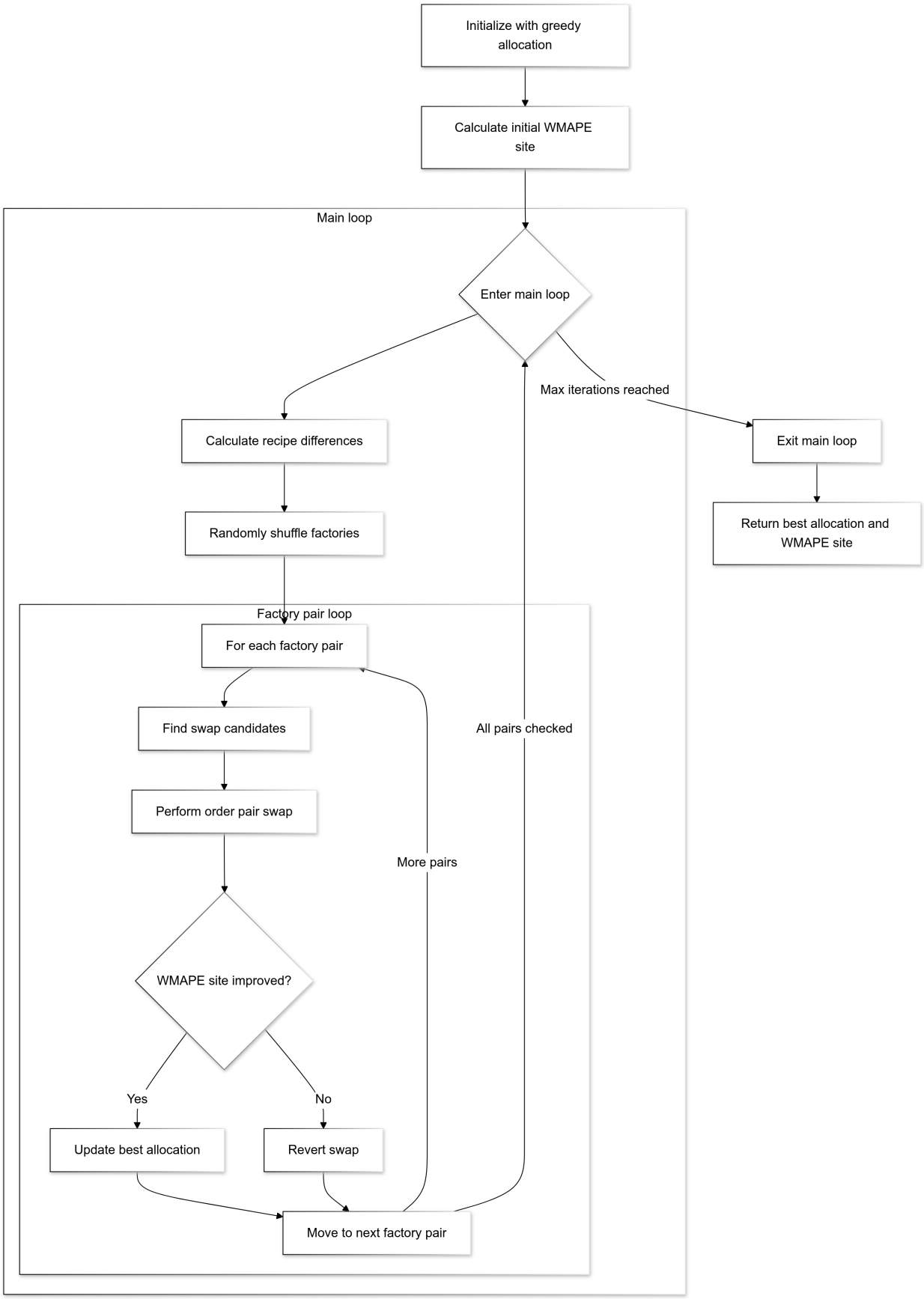



Figure 4.3. Solving process of ITPS

### 4.3.2 Tabu search (TS)

Tabu search is a metaheuristic inspired by adaptive memory techniques used in LS. It is designed to overcome the limitations of basic LS methods, which often get trapped in local optima (Gendreau and Potvin, 2010). While TS focuses on iterative improvements through small changes to the solution, it distinguishes itself from LS by incorporating a memory structure, known as tabu list, to guide the search process more intelligently (Glover and Laguna, 1997).

The term *tabu* originates from Polynesian cultures, where it refers to prohibited things (Radcliffe-Brown, 1939). In TS, tabu moves are those that are temporarily forbidden to prevent cycling and encourage the exploration of new areas in the solution space (Glover, 1989).

A key concept in TS is tenure, which refers to the duration for which a move remains in the tabu list (Gendreau, 2003). The tabu tenure determines how long a recently performed move is forbidden, avoiding short-term cycling and allowing the search to visit the previously explored areas after sufficient exploration elsewhere (Battiti and Tecchiolli, 1994).

Figure 4.4 presents the pseudo-code of TS. Due to the high computational cost of TS, the full WMAPE site is not used to evaluate performance. Instead, only the numerator of WMAPE site, which represents the absolute difference in recipes between two days, is considered. At the end of the process, the lowest absolute recipe difference will be divided by the total recipes of the current day (constant each day) to determine the best WMAPE site.

Figure 4.5 visualizes the solving process with explanation for each step as follows:

1. The process starts with the previous day's order allocation and the current day's greedy order allocation. Based on these inputs, the algorithm calculates the recipe counts for both allocations and determines the initial difference between them.
2. The algorithm then enters its main loop, which continues until reaching the maximum number of iterations:
   (a) Instead of evaluating all orders, the algorithm randomly selects a subset, such as 100 out of 10,000, to assess. This approach strikes a balance between thorough exploration and computational efficiency.
   (b) For each selected order, the algorithm considers swapping it with orders in other factories. It does not evaluate all possible swaps, but rather a random subset from each factory. This process involves:
      i. Checking if the swap is feasible (if the order can be produced in the new factory).
      ii. Calculating how the swap would impact the overall difference in recipe counts.
      iii. Determining if the swap is allowed based on the tabu list or if it meets the criteria to override its status (a tabu move can still be accepted if it leads to a solution better than the best known so far).
   (c) After evaluating potential moves, the algorithm selects the best one – typically the move that most reduces the difference in recipe counts. If a good move is found, it:
      i. Executes the swap by updating the current allocation and recipe counts.
      ii. Updates the best-known solution if the current solution is better.

iii. Adds the move to the tabu list.

(d) The algorithm dynamically manages the tabu list by removing moves whose tabu tenure has expired.

(e) Every 100 iterations, the algorithm performs a random swap. This helps to explore different areas of the solution space and avoid getting stuck in local optima.

3. After completing all iterations, the algorithm calculates the final WMAPE site and reports it along with the best allocation.

Overall, TS combines the benefits of LS with some mechanisms to avoid local optima. The tabu list helps to escape local optima by forcing it to explore new areas of the solution space. At the same time, the aspiration criteria allow promising moves even if they are tabu. The balance between intensification (through the best move selection) and diversification (through the periodic random swaps) provides a thorough exploration of the solution space.

```
target_allocation = allocation_t_minus_1                    # Allocation of previous day's orders (Input)
current_allocation = allocation_t                           # Greedy allocation of current day's orders (Input)
best_allocation = current_allocation
current_total_abs_diff = calculate_total_abs_diff(current_allocation, target_allocation)
best_total_abs_diff = current_total_abs_diff
tabu_list = {}                                              # Initialize empty tabu list
tabu_tenure = 20                                            # Set the number of iterations a move remains in the tabu list

While not reached max_iterations do
    best_move = None
    best_move_diff = 0

    orders_to_consider = randomly_select_subset_of_orders(current_allocation)

    For each order1 in orders_to_consider do
        factory1 = find_current_factory(order1)
        For each factory2 in factories do
            If factory1 ≠ factory2 then
                For each order2 in randomly_select_subset_of_orders(factory2) do
                    If is_swap_eligible(order1, factory2) and is_swap_eligible(order2, factory1) then
                        move = (order1.id, factory1, order2.id, factory2)
                        move_impact = calculate_move_impact(order1, factory1, order2, factory2)
                        If move not in tabu_list or satisfies_aspiration_criteria(move, current_total_abs_diff, best_total_abs_diff) then
                            If move_impact < best_move_diff then
                                best_move = (order1, factory1, order2, factory2)
                                best_move_diff = move_impact
                    End If
                End If
            End For
        End If
    End For
End For
```

```
30:   If best_move is found then
31:       Perform_swap(best_move)
32:       current_total_abs_diff += best_move_diff

33:       If current_total_abs_diff < best_total_abs_diff then
34:           best_total_abs_diff = current_total_abs_diff
35:           best_allocation = current_allocation.copy()

36:       Add_to_tabu_list(best_move, current_iteration, tabu_tenure)

37:   Remove_expired_tabu_moves(tabu_list, current_iteration)

38:   If current_iteration % 100 == 0 then
39:       Perform_diversification_move()

40: End While

41: best_wmape_site = calculate_wmape(best_allocation, target_allocation)
42: Return best_allocation, best_wmape_site                    # Output
```

Figure 4.4. Pseudo-code of TS

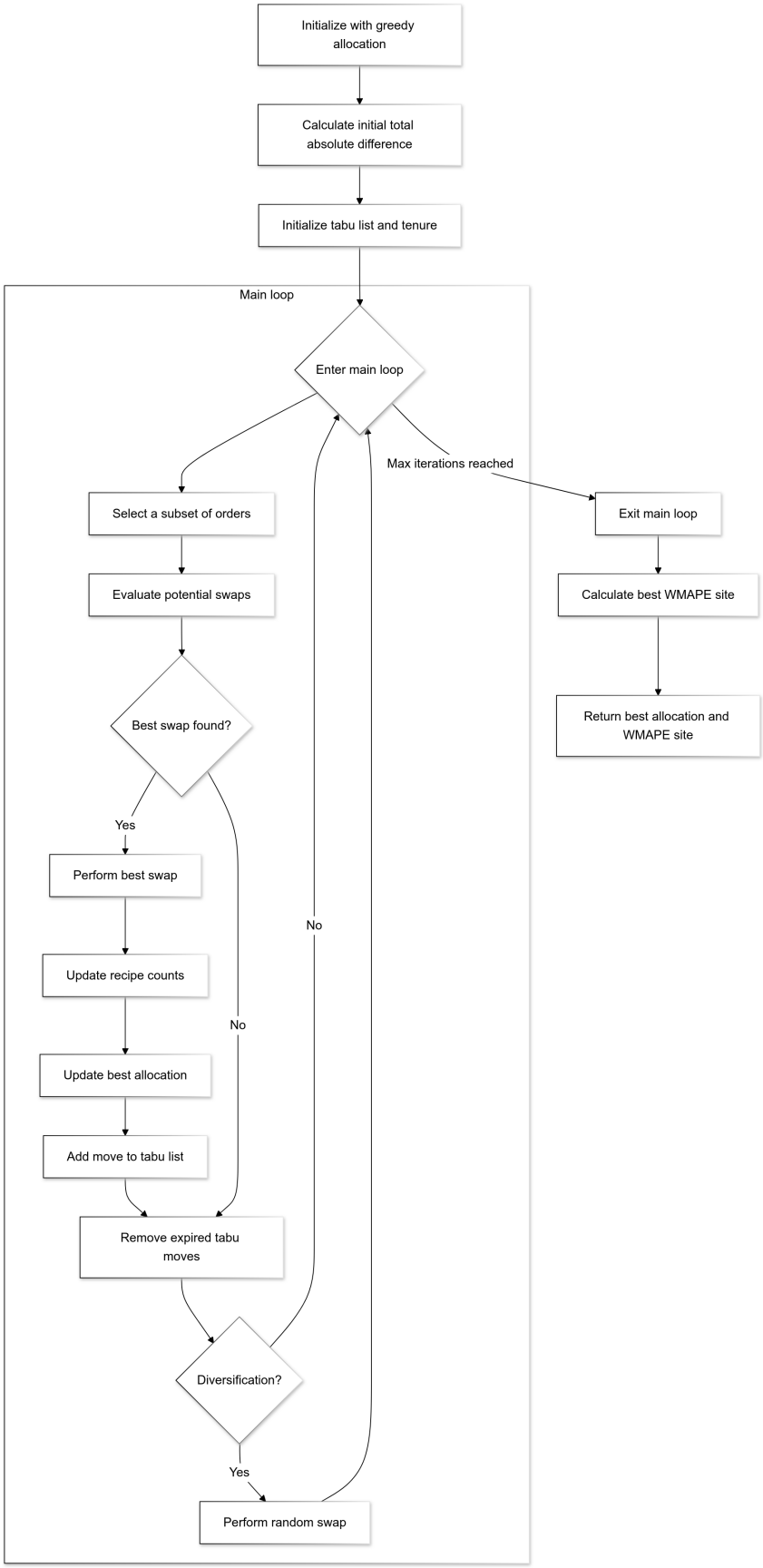



Figure 4.5. Solving process of TS

# Chapter 5

# Results and Discussion

This chapter outlines the experimental design and presents the computational results of four tests. The initial two tests compare the performance of B&B, ITPS, and TS to identify the most effective method. The superior algorithm is then applied to the temporal tests. This chapter concludes by discussing managerial implications and offering recommendations based on findings.

## 5.1 Experimental design

### 5.1.1 Data structure

In this dissertation, simulated data that matches the characteristics of real data is used to test the effectiveness of the proposed methods. Figure 5.1 illustrates how data is generated with the explanation as follows:

1. Set the targeted proportions for each type: F1-F3, F2-F3, F1-F2-F3, and F3 only in the total quantity. Here, F1 and F2 represent real factories, while F3 is a simulated factory that captures all orders not assigned to the real factories.

2. Check if any pre-existing real orders need to be included in the final list. If real orders exist, they are added to the order list, keeping the same IDs.

3. Main loop (create new orders to meet the total quantity):

   - After each new order is generated, check if the total number of orders has been reached. If more orders are needed, determine which type of order to create next, consider the targets set for each type, and select the appropriate category: F1-F3, F2-F3, F1-F2-F3, or F3 only.
   - Select the appropriate recipes and assign eligible factories for each new order based on its type. Then, assign an ID to the order and add it to the order list.

   This loop continues until the total number of orders is reached.

4. Once enough orders are generated, the algorithm adjusts the balance between real and simulated orders to ensure the final list matches the desired ratio.

5. The order list is shuffled to randomly distribute real and simulated orders before exporting the complete list.

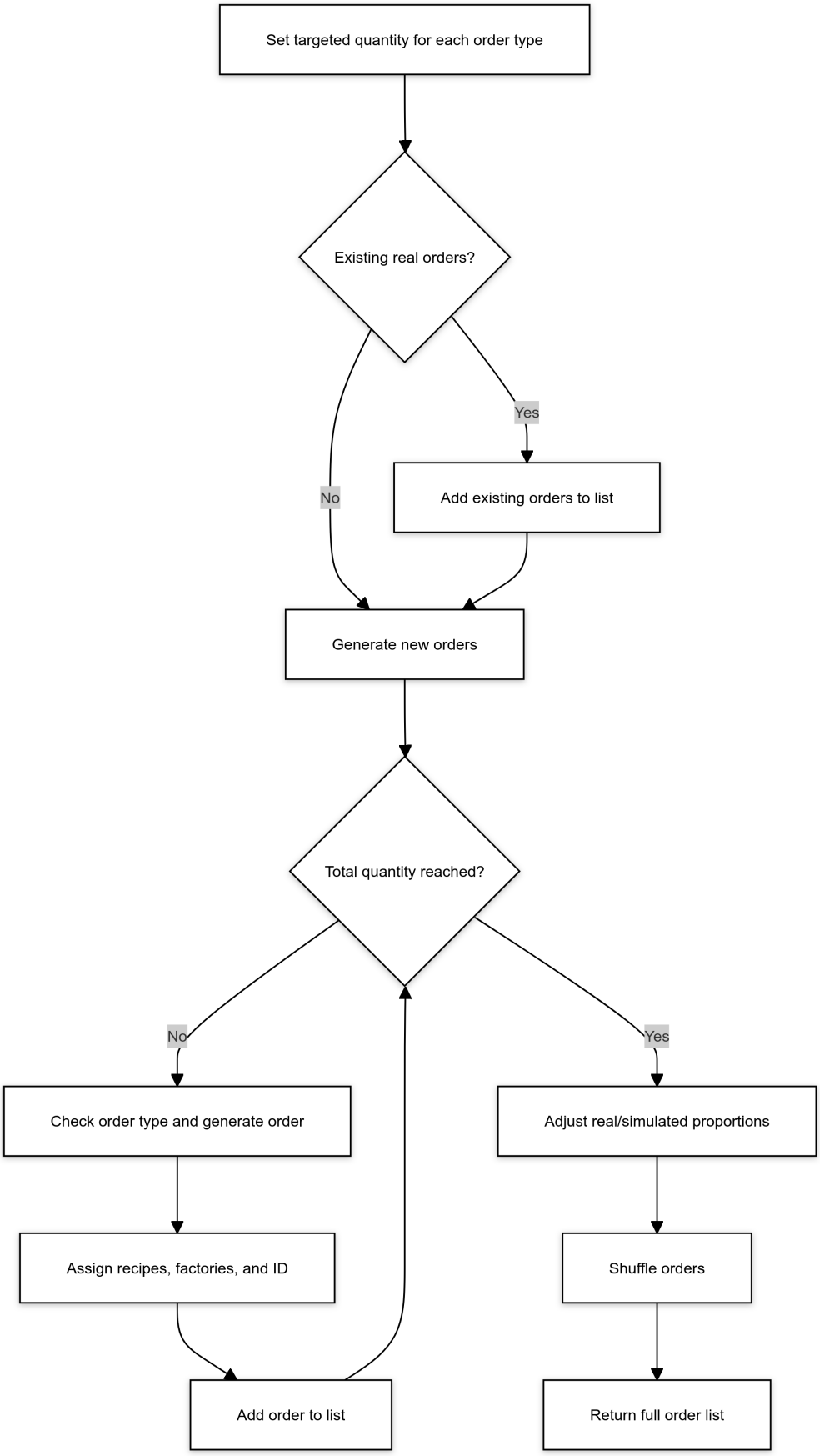



Figure 5.1. Order generation process

The above-generation way helps to ensure the created order list meets many rules in reality:

- Typically, 30% of the total orders are eligible for F1, and 60% are eligible for F2. All orders are eligible for F3, as this factory has no constraints. Figure 5.2 illustrates the distribution of order types in a sample of 100 orders.
- As the proportion of real orders in the total quantity increases over time, all real orders from the previous day are carried over to the current day's order list with their IDs unchanged, facilitating allocation tracking.
- Simulated orders are not moved to the next day. Instead, they are generated anew daily to compensate for any shortfall in the total quantity, so their IDs are not fixed.
- Each order has from 1 to 4 recipes, chosen from 100 available recipes with below eligibility:
  - Group 1: recipes from 1 to 29 are eligible for only F1.
  - Group 2: recipes from 30 to 49 are eligible for both F1 and F2.
  - Group 3: recipes from 50 to 89 are eligible for only F2.
  - Group 4: recipes from 90 to 100 are eligible for only F3.
  - An order is eligible for a factory only if all the recipes in it are eligible for that factory. However, for order type *F3 only*, any recipe can be included, as long as the order contains at least one recipe from group 4.
- The real and simulated orders are mixed in the company's database. Table 5.1 presents exemplary data, where IDs from the original list have been sorted for clarity. The upper two tables show the order details, while the lower two tables present the solution that satisfies the capacity constraints of F1 (25% of the total quantity) and F2 (50% of the total quantity). It can be seen that the allocated factories for four existing real orders remain unchanged, ensuring the minimum WMAPE site.

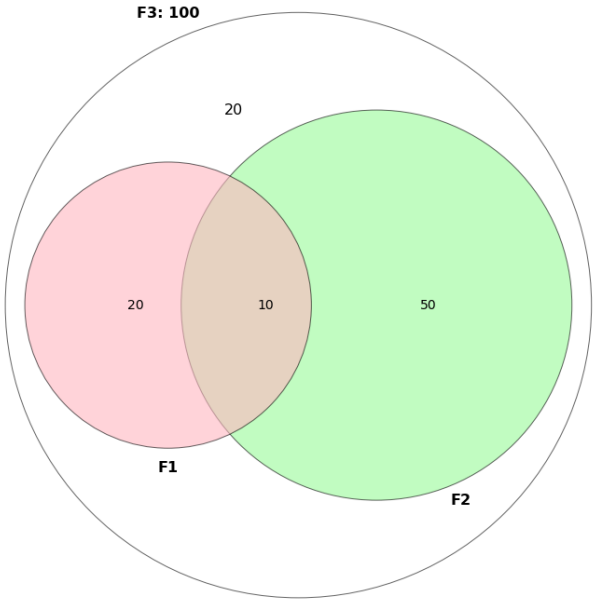



Figure 5.2. Order distribution among factories

Table 5.1. Exemplary order data and solution

Day -12 *(46% real orders)*

| Order ID | Recipe IDs | Is real | Eligible factories |
|---|---|---|---|
| 1 | 30 | TRUE | F1, F2, F3 |
| 2 | 8, 5, 24 | TRUE | F1, F3 |
| 3 | 22 | TRUE | F1, F3 |
| 4 | 87 | TRUE | F2, F3 |
| 5 | 52, 51, 55, 63 | FALSE | F2, F3 |
| 6 | 82, 88 | FALSE | F2, F3 |
| 7 | 85 | FALSE | F2, F3 |
| 8 | 84, 76 | FALSE | F2, F3 |
| 9 | 93, 1, 36, 76 | FALSE | F3 |
| 10 | 28, 95, 20 | FALSE | F3 |

| Factory | Allocated orders |
|---|---|
| F1 | 2, 3 |
| F2 | 1, 4, 5, 6, 8 |
| F3 | 7, 9, 10 |

Day -11 *(52% real orders)*

| Order ID | Recipe IDs | Is real | Eligible factories |
|---|---|---|---|
| 1 | 30 | TRUE | F1, F2, F3 |
| 2 | 8, 5, 24 | TRUE | F1, F3 |
| 3 | 22 | TRUE | F1, F3 |
| 4 | 87 | TRUE | F2, F3 |
| 5 | 74 | TRUE | F2, F3 |
| 6 | 85 | FALSE | F2, F3 |
| 7 | 89, 73, 86 | FALSE | F2, F3 |
| 8 | 54, 52 | FALSE | F2, F3 |
| 9 | 100, 99 | FALSE | F3 |
| 10 | 91, 13 | FALSE | F3 |

| Factory | Allocated orders |
|---|---|
| F1 | 2, 3 |
| F2 | 1, 4, 5, 7, 8 |
| F3 | 6, 9, 10 |

### 5.1.2 Evaluation methods

To evaluate the performance of the proposed methods, the following indicators are used:

- Optimality gap: This measures the difference between WMAPE site and WMAPE global (LB). B&B is expected to achieve WMAPE site equal to WMAPE global, as this exact method aims to find the optimal solution.
- Improvement percentage: For heuristic methods, the percentage of WMAPE site improvement over the initial solution is calculated to compare different heuristics, as they may not yield the optimal result.
- Optimization time: The time required by each method to find the best solution is measured in seconds. The maximum allowed time by the company is 10 minutes (600 seconds).

Before conducting the evaluation tests, it is essential to determine the total order quantity being used. The company has a maximum order quantity of 100,000 boxes, though this limit is seldom reached. Typically, the total quantity falls between 10,000 and 70,000 boxes. Accordingly, the quantities used for each test in Section 5.2 are as follows:

- 1st test: The performance of three methods is compared at the starting quantity of 10,000 boxes. This quantity is also used to determine the suitable number of iterations for two heuristics.
- 2nd test: Quantities ranging from 10,000 to 100,000 boxes are used to evaluate the scalability of all methods and identify the most effective approach for large-scale operation.
- 3rd test: The best method is applied across the entire planning period to evaluate its effectiveness in continuous optimization, assuming no changes in factory capacities or real orders. Due to the high computational cost, this test is conducted at the minimum order quantity.

- 4th test: A similar test to the third one is performed, but with varying capacities and orders, to assess how well the best method adapts to fluctuations.

All tests are implemented in Python, and their codes are available on Github. The tests are run on a PC with an Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz and 8 GB of RAM. To ensure fair comparisons and reproducibility, a seed is used in all codes, except for the iteration test.

## 5.2 Computational results

### 5.2.1 Benchmark test

The following results are based on the total quantity of 10,000 orders over two consecutive days: LD12 (46% real orders) and LD11 (52% real orders).

For B&B which is an exact method, it uses the allocation decision of LD12 to directly determine the best allocation for LD11, which minimizes WMAPE site between two days (no swaps like heuristics). The allocations of orders on LD12 and LD11 are shown in Figure 5.3 while more detailed information can be found in Table 5.3.

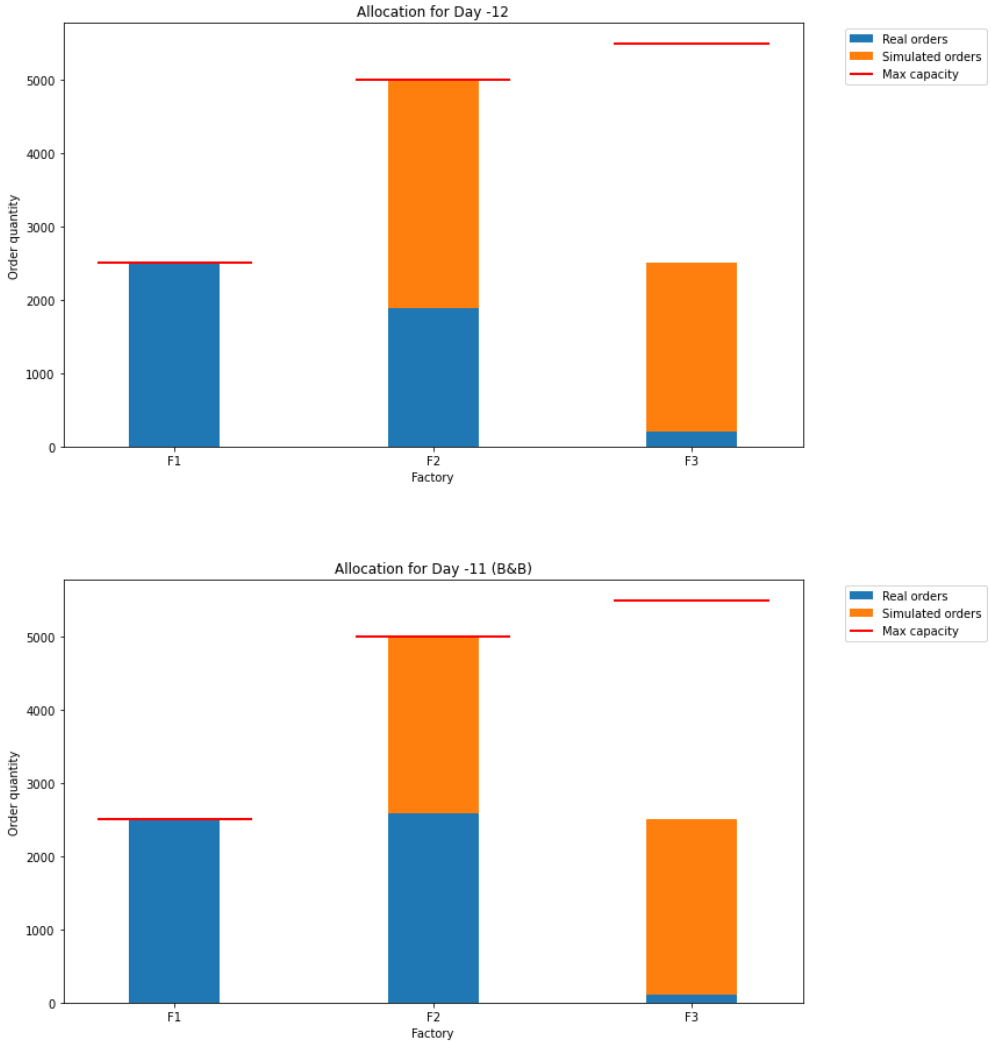



Figure 5.3. Allocation result of B&B

For heuristics, the number of iterations significantly influences their ability to find the optimal solution within a reasonable time. Therefore, the scalability test of iterations is conducted on 30 different sets of 10,000 orders to identify the best parameter for each heuristic.

The most typical results are as follows:

- Figure 5.4 demonstrates that 1,500 is the best number of iterations for ITPS because, beyond this point, there is nearly no improvement in WMAPE site, while computational time increases substantially.
- Similarly, Figure 5.5 shows that an iteration of 500 times is ideal for TS.

.

Table 5.2 summarizes the iteration test results.

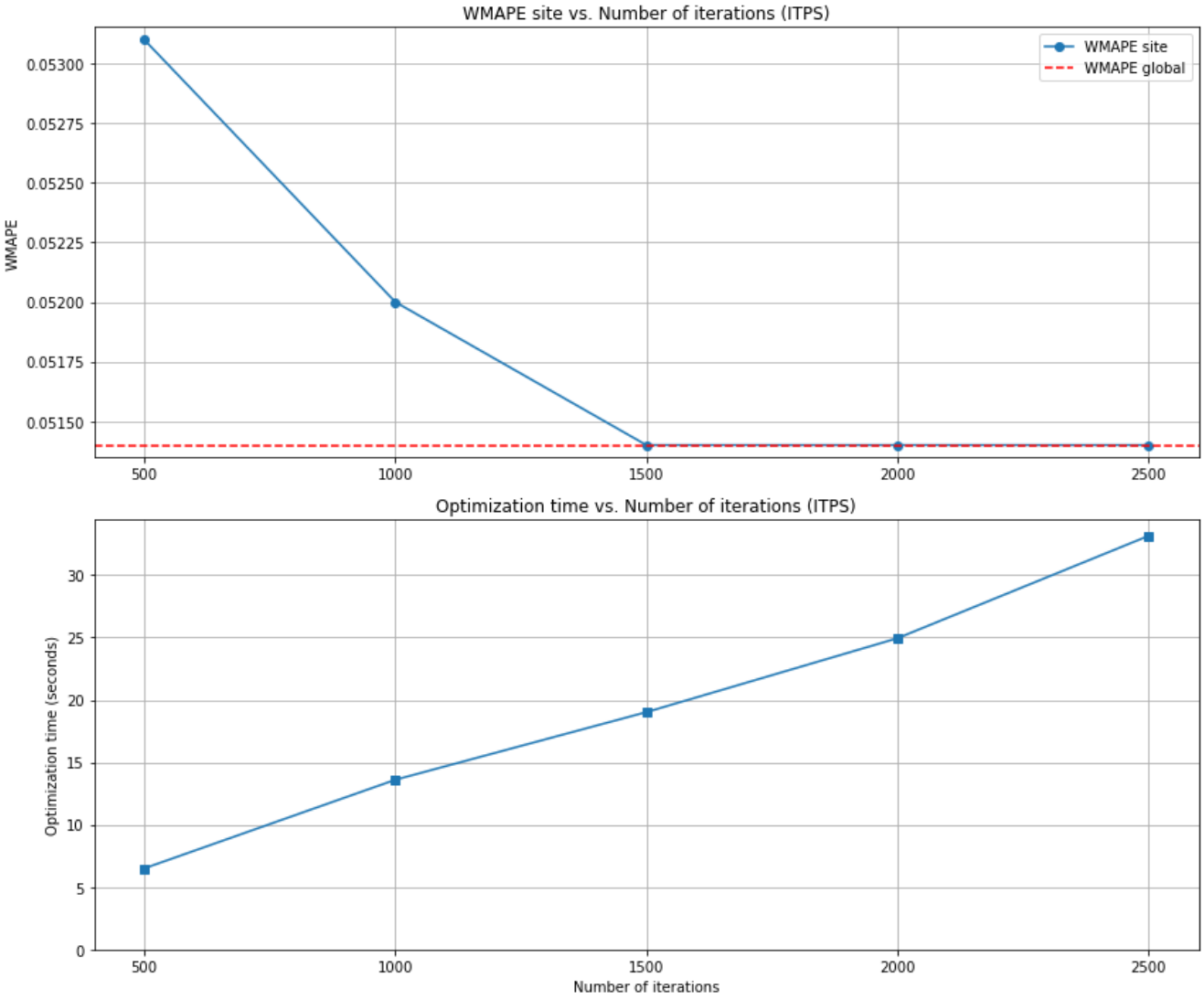



Figure 5.4. Iteration test of ITPS

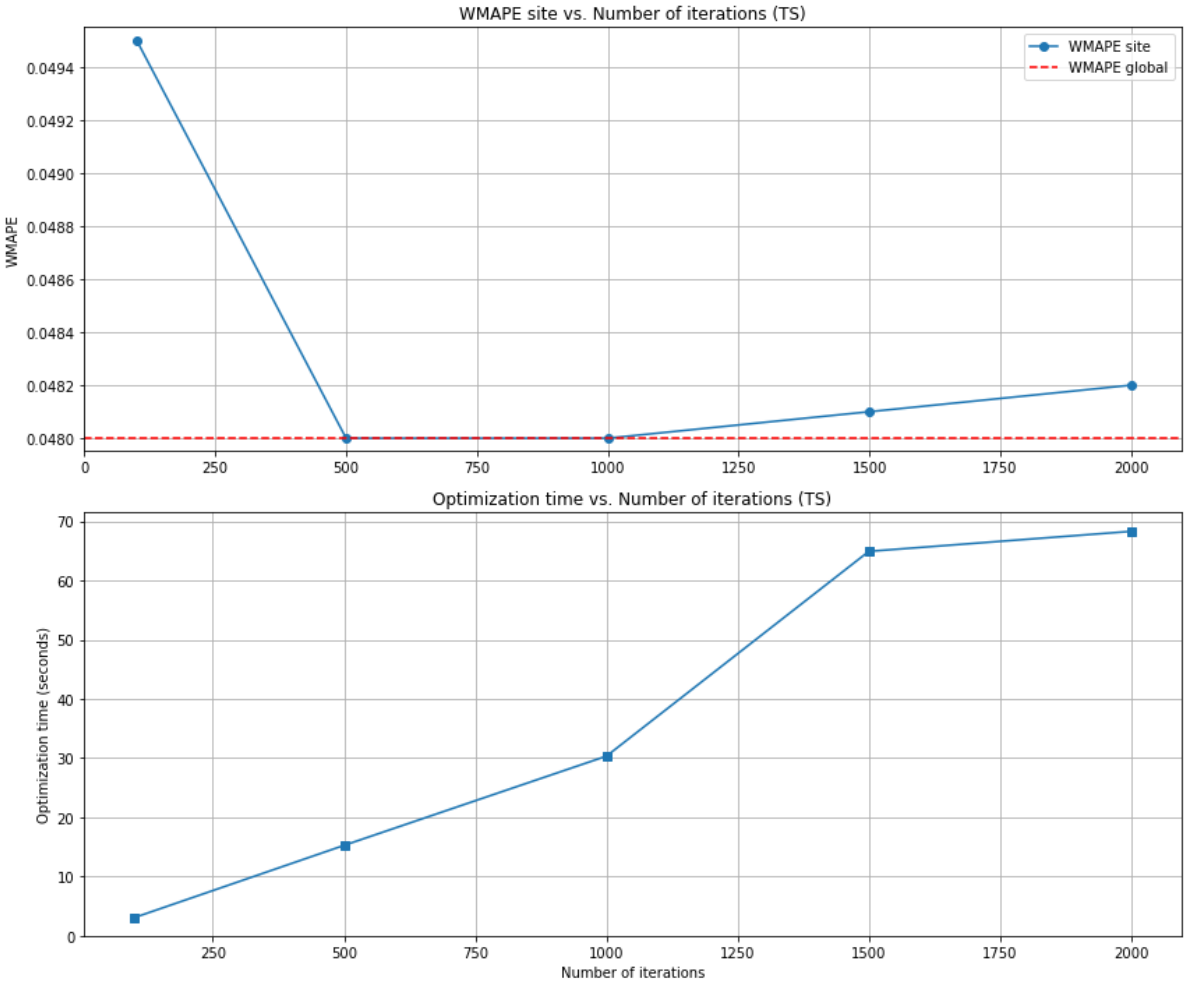



Figure 5.5. Iteration test of TS

Table 5.2. Summary of iteration test results

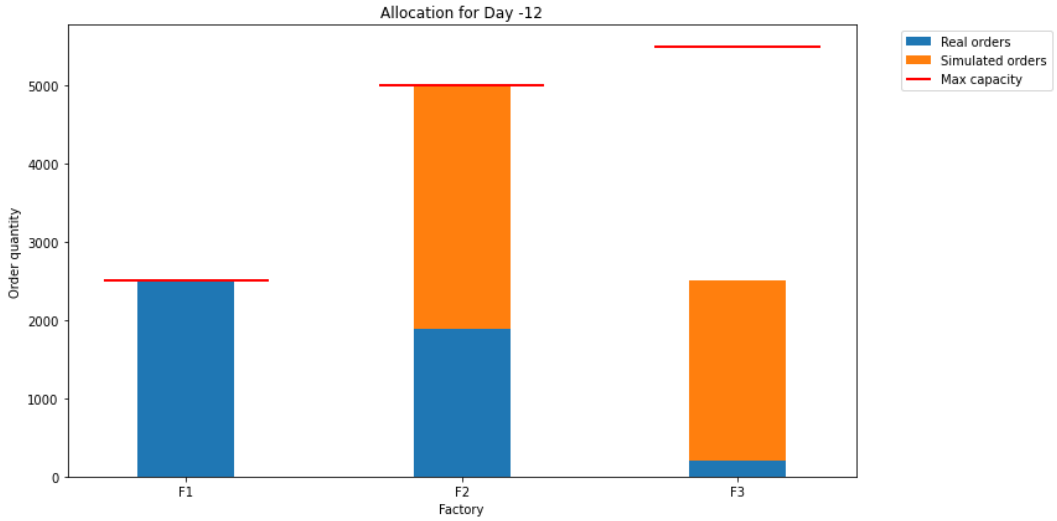
**ITPS**

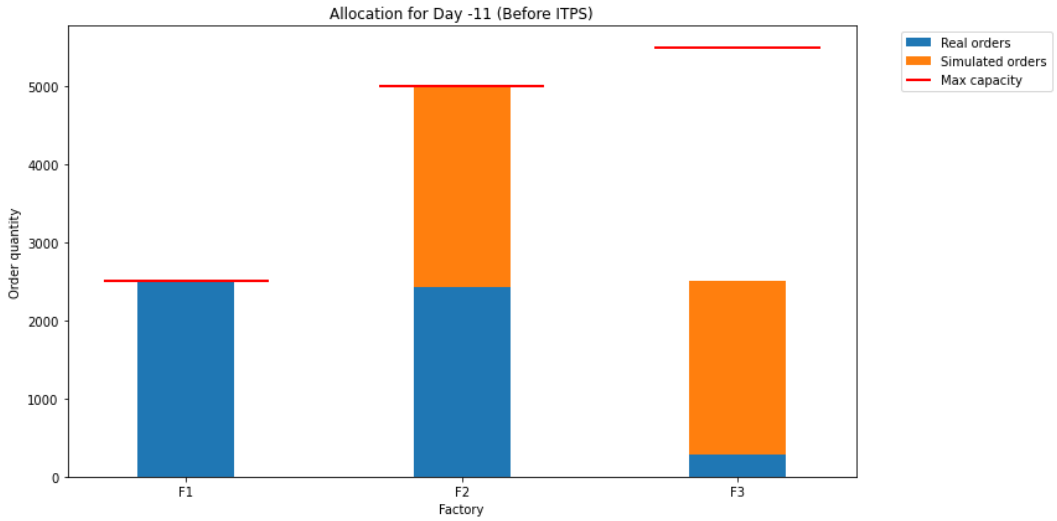
| Iterations | Initial WMAPE site | Optimized WMAPE site | WMAPE site improvement (%) | WMAPE global | Time (seconds) |
|---|---|---|---|---|---|
| 500 | 0.0675 | 0.0531 | 21.33 | 0.0514 | 6.5 |
| 1000 | 0.0675 | 0.052 | 22.96 | 0.0514 | 13.6 |
| 1500 | 0.0675 | 0.0514 | 23.85 | 0.0514 | 19.02 |
| 2000 | 0.0675 | 0.0514 | 23.85 | 0.0514 | 24.94 |
| 2500 | 0.0675 | 0.0514 | 23.85 | 0.0514 | 33.14 |

**TS**

| Iterations | Initial WMAPE site | Optimized WMAPE site | WMAPE site improvement (%) | WMAPE global | Time (seconds) |
|---|---|---|---|---|---|
| 100 | 0.0694 | 0.0495 | 28.67 | 0.048 | 3.05 |
| 500 | 0.0694 | 0.048 | 30.84 | 0.048 | 15.26 |
| 1000 | 0.0694 | 0.048 | 30.84 | 0.048 | 30.33 |
| 1500 | 0.0694 | 0.0481 | 30.69 | 0.048 | 64.91 |
| 2000 | 0.0694 | 0.0482 | 30.55 | 0.048 | 68.29 |

By using the suitable number of iterations, the allocations for LD11 before and after applying ITPS and TS are obtained in Figure 5.6 and Figure 5.7 respectively.



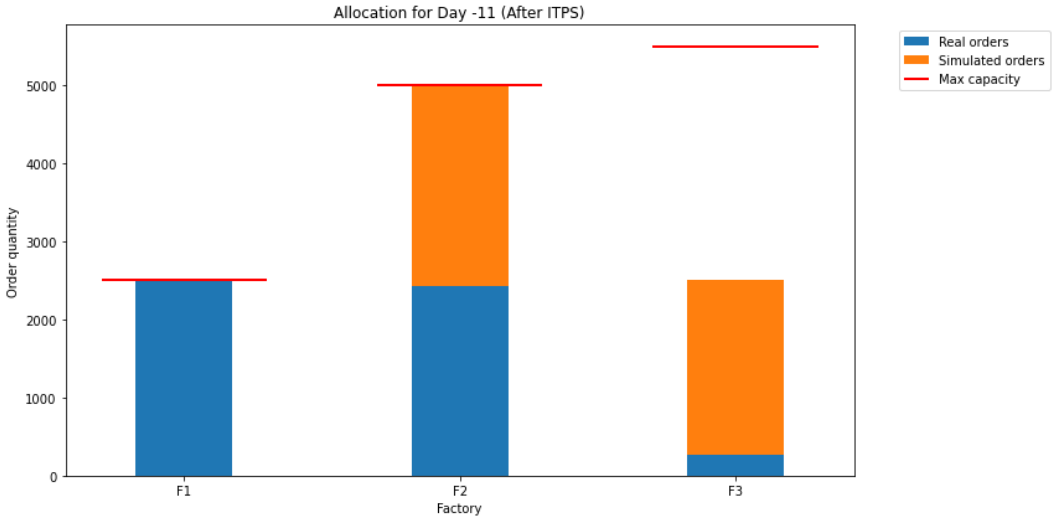
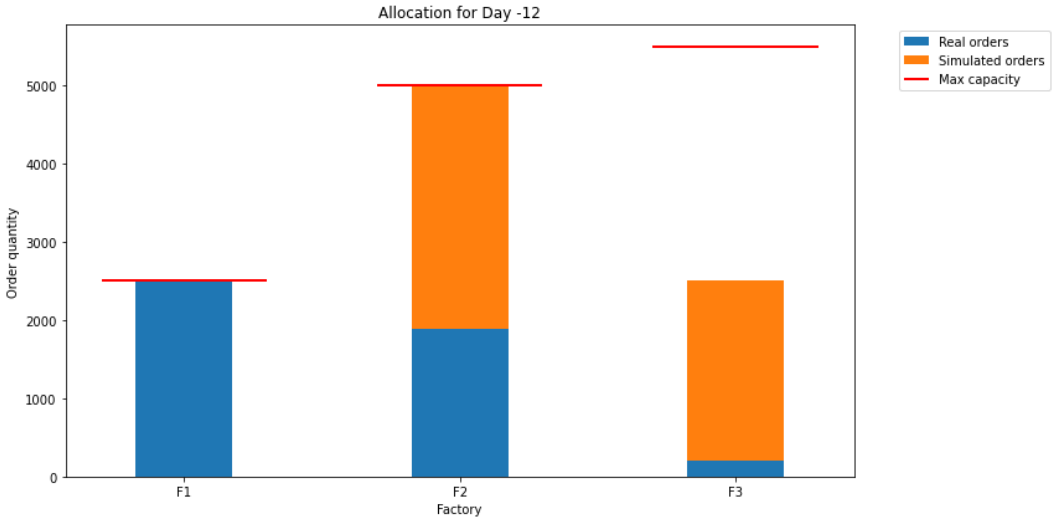
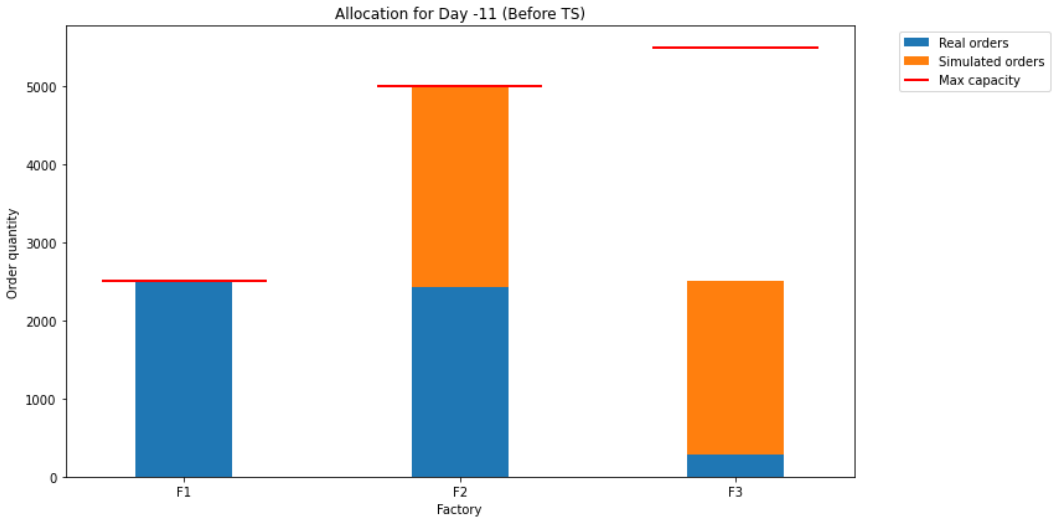



Figure 5.6. Allocation result of ITPS



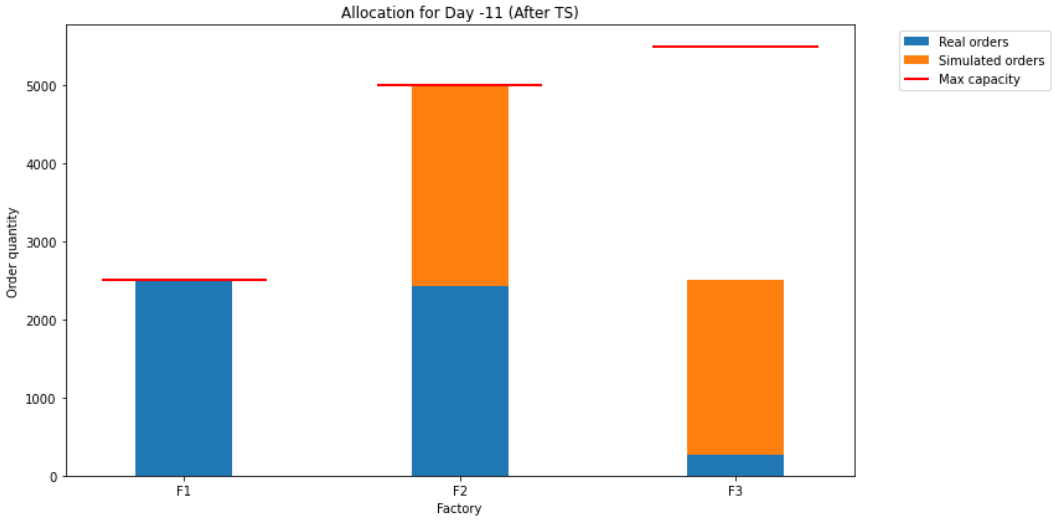



Figure 5.7. Allocation result of TS

Table 5.3. Summary of allocation results

| B&B | |
|---|---|
| **LD12** | |
| F1 | 2500 orders, 2500 real |
| F2 | 5000 orders, 1890 real |
| F3 | 2500 orders, 210 real |
| **LD11** | |
| F1 | 2500 orders, 2500 real |
| F2 | 5000 orders, 2593 real |
| F3 | 2500 orders, 107 real |
| **WMAPE site** | 0.054 |
| **WMAPE global** | 0.054 |
| **Optimization time** | 2.89 seconds |

| | ITPS | TS |
|---|---|---|
| **LD12** | | |
| F1 | 2500 orders, 2500 real | |
| F2 | 5000 orders, 1890 real | |
| F3 | 2500 orders, 210 real | |
| **LD11 (Before)** | | |
| F1 | 2500 orders, 2500 real | |
| F2 | 5000 orders, 2422 real | |
| F3 | 2500 orders, 278 real | |
| **LD11 (After)** | | |
| F1 | 2500 orders, 2500 real | 2500 orders, 2500 real |
| F2 | 5000 orders, 2436 real | 5000 orders, 2429 real |
| F3 | 2500 orders, 264 real | 2500 orders, 271 real |
| **WMAPE site** | | |
| Before | 0.074 | 0.074 |
| After | 0.054 | 0.054 |
| Improvement | 26.21% | 26.97% |
| **WMAPE global** | 0.054 | |
| **Optimization time** | 19.45 seconds | 15.35 seconds |

Table 5.3 points out that B&B is the best method, as it provides optimal solution in the shortest time. Among heuristics, TS slightly outperforms ITPS in both solution quality and speed.

While both B&B and heuristics provide solutions that match WMAPE global, their optimization ways differ significantly (explained in Chapter 4), leading to the following insights:

- Regardless of data complexity, B&B can consistently deliver the optimal solution by directly identifying the allocation that minimizes the recipe difference between the two days. Its approach ensures the best possible outcome.

- ITPS and TS improve the initial allocation iteratively. However, due to their tendency to converge on near-optimal solutions, they may fail to find the best solution if the initial allocation is poor or if the datasets have complex features. This limitation suggests that heuristics should be used only when exact method is impractical due to time constraint.

In our case, B&B appears to optimize BAP effectively, suggesting that heuristics may not compete with exact method if the upcoming scalability test confirms the same.

### 5.2.2 Scalibility test

In this test, each method will be applied to five different order quantities, ranging from 10,000 to 100,000, over two days: LD12 and LD11. Figure 5.8 compares the optimization times of three algorithms, while Figure 5.9 illustrates the gap between WMAPE site and WMAPE global for each method. A summary of results is provided in Table 5.4.

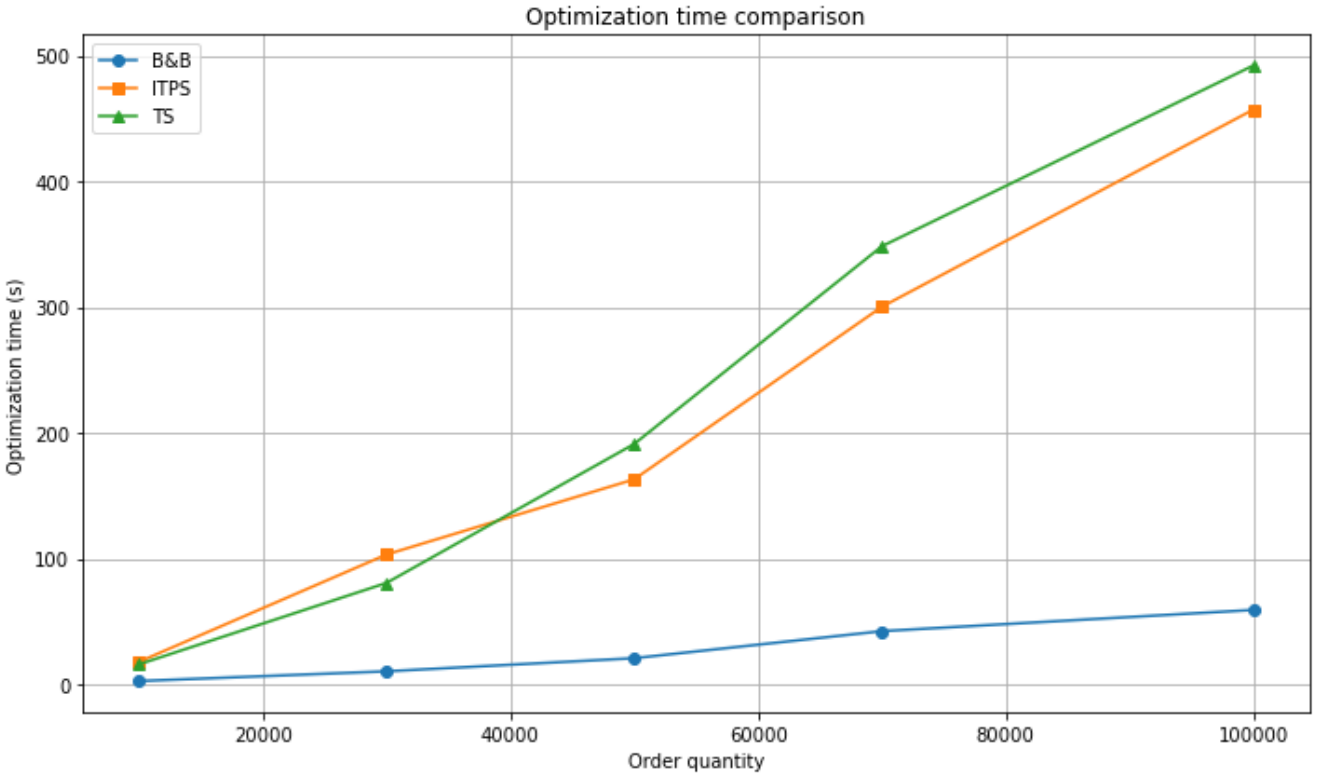
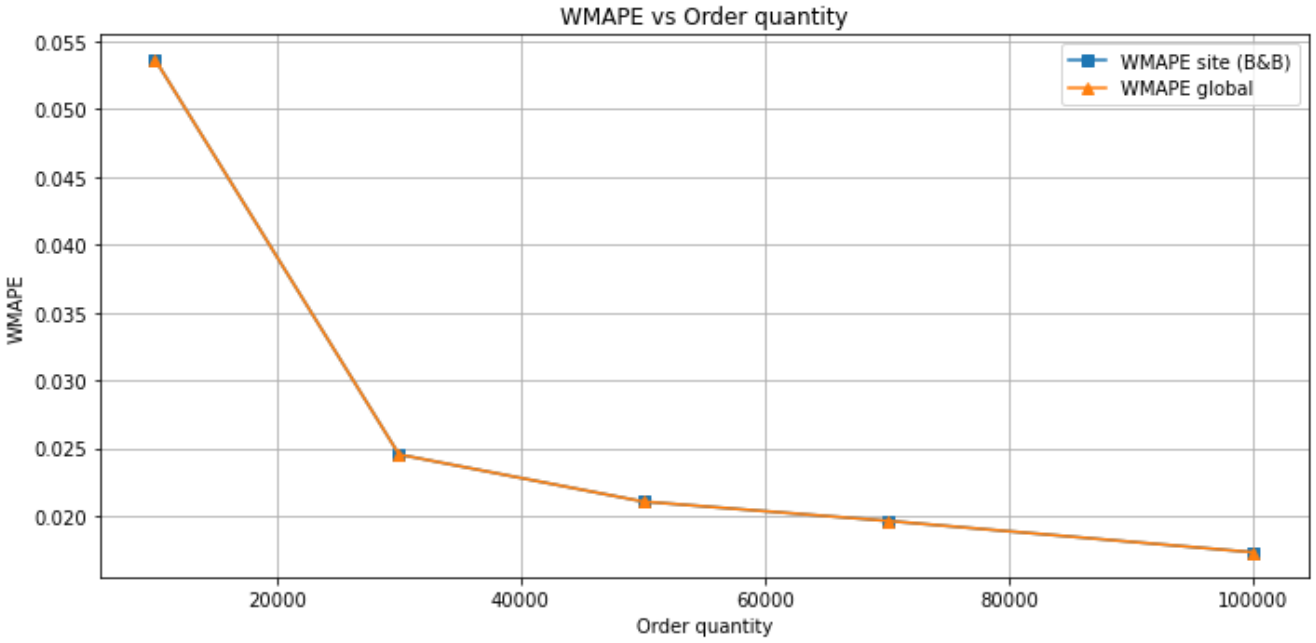



Figure 5.8. Comparison of optimization time


WMAPE vs Order quantity
WMAPE site (B&B)
WMAPE global
WMAPE
Order quantity
0.055
0.050
0.045
0.040
0.035
0.030
0.025
0.020
20000
40000
60000
80000
100000


WMAPE vs Order quantity
WMAPE site (Before ITPS)
WMAPE site (After ITPS)
WMAPE global
WMAPE
Order quantity
0.07
0.06
0.05
0.04
0.03
0.02
20000
40000
60000
80000
100000


Figure 5.9. Gap between WMAPE site and WMAPE global

Table 5.4. Summary of scalability test results (LD11 versus LD12)

| Order quantity | Optimization time (s) | WMAPE site (B&B) | WMAPE global |
|---|---|---|---|
| 10,000 | 2.67 | 0.054 | 0.054 |
| 30,000 | 10.41 | 0.025 | 0.025 |
| 50,000 | 20.84 | 0.021 | 0.021 |
| 70,000 | 42.35 | 0.020 | 0.020 |
| 100,000 | 59.2 | 0.017 | 0.017 |

| Order quantity | Optimization time (s) | WMAPE site (Before ITPS) | WMAPE site (After ITPS) | % Improvement | WMAPE global |
|---|---|---|---|---|---|
| 10,000 | 17.91 | 0.074 | 0.054 | 26.21% | 0.054 |
| 30,000 | 103.29 | 0.041 | 0.027 | 33.33% | 0.026 |
| 50,000 | 163.09 | 0.034 | 0.024 | 29.98% | 0.024 |
| 70,000 | 300.28 | 0.026 | 0.019 | 27.14% | 0.018 |
| 100,000 | 457.09 | 0.022 | 0.016 | 27.36% | 0.015 |

| Order quantity | Optimization time (s) | WMAPE site (Before TS) | WMAPE site (After TS) | % Improvement | WMAPE global |
|---|---|---|---|---|---|
| 10,000 | 15.91 | 0.074 | 0.054 | 26.97% | 0.054 |
| 30,000 | 80.7 | 0.042 | 0.030 | 29.23% | 0.030 |
| 50,000 | 191.19 | 0.031 | 0.024 | 23.24% | 0.024 |
| 70,000 | 348.38 | 0.025 | 0.017 | 29.42% | 0.017 |
| 100,000 | 492.15 | 0.022 | 0.016 | 28.02% | 0.016 |

This test validates that B&B is the best method, consistently delivering optimal solutions across all quantities in the shortest time. Between two heuristics, although TS requires more time for quantities exceeding 40,000, it generally outperforms ITPS because TS can reduce the initial WMAPE sites to the optimal values across all quantities. This highlights the effectiveness of a hybrid approach that integrates TS with targeted LS. For ITPS, increasing the number of iterations may help achieve optimal values, but it also extends the already lengthy optimization time.

This suggests while LS is primarily effective for small quantities, BAP requires more advanced techniques for large instances. A positive outcome is that all methods can provide good allocation decisions for the maximum order quantity in less than 10 minutes.

One noteworthy finding is that when the total order quantity increases, both WMAPE decreases. This can be explained by the principles of aggregate forecasting in supply chain management which states that forecasts are more accurate for groups of items than for individual items because the variability within a group tends to cancel out, leading to more stable and reliable predictions (O'Reilly, no date). Moreover, as the order quantity increases, individual fluctuations have less impact on the overall error while in smaller samples, the change of each order has a proportionally larger effect.

### 5.2.3 Temporal fixed test

In this test, the best method will be applied to address the temporal aspect of BAP. Specifically, B&B will continuously make allocation decisions for a 15-day planning period under ideal conditions, where there are no changes in capacity or real orders. Due to the high computational cost, the total quantity of 10,000 orders is used. Figures 5.10 illustrate the increasing proportions of real orders through days. As the timeline approaches LD3, the composition of orders between two consecutive days becomes increasingly similar, since each day's orders include those from the previous day. This explains the decreasing trend in two error graphs.

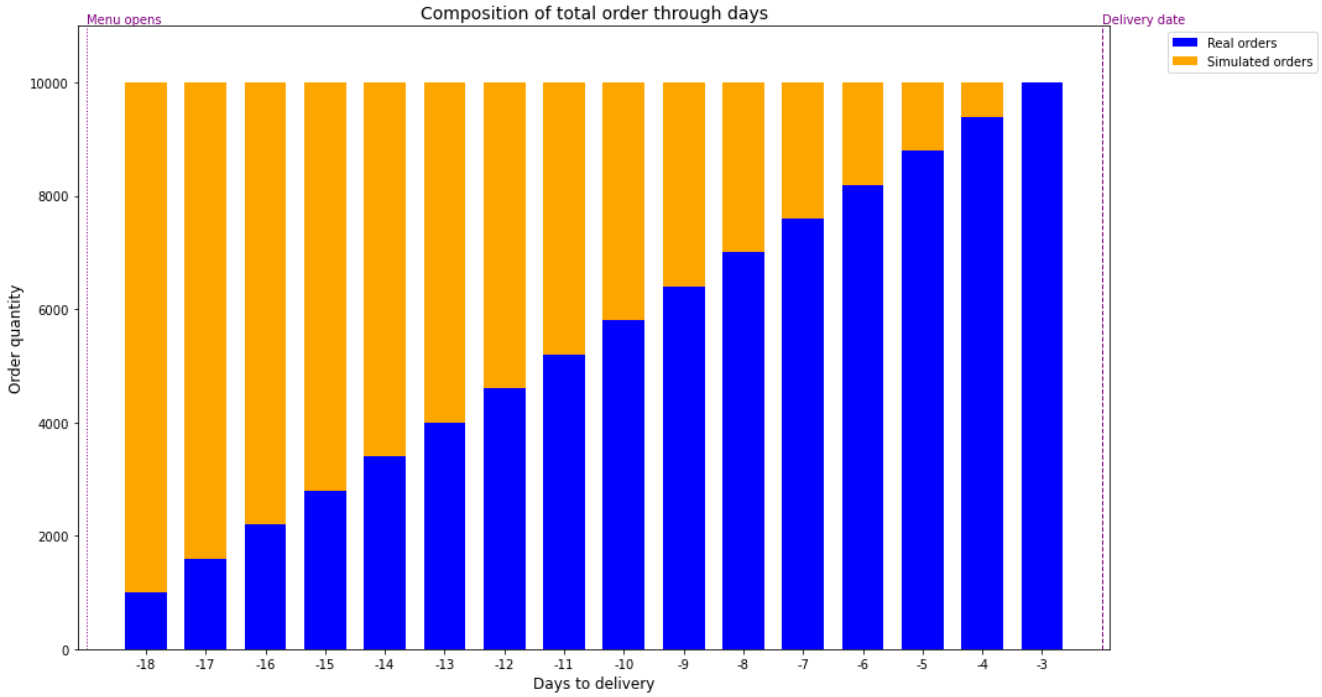



Figure 5.10. Composition of total orders through days

- Figure 5.11 shows WMAPE site achieved when B&B continuously optimizes allocation between two consecutive days. For example, LD17 is optimized based on the greedy allocation of LD18 (greedy because there is no previous day to compare with), LD16 is optimized based on the B&B allocation of LD17, and so on. It can be seen that, except for some initial

days when the real proportion is still low, B&B consistently provides optimal solutions from LD13 to LD3. The WMAPE site (Greedy) line, representing results of the greedy algorithm described in Section 4.1, shows that without B&B, the recipe differences between two days are significantly higher.

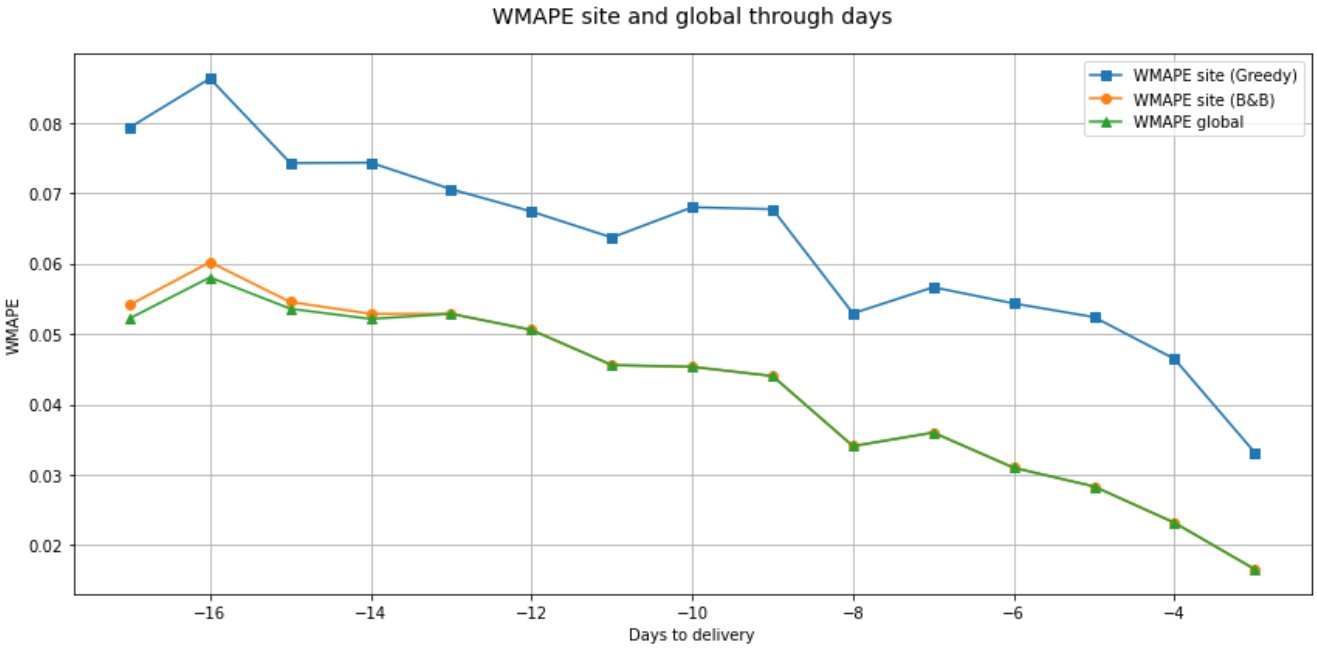



Figure 5.11. WMAPE site and WMAPE global through days

- In Figure 5.12, after B&B completes the order allocation for all days, the allocation of each past day is compared with the final day's allocation. This comparison evaluates whether day-by-day optimization in Figure 5.11 can ensure a smooth transition from soft allocations to hard allocation despite limited information about future orders. The gradual decrease in WMAPE site indicates that B&B has effectively contributed to stable production planning, minimizing abrupt changes in recipe differences. The ultimate goal is to minimize the area under WMAPE site curve, ideally making it match the area under WMAPE global curve. Although there is a big gap between the two lines in the early days, it narrows over time. As a result, the area under WMAPE site curve is not much larger than that of WMAPE global curve, confirming the effectiveness of B&B in achieving proxy optimization, as discussed in Section 3.2.

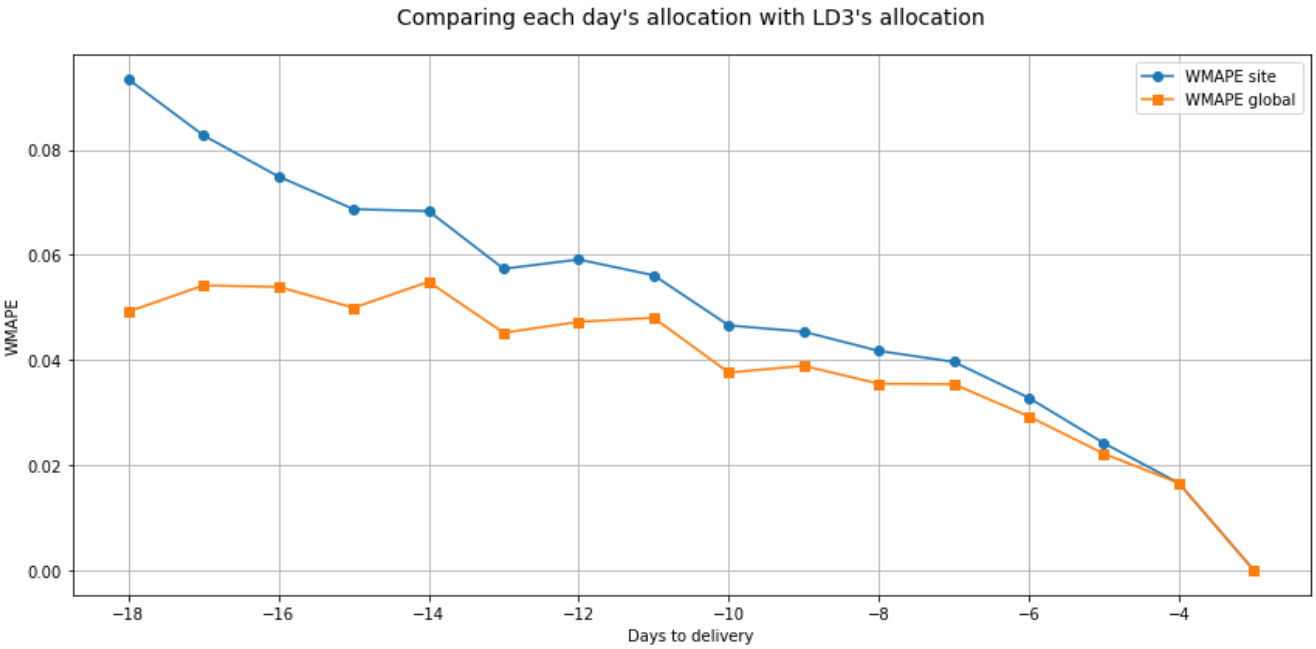



Figure 5.12. Comparison between each day's allocation and final day's allocation

Table 5.5 provides the detailed data of the three above graphs.

Table 5.5. Summary of temporal fixed test results

| Day | Real orders proportion | WMAPE site (B&B) | WMAPE site (Greedy) | WMAPE global | WMAPE site vs LD3 | WMAPE global vs LD3 |
|---|---|---|---|---|---|---|
| -18 | 10% | N/A | N/A | N/A | 0.093 | 0.049 |
| -17 | 16% | 0.054 | 0.079 | 0.052 | 0.083 | 0.054 |
| -16 | 22% | 0.060 | 0.086 | 0.058 | 0.075 | 0.054 |
| -15 | 28% | 0.055 | 0.074 | 0.054 | 0.069 | 0.050 |
| -14 | 34% | 0.053 | 0.074 | 0.052 | 0.068 | 0.055 |
| -13 | 40% | 0.053 | 0.071 | 0.053 | 0.057 | 0.045 |
| -12 | 46% | 0.051 | 0.067 | 0.051 | 0.059 | 0.047 |
| -11 | 52% | 0.046 | 0.064 | 0.046 | 0.056 | 0.048 |
| -10 | 58% | 0.045 | 0.068 | 0.045 | 0.047 | 0.038 |
| -9 | 64% | 0.044 | 0.068 | 0.044 | 0.045 | 0.039 |
| -8 | 70% | 0.034 | 0.053 | 0.034 | 0.042 | 0.035 |
| -7 | 76% | 0.036 | 0.057 | 0.036 | 0.040 | 0.035 |
| -6 | 82% | 0.031 | 0.054 | 0.031 | 0.033 | 0.029 |
| -5 | 88% | 0.028 | 0.052 | 0.028 | 0.024 | 0.022 |
| -4 | 94% | 0.023 | 0.046 | 0.023 | 0.017 | 0.017 |
| -3 | 100% | 0.017 | 0.033 | 0 | 0 | 0 |

TS, which effectively minimized WMAPE site in the previous two tests, is also applied to this test. However, the results show that TS performs slightly worse than B&B, with detailed graphs available in Appendix III.

### 5.2.4 Temporal variation test

In the previous test, factory capacities, and real orders are assumed to remain unchanged throughout the entire planning period. However, in reality, the following scenarios can happen:

1. From LD18 to LD10, F1 was initially assigned a capacity of 3,000. However, by LD10, the volume of orders is lower than expected. Therefore, F1's capacity is reduced from 3,000 to 1,000. This adjustment is anticipated to generate site errors. The real occurrence of this scenario necessitates evaluating how the proposed methods adapt to capacity changes.

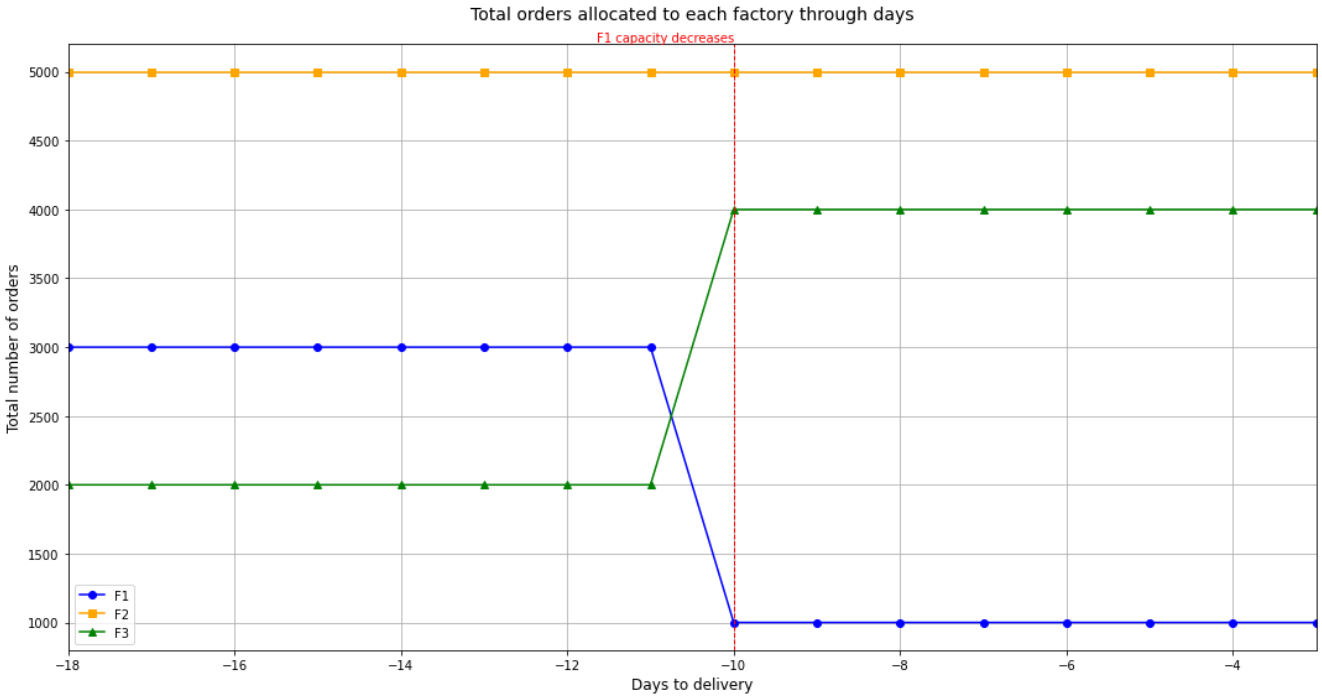
2. Various changes to real orders can also occur, such as customers adjusting recipes inside their orders or canceling orders. Investigating order change is important to understand its impact on WMAPE and to assess how our methods adapt.

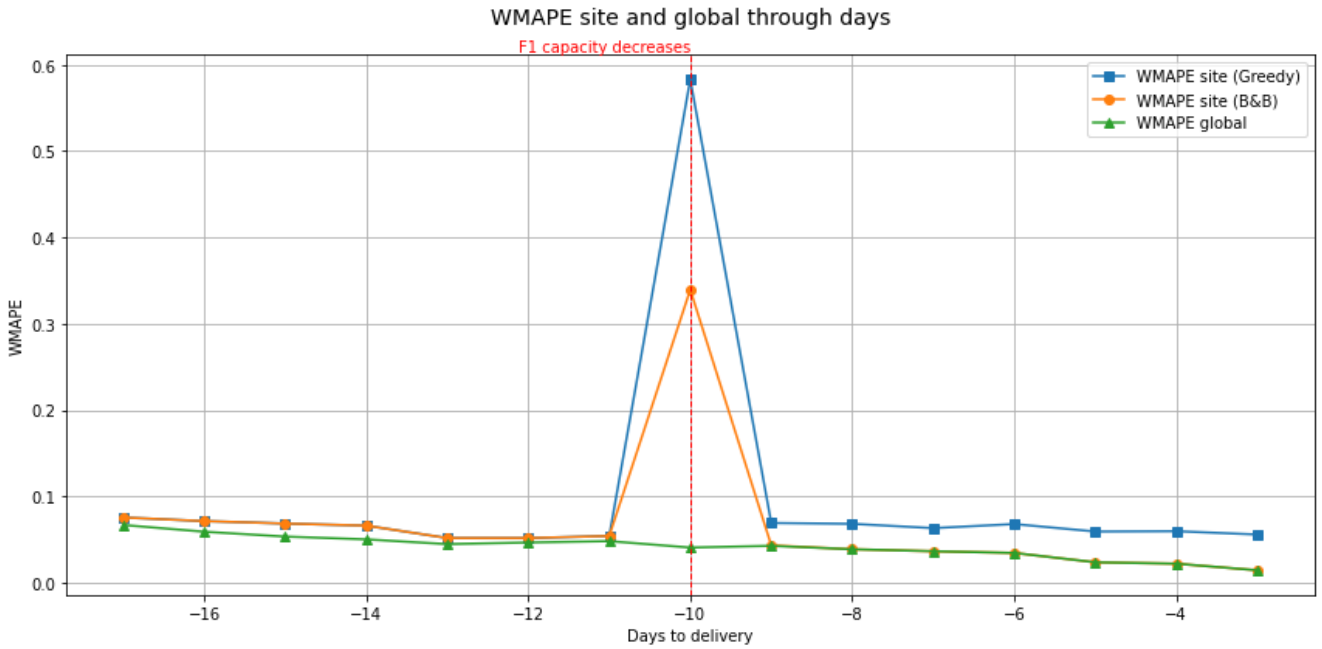
3. On some days, both factory capacities and order details can change. This further complicates the situation and requires investigation.

To evaluate how exact and heuristic methods respond to the above variations, both B&B and TS are applied. The results show that while TS effectively handles capacity changes, it struggles to minimize WMAPE site when orders change. Given the superior performance of B&B, its results are presented here for further analysis, with TS's results available in Appendix III:

1. **Capacity change**: Figure 5.13 shows the impact of a significant capacity reduction in F1 from 3,000 to 1,000 orders on LD10. This reduction causes a spike in WMAPE site due to the sudden change in the number of orders allocated to each factory. While B&B demonstrates impressive agility by quickly adjusting allocations and reaching the optimal WMAPE site by LD9, the greedy method results in a wider gap between WMAPE site and WMAPE global after the capacity change, highlighting its lack of flexibility.

   The comparison between each day's allocation and that of the final day shows a smooth convergence to the optimal value of WMAPE site starting from LD10. The sharp drop in error between LD11 and LD10 marks a clear transition between two different operational scenarios. As a result, all allocation decisions made before LD10 have significant discrepancies compared to LD3. This sudden shift highlights B&B's ability to adapt to major change in operational conditions quickly.


Total orders allocated to each factory through days
F1 capacity decreases
Total number of orders
Days to delivery
F1
F2
F3


WMAPE site and global through days
F1 capacity decreases
WMAPE
Days to delivery
WMAPE site (Greedy)
WMAPE site (B&B)
WMAPE global

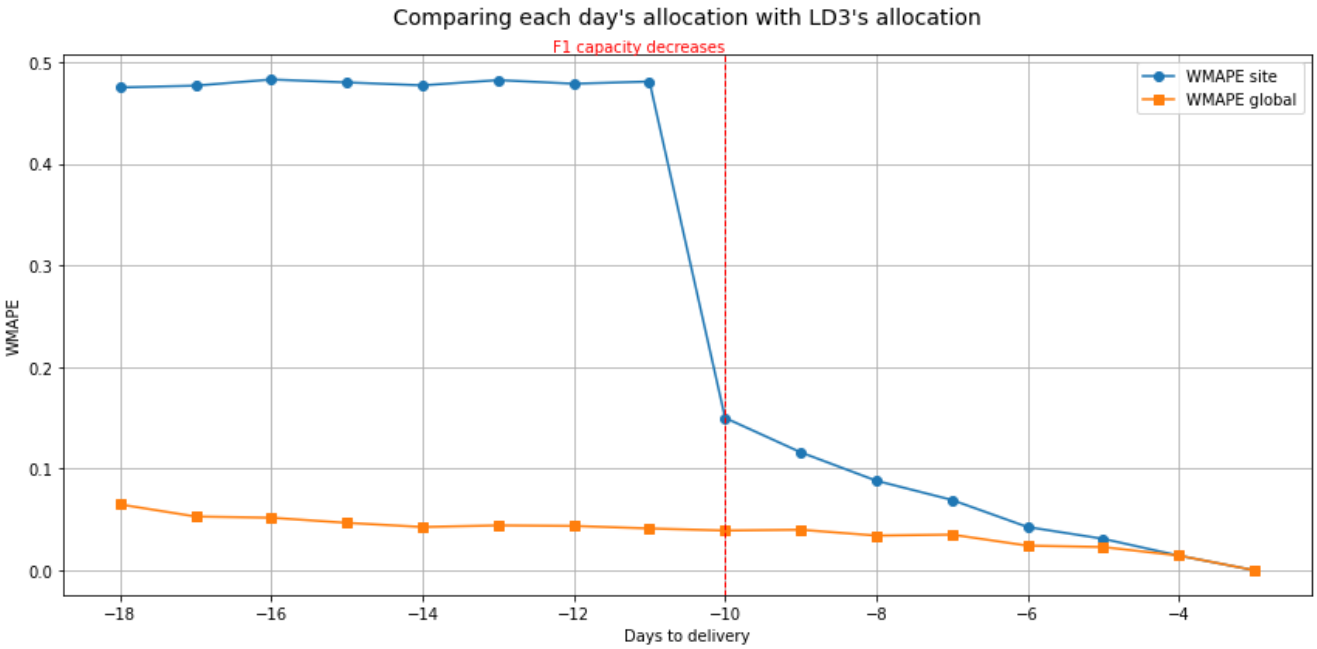



Figure 5.13. Temporal test with F1's capacity change

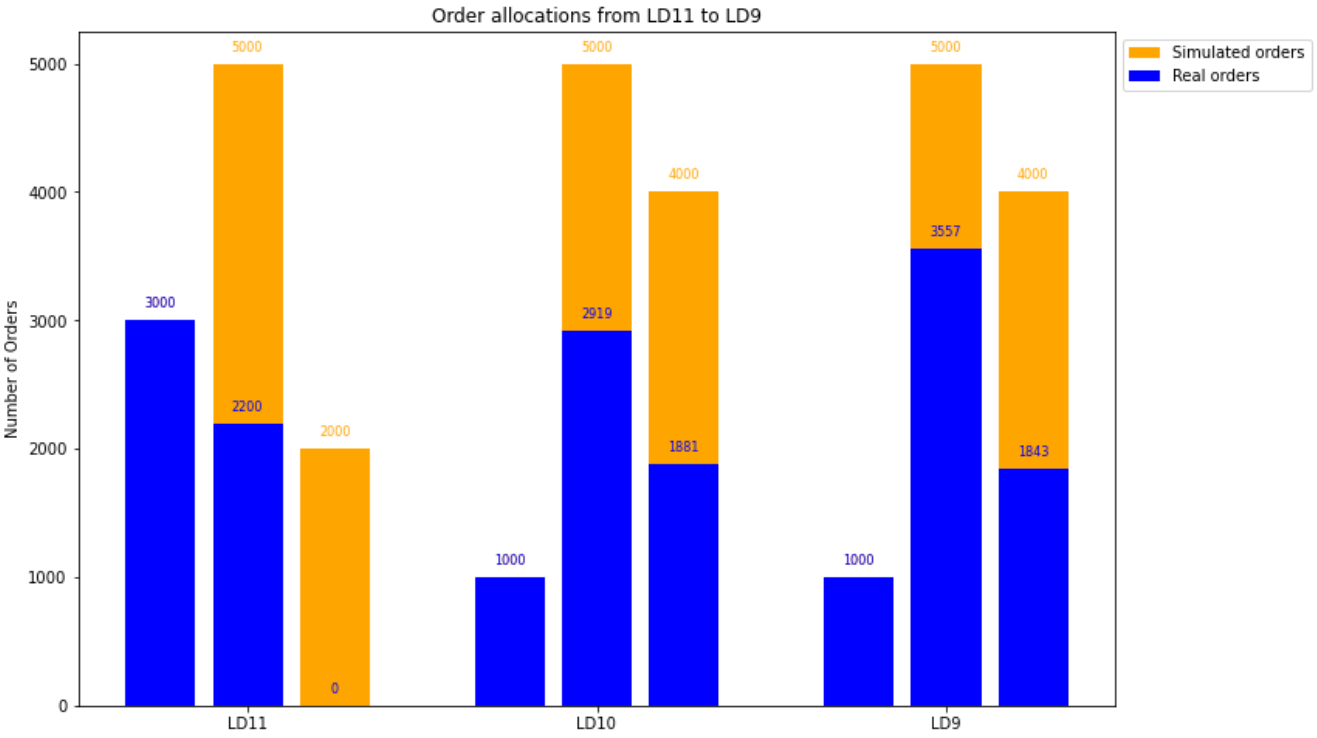



Figure 5.14. Order allocations from LD11 to LD9

To provide a more detailed analysis, B&B's order allocations around the capacity-changing point are extracted and shown in Figure 5.14. This visualization confirms that the solutions adhere to the full capacity constraint and illustrates the transfer of 2,000 real orders from F1 to F2 and F3. It also highlights the ongoing refinement of the solution from LD10 to LD9, evidenced by fluctuations in real order quantities, with detailed WMAPE as follows:

| Day | WMAPE site (B&B) | WMAPE site (Greedy) | WMAPE global |
|---|---|---|---|
| -11 | 0.054 | 0.054 | 0.048 |
| -10 | 0.339 | 0.583 | 0.041 |
| -9 | 0.043 | 0.069 | 0.043 |

2. **Order changes**: This study assumes that each day, 5% of real orders are deleted and 30% (a figure higher than reality to highlight the effectiveness of B&B) have recipe adjustments by customers. The daily progression of changed and deleted orders is illustrated in Figure 5.15.

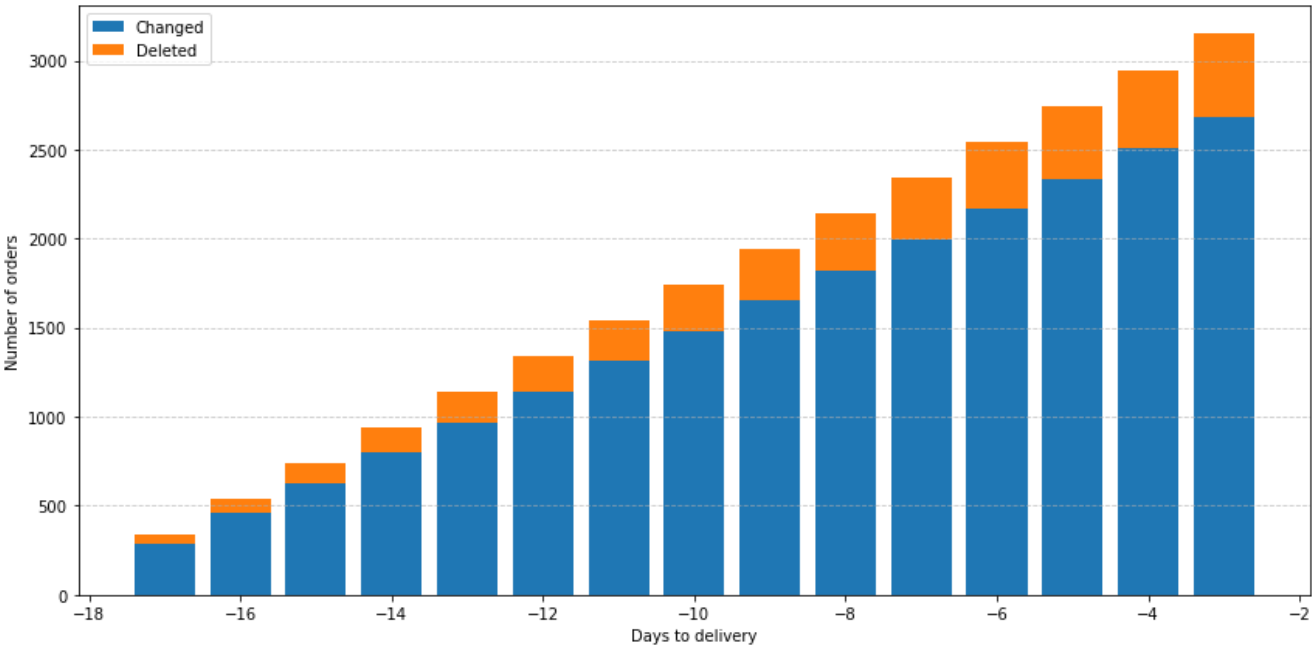



Figure 5.15. Quantity of changed and deleted real orders through days

Based on the above assumption, the following results are achieved:

- Figure 5.16 illustrates how order changes cause fluctuations in both WMAPE site and WMAPE global. It can be observed that, without order changes, two WMAPEs show a clear decreasing trend. However, when changes occur, two WMAPEs fluctuate significantly, depending on the magnitude of these changes. This highlights the challenges in real business when managing errors. However, B&B consistently achieves the optimal WMAPE site by matching WMAPE global in all situations.

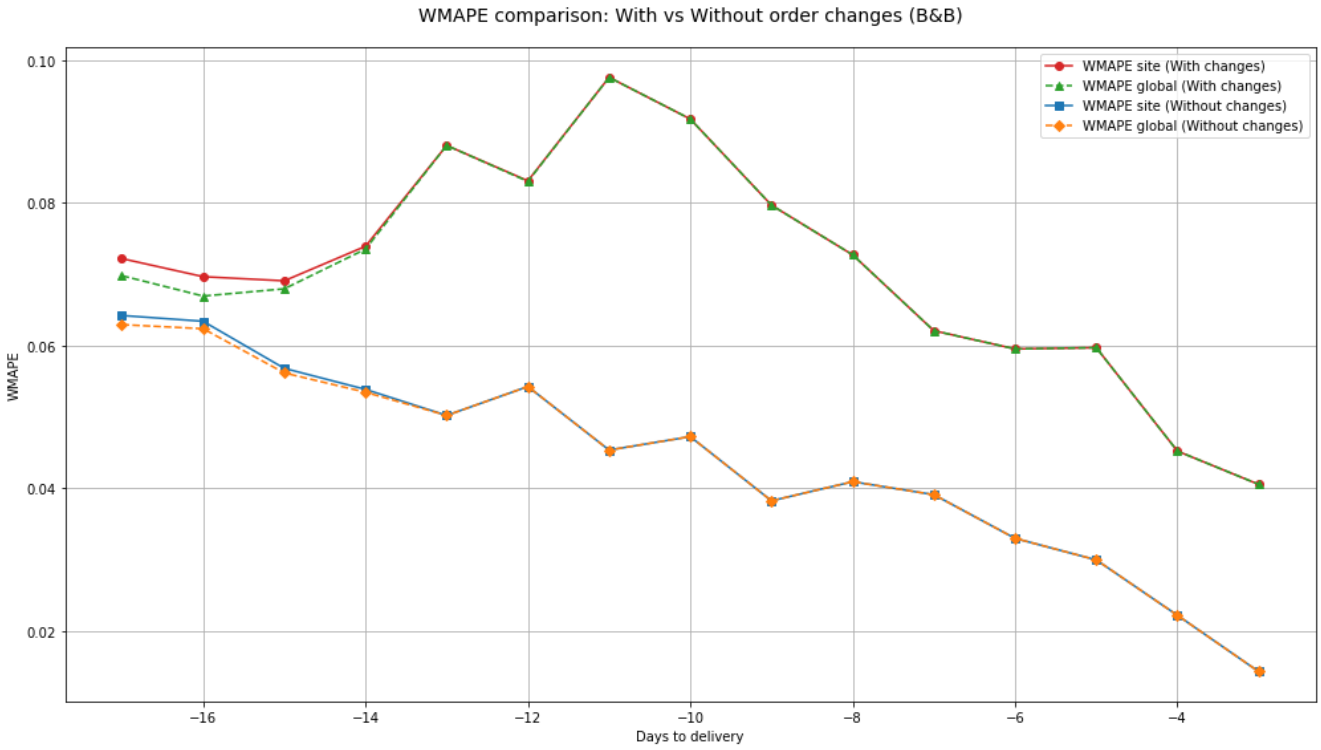



Figure 5.16. WMAPE site and WMAPE global with and without order changes

- Since IDs of real orders remain unchanged over time (as mentioned in Section 5.1.1), it is logical to allocate orders of the next day based on their positions from the previous day. This is explained in the following example:
  - LD14:

| Real order ID | Allocated factory |
|---|---|
| R1 | F1 |
| R2 | F2 |
| R3 | F2 |
| R4 | F3 |

  - LD15: The previous allocation decisions for real orders are reused to ensure these orders are consistently sent to the same factory, and minimize changes.

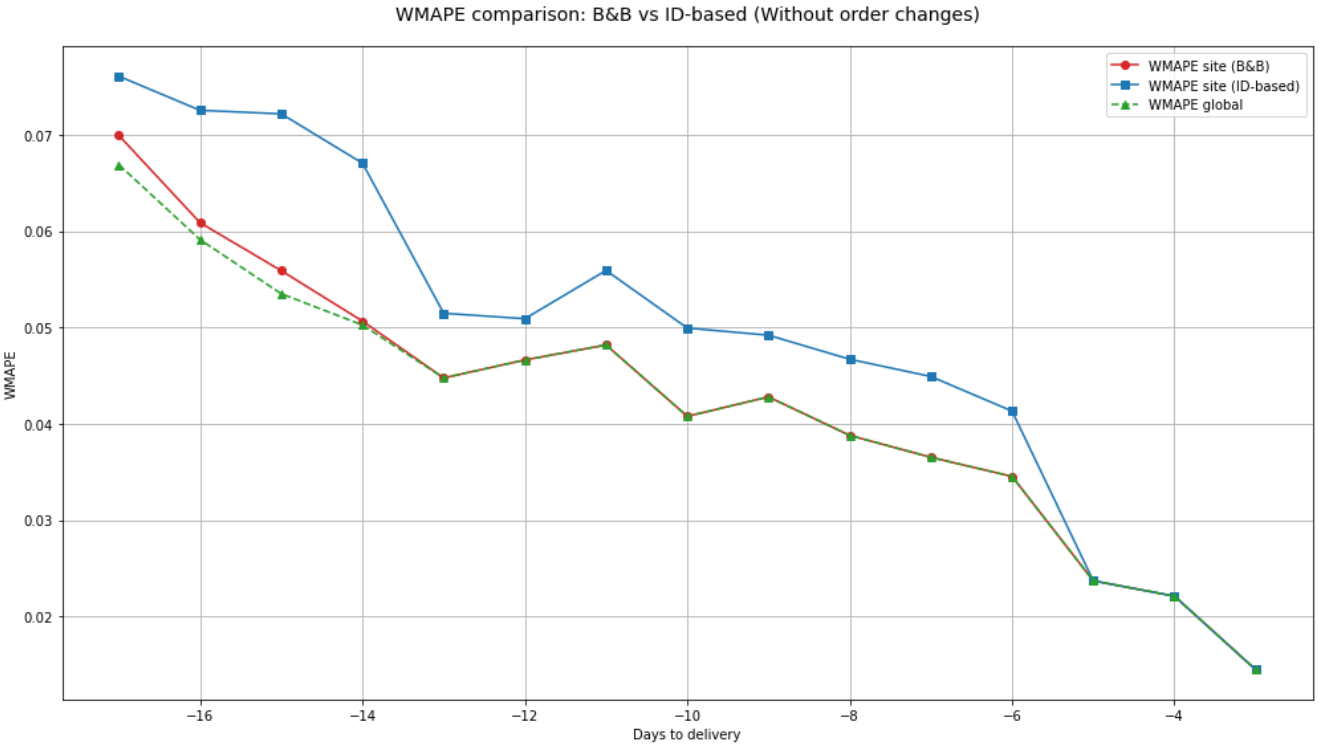
  However, the ID-based method has some problems:

  - Changes of recipes within each order (for example, R1) may alter its eligibility, potentially making the order no longer suitable for F1. This could confuse the staff at F1 if R1 is still assigned to this factory.
  - Deleted orders can further complicate the situation, as IDs of previous real orders can no longer be retrieved today.

  Even if the company can detect IDs of all changed and deleted orders and re-evaluate their eligibility to prevent factory confusion, issues still persist. Figure 5.17 illustrates

the above problems by comparing the ID-based allocation method with B&B. The results show that, regardless of order changes, WMAPE site of B&B consistently aligns closely with WMAPE global. This helps to minimize variations in number of each recipe assigned to factories, allowing for more accurate and efficient ingredient stock preparation.

Notably, when no order changes exist, the ID-based method can match WMAPE global on some final days when the percentage of real orders is high. However, when order changes are present, the gap between WMAPE site of ID-based method and WMAPE global widens over time. This underscores the limitations of the conventional allocation method and highlights the advantages of more advanced techniques like B&B.



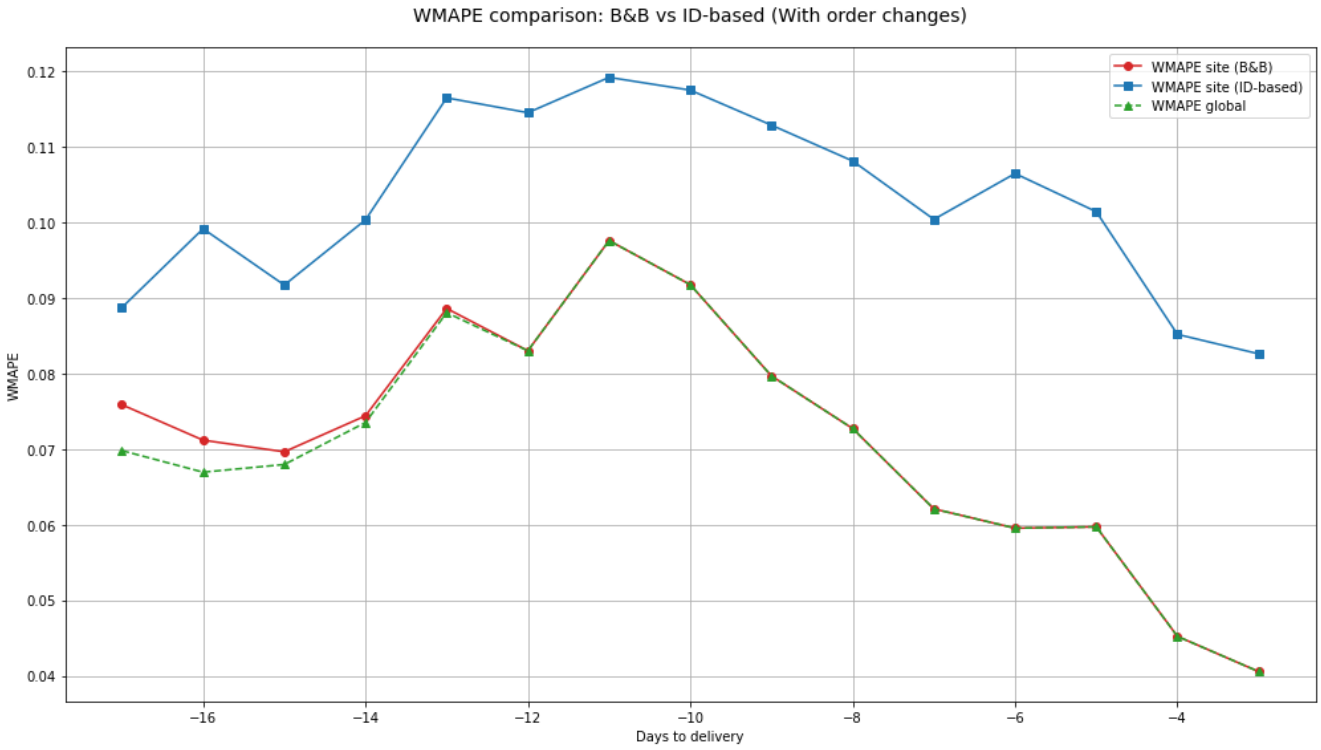
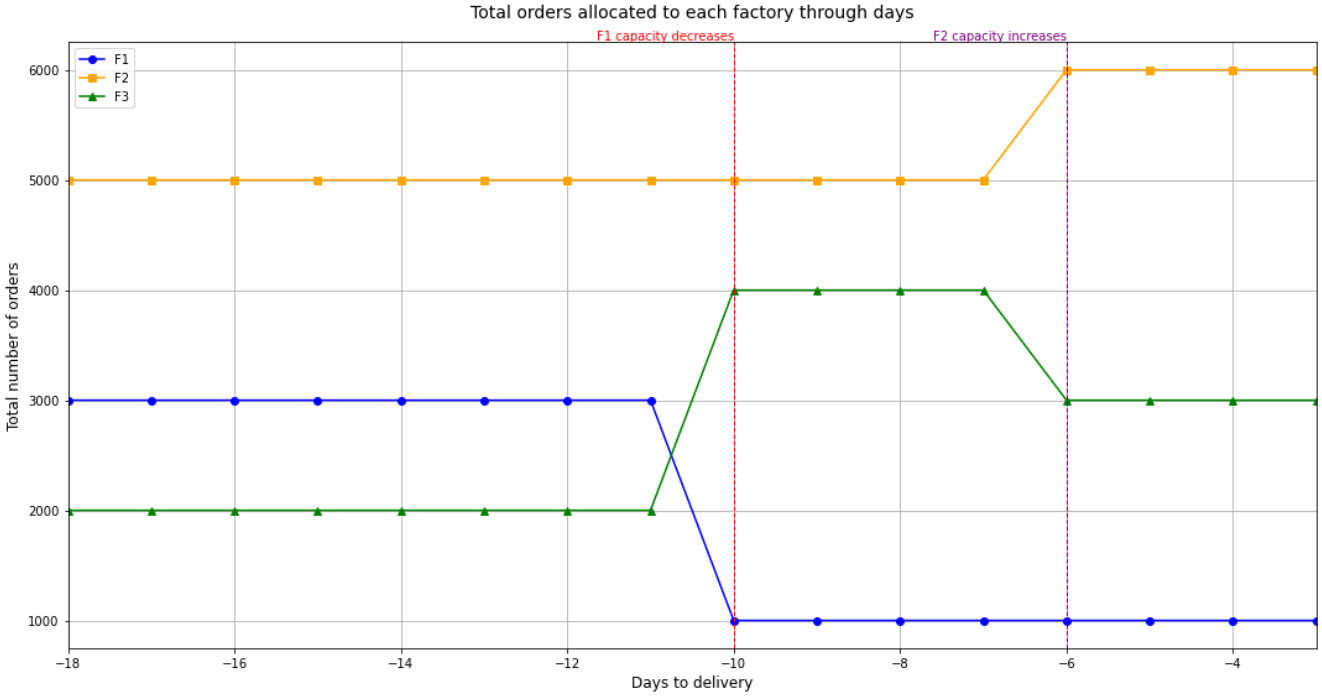


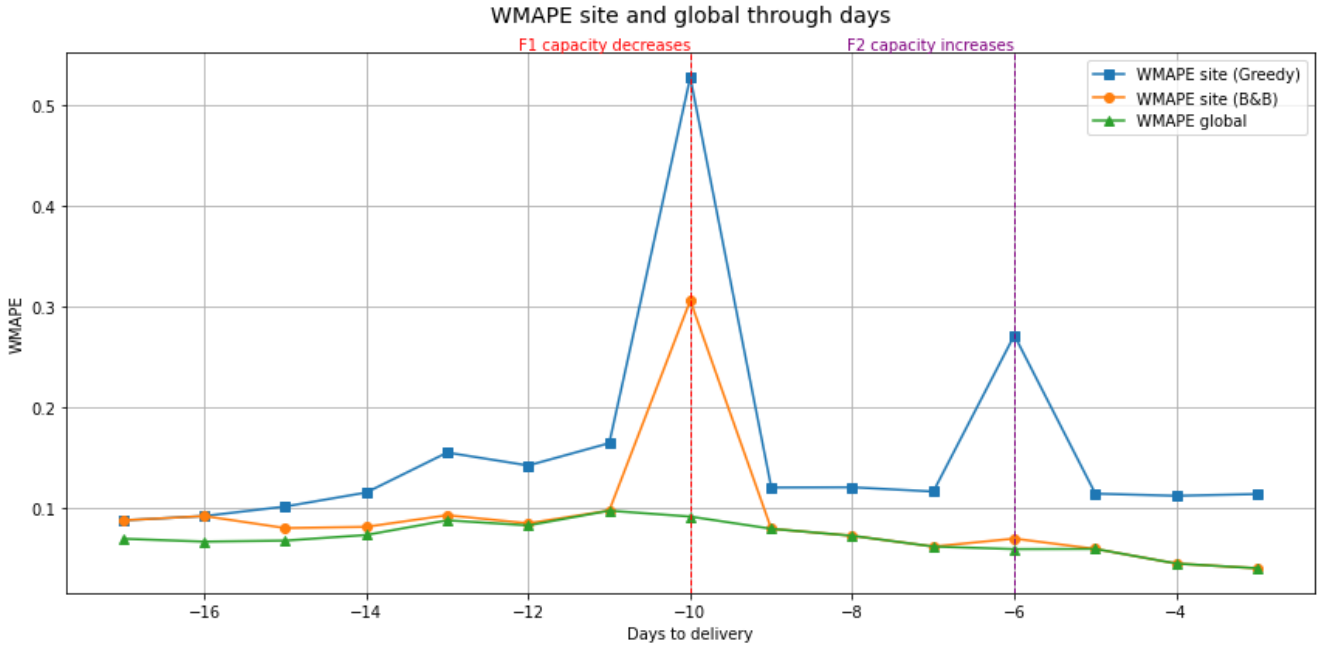

Figure 5.17. Comparison between B&B and ID-based allocation method

3. **Both capacity and order change**: In the final scenario, responses of greedy method and B&B are examined in the context of changing capacities of both F1 and F2, along with daily deletions and modifications of real orders.

   Figure 5.18 illustrates a consistent and significant gap between WMAPE site obtained by greedy allocation and WMAPE global. In contrast, B&B achieves the optimal WMAPE site on most days when there are no capacity changes. Notably, while WMAPE site of greedy allocation tends to spike whenever there is a capacity change, B&B is primarily concerned with the decrease in factory capacity, as it limits the quantity of orders that can be allocated to the most suitable factory.

   For example, on LD6, when the capacity of F2 is increased to accommodate more orders, compensating for the earlier reduction in F1's capacity, the impact of this change on the WMAPE site of B&B is minimal. This explains the smooth decreasing trend from LD10 to LD3 when comparing each day's allocation with LD3's allocation, highlighting the adaptability and effectiveness of B&B, even in scenarios with both order and capacity changes.


Total orders allocated to each factory through days
F1 capacity decreases
F2 capacity increases
F1
F2
F3
Total number of orders
Days to delivery


WMAPE site and global through days
F1 capacity decreases
F2 capacity increases
WMAPE site (Greedy)
WMAPE site (B&B)
WMAPE global
WMAPE
Days to delivery

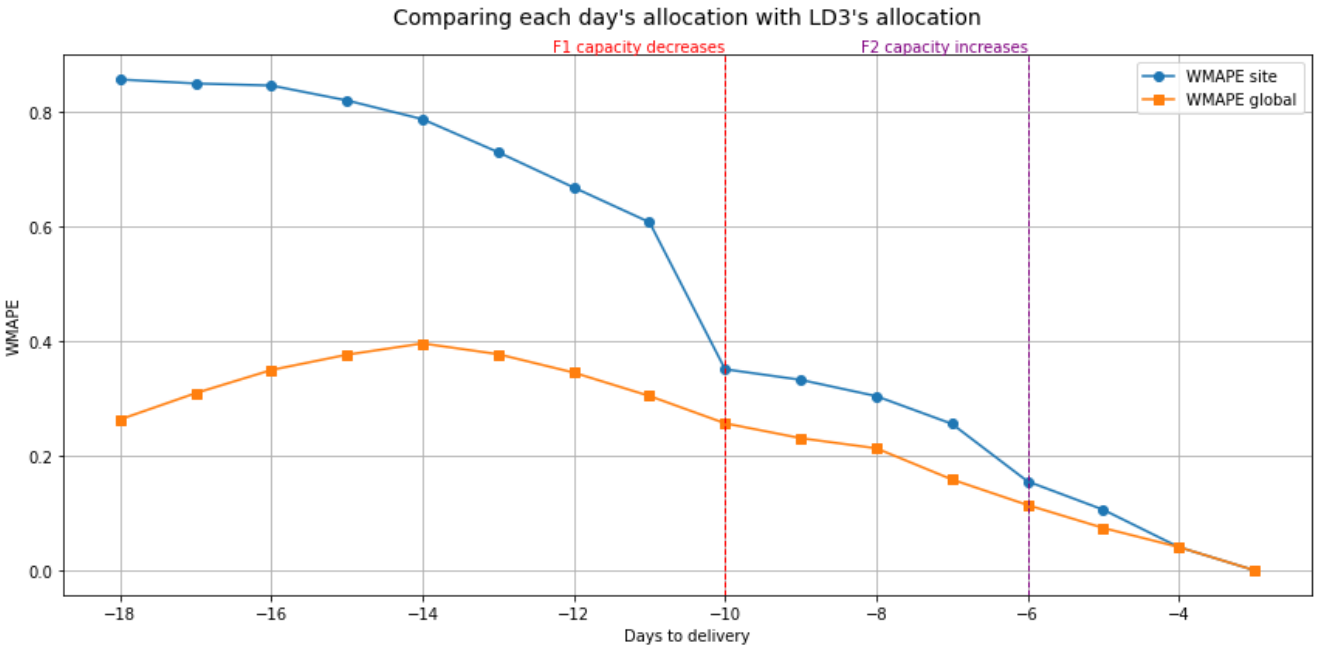



Figure 5.18. Temporal test with both capacity and order changes

Data for all above graphs can be found in Appendix II.

## 5.3 Managerial implications and recommendations

The results of this study have several important implications for managers and decision-makers in meal kit delivery:

- The superior performance of B&B demonstrates the value of implementing advanced optimization techniques in the box allocation process. Managers should consider integrating this method with their inventory management system to improve operational efficiency and reduce food waste, as smaller recipe differences between days lead to better inventory management and less food spoilage.
- While B&B consistently delivers optimal solutions in a short time on the simulated data, companies should apply it to real data to assess its effectiveness in more complex data structures. An instruction for its implementation in Python is provided in Appendix I to support this process.
- The temporal test highlights the importance of continuous optimization throughout the planning period. Managers should optimize decisions daily to ensure a smooth transition from soft to hard allocations, minimizing disruptions in production planning and ingredient procurement.
- The variation tests demonstrate the robustness of the proposed methods in handling capacity changes and order adjustments, preventing overproduction or underutilization of resources. Managers should prioritize flexible methods that can quickly adapt to changes, ensuring minimal impact on overall efficiency.

- The comparison between B&B and ID-based allocation method reveals the shortcomings of simpler, rule-based approaches. If ID-based method, or a similar approach, is currently used, managers should be cautious, especially when handling frequent order changes or capacity fluctuations.

- Companies should highlight improved allocation accuracy and waste reduction achievements in sustainability reporting, as this can enhance their reputation and potentially lead to increased customer loyalty and market share.

- One noteworthy weakness in the current operation is the company's method of generating new simulated orders every day. This approach introduces unnecessary variability in recipe distribution between allocation runs, potentially leading to increasing errors at each factory. A more effective strategy would be to maintain consistency by preserving some simulated orders from previous allocations. This could involve selectively retaining orders that closely match the desired recipe distribution and reallocating them to the same factories as before. New simulated orders would be generated only as needed to fill gaps or adjust the overall distribution. This approach will leverage information from previous runs, resulting in more stable allocations and reduced errors over time.

By implementing these recommendations, meal kit delivery companies can enhance operational efficiency and significantly reduce waste across their supply chain. The proper implementation of advanced optimization techniques serves as a powerful tool for achieving more sustainable and cost-effective operations.

# Chapter 6

# Conclusion

This dissertation has addressed the box allocation problem (BAP) faced by a leading meal kit delivery company in the UK. The study focused on developing effective approaches to minimize recipe differences between two days while respecting factory capacity and eligibility constraints. The literature review revealed that BAP shares similarities with some classic COPs including BPP and CVRP. Recent advancements in solving these problems, particularly using B&B, heuristic (LS), and metaheuristic (TS), suggested their potential for BAP. The application of these methods has led to several findings, which are explained in the next section.

## 6.1 Key findings and contributions

The key findings and contributions of this study are as follows:

- A mathematical model is built to capture the essential constraints and objectives of BAP. This model provides a solid foundation for understanding the problem and developing the solving strategy.
- The comparison of three proposed methods: B&B, ITPS, and TS provides insights into the strengths and limitations of each optimization approach:
  - B&B consistently delivered optimal solutions for all order quantities in under one minute, demonstrating its effectiveness in solving BAP, where achieving optimality is crucial for reducing food waste. Although exact methods are typically known for taking extended time to find optimal solutions, B&B in CBC solver is not a traditional exact method. Instead, it is a hybrid approach that combines various primal heuristics, such as FP and Diving Coefficient, to achieve optimal solutions in impressively short times.
  - Since ITPS and TS are developed specifically for BAP, both heuristics demonstrated significant improvements over initial greedy solutions. However, TS generally outperforms ITPS and nearly matches B&B in achieving optimal solutions. This result is supported by previous research, which suggests that combining metaheuristics with problem-specific knowledge and LS can deliver outstanding performance.
  - All proposed methods can handle the maximum quantity of 100,000 orders within 10 minutes. This ability is valuable for large-scale operations.

- The temporal fixed test confirms B&B can effectively minimize day-to-day variations and ensure a smooth transition from soft to hard allocations. This capability helps the company to achieve their objective of proxy optimization.
- B&B also shows remarkable adaptability to changes in factory capacities and order details. This finding highlights its robustness for real-world applications, where such variations are common. TS is also a viable method for handling capacity changes, though it is not as effective as B&B when addressing order changes.
- The comparison between B&B and ID-based allocation method highlights the distinct advantages of the optimization technique, particularly in the scenario of order changes.
- This study also revealed that WMAPE tends to decrease as order quantity increases, providing valuable insights into the relationship between order volume and allocation error in meal kit delivery.

These findings and contributions not only advance the academic understanding of BAP but also provide practical solutions and insights for meal kit delivery service as a whole. The developed model and methods can be applied to similar resource allocation problems in other domains, extending the impact of this research beyond its immediate context.

## 6.2 Limitations and future research directions

While this dissertation has effectively addressed BAP, several limitations remain, opening opportunities for future research:

- The current study focused on specific types of variations, namely changes in factory capacities and order details. Future research can explore a wider range of variations, such as seasonal demand fluctuations, or the introduction of new recipes and dietary restrictions.
- The current model optimizes allocation decisions for each day independently. Future studies could investigate dynamic optimization approaches that consider the entire planning horizon simultaneously, potentially leading to better long-term allocation strategies.
- The current model assumes deterministic demand. Future studies could incorporate stochastic elements to account for uncertainties in demand forecasts. Besides, it also assumes fixed recipe eligibility for each factory. Future studies could explore dynamic eligibility scenarios where factories can adapt to produce different recipes over time.
- While single-swap methods like ITPS and TS have been tested and shown effectiveness, other metaheuristics involving larger moves, such as GA or LNS, have not been explored. Future research could investigate their potential for BAP.
- This dissertation is based on simulated data. Although they match real-world characteristics, future research should apply optimization methods to the real data to validate their effectiveness in practice.
- This study's experiments use a data structure similar to the company's current practice. However, generating entirely new simulated orders each day introduces unnecessary variability and potential errors in recipe distribution across factories. Future research should

develop an algorithm that preserves suitable simulated orders between allocation runs, generating new ones only as needed.

- Although this study has solved up to 100,000 orders, even larger volumes may be relevant when the company expands its business. Future research could explore techniques such as machine learning, parallel optimization, or more effective commercial solvers to tackle ultra-large instances of BAP.

By addressing these limitations and exploring new research directions, scholars can further advance the optimization methods for meal kit delivery and related sectors, enabling more effective management of the complexities and uncertainties inherent in real-world operations.

# References


Aarts, E. and Lenstra, J.K. eds. (2003) *Local search in combinatorial optimization*. Princeton: Princeton University Press. Available at: https://press.princeton.edu/books/paperback/9780691115221/local-search-in-combinatorial-optimization (Accessed: 1 August 2024).

Achterberg, T., Koch, T. and Martin, A. (2005) 'Branching rules revisited', *Operations Research Letters*, 33(1), pp. 42-54. Available at: https://doi.org/10.1016/j.orl.2004.04.002.

Achterberg, T. (2009) 'SCIP: Solving Constraint Integer Programs', *Mathematics of Operations Research*, 34(1), pp. 1–41. Available at: https://doi.org/10.1007/s12532-008-0001-1.

Alvim, A.C.F., Glover, F.S., Ribeiro, C.C. and Aloise, D.J. (1999) 'Local Search for the Bin Packing Problem', *MIC'99 - III Metaheuristics International Conference*. Angra dos Reis, Brazil, 19-22 July. Available at: https://www.researchgate.net/publication/2286877_Local_Search_For_The_Bin_Packing_Problem (Accessed: 10 August 2024).

Alvim, A.C.F., Ribeiro, C.C., Glover, F., and Aloise, D.J. (2004) 'A hybrid improvement heuristic for the one-dimensional bin packing problem', *Journal of Heuristics*, 10(2), pp. 205-229. Available at: http://dx.doi.org/10.1023/B:HEUR.0000026267.44673.ed.

Baker, B.M., and Ayechew, M.A. (2003) 'A genetic algorithm for the vehicle routing problem', *Computers & Operations Research*, 30(5), pp. 787-800. Available at: http://dx.doi.org/10.1016/S0305-0548(02)00051-5.

Balinski, M.L., and Quandt, R.E. (1964) 'On an integer program for a delivery problem', *Operations Research*, 12(2), pp. 300-304. Available at: https://doi.org/10.1287/opre.12.2.300.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., and Vance, P. H. (1998) 'Branch-and-Price: Column Generation for Solving Huge Integer Programs', *Operations Research*, 46(3), pp. 316-329. Available at: https://doi.org/10.1287/opre.46.3.316.

Barnes, J.W. and Laguna, M. (1993) 'A tabu search experience in production scheduling', *Annals of Operations Research*, 41(3), pp. 139-156. Available at: https://doi.org/10.1007/BF02023072.

Battiti, R., and Tecchiolli, G. (1994) 'The reactive tabu search', *ORSA Journal on Computing*, 6(2), pp. 126-140. Available at: https://doi.org/10.1287/ijoc.6.2.126.

Berthold, T. (2006) *Primal heuristics for mixed integer programs*. Diploma thesis. Technische Universität Berlin. Available at: https://www.researchgate.net/publication/258846101_Primal_Heuristics_for_Mixed_Integer_Programs (Accessed: 1 August 2024).

Bullnheimer, B., Hartl, R.F. and Strauss, C. (1999) 'An improved ant system algorithm for the vehicle routing problem', *Annals of Operations Research*, 89, pp. 319-328. Available at: https://doi.org/10.1023/A:1018940026670.

Clausen, J. (1999) 'Branch and bound algorithms-principles and examples', *Department of Computer Science, University of Copenhagen*, pp. 1-30. Available at: https://imada.sdu.dk/~jbj/DM85/TSPtext.pdf (Accessed: 1 August 2024).

Coffman Jr, E.G., Garey, M.R., and Johnson, D.S. (1997) 'Approximation algorithms for bin packing: a survey', in Hochbaum, D.S. (ed.) *Approximation algorithms for NP-hard problems*. Boston: PWS Publishing Co., pp. 46-93. Available at: https://www.labri.fr/perso/eyraud/pmwiki/uploads/Main/approx-bin-packing.pdf (Accessed: 1 August 2024).

Contardo, C. and Martinelli, R. (2014) 'A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints', *Discrete Optimization*, 12, pp. 129-146. Available at: https://doi.org/10.1016/j.disopt.2014.03.001.

Cordeau, J., Gendreau, M., and Laporte, G. (1997) 'A tabu search heuristic for periodic and multi-depot vehicle routing problems', *Networks*, 30, pp. 105-119. Available at: https://doi.org/10.1002/(SICI)1097-0037(199709)30:2%3C105::AID-NET5%3E3.0.CO;2-G.

Costa, L., Contardo, C. and Desaulniers, G. (2019) 'Exact branch-price-and-cut algorithms for vehicle routing', *Transportation Science*, 53(4), pp. 946-985. Available at: https://doi.org/10.1287/trsc.2018.0878.

Croes, G.A. (1958) 'A method for solving traveling-salesman problems', *Operations Research*, 6(6), pp. 791-812. Available at: https://doi.org/10.1287/opre.6.6.791.

Dantzig, G.B. and Ramser, J.H. (1959) 'The truck dispatching problem', *Management Science*, 6(1), pp. 80-91. Available at: https://doi.org/10.1287/mnsc.6.1.80.

Delorme, M., Iori, M. and Martello, S. (2016) 'Bin packing and cutting stock problems: Mathematical models and exact algorithms', *European Journal of Operational Research*, 255(1), pp. 1-20. Available at: https://doi.org/10.1016/j.ejor.2016.04.030.

Díaz, J.A. and Fernández, E. (2001) 'A tabu search heuristic for the generalized assignment problem', *European Journal of Operational Research*, 132(1), pp. 22-38. Available at: https:

//doi.org/10.1016/S0377-2217(00)00108-9.

Adiba, I., Elhassania, M., Ahemd, H.A., and Abdelah, I. (2013) 'A Hybrid Ant Colony System for Green capacitated vehicle routing problem in sustainable transport', *Journal of Theoretical and Applied Information Technology*, 54(2), pp. 198-208. Available at: https://www.jatit.org/volumes/Vol54No2/1Vol54No2.pdf (Accessed: 1 August 2024).

Falkenauer, E. (1996) 'A hybrid grouping genetic algorithm for bin packing', *Journal of Heuristics*, 2(1), pp. 5-30. Available at: https://doi.org/10.1007/BF00226291.

Fischetti, M., Glover, F. and Lodi, A. (2005) 'The feasibility pump', *Mathematical Programming*, 104(1), pp. 91-104. Available at: https://doi.org/10.1007/s10107-004-0570-3.

Forrest, J. and Lougee-Heimer, R. (2005) 'CBC user guide', in *Emerging theory, methods, and applications*. INFORMS, pp. 257-277. Available at: https://doi.org/10.1287/educ.1053.0020.

Fukasawa, R., Longo, H., Lysgaard, J., de Aragão, M.P., Reis, M., Uchoa, E. and Werneck, R.F. (2006) 'Robust branch-and-cut-and-price for the capacitated vehicle routing problem', *Mathematical Programming*, 106(3), pp. 491-511. Available at: http://dx.doi.org/10.1007/s10107-005-0644-x.

GeeksforGeeks (2021) *How to Calculate Weighted MAPE in Excel?* Available at: https://www.geeksforgeeks.org/how-to-calculate-weighted-mape-in-excel/ (Accessed: 24 August 2024).

Gendreau, M. (2003) 'An introduction to tabu search', in Glover, F. and Kochenberger, G.A. (eds.) *Handbook of metaheuristics*. Boston: Springer, pp. 37-54. Available at: https://doi.org/10.1007/0-306-48056-5_2.

Gendreau, M., Hertz, A. and Laporte, G. (1994) 'A tabu search heuristic for the vehicle routing problem', *Management Science*, 40(10), pp. 1276-1290. Available at: http://dx.doi.org/10.1287/mnsc.40.10.1276.

Gendreau, M. and Potvin, J.-Y. eds. (2010) *Handbook of metaheuristics*. 2nd edn. New York: Springer. Available at: http://dx.doi.org/10.1007/978-1-4419-1665-5.

Glover, F. (1986) 'Future paths for integer programming and links to artificial intelligence', *Computers & Operations Research*, 13(5), pp. 533-549. Available at: https://doi.org/10.1016/0305-0548(86)90048-1.

Glover, F. (1989) 'Tabu search—part I', *ORSA Journal on Computing*, 1(3), pp. 190-206. Available at: https://courses.cs.umass.edu/cics521-cg/docs/tabu-glover-1.pdf (Accessed: 1 August 2024).

Glover, F., and Laguna, M. (1997) *Tabu search*. Boston: Springer Science & Business Media. Available at: http://dx.doi.org/10.1007/978-1-4615-6089-0.

Golden, B.L., Wasil, E.A., Kelly, J.P., and Chao, I.M. (1998) 'The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results', in Crainic, T.G. and Laporte, G. (eds) *Fleet management and logistics*. Boston, MA: Springer. Available at: https://doi.org/10.1007/978-1-4615-5755-5_2.

Hahn, P.M. and Grant, T.L. (1998) 'Lower bounds for the quadratic assignment problem based upon a dual formulation', *Operations Research*, 46(6), pp. 912-922. Available at: https://www.jstor.org/stable/222943.

Hanafi, S. and Freville, A. (1998) 'An efficient tabu search approach for the 0–1 multidimensional knapsack problem', *European Journal of Operational Research*, 106(2-3), pp. 659-675. Available at: https://doi.org/10.1016/S0377-2217(97)00296-8.

Helsgaun, K. (2000) 'An effective implementation of the Lin-Kernighan traveling salesman heuristic', *European Journal of Operational Research*, 126(1), pp. 106-130. Available at: https://doi.org/10.1016/S0377-2217(99)00284-2.

Johnson, D.S., Demers, A., Ullman, J.D., Garey, M.R. and Graham, R.L. (1974) 'Worst-case performance bounds for simple one-dimensional packing algorithms', *SIAM Journal on Computing*, 3(4), pp. 299-325. Available at: https://doi.org/10.1137/0203025.

Kernighan, B.W. and Lin, S. (1970) 'An efficient heuristic procedure for partitioning graphs', *The Bell System Technical Journal*, 49(2), pp. 291-307. Available at: https://doi.org/10.1002/j.1538-7305.1970.tb01770.x.

Kolesar, P.J. (1967) 'A branch and bound algorithm for the knapsack problem', *Management Science*, 13(9), pp. 723-735. Available at: http://dx.doi.org/10.1287/mnsc.13.9.723.

Lageweg, B.J., Lenstra, J.K. and Kan, A.R. (1977) 'Job-shop scheduling by implicit enumeration', *Management Science*, 24(4), pp. 441-450. Available at: https://doi.org/10.1287/MNSC.24.4.441.

Land, A.H. and Doig, A.G. (1960) 'An automatic method of solving discrete programming problems', *Econometrica*, 28(3), pp. 497-520. Available at: https://doi.org/10.2307/1910129.

Laporte, G. (2009) 'Fifty years of vehicle routing', *Transportation Science*, 43(4), pp. 408-416. Available at: https://www.jstor.org/stable/25769465.

Laporte, G., Mercure, H. and Nobert, Y. (1986) 'An exact algorithm for the asymmetrical capacitated vehicle routing problem', *Networks*, 16(1), pp. 33-46. Available at: https://doi.org/10.1002/net.3230160104.

Laporte, G. and Nobert, Y. (1987) 'Exact algorithms for the vehicle routing problem', *Annals of Discrete Mathematics*, 132, pp. 147-184. Available at: https://doi.org/10.1016/S0304-0208(08)73235-3.

Levine, J., and Ducatelle, F. (2004) 'Ant colony optimization and local search for bin packing and cutting stock problems', *Journal of the Operational Research Society*, 55(7), pp. 705-716. Available at: https://www.researchgate.net/publication/2563550_Ant_Colony_Optimisation_and_Local_Search_for_Bin_Packing_and_Cutting_Stock_Problems (Accessed: 10 August 2024).

Lin, S. (1965) 'Computer solutions of the traveling salesman problem', *Bell System Technical Journal*, 44(10), pp. 2245-2269. Available at: https://doi.org/10.1002/j.1538-7305.1965.tb04146.x.

Lin, S. and Kernighan, B.W. (1973) 'An effective heuristic algorithm for the traveling-salesman problem', *Operations Research*, 21(2), pp. 498-516. Available at: https://doi.org/10.1287/opre.21.2.498.

Little, J.D., Murty, K.G., Sweeney, D.W. and Karel, C. (1963) 'An algorithm for the traveling salesman problem', *Operations Research*, 11(6), pp. 972-989. Available at: https://doi.org/10.1287/opre.11.6.972.

Lodi, A., Martello, S. and Vigo, D. (1999) 'Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems', *INFORMS Journal on Computing*, 11(4), pp. 345-357. Available at: https://doi.org/10.1287/ijoc.11.4.345.

Marchand, H., Martin, A., Weismantel, R. and Wolsey, L. (2002) 'Cutting planes in integer and mixed integer programming', *Discrete Applied Mathematics*, 123(1-3), pp. 397-446. Available at: https://doi.org/10.1016/S0166-218X(01)00348-1.

Martello, S., and Toth, P. (1990) 'Lower bounds and reduction procedures for the bin packing problem', *Discrete Applied Mathematics*, 28(1), pp. 59-70. Available at: https://doi.org/10.1016/0166-218X(90)90094-S.

Nowicki, E. and Smutnicki, C. (1996) 'A fast taboo search algorithm for the job shop problem', *Management Science*, 42(6), pp. 797-813. Available at: https://doi.org/10.1287/mnsc.42.6.797.

O'Reilly (no date) *Operations Management: An Integrated Approach*. Available at: https://www.oreilly.com/library/view/operations-management-an/9781118122679/ch8-sec004.html (Accessed: 23 August 2024).

Osman, I.H. (1993)'Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem', *Annals of Operations Research*, 41(4), pp. 421-451. Available at: https://doi.org/10.1007/BF02023004.

Paulus, M.B. and Krause, A. (2023) 'Learning To Dive In Branch And Bound', *arXiv preprint arXiv:2301.09943*. Available at: https://doi.org/10.48550/arXiv.2301.09943.

Pecin, D., Pessoa, A., Poggi, M. and Uchoa, E. (2017) 'Improved branch-cut-and-price for capacitated vehicle routing', *Mathematical Programming Computation*, 9(1), pp. 61-100. Available at: https://doi.org/10.1007/s12532-016-0108-8.

Pisinger, D., and Ropke, S. (2007) 'A general heuristic for vehicle routing problems', *Computers & Operations Research*, 34(8), pp. 2403-2435. Available at: https://doi.org/10.1016/j.cor.2005.09.012.

Potts, C.N. and Van Wassenhove, L.N. (1985) 'A branch and bound algorithm for the total weighted tardiness problem', *Operations Research*, 33(2), pp. 363-377. Available at: https://doi.org/10.1287/opre.33.2.363.

Prins, C. (2004) 'A simple and effective evolutionary algorithm for the vehicle routing problem', *Computers & Operations Research*, 31(12), pp. 1985-2002. Available at: https://doi.org/10.1016/S0305-0548(03)00158-8.

Radcliffe-Brown, A.R. (1939) *Taboo*. Cambridge: Cambridge University Press. Available at: https://www.amazon.co.uk/Taboo-Frazer-Lecture-R-Radcliffe-Brown/dp/1107695791 (Accessed: 20 August 2024).

Rao, R.L. and Iyengar, S.S. (1994) 'Bin-packing by simulated annealing', *Computers & Mathematics with Applications*, 27(5), pp. 71-82. Available at: https://doi.org/10.1016/0898-1221(94)90077-9.

Rosenberger, J.M., Hwang, H.S., Pallerla, R.P., Yucel, A., Wilson, R.L. and Brungardt, E.G. (2005) 'The generalized weapon target assignment problem', in *10th International Command and Control Research and Technology Symposium: The Future of C2*. McLean, VA, 13-16 June. Available at: https://apps.dtic.mil/sti/pdfs/ADA463988.pdf (Accessed: 5 August 2024).

Savelsbergh, M.W. (1992) 'The vehicle routing problem with time windows: Minimizing route duration', *ORSA Journal on Computing*, 4(2), pp. 146-154. Available at: https://doi.org/10.1287/ijoc.4.2.146.

Scholl, A., Klein, R. and Jürgens, C. (1997) 'BISON: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem', *Computers & Operations Research*, 24(7), pp. 627-645. Available at: https://doi.org/10.1016/S0305-0548(96)00082-2.

Shaw, P. (1998) 'Using constraint programming and local search methods to solve vehicle routing problems', in Maher, M. and Puget, J.F. (eds.) *Principles and Practice of Constraint Programming — CP98*. Berlin: Springer, pp. 417-431. Available at: https://doi.org/10.1007/3-540-49481-2_30.

Sonuc, E., Sen, B., and Bayir, S. (2017) 'Solving Bin Packing Problem Using Simulated Annealing', in *Proceedings of 65th ISERD International Conference*. Mecca, Saudi Arabia, 23-24 January. Available at: https://www.worldresearchlibrary.org/up_proc/pdf/635-148739346283-85.pdf (Accessed: 2 August 2024).

Statista (2024) *Meal Kits in the UK - Overview*. Available at: https://www.statista.com/topics/8895/meal-kits-in-the-uk/ (Accessed: 7 August 2024).

Talbi, E.G. (2009) *Metaheuristics: From Design to Implementation*. Available at: https://typeset.io/papers/metaheuristics-from-design-to-implementation-1vqdfv8nk9 (Accessed: 1 August 2024).

Toth, P. and Vigo, D. (2002) *The vehicle routing problem*. Philadelphia: Society for Industrial and Applied Mathematics. Available at: https://doi.org/10.1137/1.9780898718515.

Toth, P. and Vigo, D. (2003) 'The granular tabu search and its application to the vehicle-routing problem', *INFORMS Journal on Computing*, 15(4), pp. 333-346. Available at: https://doi.org/10.1287/ijoc.15.4.333.24890.

Valério de Carvalho, J.M. (1999) 'Exact solution of bin-packing problems using column generation and branch-and-bound', *Annals of Operations Research*, 86, pp. 629-659. Available at: https://doi.org/10.1023/A:1018952112615.

Woop (no date) *Home*. Available at: https://woop.co.nz/ (Accessed: 7 August 2024).

Yu, B., Yang, Z.Z. and Yao, B. (2009) 'An improved ant colony optimization for vehicle routing problem', *European Journal of Operational Research*, 196(1), pp. 171-176. Available at: https://doi.org/10.1016/j.ejor.2008.02.028.

Zangl, G., Graf, T. and Al-Kinani, A. (2006) 'Proxy Modeling in Production Optimization', *Paper presented at the SPE Europec/EAGE Annual Conference and Exhibition*, Vienna, Austria, 12-15 June. Available at: https://doi.org/10.2118/100131-MS.

# Appendix I: User guide for solving BAP with CBC Solver

This guide provides a comprehensive approach to using CBC solver in PuLP to solve BAP in meal kit delivery.

## Prerequisites

Ensure the following libraries are installed:

```
pip install pulp pandas matplotlib
```

## Step 1: Import required libraries

Begin by importing the necessary Python libraries:

```
from pulp import *
import pandas as pd
import matplotlib.pyplot as plt
import time
```

## Step 2: Data preparation

Prepare data for the optimization problem. This typically involves:

1. Loading data from the source, such as Excel files, CSV files, or databases.
2. Processing the data into a format suitable for the optimization model

Example:

```
def prepare_data(file_path):
    # Load data
    data = pd.read_excel(file_path)  # or another method depending on data
        structure

    # Process data into the required format
    orders = process_orders(data)
    previous_allocation = process_previous_allocation(data)
    recipe_counts = calculate_recipe_counts(orders, previous_allocation)

    return orders, previous_allocation, recipe_counts
```

The *recipe_counts* is a crucial part of the data preparation. It represents the number of times each recipe is assigned to each factory based on the previous day's allocation. This information is used to calculate WMAPE site in the optimization model.

The structure of *recipe_counts* might look like this:

```
recipe_counts = {
    'F1': {'recipe1': 10, 'recipe2': 5, 'recipe3': 8},
    'F2': {'recipe1': 7, 'recipe2': 12, 'recipe4': 3},
    'F3': {'recipe2': 6, 'recipe3': 9, 'recipe5': 4}
}
```

In this example, each factory has a dictionary of recipes and their corresponding counts. For instance, there are 10 recipe 1, 5 recipe 2, and 8 recipe 3 in F1. This information helps the optimization model to minimize change in recipes between consecutive days, which is often desirable in production planning to maintain consistency.

## Step 3: Set up the optimization problem

Create a function to set up and solve the optimization problem:

```
def setup_and_solve_optimization(orders, previous_allocation,
    factory_capacities):
    # Create the model
    prob = LpProblem("Order_Allocation", LpMinimize)

    # Define decision variables
    # Binary variables for order allocation: 1 if order i is allocated to
       factory f, 0 otherwise
    x = LpVariable.dicts("allocation",
                         [(o['id'], f) for o in orders for f in
                            factory_capacities.keys()],
                         cat='Binary')

    # Add constraints
    # Ensure each order is allocated to exactly one factory
    for o in orders:
        prob += lpSum([x[o['id'], f] for f in factory_capacities.keys()]) == 1

    # Ensure orders are only allocated to eligible factories
    for o in orders:
        for f in factory_capacities.keys():
            if f not in o['eligible_factories']:
                prob += x[o['id'], f] == 0

    # Enforce factory capacity constraints
    for f, capacity in factory_capacities.items():
        if capacity != float('inf'):
            prob += lpSum([x[o['id'], f] for o in orders]) <= capacity

    # Set up variables for WMAPE site calculation
    recipe_counts = {}
    for f in factory_capacities.keys():
        for o in orders:
            for r in o['recipe_ids']:
```

```
                if r not in recipe_counts:
                    recipe_counts[r] = {}
                if f not in recipe_counts[r]:
                    recipe_counts[r][f] = 0
                recipe_counts[r][f] += x[o['id'], f]

    # Calculate absolute differences for WMAPE site
    abs_diffs = []
    for r in set(recipe_counts.keys()) | set(previous_allocation['recipe_counts'].keys()):
        for f in factory_capacities.keys():
            prev_count = previous_allocation['recipe_counts'].get(r, {}).get(f, 0)
            current_count = recipe_counts.get(r, {}).get(f, 0)

            pos_diff = LpVariable(f"pos_diff_{r}_{f}", lowBound=0)
            neg_diff = LpVariable(f"neg_diff_{r}_{f}", lowBound=0)

            prob += current_count - prev_count == pos_diff - neg_diff

            abs_diffs.append(pos_diff + neg_diff)

    # Set the objective to minimize the sum of absolute differences (WMAPE site numerator)
    prob += lpSum(abs_diffs)

    # Solve the problem
    prob.solve()

    return prob, x
```

## Step 4: Execute the optimization and verify results

After running the optimization, companies can verify the results by:

- Checking the optimization status and objective value to ensure an optimal solution is found.
- Reviewing the total orders allocated to each factory to ensure capacity constraints are met.

```
def run_optimization(file_path):
    # Prepare data
    orders, previous_allocation, recipe_counts = prepare_data(file_path)

    # Set up factory capacities
    factory_capacities = {
        'F1': int(0.25 * len(orders)),
        'F2': int(0.5 * len(orders)),
        'F3': float('inf')
    }

    # Run optimization
    prob, x = setup_and_solve_optimization(orders,
                                           {'allocation': previous_allocation,
                                            'recipe_counts': recipe_counts},
```

```
                                            factory_capacities)

    # Extract results
    optimal_allocation = {f: [] for f in factory_capacities.keys()}
    for o in orders:
        for f in factory_capacities.keys():
            if value(x[o['id'], f]) == 1:
                optimal_allocation[f].append(o['id'])

    return prob, optimal_allocation

# Example usage
file_path = 'data_file.xlsx'
prob, optimal_allocation = run_optimization(file_path)

print(f"Optimization status: {LpStatus[prob.status]}")
print(f"Objective value: {value(prob.objective)}")
print("\nOptimal allocation:")
for factory, order_ids in optimal_allocation.items():
    print(f"{factory}: {len(order_ids)} orders")
```

## Step 5: Analyze results

After running the optimization, perform an analysis of the results with some possible steps:

1. Calculate WMAPE site and global to evaluate the quality of the solution. This helps to understand how well the current allocation matches the previous day's allocation.

2. Track the solving time to evaluate the scalability of this method and identify potential bottlenecks, especially when dealing with large-scale instances.

3. Use visualization tools such as Matplotlib to aid in pattern recognition and communicate findings. Examples include bar charts for order allocations per factory, Venn diagrams for order distributions, and line graphs for tracking WMAPE trends over time.

4. Export both input and output to confirm eligibility constraints are satisfied, and send the data file to factories if necessary. Below is an example function to export orders of both days, the previous day's allocation, the current day's allocation, and the detailed WMAPE calculations to an Excel file:

```
def export_to_excel(orders_t_minus_1, orders_t, allocation_t_minus_1,
    allocation_t, wmape_site, wmape_global):
    workbook = Workbook()

    # Create sheets
    sheet_t_minus_1_orders = workbook.create_sheet("Day t-1 orders", 0)
    sheet_t_orders = workbook.create_sheet("Day t orders", 1)
    sheet_t_minus_1_allocation = workbook.create_sheet("Day t-1 allocation
        ", 2)
    sheet_optimal_allocation = workbook.create_sheet("Day t allocation",
        3)
    sheet_wmape = workbook.create_sheet("WMAPE", 4)
```

```
    # Populate Day t-1 orders sheet
    sheet_t_minus_1_orders.append(["Order ID", "Recipe IDs", "Is real", "
        Eligible factories"])
    for order in orders_t_minus_1:
        sheet_t_minus_1_orders.append([order['id'], ", ".join(map(str,
            order['recipe_ids'])), order['is_real'], ", ".join(order['
            eligible_factories'])])

    # Populate Day t orders sheet
    sheet_t_orders.append(["Order ID", "Recipe IDs", "Is real", "Eligible
        factories"])
    for order in orders_t:
        sheet_t_orders.append([order['id'], ", ".join(map(str, order['
            recipe_ids'])), order['is_real'], ", ".join(order['
            eligible_factories'])])

    # Function to write allocation data
    def write_allocation(sheet, allocation):
        sheet.append(["Factory", "Allocated orders"])
        for factory, orders in allocation.items():
            order_ids = [str(order['id']) if isinstance(order, dict) else
                str(order) for order in orders]
            sheet.append([factory, ", ".join(order_ids)])

    # Populate Day t-1 allocation sheet
    write_allocation(sheet_t_minus_1_allocation, allocation_t_minus_1)

    # Populate Day t allocation sheet
    write_allocation(sheet_t_allocation, allocation_t)

    # Populate WMAPE sheet
    sheet_wmape.append(["WMAPE site", wmape_site])
    sheet_wmape.append(["WMAPE global", wmape_global])

    # Save the workbook
    workbook.save("allocation_results.xlsx")
```

By following this guide, CBC solver in PuLP library can be applied to solve BAP. Each step should be adapted to the specific requirements and data structure of reality.

# Appendix II: B&B variation test results

Table 1. WMAPE when F1's capacity changes

| Day | Real orders | Site (B&B) | Site (Greedy) | Global | Site vs LD3 | Global vs LD3 |
|---|---|---|---|---|---|---|
| -18 | 10% | N/A | N/A | N/A | 0.4751 | 0.0648 |
| -17 | 16% | 0.0756 | 0.0756 | 0.0669 | 0.4769 | 0.0529 |
| -16 | 22% | 0.0714 | 0.0714 | 0.0591 | 0.4829 | 0.0517 |
| -15 | 28% | 0.0686 | 0.0686 | 0.0535 | 0.4801 | 0.0467 |
| -14 | 34% | 0.0662 | 0.0662 | 0.0503 | 0.4772 | 0.0426 |
| -13 | 40% | 0.0518 | 0.0518 | 0.0448 | 0.4823 | 0.0441 |
| -12 | 46% | 0.0518 | 0.0518 | 0.0466 | 0.4788 | 0.0436 |
| -11 | 52% | 0.0542 | 0.0542 | 0.0482 | 0.4809 | 0.0411 |
| -10 | 58% | 0.3395 | 0.5830 | 0.0408 | 0.1500 | 0.0391 |
| -9 | 64% | 0.0433 | 0.0692 | 0.0428 | 0.1161 | 0.0399 |
| -8 | 70% | 0.0388 | 0.0683 | 0.0388 | 0.0881 | 0.0341 |
| -7 | 76% | 0.0365 | 0.0633 | 0.0365 | 0.0692 | 0.0351 |
| -6 | 82% | 0.0345 | 0.0680 | 0.0345 | 0.0426 | 0.0242 |
| -5 | 88% | 0.0237 | 0.0593 | 0.0237 | 0.0310 | 0.0229 |
| -4 | 94% | 0.0221 | 0.0597 | 0.0221 | 0.0145 | 0.0145 |
| -3 | 100% | 0.0145 | 0.0559 | 0.0145 | 0 | 0 |

Table 2. Changed and deleted orders' quantity

| Days to delivery | Changed orders | Deleted orders |
|---|---|---|
| -17 | 285 | 50 |
| -16 | 456 | 80 |
| -15 | 627 | 110 |
| -14 | 798 | 140 |
| -13 | 969 | 170 |
| -12 | 1140 | 200 |
| -11 | 1311 | 230 |
| -10 | 1482 | 260 |
| -9 | 1653 | 290 |
| -8 | 1824 | 320 |
| -7 | 1995 | 350 |
| -6 | 2166 | 380 |
| -5 | 2337 | 410 |
| -4 | 2508 | 440 |
| -3 | 2679 | 470 |

Table 3. WMAPE with and without order changes

| Days to delivery | Site (with) | Global (with) | Site (without) | Global (without) |
|---|---|---|---|---|
| -17 | 0.072 | 0.070 | 0.064 | 0.063 |
| -16 | 0.070 | 0.067 | 0.063 | 0.062 |
| -15 | 0.069 | 0.068 | 0.057 | 0.056 |
| -14 | 0.074 | 0.074 | 0.054 | 0.053 |
| -13 | 0.088 | 0.088 | 0.050 | 0.050 |
| -12 | 0.083 | 0.083 | 0.054 | 0.054 |
| -11 | 0.098 | 0.098 | 0.045 | 0.045 |
| -10 | 0.092 | 0.092 | 0.047 | 0.047 |
| -9 | 0.080 | 0.080 | 0.038 | 0.038 |
| -8 | 0.073 | 0.073 | 0.041 | 0.041 |
| -7 | 0.062 | 0.062 | 0.039 | 0.039 |
| -6 | 0.060 | 0.060 | 0.033 | 0.033 |
| -5 | 0.060 | 0.060 | 0.030 | 0.030 |
| -4 | 0.045 | 0.045 | 0.022 | 0.022 |
| -3 | 0.041 | 0.041 | 0.014 | 0.014 |

Table 4. WMAPE comparison between B&B and ID-based method (Without order changes)

| Days to delivery | Site (B&B) | Site (ID-based) | Global |
|---|---|---|---|
| -17 | 0.070 | 0.076 | 0.067 |
| -16 | 0.061 | 0.073 | 0.059 |
| -15 | 0.056 | 0.072 | 0.053 |
| -14 | 0.051 | 0.067 | 0.050 |
| -13 | 0.045 | 0.051 | 0.045 |
| -12 | 0.047 | 0.051 | 0.047 |
| -11 | 0.048 | 0.056 | 0.048 |
| -10 | 0.041 | 0.050 | 0.041 |
| -9 | 0.043 | 0.049 | 0.043 |
| -8 | 0.039 | 0.047 | 0.039 |
| -7 | 0.037 | 0.045 | 0.037 |
| -6 | 0.035 | 0.041 | 0.035 |
| -5 | 0.024 | 0.024 | 0.024 |
| -4 | 0.022 | 0.022 | 0.022 |
| -3 | 0.014 | 0.014 | 0.014 |

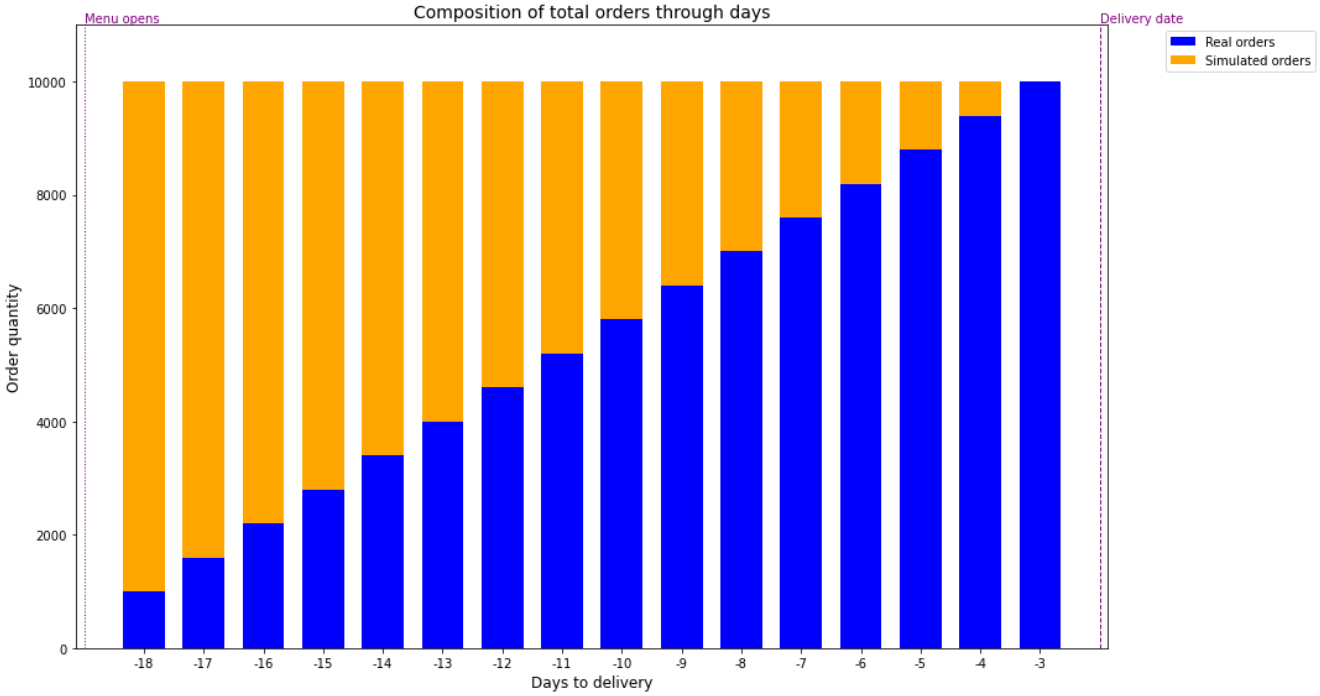
Table 5. WMAPE comparison between B&B and ID-based method (With order changes)

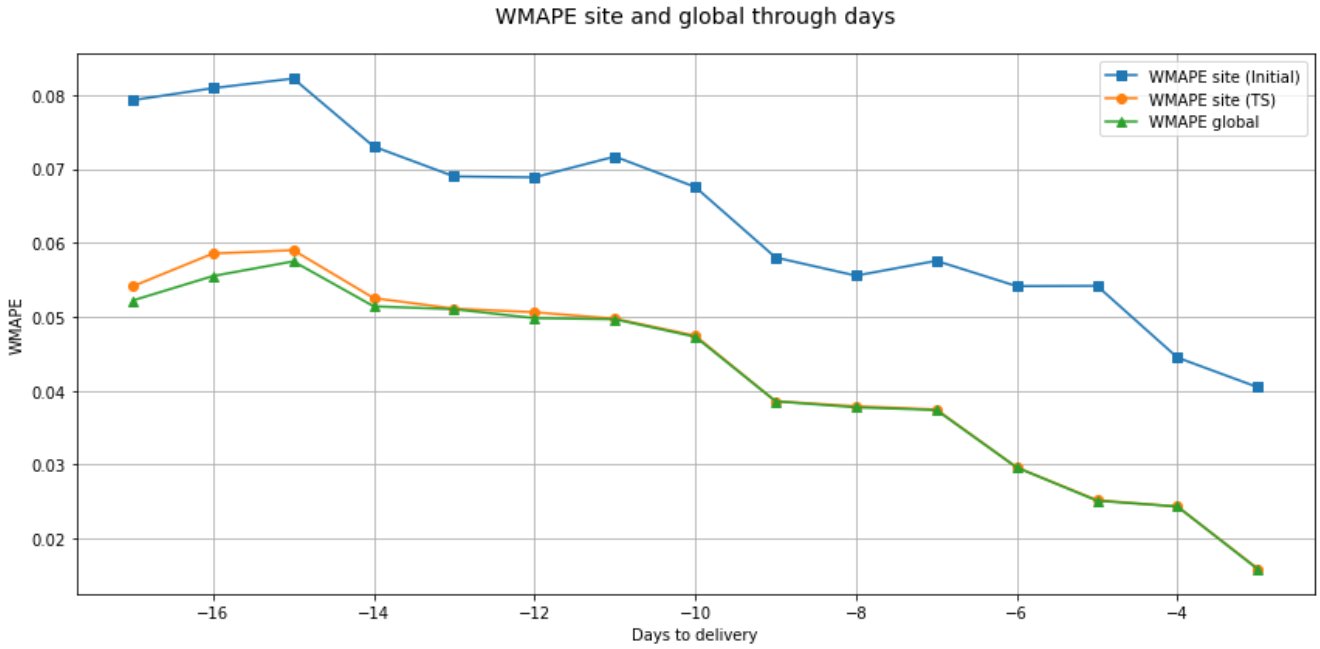
| Days to delivery | Site (B&B) | Site (ID-based) | Global |
|---|---|---|---|
| -17 | 0.076 | 0.089 | 0.070 |
| -16 | 0.071 | 0.099 | 0.067 |
| -15 | 0.070 | 0.092 | 0.068 |
| -14 | 0.074 | 0.100 | 0.074 |
| -13 | 0.089 | 0.117 | 0.088 |
| -12 | 0.083 | 0.115 | 0.083 |
| -11 | 0.098 | 0.119 | 0.098 |
| -10 | 0.092 | 0.118 | 0.092 |
| -9 | 0.080 | 0.113 | 0.080 |
| -8 | 0.073 | 0.108 | 0.073 |
| -7 | 0.062 | 0.100 | 0.062 |
| -6 | 0.060 | 0.107 | 0.060 |
| -5 | 0.060 | 0.101 | 0.060 |
| -4 | 0.045 | 0.085 | 0.045 |
| -3 | 0.041 | 0.083 | 0.041 |

Table 6. WMAPE when both capacity and order change

| **Day** | **Real orders** | **Site (B&B)** | **Site (Greedy)** | **Global** | **Site vs LD3** | **Global vs LD3** |
|---|---|---|---|---|---|---|
| -18 | 10% | N/A | N/A | N/A | 0.8573 | 0.2635 |
| -17 | 16% | 0.0880 | 0.0880 | 0.0698 | 0.8504 | 0.3095 |
| -16 | 22% | 0.0923 | 0.0923 | 0.0670 | 0.8468 | 0.3499 |
| -15 | 28% | 0.0804 | 0.1017 | 0.0680 | 0.8208 | 0.3766 |
| -14 | 34% | 0.0816 | 0.1156 | 0.0735 | 0.7879 | 0.3963 |
| -13 | 40% | 0.0931 | 0.1552 | 0.0881 | 0.7302 | 0.3775 |
| -12 | 46% | 0.0851 | 0.1424 | 0.0830 | 0.6686 | 0.3453 |
| -11 | 52% | 0.0978 | 0.1646 | 0.0976 | 0.6080 | 0.3045 |
| -10 | 58% | 0.3059 | 0.5270 | 0.0918 | 0.3512 | 0.2567 |
| -9 | 64% | 0.0797 | 0.1206 | 0.0797 | 0.3327 | 0.2308 |
| -8 | 70% | 0.0727 | 0.1208 | 0.0727 | 0.3042 | 0.2131 |
| -7 | 76% | 0.0621 | 0.1166 | 0.0621 | 0.2558 | 0.1586 |
| -6 | 82% | 0.0700 | 0.2714 | 0.0595 | 0.1553 | 0.1140 |
| -5 | 88% | 0.0597 | 0.1145 | 0.0597 | 0.1058 | 0.0740 |
| -4 | 94% | 0.0452 | 0.1124 | 0.0452 | 0.0406 | 0.0406 |
| -3 | 100% | 0.0406 | 0.1141 | 0.0406 | 0 | 0 |

# Appendix III: TS temporal test results



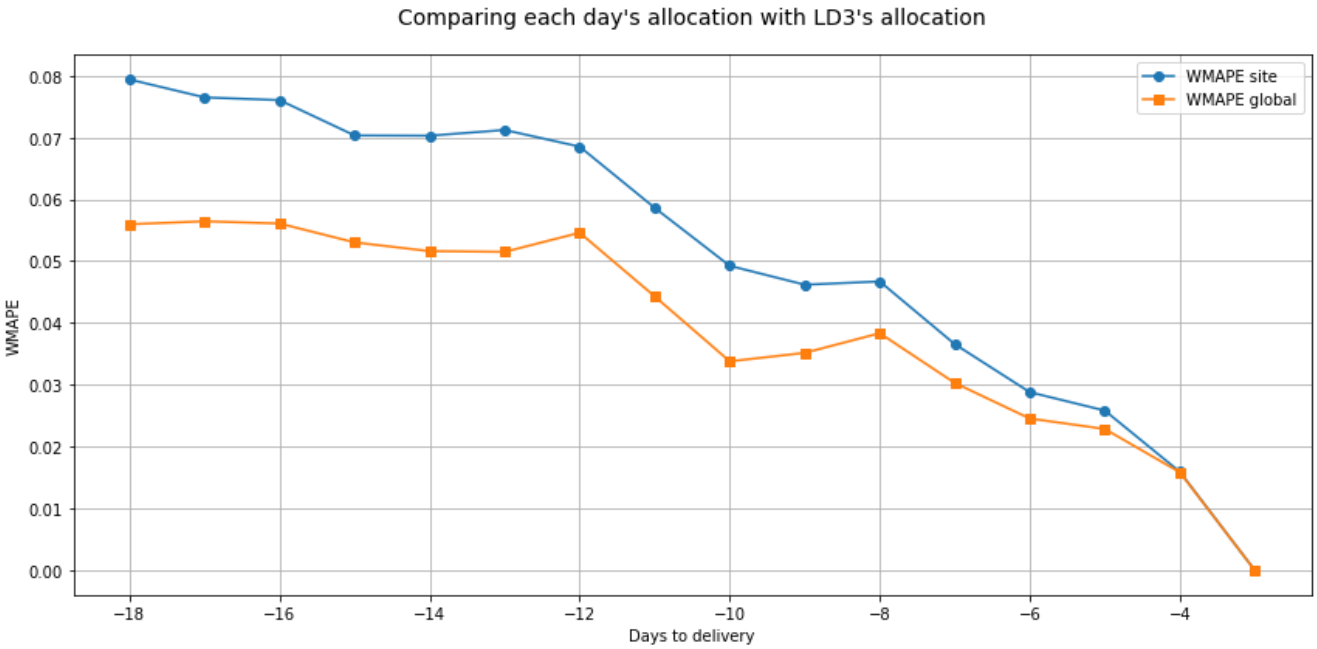
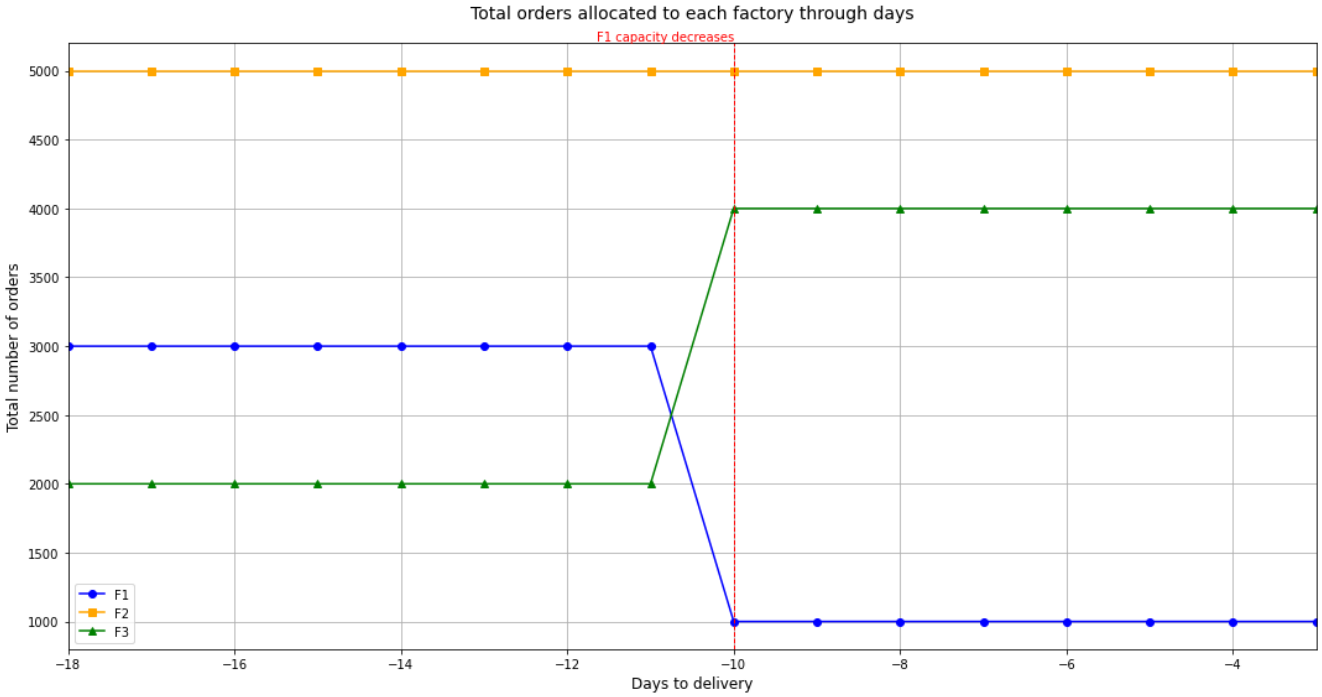
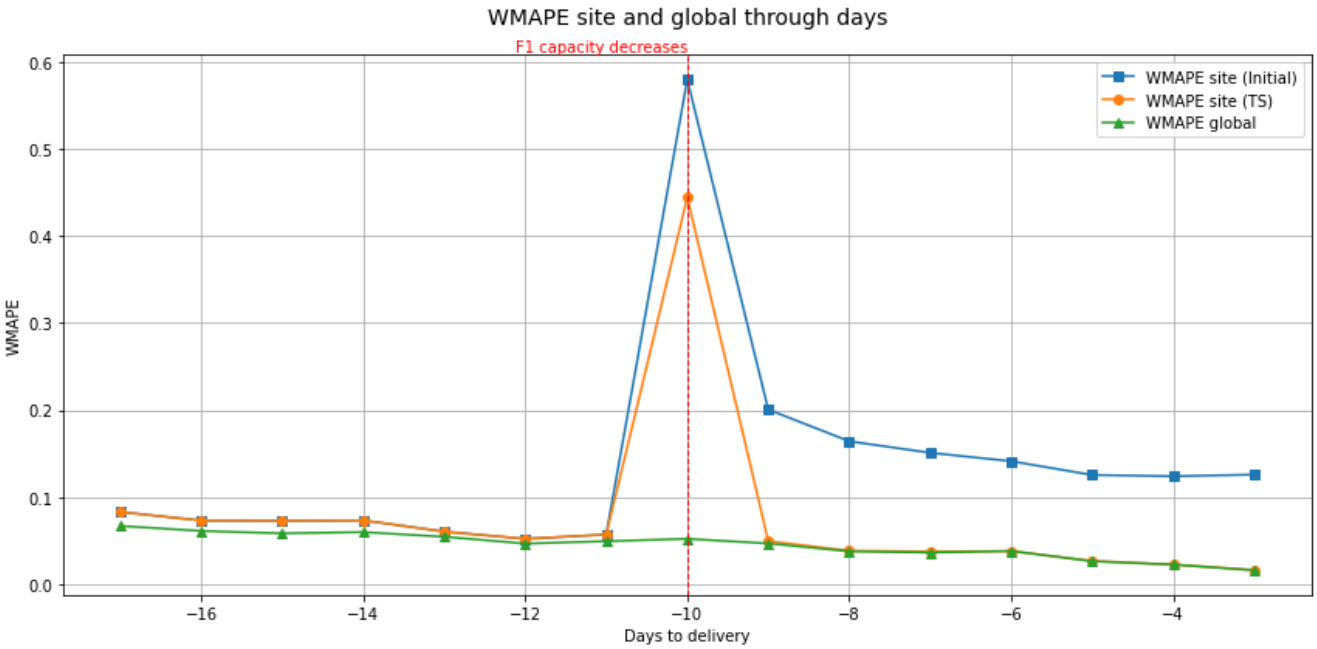
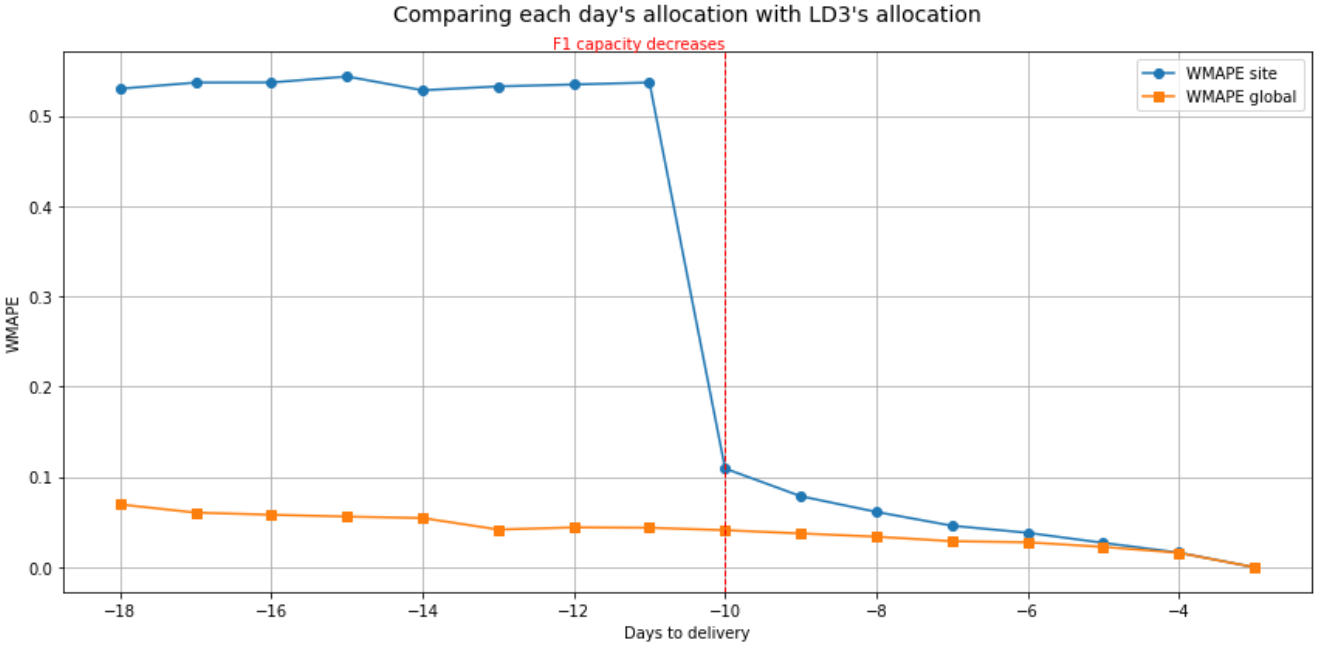



Figure 1. Temporal fixed test of TS




WMAPE site and global through days
F1 capacity decreases
WMAPE site (Initial)
WMAPE site (TS)
WMAPE global
WMAPE
Days to delivery


Comparing each day's allocation with LD3's allocation
F1 capacity decreases
WMAPE site
WMAPE global
WMAPE
Days to delivery

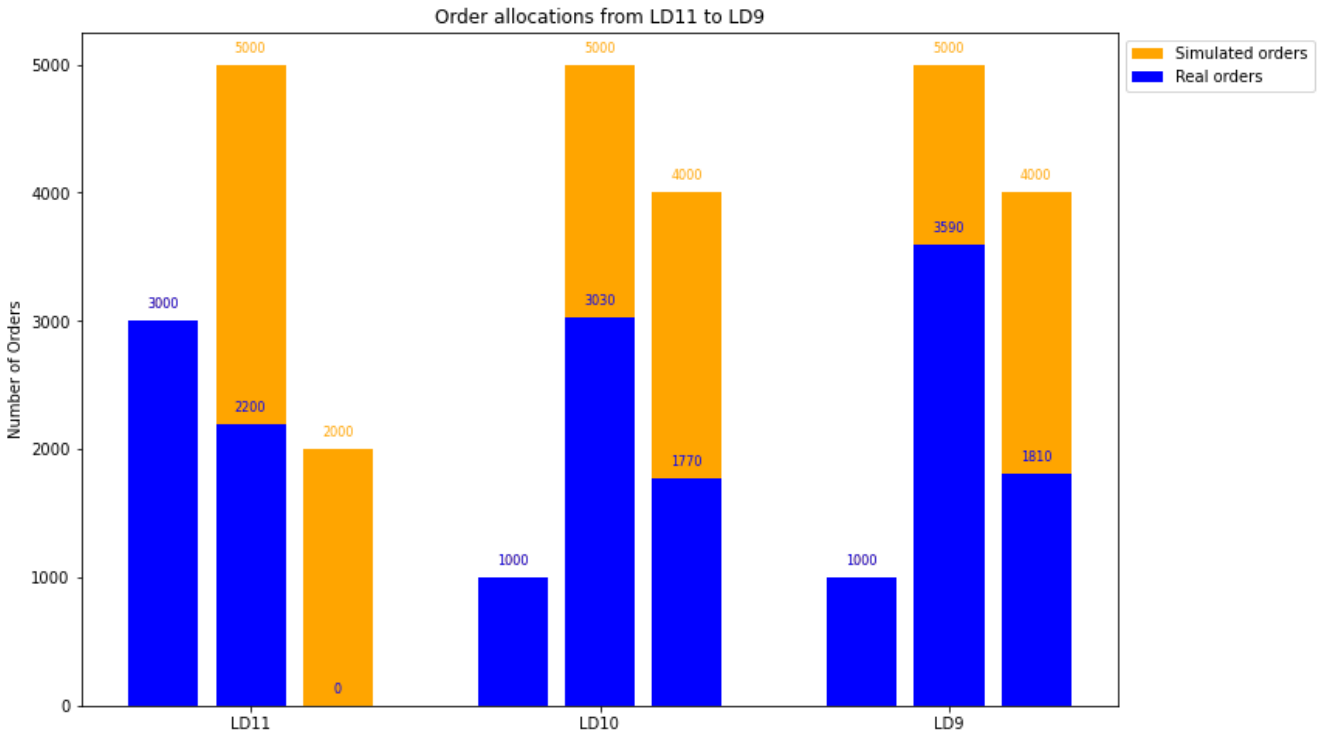
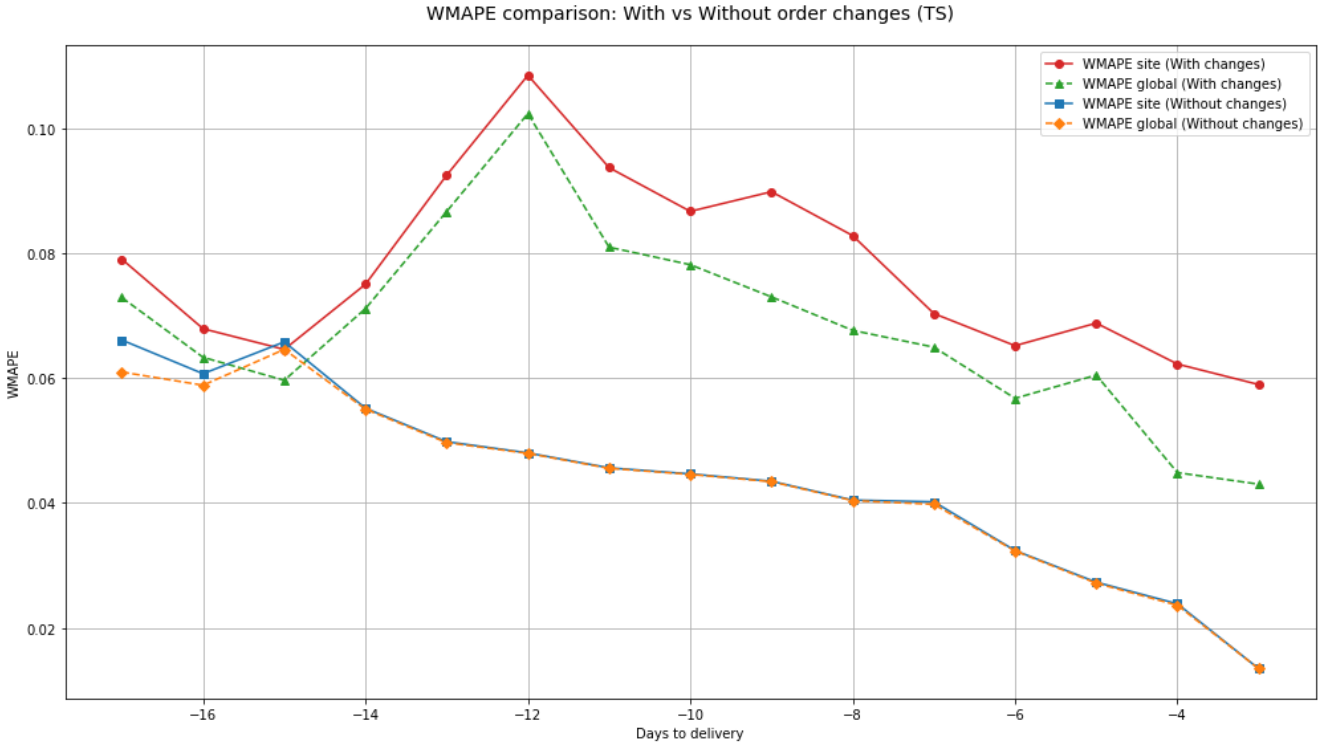
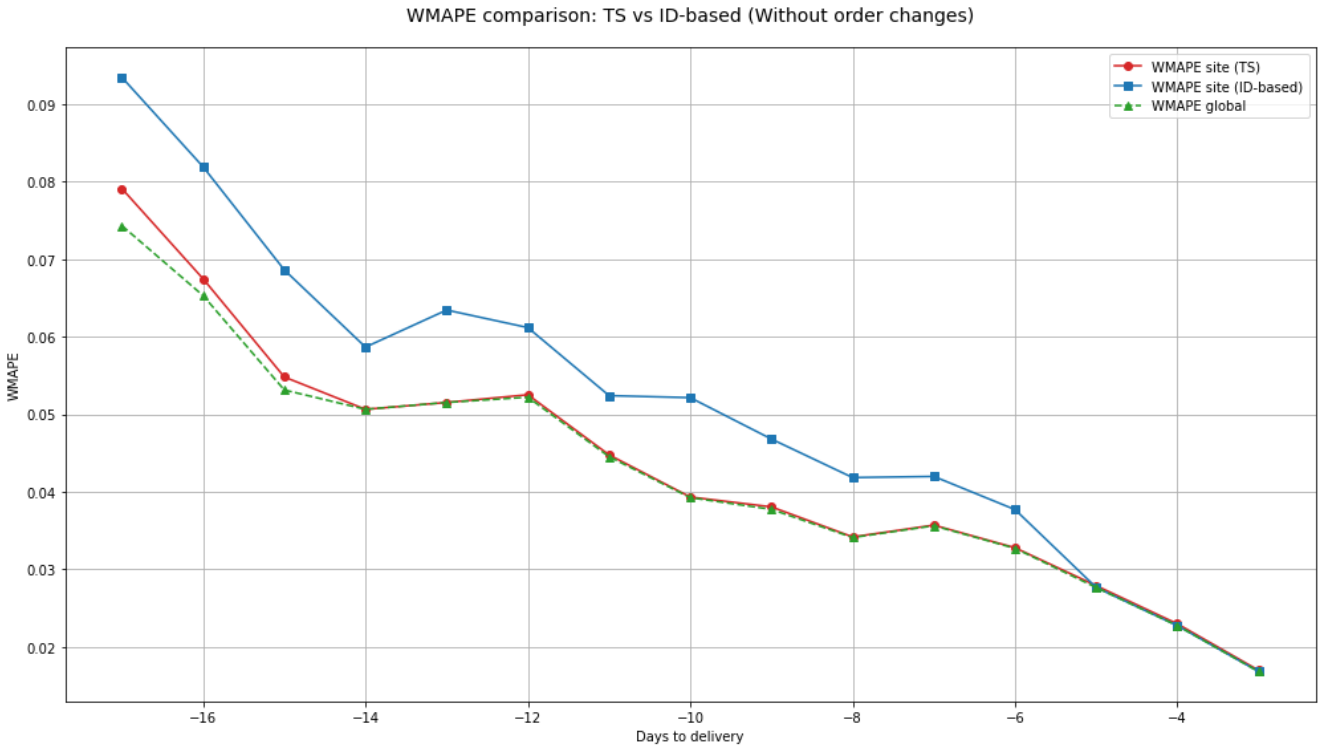



Figure 2. TS when F1's capacity changes





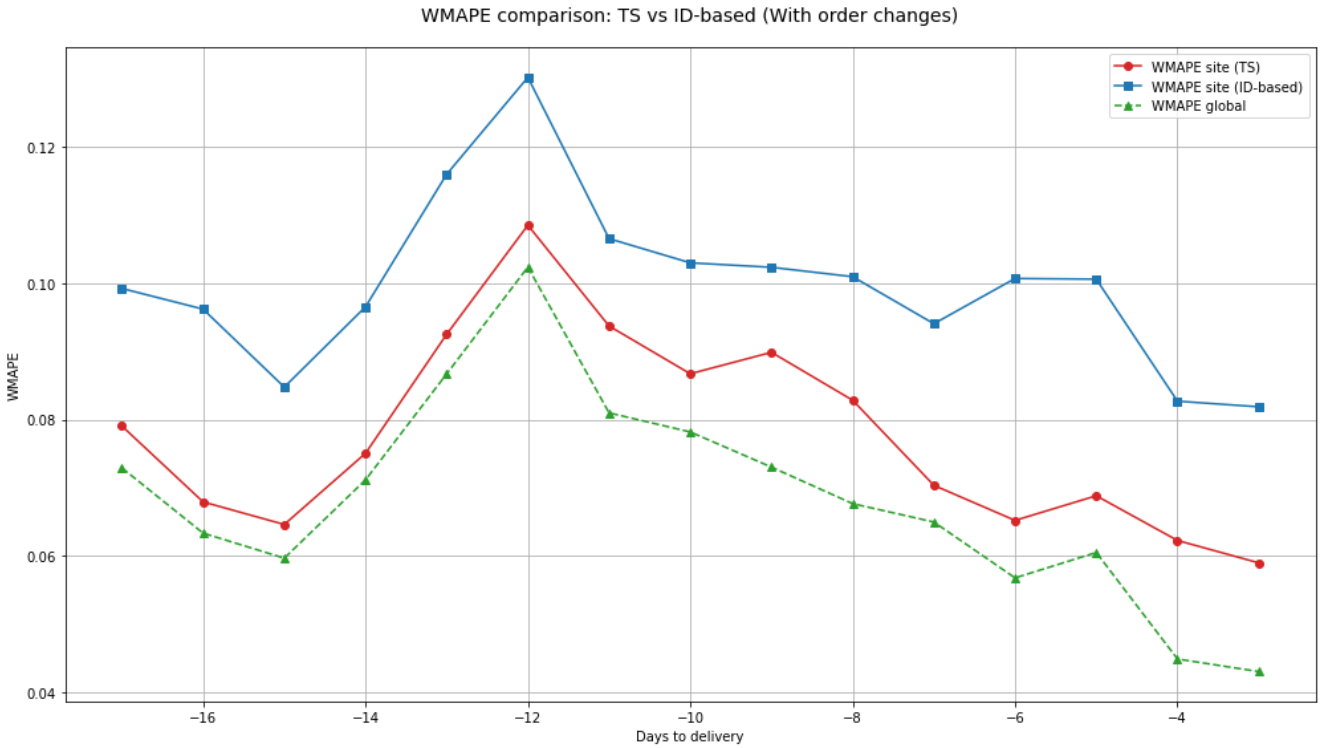
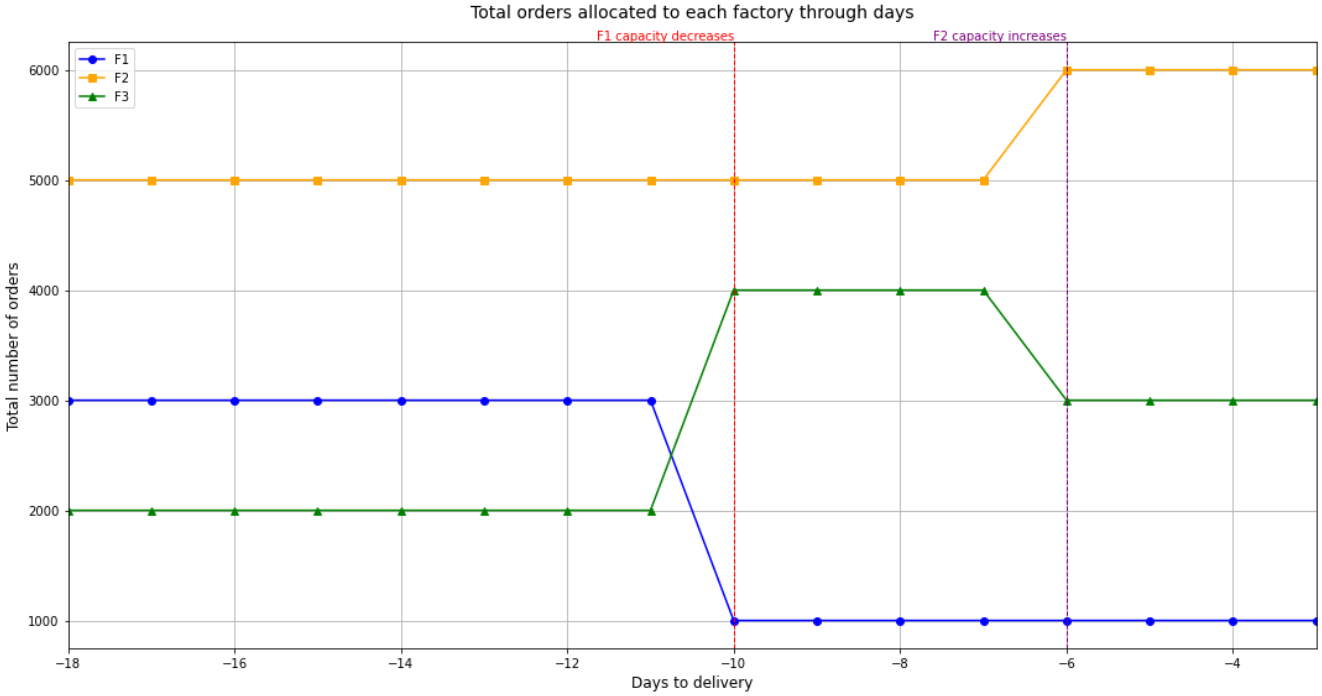
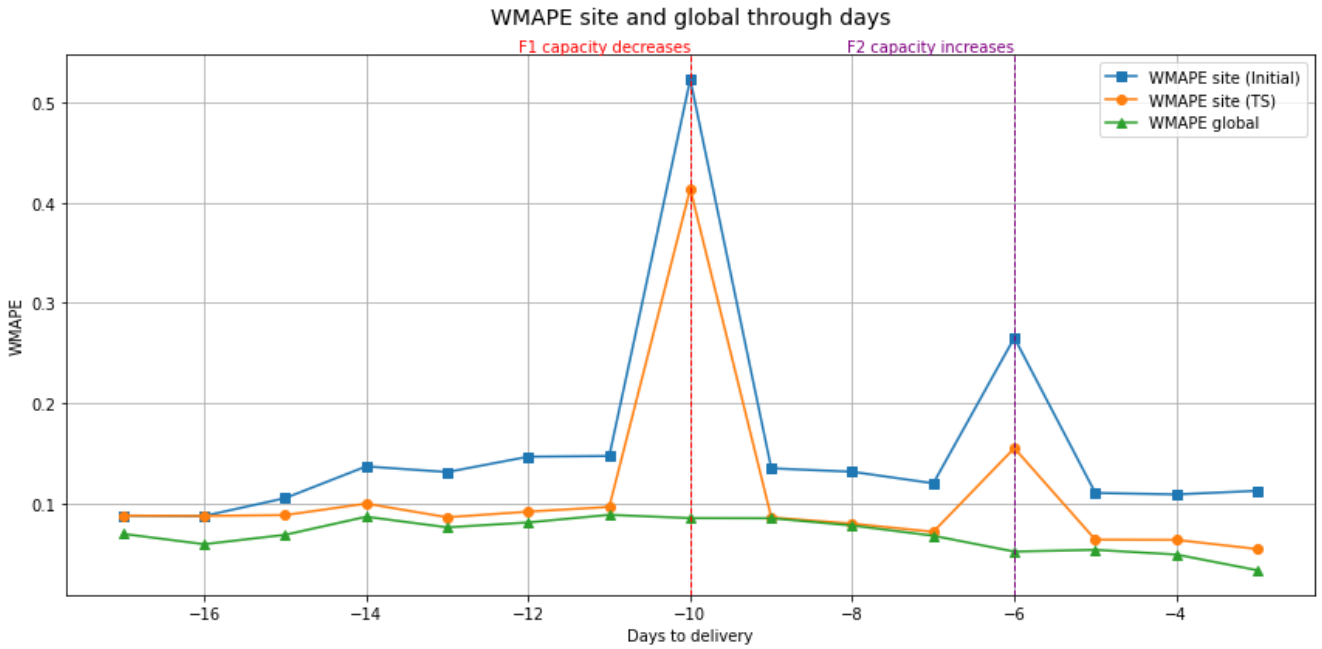




Figure 3. TS when orders change


Total orders allocated to each factory through days
F1 capacity decreases
F2 capacity increases
F1
F2
F3
Total number of orders
Days to delivery


WMAPE site and global through days
F1 capacity decreases
F2 capacity increases
WMAPE site (Initial)
WMAPE site (TS)
WMAPE global
WMAPE
Days to delivery

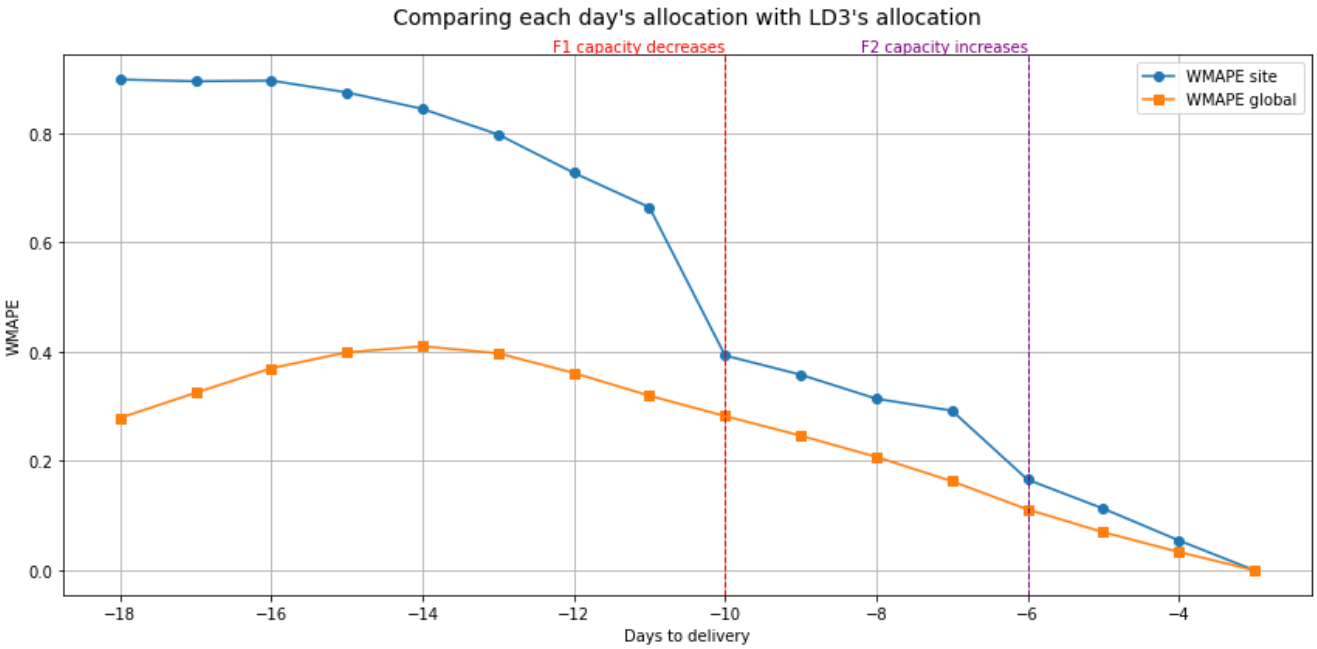



Figure 4. TS when both capacity and order change