Politecnico di Torino

Corso di Laurea Magistrale in

data science and engineering

Tesi di Laurea Magistrale

Bilevel Stochastic Optimization in Decentralized Assembly-To-Order Supply Chain

Relatore:
Edoardo Fadda

Candidato:
MINH TRIET NGO

Anno Accademico 2024/2025

Acknowledgment

First and foremost, I would like to express my deepest gratitude to my supervisor, professor Edoardo Fadda, for their invaluable guidance, patience, and encouragement throughout not only the writing of my thesis but also my academic journey at Politecnico di Torino. His expertise, insightful feedback, unwavering support and the trust he gave me to freely explore the topic were instrumental in shaping this work.

My appreciation extends to the faculty members and professors at DAUIN and DISMA, whose lectures, discussions, and mentorship have enriched my academic experience.

Contents

Chapter 1

Background

1.1 Introduction

The Assemble-to-Order (ATO) problem is a production and inventory management challenge commonly encountered in industries where products are assembled from various components or subassemblies. In an ATO environment, a company has a stock of components and assembles the final products only when customer orders are received. This approach balances customization (ince products are constructed based on specific customer requirements and efficiency since the company does not need to stock large quantities of finished goods. There has been a rich source of literature regarding the problem that deals with a variety of versions of the ATO model. However, the main focus of the mainstream literature is to prove theoretically some certain properties of the model and design control policies to optimize predefined metrics. This approach isolates the assembler as the sole rational decision maker of the process and ignores the complexity and the contractual nature of supply chain coordination.The supplier cannot force the assembler to receive more components than the ordered amount but in return, there is not a strict compliance regime to force the supplier to deliver the desirable amount of components to the assembler due to objective and subjective factor Kok, de and Graves [2003]. Therefore, it is necessary to create incentives by sharing information on the uncertainty of demand so that each actor in the supply chain behaves predictably.

1.2 Literature review

The general formulation of the ATO problem with multiple products over multiple periods has not been analytically solved since it was first mentioned in Song and Zipkin [2003]. Some specific cases have been solved, including the single product problem by Rosling [1989] and the single period problem by Song and Zipkin [2003].It is important to notice the general formulation in Song and Zipkin [2003] assume no procurement lead time and backlogged demand is merged with current demand which means that no priority is given to the back ordered demand. Here is the formulation of the problem:

$$
\begin{aligned}
\text{(P)} \quad & \text{minimize} \quad \mathbb{E}\left\{\sum_{t=0}^{T}\left[c(y_t - x_t) + hx_{t+1} + pw_{t+1}\right]\right\} \\
& \text{subject to} \\
& x_{t+1} = y_t - Az_t, \quad t = 0, \ldots, T, \\
& w_{t+1} = w_t + d_t - z_t, \quad t = 0, \ldots, T, \\
& x_t, w_t, z_t \geq 0, \quad t = 0, \ldots, T, \\
& y_t \geq x_t, \quad t = 0, \ldots, T.
\end{aligned}
$$

where:

- $c(y_t - x_t)$: **Ordering cost** for components, where $y_t - x_t$ is the order quantity in period t.
- hx_{t+1}: **Holding cost** for excess inventory at the end of period $t+1$.
- pw_{t+1}: **Backorder cost** for unmet demand at the end of period $t+1$.
- x_{t+1}: Net inventory of components at the start of period $t+1$.
- Az_t: Components consumed to assemble products in period t (where A is the bill-of-materials matrix).
- w_{t+1}: Backlogged demand at the end of period $t+1$.
- d_t: Random demand for products in period t.
- z_t: Production (assembly) of products in period t.

The mathematical program above consists of two processes. The first one is components procurement process in which the assembler (or the manufacturer as mentioned in the majority of the literature. Here we use the term the assembler to underline their relationship with the suppliers that would be later explained in this thesis). The second process is the components allocation process in which the assembler makes a decision to distribute the procured components into the production of each end-item once the demand is realized. We cannot see clearly the distinction between these two processes in the above mathematical formulation. However, there are papers that tackle the same problem with different specifications. For example, there are papers dealing with the ATO model with positive lead time. In this case, it is more convenient to model the problem by separating the two process and deal with them once at a time. An early paper that models positive lead time into the ATO problem is Agrawal and Cohen [2001], in which the author consignment policy in conjunction with a fair shares allocation scheme.

We can also simplify the model by removing the backlogged demand and consider a lost-sale model. Gerchak and Henig [1989] study a lost-sales model with stationary demand and linear order cost. Here is the formulation:

$$\text{(Lost Sales)} \quad \text{minimize} \quad \mathbb{E}\left\{\sum_{t=0}^{T}\left[c(y_t - x_t) + hx_{t+1} - rz_t\right]\right\},$$

where:

- $c(y_t - x_t)$: **Ordering cost** for components.
- hx_{t+1}: **Holding cost** for excess inventory.
- rz_t: **Revenue** from selling z_t units of products (negative cost).

Constraints

1. **Inventory balance**:

$$x_{t+1} = y_t - Az_t, \quad t = 0, \ldots, T.$$

2. **Demand fulfillment** (lost sales if unmet):

$$z_t \leq d_t, \quad t = 0, \ldots, T.$$

3. **Non-negativity**:

$$x_t, z_t \geq 0, \quad y_t \geq x_t, \quad t = 0, \ldots, T.$$

Key Differences from Backlog Model

- **No backorders**: Unmet demand $(d_t - z_t)$ is lost (no w_t term).
- **Revenue term**: Objective includes $-rz_t$ to reflect revenue from sales.

The authors proved that above formulation is actually can be solved equivalently by obtaining the single-period solution. In other words, the procurement policy is a base-stock order policy in which at the beginning of each period the assembler decides the amount of components to be procured and once the demand is realized a linear program is run to solve the allocation problem. This allocation policy is myopic which means that it does not take into consideration what might happen in the next periods but the authors has shown that under stationary demand, the base-stock order policy in conjunction with the myopic-allocation policy is optimal. However, it is noted that the paper only prove which policy would optimal profit under the given settings without providing a clear way to actually compute the solution and obtain a numerical result.

Based on the lost-sales model in Gerchak and Henig [1989], a line of research using the stochastic programming technique was first established by Brandimarte et al. [2021] regarding a one period model formulated as a two-stage stochastic problem in which the problem is modeled and can be solved using a solver for mathematical optimization. In the paper, the authors computes the numerical result of the base-stock policy following the approach of sample average approximation (SAA) and an allocation policy is not specified. Here is the formulation used in the paper:

$$\begin{aligned}
\max \quad & -\sum_{i=1}^{I} C_i x_i + \sum_{s=1}^{S} \pi^s \left(\sum_{j=1}^{J} P_j y_j^s \right) \\
\text{s.t.} \quad & \sum_{i=1}^{I} T_{im} x_i \leq L_m \quad \forall m \in \mathcal{M} \\
& y_j^s \leq d_j^s \quad \forall j \in \mathcal{J}, s \in \mathcal{S} \\
& \sum_{j=1}^{J} G_{ij} y_j^s \leq x_i \quad \forall i \in \mathcal{I}, s \in \mathcal{S} \\
& y_j^s, x_i \geq 0
\end{aligned}$$

where

- $\mathcal{I} = \{1, \ldots, I\}$: set of components
- $\mathcal{J} = \{1, \ldots, J\}$: set of end items
- $\mathcal{M} = \{1, \ldots, M\}$: set of production resources (machines)
- $\mathcal{S} = \{1, \ldots, S\}$: set of scenarios for demand uncertainty
- d_j^s: demand for end item $j \in \mathcal{J}$ in scenario $s \in \mathcal{S}$
- π^s: probability of scenario s
- C_i: cost of component $i \in \mathcal{I}$
- P_j: price of end item $j \in \mathcal{J}$
- L_m: available time for machine $m \in \mathcal{M}$
- T_{im}: processing time for component $i \in \mathcal{I}$ on machine $m \in \mathcal{M}$
- G_{ij}: number of components of type i needed to assemble one end item of type j (gozinto factors)
- x_i: first-stage variable representing the amount of component i to produce
- y_j^s: second-stage variable representing the amount of end item j assembled and sold in scenario s

Later Fadda et al. [2023] developed risk-averse approach for the same two-stage model.

$$
\begin{aligned}
\min_{x\in\mathbb{R}^I, y\in\mathbb{R}^J} \quad & \zeta + \frac{1}{1-\alpha}\sum_{s=1}^{S}\pi_s z^s \\
\text{s.t.} \quad & -\sum_{i\in\mathcal{I}} C_i x_i + \sum_{s\in\mathcal{S}} \pi^s \left(\sum_{j\in\mathcal{J}} P_j y_j^s\right) \geq \Psi \\
& z^s \geq 0 \quad \forall s \in \mathcal{S} \\
& z^s \geq \sum_{i\in\mathcal{I}} C_i x_i - \sum_{j\in\mathcal{J}} P_j y_j^s - \zeta \quad \forall s \in \mathcal{S} \\
& \sum_{i\in\mathcal{I}} T_{im} x_i \leq L_m \quad \forall m \in \mathcal{M} \\
& y_j^s \leq d_j^s \quad \forall j \in \mathcal{J}, \forall s \in \mathcal{S} \\
& \sum_{j\in\mathcal{J}} G_{ij} y_j^s \leq x_i \quad \forall i \in \mathcal{I}, \forall s \in \mathcal{S} \\
& y_j^s, x_i \geq 0 \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{J}, \forall s \in \mathcal{S} \\
& \zeta \in \mathbb{R}, \quad z^s \geq 0 \quad \forall s \in \mathcal{S}
\end{aligned}
$$

Finally, the approach is generalized to the multistage formulation by Gioia et al. [2024].

$$
\begin{aligned}
\max_{x,y} \quad & \sum_{n\in\mathcal{N}} \pi^{[n]} \left[\sum_{j\in\mathcal{J}} (P_j y_j^{[n]} - K_j l_j^{[n]}) - \sum_{i\in\mathcal{I}} (C_i x_i^{[n]} + H_i I_i^{[n]})\right] & \\
\text{s.t.} \quad & \sum_{i\in\mathcal{I}} T_{im} x_i^{[n]} \leq L_m & \forall m \in \mathcal{M}, \forall n \in \mathcal{N} \\
& I_i^{[n]} = I_i^{[p(n)]} + x_i^{[p(n)]} - \sum_{j\in\mathcal{J}} G_{ij} y_j^{[n]} & \forall i \in \mathcal{I}, \forall n \in \mathcal{N}^+ \\
& I_i^{[0]} = I_i^0 - \sum_{j\in\mathcal{J}} G_{ij} y_j^{[0]} & \forall i \in \mathcal{I} \\
& y_j^{[n]} + l_j^{[n]} = d_j^{[n]} & \forall j \in \mathcal{J}, \forall n \in \mathcal{N} \\
& x_i^{[n]}, I_i^{[n]} \in \mathbb{Z}^+ & \forall i \in \mathcal{I}, \forall n \in \mathcal{N} \\
& y_j^{[n]}, l_j^{[n]} \in \mathbb{Z}^+ & \forall j \in \mathcal{J}, \forall n \in \mathcal{N}
\end{aligned}
$$

As mentioned in the introduction, the mainstream literature regarding the ATO problem isolates the assembler as the sole decision maker, which

might not be the case in many situations. There are papers in the line of research in which there is information sharing between the assembler and its suppliers of components and the assembler has to take into consideration the decision made by its suppliers to optimize their own gain. One of them is Gerchak and Wang [2004] which investigates a one-product model with vendor-managed inventory with revenue sharing. The paper shows that the channel performance of a wholesale-price-only scheme is shown to degrade with the number of suppliers, which is not the case with a revenue-share-only contract. Here is the formulation of the problem:

The assembler shares revenue with the suppliers. The contract specifies that for each unit sold:

- Supplier i receives α_i $(0 < \alpha_i < 1)$.
- The assembler keeps $\alpha_0 = 1 - \sum_{i=1}^{n} \alpha_i$.

The condition for participation is:

$$\alpha_i > c_i, \quad i = 0, 1, \ldots, n.$$

Each supplier i chooses a production quantity Q_i to maximize their expected profit, solving:

$$\bar{F}(Q_i^*) = \frac{c_i}{\alpha_i}, \quad i = 1, \ldots, n.$$

The assembler's optimal assembly quantity Q_0^* is similarly determined by:

$$\bar{F}(Q_0^*) = \frac{c_0}{\alpha_0}.$$

At equilibrium, all suppliers and the assembler produce the same quantity, determined by the "critical supplier" with the smallest Q_i^*:

$$Q_d^* = \min\{Q_1^*, \ldots, Q_n^*, Q_0^*\}.$$

The assembler sets revenue shares α_i to maximize its expected profit:

$$\max_{\alpha_1,\ldots,\alpha_n} \pi_0(\alpha_1, \ldots, \alpha_n) = E\left[-c_0 Q_d^* + \left(1 - \sum_{i=1}^{n} \alpha_i\right) \min(Q_d^*, D)\right].$$

The optimal revenue shares satisfy:

$$\frac{c_1}{\alpha_1} = \cdots = \frac{c_n}{\alpha_n} \geq \frac{c_0}{\alpha_0}.$$

Another paper is Bernstein et al. [2007] in which the authors investigate a 2-product, 3-components (an M system) that shows possible capacity imbalance and inefficiency caused by decentralized decision-making. Specifically, the result indicates that outsourcing the management of component supplies may inhibit the use of operational hedging approaches for managing uncertainty. Our interest is to build a large-scale bilevel stochastic model to handle the size of a realistic instance that optimizes the profit of the assembler.

1.3 Stochastic Programming

In this model, the assembler produces its own components instead of ordering them from other suppliers. Conceptually, there is no difference between producing components and ordering them with zero lead time except that the number of produced component is limited by the assembler's capacity. The limit of the assembler's capacity would lead to the need of the development of a decentralized model that is characterized later in chapter 3.

1.3.1 Two Stage Problems

Two-stage stochastic programming is a powerful framework for optimization under uncertainty, where decisions are made in two sequential stages: before and after observing random parameters. We discuss the formulation, solution methods, and challenges of two-stage stochastic programming, with an emphasis on practical applications such as inventory management. We also review scenario approximation techniques, decomposition methods, and risk-averse extensions.

Stochastic programming provides a systematic approach to decision-making under uncertainty. Among its variants, two-stage stochastic programming is widely used in operations research, finance, and logistics. The key idea is to separate decisions into:

- **First-stage decisions** ("here-and-now"): Made before uncertainty is resolved.
- **Second-stage decisions** ("recourse actions"): Adjustments made after observing the random outcomes.

The first-stage decision variable $x \in X$ is chosen to minimize an immediate cost $c^\top x$ plus the expected cost of future recourse:

$$\min_{x \in X} \left\{ c^\top x + \mathbb{E}[Q(x, \xi)] \right\},$$

where $Q(x, \xi)$ is the optimal value of the second-stage problem. Given a realization of random parameters $\xi = (q, T, W, h)$, the second-stage problem is:

$$Q(x, \xi) = \min_y \left\{ q^\top y \mid Tx + Wy \leq h \right\}.$$

The term $\mathbb{E}[Q(x, \xi)]$ averages the recourse cost over all possible scenarios, ensuring robustness.

Example: Inventory Management Consider a newsvendor problem where:

- x: Order quantity (first-stage decision).
- D: Random demand.
- Costs: c (ordering), b (backorder), h (holding).

The total cost is:

$$G(x, D) = cx + b \max(D - x, 0) + h \max(x - D, 0).$$

The stochastic optimization problem:

$$\min_{x \geq 0} \mathbb{E}[G(x, D)],$$

has a closed-form solution:

$$\bar{x} = F^{-1} \left(\frac{b - c}{b + h} \right),$$

where F is the CDF of demand.

Solution Methods Scenario Approximation Discretize ξ into K scenarios $\{\xi_k\}_{k=1}^K$ with probabilities p_k. The problem becomes a deterministic LP:

$$\min_{x, y_1, \ldots, y_K} c^\top x + \sum_{k=1}^K p_k q_k^\top y_k \quad \text{s.t.} \quad T_k x + W_k y_k \leq h_k, \quad \forall k.$$

Sample Average Approximation (SAA)

- Generate N Monte Carlo samples $\xi^1, \ldots, \xi^N$.
- Approximate $\mathbb{E}[Q(x, \xi)]$ with $\frac{1}{N} \sum_{j=1}^N Q(x, \xi^j)$.

- Statistical bounds validate solution quality.

The two-stage problem can be solved using the following techniques.

- **Benders decomposition (L-shaped method)**: Separates first-stage and second-stage problems.
- **Lagrangian relaxation**: Dualizes coupling constraints.

In conclusion, two-stage stochastic programming provides a flexible and robust framework for optimization under uncertainty. Advances in decomposition methods and sampling techniques have made large-scale problems tractable. Future work includes integrating machine learning for scenario generation and improving risk-averse formulations.

1.3.2 Sample Average Approximation (SAA)

The Sample Average Approximation (SAA) method is a widely used approach for solving stochastic optimization problems, where the objective function involves expectations that are difficult to evaluate directly. The core idea of SAA is to approximate the expected objective function by a sample-based empirical average, transforming the stochastic problem into a deterministic optimization problem that can be solved using conventional techniques. Given a stochastic optimization problem of the form

$$\min_{x \in X} \mathbb{E}[F(x, \xi)],$$

where ξ represents a random variable with an unknown probability distribution, the SAA method estimates the expectation by drawing an independent and identically distributed (i.i.d.) sample $\{\xi_1, \xi_2, \ldots, \xi_N\}$ of size N. The expected objective function is then approximated as

$$\hat{F}_N(x) = \frac{1}{N} \sum_{i=1}^{N} F(x, \xi_i),$$

leading to the deterministic optimization problem

$$\min_{x \in X} \hat{F}_N(x).$$

As the sample size N increases, the SAA solution converges to the true optimal solution under appropriate regularity conditions, such as continuity and convexity of the function $F(x, \xi)$. Theoretical guarantees, including

the consistency and asymptotic normality of SAA estimators, make this approach attractive for practical applications. The efficiency of SAA depends on the trade-off between computational complexity and solution accuracy, as larger sample sizes provide better approximations but increase computational cost. The method has been successfully applied in various fields, including supply chain management, financial optimization, and machine learning, where decision-making under uncertainty is crucial. Moreover, advanced techniques such as variance reduction and adaptive sampling can be incorporated to enhance the performance of SAA. Despite its advantages, one of the main challenges of SAA is selecting an appropriate sample size N, as an insufficient sample may lead to suboptimal solutions, while an excessively large sample increases computational burden. Nonetheless, the SAA method remains a powerful and flexible approach for addressing stochastic optimization problems, particularly when exact analytical solutions are intractable.

1.4 Bilevel optimization

An optimization problem can generally be denoted in the following way:

$$\min_{x \in \mathbb{R}^n} \; f(x) \tag{1.1}$$

$$s.t. \; g(x) \geq 0, \tag{1.2}$$

$$h(x) = 0 \tag{1.3}$$

There is only one objective function f, a variable $x \in \mathbb{R}^n$, and constraints h and g. This type of model is appropriate in several cases in which there is only one rational decision maker, which means that all the parameters affecting the decision are either constant or stochastically known through a distribution. The model is, however, less appropriate when the problem involves more than one decision maker, each of whose decisions affects the parameters of the other's optimization problem. A special case of the problem is the situation in which some decision-makers follow one decision-maker in the temporal order of their decisions. It is called a leader-follower game by von Stackelberg, or a bilevel optimization problem. Here is the formal

definition of a bilevel optimization problem:

$$\min_{x \in X, y} \; F(x, y) \tag{1.4}$$

$$s.t. \; G(x, y) \geq 0, \tag{1.5}$$

$$y \in S(x), \tag{1.6}$$

$$\tag{1.7}$$

where S(x) is the set of optimal solutions of the x-parameterized problem

$$\min_{y \in Y} \; f(x, y) \tag{1.8}$$

$$s.t. \; g(x, y) \geq 0. \tag{1.9}$$

1.4.1 Linear Bilevel optimization

Bilevel optimization is a powerful mathematical framework used to model hierarchical decision-making processes where two decision-makers, referred to as the **leader** and the **follower**, interact in a sequential manner. The leader makes decisions first, anticipating the follower's optimal response, which in turn affects the leader's outcome. This structure is particularly useful in modeling real-world scenarios such as pricing strategies, toll setting, energy markets, and critical infrastructure defense.

Among the various classes of bilevel optimization problems, **linear bilevel optimization** is one of the most studied and well-understood. In linear bilevel optimization, both the leader's and the follower's problems are linear programs (LPs). Despite the linearity of the individual problems, the bilevel structure introduces significant complexity, making even linear bilevel optimization problems challenging to solve.

This article provides a comprehensive overview of linear bilevel optimization, covering its mathematical formulation, properties, solution concepts, and algorithms. We will also discuss some of the challenges and applications of linear bilevel optimization in real-world scenarios.

A **linear bilevel optimization problem** can be formally defined as follows:

Upper-Level Problem (Leader's Problem):

$$\min_{x \in X, y} \quad c_x^\top x + c_y^\top y$$

$$\text{subject to:} \quad Ax + By \geq a,$$

$$y \in S(x),$$

where:

- $x \in \mathbb{R}^{n_x}$ is the leader's decision variable.
- $y \in \mathbb{R}^{n_y}$ is the follower's decision variable.
- $c_x \in \mathbb{R}^{n_x}$ and $c_y \in \mathbb{R}^{n_y}$ are the leader's objective coefficients.
- $A \in \mathbb{R}^{m \times n_x}$ and $B \in \mathbb{R}^{m \times n_y}$ are the leader's constraint matrices.
- $a \in \mathbb{R}^m$ is the right-hand side vector of the leader's constraints.
- $S(x)$ is the set of optimal solutions to the follower's problem.

Lower-Level Problem (Follower's Problem):

$$S(x) = \arg\min_{y \in Y} \quad d^\top y$$

$$\text{subject to:} \quad Cx + Dy \geq b,$$

where:

- $d \in \mathbb{R}^{n_y}$ is the follower's objective coefficient vector.
- $C \in \mathbb{R}^{\ell \times n_x}$ and $D \in \mathbb{R}^{\ell \times n_y}$ are the follower's constraint matrices.
- $b \in \mathbb{R}^\ell$ is the right-hand side vector of the follower's constraints.

Key Components:

1. **Leader's Decision (x)**: The leader chooses x to minimize their objective function, anticipating the follower's optimal response $y \in S(x)$.
2. **Follower's Reaction (y)**: The follower reacts to the leader's decision x by solving their own optimization problem, which is parameterized by x.
3. **Coupling Constraints**: The constraints $Ax + By \geq a$ and $Cx + Dy \geq b$ link the leader's and follower's decisions, making the problem hierarchical.

Linear bilevel optimization problems exhibit several unique properties that distinguish them from standard linear programs:

- Nonconvexity: Even though both the leader's and follower's problems are linear, the bilevel structure introduces nonconvexity. The feasible region of the bilevel problem is often nonconvex and may even be disconnected. This nonconvexity arises because the follower's optimal solution $y \in S(x)$ depends on the leader's decision x, leading to a complex interaction between the two levels.

- Disconnected Feasible Set:The feasible set of a linear bilevel problem is typically a union of faces of the shared constraint set $\Omega = \{(x, y) : Ax + By \geq a, Cx + Dy \geq b\}$. This means that the feasible region is not a single convex set but rather a collection of convex polyhedra.
- NP-Hardness:Linear bilevel optimization problems are **NP-hard**, meaning that there is no known polynomial-time algorithm to solve them in general. This complexity arises from the hierarchical structure and the need to account for the follower's optimal response.
- Optimal Solutions at Vertices: Despite the nonconvexity, the optimal solution of a linear bilevel problem is often found at a vertex of the shared constraint set Ω. This property is exploited by algorithms such as the Kth-best algorithm and branch-and-bound methods.

Now we discuss solution concepts in Bilevel Optimization:

1. Optimistic vs. Pessimistic Solutions
    - **Optimistic Approach**: The leader assumes that the follower will choose the best response for the leader. This is the most common approach in bilevel optimization.
    - **Pessimistic Approach**: The leader assumes that the follower may choose the worst possible response, leading to a more conservative solution.
2. Local vs. Global Solutions
    - **Local Solution**: A feasible point (x^*, y^*) is a local solution if there exists a neighborhood around it where no better solution exists.
    - **Global Solution**: A feasible point (x^*, y^*) is a global solution if it has the best objective value among all feasible points.
3. Inducible Region
    - The **inducible region** is the set of all feasible points (x, y) where y is an optimal solution to the follower's problem for a given x.
    - The bilevel feasible set is a subset of the shared constraint set Ω and is defined by the inducible region.

Solving linear bilevel optimization problems is challenging due to their nonconvexity and NP-hardness. Several algorithms have been developed to tackle these problems:

1. Single-Level Reformulations

   - **KKT Reformulation**: The follower's problem is replaced by its Karush-Kuhn-Tucker (KKT) conditions, leading to a single-level mathematical program with complementarity constraints (MPCC).
   - **Optimal-Value Function Reformulation**: The bilevel problem is reformulated using the optimal-value function of the follower's problem, resulting in a single-level problem with a nonsmooth constraint.
   - **Strong-Duality Based Reformulation**: The follower's problem is replaced by its dual problem, and strong duality is used to derive a single-level reformulation.

2. The Kth-Best Algorithm: This algorithm enumerates the vertices of the shared constraint set Ω in order of increasing leader's objective value. The first vertex that is bilevel feasible (i.e., satisfies the follower's optimality condition) is the optimal solution. The Kth-best algorithm is simple but computationally expensive for large problems due to the need to enumerate vertices.

3. Branch-and-Bound: Branch-and-bound methods are widely used for solving linear bilevel problems. The algorithm branches on the complementarity constraints of the KKT reformulation or the binary variables introduced in the mixed-integer reformulation. This method is more efficient than the Kth-best algorithm but still faces challenges due to the nonconvexity of the problem.

4. Penalty Alternating Direction Methods (PADM): PADM is a heuristic approach that alternates between solving the leader's and follower's problems while penalizing violations of the bilevel constraints. While PADM does not guarantee global optimality, it is often effective in finding high-quality solutions for large-scale problems.

Linear bilevel optimization presents several challenges that complicate its solution process. The nonconvexity of the feasible region makes it difficult to guarantee global optimality, as most algorithms can only find local solutions. Additionally, the NP-hard nature of these problems leads to high computational complexity, making it challenging to solve large instances efficiently, even with advanced algorithms. Another difficulty arises when the follower's problem has multiple optimal solutions for a given upper-level decision, causing the bilevel problem to become ill-posed. To address this, optimistic and

pessimistic approaches are used, but they introduce additional complexity. Furthermore, the choice of single-level reformulation—such as using Karush-Kuhn-Tucker (KKT) conditions, the optimal-value function, or strong duality—plays a crucial role in determining the efficiency and tractability of the solution process.

Linear bilevel optimization has numerous applications in various fields:

1. Pricing Problems: A company (leader) sets prices for its products, anticipating how customers (followers) will react by choosing the optimal combination of products to purchase.

2. Toll Setting: A toll-setting agency (leader) determines tolls on a transportation network, while travelers (followers) choose routes that minimize their travel costs.

3. Energy Markets: A regulatory authority (leader) sets market rules, while energy producers and consumers (followers) optimize their production and consumption decisions.

4. Critical Infrastructure Defense: A defender (leader) allocates resources to protect critical infrastructure, while an attacker (follower) chooses targets to maximize damage.

5. Interdiction Problems: A defender (leader) interdicts resources to disrupt an adversary's (follower) operations, such as in drug smuggling or network interdiction.

In conclusion, Linear bilevel optimization is a powerful tool for modeling hierarchical decision-making processes in various real-world applications. Despite its linear structure, the bilevel nature of the problem introduces significant complexity, making it challenging to solve. Researchers have developed a variety of algorithms, including single-level reformulations, the Kth-best algorithm, branch-and-bound methods, and penalty alternating direction methods, to tackle these problems. While linear bilevel optimization remains an active area of research, its applications in pricing, toll setting, energy markets, and critical infrastructure defense demonstrate its practical importance. As computational techniques continue to advance, we can expect further progress in solving larger and more complex bilevel optimization problems, unlocking new possibilities for modeling and optimizing hierarchical systems.

1.5 Heuristic methods

1.5.1 CMA-ES

Optimization is a fundamental problem in many scientific, engineering, and machine learning domains. The goal is to find the best solution to a problem, often by minimizing or maximizing an objective function $f : \mathbb{R}^n \to \mathbb{R}$. Among the many optimization algorithms developed, the **Covariance Matrix Adaptation Evolution Strategy (CMA-ES)** stands out as a powerful, derivative-free method for solving difficult non-linear, non-convex optimization problems in continuous search spaces.

CMA-ES is a stochastic optimization algorithm that belongs to the class of **evolution strategies (ES)**, which are inspired by natural evolution principles such as mutation, recombination, and selection. It is particularly effective for problems where the objective function is noisy, non-differentiable, or has many local optima. The key innovation in CMA-ES is its ability to adapt the **covariance matrix** of a multivariate Gaussian distribution, which guides the search process by learning the problem's underlying structure.

In this section, we provide a detailed and comprehensive overview of CMA-ES, including its mathematical formulation, algorithmic components, practical applications, and recent developments.

Evolution Strategies: The Foundation of CMA-ES

Evolution Strategies (ES) are a class of optimization algorithms inspired by natural evolution. They maintain a population of candidate solutions, which are iteratively improved through **mutation**, **recombination**, and **selection**.

Mutation and Recombination

In ES, new candidate solutions are generated by **mutation**, which involves adding random perturbations to the current population. The perturbations are typically drawn from a multivariate normal distribution:

$$\mathbf{x}_i' = \mathbf{x}_i + \sigma \mathbf{z}_i,$$

where:

- $\mathbf{x}_i$ is a candidate solution,
- σ is the step-size (mutation strength),

- $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$ is a random vector sampled from a multivariate normal distribution with mean $\mathbf{0}$ and covariance matrix $\mathbf{C}$.

Recombination combines information from multiple parent solutions to create offspring. For example, the weighted average of μ selected parents can be computed as:

$$\mathbf{x}_{\text{recomb}} = \sum_{i=1}^{\mu} w_i \mathbf{x}_i,$$

where w_i are weights assigned to the parents.

Selection

Selection is the process of choosing the fittest solutions from the population to form the next generation. In ES, selection is typically based on the fitness values $f(\mathbf{x}_i)$. The μ best solutions are selected, and their information is used to guide the search.

Limitations of Basic ES

The performance of ES depends heavily on the choice of the covariance matrix $\mathbf{C}$ and step-size σ. If these parameters are not properly tuned, the algorithm may converge slowly or get stuck in local optima. CMA-ES addresses this limitation by introducing a mechanism to **adapt the covariance matrix** and **step-size** during the optimization process.

The CMA-ES Algorithm

CMA-ES extends the basic ES framework by dynamically adapting the covariance matrix $\mathbf{C}$ and step-size σ based on information gathered during the optimization process. The key components of CMA-ES are:

1. **Covariance Matrix Adaptation**: Adjusts the shape and orientation of the search distribution.
2. **Evolution Path**: Tracks the movement of the population over iterations.
3. **Step-Size Control**: Adjusts the overall scale of the mutation distribution.
4. **Population Update**: Generates new candidate solutions through sampling, recombination, and selection.

Let's explore each component in detail, with mathematical formulations.

Covariance Matrix Adaptation The covariance matrix $\mathbf{C}$ determines the shape and orientation of the search distribution. In CMA-ES, $\mathbf{C}$ is adapted iteratively using two mechanisms:

Rank-One Update

The rank-one update incorporates information from the **evolution path** $\mathbf{p}_\sigma$, which accumulates the steps taken by the algorithm:

$$\mathbf{p}_\sigma^{(t+1)} = (1 - c_\sigma)\mathbf{p}_\sigma^{(t)} + \sqrt{c_\sigma(2 - c_\sigma)\mu_{\text{eff}}} \cdot \frac{\mathbf{m}^{(t+1)} - \mathbf{m}^{(t)}}{\sigma^{(t)}},$$

where:

- c_σ is the learning rate for the evolution path,
- μ_{eff} is the effective selection mass,
- $\mathbf{m}^{(t)}$ is the mean of the population at iteration t.

The covariance matrix is then updated as:

$$\mathbf{C}^{(t+1)} = (1 - c_1)\mathbf{C}^{(t)} + c_1\mathbf{p}_\sigma^{(t+1)}(\mathbf{p}_\sigma^{(t+1)})^\top,$$

where c_1 is the learning rate for the rank-one update.

Rank- Update

The rank- update incorporates information from the selected candidate solutions:

$$\mathbf{C}^{(t+1)} = (1 - c_\mu)\mathbf{C}^{(t)} + c_\mu \sum_{i=1}^{\mu} w_i \mathbf{y}_i \mathbf{y}_i^\top,$$

where:

- c_μ is the learning rate for the rank- update,
- $\mathbf{y}_i = (\mathbf{x}_i - \mathbf{m}^{(t)})/\sigma^{(t)}$ is the normalized step,
- w_i are weights assigned to the selected solutions.

The combined update rule for the covariance matrix is:

$$\mathbf{C}^{(t+1)} = (1 - c_1 - c_\mu)\mathbf{C}^{(t)} + c_1\mathbf{p}_\sigma^{(t+1)}(\mathbf{p}_\sigma^{(t+1)})^\top + c_\mu \sum_{i=1}^{\mu} w_i \mathbf{y}_i \mathbf{y}_i^\top.$$

Evolution Path

The evolution path $\mathbf{p}_\sigma$ accumulates the steps taken by the algorithm over multiple iterations. It is updated as:

$$\mathbf{p}_\sigma^{(t+1)} = (1 - c_\sigma)\mathbf{p}_\sigma^{(t)} + \sqrt{c_\sigma(2 - c_\sigma)\mu_{\text{eff}}} \cdot \frac{\mathbf{m}^{(t+1)} - \mathbf{m}^{(t)}}{\sigma^{(t)}}.$$

Step-Size Control

The step-size σ controls the overall scale of the mutation distribution. It is adapted using the **cumulative step-size adaptation (CSA)** mechanism:

$$\sigma^{(t+1)} = \sigma^{(t)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\mathbf{p}_\sigma^{(t+1)}\|}{\mathbb{E}[\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|]} - 1\right)\right),$$

where:

- c_σ is the learning rate for the step-size,
- d_σ is a damping factor,
- $\mathbb{E}[\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|]$ is the expected length of a vector sampled from a standard normal distribution.

Population Update

The population is updated iteratively by sampling new candidate solutions from the multivariate normal distribution:

$$\mathbf{x}_i^{(t+1)} = \mathbf{m}^{(t)} + \sigma^{(t)}\mathbf{z}_i,$$

where $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C}^{(t)})$. The mean $\mathbf{m}$ is updated as:

$$\mathbf{m}^{(t+1)} = \sum_{i=1}^{\mu} w_i \mathbf{x}_i^{(t+1)}.$$

The CMA-ES Algorithm in Pseudocode

CMA-ES Algorithm

Initialize:
  Mean vector $\mathbf{m}^{(0)}$,
  Covariance matrix $\mathbf{C}^{(0)} = \mathbf{I}$,
  Step-size $\sigma^{(0)}$,
  Evolution path $\mathbf{p}_\sigma^{(0)} = \mathbf{0}$.
Repeat until convergence:
  Sample λ candidate solutions: $\mathbf{x}_i = \mathbf{m} + \sigma \mathbf{z}_i$, where $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$.
  Evaluate fitness $f(\mathbf{x}_i)$.
  Select the μ best solutions.
  Update mean $\mathbf{m}$.
  Update evolution path $\mathbf{p}_\sigma$.
  Update covariance matrix $\mathbf{C}$.
  Update step-size σ.

Practical Applications of CMA-ES

The CMA-ES algorithm has demonstrated successful applications across diverse domains, including **hyperparameter tuning** in machine learning, **engineering design optimization** (such as aerodynamic shapes and structural design), **robotics** (including gait optimization and control policies), **financial portfolio optimization**, and **bioinformatics** (notably protein folding and gene network inference). The method offers several key advantages: its **robustness** enables handling of noisy, non-differentiable, and non-convex functions; its **adaptability** allows it to learn problem structure through covariance matrix adaptation; and its **efficiency** makes it particularly effective for high-dimensional problems. However, CMA-ES also presents certain limitations, including significant **computational cost** with high memory and processing requirements for large dimensions n n, the need for **careful hyperparameter tuning**, and a tendency to converge to **local optima** when dealing with highly multimodal functions.

Conclusion

CMA-ES is a state-of-the-art optimization algorithm that combines robustness, adaptability, and efficiency. Its ability to adapt the covariance matrix and step-size makes it particularly effective for challenging optimization problems. Despite its computational cost, CMA-ES remains a powerful tool

for solving complex real-world problems, and ongoing research continues to expand its capabilities and applications.

1.5.2 PSO

Particle Swarm Optimization (PSO) is a population-based stochastic optimization technique inspired by the social behavior of bird flocking or fish schooling. We cover the fundamental concepts, mathematical formulations, variants, applications, and recent advances in PSO. The article provides both theoretical foundations and practical implementation guidelines, making it a valuable resource for researchers and practitioners in optimization and computational intelligence. Particle Swarm Optimization (PSO) is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. First introduced by Kennedy and Eberhart in 1995, PSO has become one of the most popular swarm intelligence algorithms due to its simplicity, efficiency, and effectiveness across a wide range of optimization problems. The algorithm simulates the social behavior observed in flocks of birds or schools of fish, where individuals (particles) adjust their movements based on their own experience and the experience of their neighbors. In PSO terminology, the "swarm" refers to the population of candidate solutions, and each "particle" represents a potential solution to the optimization problem. In the standard PSO algorithm, each particle i in the swarm is characterized by:

- A position vector $\vec{x}_i(t) = (x_{i1}, x_{i2}, ..., x_{iD})$ representing a potential solution in D-dimensional space
- A velocity vector $\vec{v}_i(t) = (v_{i1}, v_{i2}, ..., v_{iD})$ representing the direction and magnitude of movement
- A personal best position $\vec{p}_i(t) = (p_{i1}, p_{i2}, ..., p_{iD})$ representing the best solution found by the particle so far

The swarm also maintains a global best position $\vec{g}(t)$, which represents the best solution found by any particle in the swarm up to time t.

The velocity and position update equations are:

$$\vec{v}_i(t+1) = w\vec{v}_i(t) + c_1 r_1(\vec{p}_i(t) - \vec{x}_i(t)) + c_2 r_2(\vec{g}(t) - \vec{x}_i(t)) \tag{1.10}$$

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1) \tag{1.11}$$

where:

- w is the inertia weight
- c_1 and c_2 are cognitive and social acceleration coefficients
- r_1 and r_2 are random numbers uniformly distributed in [0,1]

Algorithm Pseudocode

Basic PSO Algorithm

```
Initialize swarm with random positions and velocities
while stopping criterion not met do
    for each particle i do
        Evaluate fitness f(x_i)
        if f(x_i) < f(p_i) then
            p_i ← x_i
        end if
        if f(x_i) < f(g) then
            g ← x_i
        end if
    end for
    for each particle i do
        Update velocity using equation (1)
        Update position using equation (2)
    end for
end whilereturn g as the best solution found
```

Parameters and Their Impact

The performance of PSO depends heavily on the proper setting of its parameters:

Table 1.1: Key PSO Parameters and Their Effects

Parameter	Typical Range	Effect
Swarm size	20-25	Larger swarms explore more but increase computation
Inertia weight (w)	0.4-0.9	Controls exploration-exploitation trade-off
Cognitive coefficient (c1)	1.5-2.0	Influences attraction to personal best
Social coefficient (c2)	1.5-2.0	Prevents swarm explosion
Velocity clamping	$-v_{max}, v_{max}$	Prevents swarm explosion

Particle Swarm Optimization has evolved significantly since its inception, proving to be a versatile and powerful optimization tool. Its simplicity, flexibility, and effectiveness have made it popular across numerous domains. Current research continues to expand its capabilities through hybridization,

theoretical analysis, and application to increasingly complex problems. Future directions may include deeper theoretical foundations, adaptive parameter control, and applications in emerging fields like deep learning and big data analytics.

Chapter 2

The ATO Model

2.1 Notations

To state the mathematical model describing the ATO production planning problem, let us introduce the following sets: the set of components $\mathcal{I} = \{1, \ldots, I\}$, the set of end items $\mathcal{J} = \{1, \ldots, J\}$, the set of production resources (machines) $\mathcal{M} = \{1, \ldots, M\}$.

Furthermore, let us introduce the following parameters:

- C_i cost of component $i \in \mathcal{I}$;
- P_j price of the end item $j \in \mathcal{J}$;
- L_m availability (in terms of time) of machine $m \in \mathcal{M}$;
- T_{im} processing time for end item $i \in \mathcal{I}$ on machine $m \in \mathcal{M}$;
- G_{ij} number of components of type $i \in \mathcal{I}$ needed to assemble one end item of type $j \in \mathcal{J}$; in manufacturing parlance, these numbers are called *gozinto factors*.
- H_i inventory holding cost of component $i \in \mathcal{I}$
- K_i lost sale penalty of end item $j \in \mathcal{J}$

We define $\mathcal{S}$ as the set of scenarios. For each scenario $s \in \mathcal{S}$, we have the variables:

- $x_i^s \in \mathbb{Z}^+$, the amount of components $i \in \mathcal{I}$ produced or received in the scenario $s \in \mathcal{S}$. However, since the number of components has to be determined in advance before any scenario is realized, we have to

add a nonanticipativity constraint on $x_i^s \ \forall i \in \mathcal{I}, \forall s \in \mathcal{S}$. Therefore, we simply denote the quantity as $x_i \ \forall i \in \mathcal{I}$

- $I_i^s \in \mathbb{Z}^+$, the available inventory of components $i \in \mathcal{I}$ in the scenario $s \in \mathcal{S}$.
- $y_j^s \in \mathbb{Z}^+$, the amount of end items$j \in \mathcal{J}$ assembled in the scenario $s \in \mathcal{S}$.
- $l_j^s \in \mathbb{Z}^+$, the lost sales of end items $j \in \mathcal{J}$ in the scenario $s \in \mathcal{S}$.
- $\alpha_i \in \mathbb{Z}^+$, the price of the component $i \in \mathcal{I}$ set by the assembler in the first stage of the decentralized model.

2.2 Centralized model

$$\max \ -\sum_i C_i x_i + \sum_{s\in\mathcal{S}} \pi^s \left[\sum_{j\in\mathcal{J}} P_j y_j^s - \sum_{j\in\mathcal{J}} K_j l_j^s - \sum_{i\in\mathcal{I}} H_i I_i^s \right] \tag{2.1}$$

$$\text{s.t} \ \sum T_{im} x_i \leq L_m \qquad \forall m \in \mathcal{M} \tag{2.2}$$

$$I_i^s = I_i^0 + x_i - \sum_{j\in\mathcal{J}} G_{ij} y_j^s \qquad \forall i \in \mathcal{I}, \forall s \in \mathcal{S} \tag{2.3}$$

$$d_j^s = y_j^s + l_j^s \qquad \forall j \in \mathcal{J}, \forall s \in \mathcal{S} \tag{2.4}$$

$$x_i^s, y_j^s, I_i^s, l_j^s \in \mathbb{Z}^+ \qquad \forall i \in \mathcal{I}, \forall j \in \mathcal{J}, \forall s \in \mathcal{S} \tag{2.5}$$

The objective function of the centralized model 2.1 is the net profit calculated by taking the expected revenue minus the total cost that the assembler has to pay to procure components in the first stage. Constraints 2.2 regulate the number of components that can be produced to meet the incoming demands according to the current capacity of the assembler. In this centralized model, the assembly company also plays the role of the manufacturer where they produce their components to be assembled into the end items. Constraints 2.3 represent the relation between the components produced and the number of assembled end items; the unused components in this period will be carried over to the next one, the assembler paying for the storage costs. Constraints 2.4 with the non-negativity of the decision variable guarantee that the number of assembled end items does not exceed the supposed demands in a scenario and calculate the lost sale. Constraints 2.5 specify that all decision variables are nonnegative integers.

2.3 Decentralized model

$$\max \sum_{s\in\mathcal{S}} \pi^s \left[\sum_{j\in\mathcal{J}} P_j y_j^s - \sum_{j\in\mathcal{J}} K_j l_j^s - \sum_{i\in\mathcal{I}} H_i I_i^s - \sum_{i\in\mathcal{I}} \alpha_i \sum_{j\in\mathcal{J}} G_{ij} y_j^s \right] \tag{2.6}$$

$$\text{s.t } I_i^s = I_i^0 + x_i^* - \sum_{j\in\mathcal{J}} G_{ij} y_j^s \qquad \forall i \in \mathcal{I}, \forall s \in \mathcal{S} \tag{2.7}$$

$$d_j^s = y_j^s + l_j^s \qquad \forall j \in \mathcal{J}, \forall s \in \mathcal{S} \tag{2.8}$$

$$y_j^s, I_i^s, l_j^s \in \mathbb{Z}^+ \qquad \forall i \in \mathcal{I}, \forall j \in \mathcal{J}, \forall s \in \mathcal{S} \tag{2.9}$$

$$x_i^* \in B_i(\alpha_i) \qquad \forall i \in \mathcal{I} \tag{2.10}$$

$$\max \; -C_i x_i + \alpha_i E \left[\min \left(x_i, \sum_{j\in\mathcal{J}} G_{ij} d_j^s \right) \right] \tag{2.11}$$

$$x_i \in \mathbb{Z}^+ \tag{2.12}$$

In the decentralized model, the objective function 2.6 is still the net profit made by the manufacturer, but the decision variable in the first stage is no longer the number of components produced determined by the manufacturer. Instead, in this version of the problem, the assembler, in the first stage, sets a price level α_i for each component $i \in \mathcal{I}$. In response, each supplier will produce the number of components according to the information shared about the incoming demands that optimize their objective 2.11. The supplier problem is a Newsvendor model that has an analytical solution:

$$x_i^* = F^{-1}\left(\frac{\alpha_i - C_i}{\alpha_i}\right),$$

for each $i \in \mathcal{I}$ and $F(\cdot)$ is the cumulative distribution function of demand. The constraints 2.7 represent the inventory of the assembler, and the number of received components is a constant from the assembler's point of view. We assume that the assembler keeps all the unused components and will pay for the incurred holding costs, but as we can see in the objective function 2.6, the assembler will only pay for the components that are used to make the end items. The constraints 2.8 and 2.9 are the same as in the centralized version of the model. This can be seen as a bilevel optimization problem or the so-called Stalkerberg game where the leader makes the first move and all

the followers respond accordingly; all actors in the game decide to optimize their objective. In our model, the assembler is the leader who sets the price they will pay to each of their suppliers, and the suppliers are the followers who will respond by producing an appropriate number of components.

2.4 Heuristic Ascent Direction Algorithm

The most direct approach to solving a bilevel optimization problem involves reformulating it as a single-level problem. This process typically begins by discretizing and linearizing the lower-level problem. Subsequently, the Karush-Kuhn-Tucker (KKT) conditions are applied to derive an equivalent set of constraints for the resulting single-level formulation.

$$\max \ -C_i x_i + \alpha_i \sum_{s \in \mathcal{S}} \pi^s t^s \tag{2.13}$$

$$t^s \leq x_i \qquad \forall s \in \mathcal{S} \tag{2.14}$$
$$t^s \leq \sum_{j \in \mathcal{J}} G_{ij} d_j^s \qquad \forall s \in \mathcal{S} \tag{2.15}$$
$$x_i, t^s \in \mathbb{Z}^+ \qquad \forall s \in \mathcal{S} \tag{2.16}$$

$$\mathcal{L}(x,t,\lambda) = -C_i x_i + \alpha_i \sum_{s \in \mathcal{S}} \pi^s t^s + \sum_{s \in \mathcal{S}} \lambda_1^s (x_i - t^s) + \sum_{s \in \mathcal{S}} \lambda_2^s (-t^s + \sum_{j \in \mathcal{J}} G_{ij} d_j^s) + \lambda_3 x_i + \sum_{s \in \mathcal{S}} \lambda_4^s t^s$$

$$\nabla_{x_i} = -C_i + \sum_{s \in \mathcal{S}} \lambda_1^s + \lambda_3$$
$$\nabla_{t^s} = \alpha_i \pi^s - \lambda_1^s - \lambda_2^s + \lambda_4^s$$

- 1. Primal feasibility

$$t^s \leq x_i \qquad \forall s \in \mathcal{S} \tag{2.17}$$
$$t^s \leq \sum_{j \in \mathcal{J}} G_{ij} d_j^s \qquad \forall s \in \mathcal{S} \tag{2.18}$$
$$x_i, t^s \geq 0 \qquad \forall s \in \mathcal{S} \tag{2.19}$$

- 2. Complementarity

$$\lambda_1^s(x_i - t^s) = 0 \qquad \forall s \in \mathcal{S} \qquad (2.20)$$

$$\lambda_2^s(-t^s + \sum_{j \in \mathcal{J}} G_{ij} d_j^s) = 0 \qquad \forall s \in \mathcal{S} \qquad (2.21)$$

$$\lambda_3 x_i = 0 \qquad (2.22)$$

$$\lambda_4^s t^s = 0 \qquad \forall s \in \mathcal{S} \qquad (2.23)$$

- 3. Dual feasibility

$$-C_i + \sum_{s \in \mathcal{S}} \lambda_1^s + \lambda_3 = 0 \qquad (2.24)$$

$$\alpha_i \pi^s - \lambda_1^s - \lambda_2^s + \lambda_4^s = 0 \qquad \forall s \in \mathcal{S} \qquad (2.25)$$

$$\lambda^s \geq 0 \qquad \forall s \in \mathcal{S} \qquad (2.26)$$

The reformulated problem introduces nonlinearity into the objective function along with a set of complementarity constraints. While commercial optimization solvers such as Gurobi can handle small-scale instances of this problem, solving larger, more realistic instances using exact methods remains computationally infeasible. To address this limitation, we propose a heuristic approach to obtain high-quality solutions. The method is grounded in the observation that the assembler must offer a strictly higher price to acquire the necessary components for producing the final products. Specifically, we initialize the parameter $\alpha^0 = C$ and iteratively update it according to $\alpha^{n+1} = \alpha^n + \Delta\alpha$, where $\Delta\alpha > 0$, after a certain number of iterations, we choose the best value achieved in the process. Finally, we refine the solution using a local search technique to further improve the outcome.

Heuristic Ascent Direction

Require: Initial cost C, ascent direction $\Delta\alpha > 0$, number of iterations N
Ensure: Refined solution maximizing expected profit
1: Initialize $\alpha_0 \leftarrow C$
2: **repeat**
3: Update $\alpha_{n+1} \leftarrow \alpha_n + \Delta\alpha$
4: Compute the expected profit and store the result
5: **until** repeated N times
6: $\alpha^* = \alpha_0 + argmax_n\{\alpha_n\} * \Delta\alpha$
7: **return** Refined solution.

We can restrict the search space by finding an upper bound for α using the following lemma.

Lemma 2.1. *For each component $i \in \mathcal{I}$, $\mathcal{J}_i = \{j \in \mathcal{J} \mid G_{ij} > 0\}$, if $\delta_i = \max_{j \in \mathcal{J}_i} \frac{P_j - \sum_{i \in \mathcal{I}} G_{ij} C_i}{G_{ij} C_i}$, then there exists $\alpha_i^* \leq (1 + \delta_i) C_i$.*

Proof. In the context of manufacturing, it is crucial to ensure that any increase in the cost of a component does not exceed the profit margins of the products that incorporate this component. This principle is essential because if the cost increase leads to selling the final products at a loss, the optimal strategy would be to cease assembling those products, thereby eliminating the need for the cost increase in the first place. To formalize this, let's consider the optimal offer value for a component i, denoted as $\alpha_i^* = (1 + \delta_i^*) C_i$, where C_i is the base cost of the component and δ_i^* is the margin that represents the additional profit above the base cost. The goal is to determine the conditions under which δ_i^* ensures that the products remain profitable. Let G_{ij} represent the quantity of component i used in product j, and let P_j be the selling price of product j. The margin for product j is calculated as $M_j = P_j - \sum_{i \in \mathcal{I}} G_{ij} C_i$, which is the difference between the selling price and the total cost of the components used in the product. For any product j that includes component i (i.e., $G_{ij} > 0$), the increase in the component cost, given by $G_{ij} \delta_i^* C_i$, must be less than the margin M_j to ensure that the product is not sold at a loss. This condition can be expressed mathematically as $G_{ij} \delta_i^* C_i < P_j - \sum_{i \in \mathcal{I}} G_{ij} C_i$. By rearranging this inequality, we find that there exists a product j in the set of products $\mathcal{J}_i$ that use component i such that $\delta_i^* \leq \frac{P_j - \sum_{i \in \mathcal{I}} G_{ij} C_i}{G_{ij} C_i}$. This implies that the margin δ_i^* must be less than or equal to the maximum margin ratio across all products that include component i, which can be written as $\delta_i^* \leq \max_{j \in \mathcal{J}_i} \frac{P_j - \sum_{i \in \mathcal{I}} G_{ij} C_i}{G_{ij} C_i}$. Therefore, by ensuring that δ_i^* satisfies this condition, we can guarantee that the increase in the component cost will not lead to selling any of the products at a loss, thereby maintaining overall profitability. □

Chapter 3

Experiment Design

This chapter provides an overview of the technological and economic data utilized to support our numerical experiments. The data encompass a range of demand distributions, varying in complexity from relatively straightforward to challenging scenarios for identifying optimal solutions. In addition, the configuration of the heuristic optimization framework employed in this study is detailed.

3.1 Problem instance generation

The instance utilized in our numerical experiments is taken from Gioia et al. [2024]. The problem instance is described by the following data:

- capacity requirements and availability (T_{im}, L_m).
- economic parameters, i.e., costs, penalties, and sale prices (C_i, H_i, K_i, P_j)
- product structures, i.e., the gozinto factors (G_{ij})

First of all, we set the number of components I and end items J according to 3.1. In the centralized setting, we have the number of machines M and tightness factor γ to generate the available capacity for each machine L_m, concerning the average capacity requirement. In formulas:

$$L_m = \gamma \sum_{i \in \mathcal{I}} T_{im} \left(\sum_{j \in \mathcal{J}} G_{ij} \bar{d}_j \right) \tag{3.1}$$

The expected demand for end item j, denoted by $\bar{d}_j$, is estimated through a sampling process involving 5,000 demand scenarios, given the complexity

of the demand generation mechanism described subsequently. In the experimental setup, the parameter γ is specified according to Table 3.1, representing instances where production capacity is defined as the product of average demand and γ. Higher values of γ facilitate the assembly company's ability to manufacture sufficient components to meet anticipated demand.

Regarding economic parameters, component costs are randomly generated following a uniform distribution within the range [1, 50]. The end items are classified into three categories of profit margins, low, medium, and high, whose proportions and respective ranges for uniform sampling are detailed in Table 3.1. The cost of each end item is derived by combining the bill of materials with the associated component costs. The corresponding sales price is calculated by applying a profit mark-up consistent with the designated profit margin category. In addition, inventory holding costs for components and lost sales penalties for end items are determined as fixed percentages of component costs and sales prices, respectively.

In the literature, several standard structures for the gozinto matrix have been proposed Atan et al. [2017]. Nevertheless, these structures are usually considered for their theoretical properties rather than their realism. We define the gozinto matrix as follows. First, we define families by partitioning the set of items: each family has a given number of end items, and, within a family, end items have some required components, either specific or common with other members of the family. Moreover, we also introduce degenerate families consisting of a single end item. Such end items are, in a sense, different from the others and are called outcast items. In section 3.1, we report the characteristics of standard families (number of common vs. total components and number of items per family) and the number of outcast items. Once the total number of components of a (standard) family is known, we select the common components and the specific ones for each end item. Then, we sample the number of each component required by the end item, according to a discrete uniform distribution ranging from 1 to 9. On the contrary, outcast items do not share any specific pattern with others, and their components are picked according to a binomial distribution so that each component is included and adopted for the computational experiments, as depicted in ... The diagonal blocks correspond to the standard families (five in this case), while the bottom rows correspond to the outcast items. Each standard family shares several common components (the first columns of each block).

The literature presents several standard structures for the gozinto matrix, as discussed by Atan et al. Atan et al. [2017]. However, these structures are often evaluated for their theoretical properties rather than their practical

Parameter	Value	Parameter	Value
No. end items (I)	35	No. components (J)	60
No. machines (M)	5	Tightness factor γ	0.25, 0.5, 0.75, 1.0, 1.25
Low margin profit	0.05, 0.2	% low margin items	40%
Medium margin profit	0.2,0.4	% medium margin items	30%
High margin profit	0.4, 0.6	% high margin items	30%
No. outcast items	5	No. families (R)	5
No. items per family	[12, 7, 5, 3, 3]	No. common components	2
No. components per family	[11, 17, 12, 6, 9]		
Lost sale penalty K_j	$0.2P_j$	Holding cost H_i	$0.1C_i$

Table 3.1: Parameters defining a problem instance

applicability. In this work, we define the gozinto matrix as follows: items are partitioned into families, each comprising a specified number of end items. Within a family, end items share some required components, which may be common across the family or specific to individual items. In addition, we introduce degenerate families, which consist of a single end item. These items referred to as outcast items, are distinct from other end items in terms of structure.

The characteristics of the standard families, including the ratio of common to total components, the number of items per family, and the number of outcast items, are summarized in Table 3.1. For standard families, once the total number of components is determined, the common components are identified and specific components are assigned to each end item. The quantity of each component required by an end item is then sampled from a discrete uniform distribution over the range [1, 9]. In contrast, outcast items exhibit no shared structural pattern; their components are selected independently following a binomial distribution.

For computational experiments, the resulting gozinto matrix is structured as depicted in 3.1 Diagonal blocks in the matrix correspond to standard families (five in this example), where each block includes columns representing shared common components and rows for specific components. The bottom rows of the matrix represent the outcast items, which are characterized by their unique sets of components.

3.2 Demand distributions

n the following, we test the effect of the aforementioned instance settings with respect to different i.i.d. demand distributions. In particular, we consider $\beta_{4,1}(0, 1300)$ and $\beta_{1,4}(0, 1300)$, $\mathcal{N}(300, 50)$, and a normal mixture (bi-gaussian distribution) $0.8\mathcal{N}(300, 50) + 0.2\mathcal{N}(50, 15)$. We use the notation

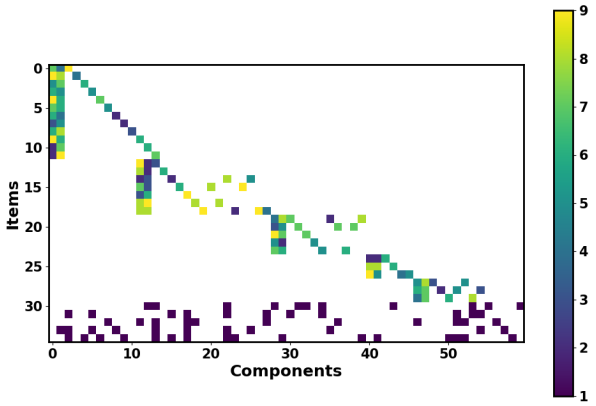

Figure 3.1: The generated gozinto matrix from the configuration

$\beta_{\alpha_1,\alpha_2}(d_{min}, d_{max})$to denote the a standard beta random variate X with support (0, 1) with given parameters $\alpha_1, \alpha_2 \geq 1$, re-scaled in the interval $[d_{min}, d_{max}]$ by $d_{min} + (d_{max} - d_{min})X$. In the case of a Gaussian distribution, we write variance in such a way that the standard deviation is evident. To generate the problem instances, we use a Python 3.9 script that directly solves the optimization problem by Gurobi 9.5.2 interior point solver, with default settings. By checking the out-of-sample stability of the problem, it is possible to see that 100 scenarios are enough for achieving reasonable results under given distributions for demands of our products. This is true under all other settings used in our instance (this does not necessarily apply to other settings). The number of out-of-sample scenarios $S^{'}$ has been set to 2000. This number of scenarios is adequate to get a good estimate of expected revenue for a given first-stage solution.

Chapter 4

Numerical Result

In this chapter, we present a comprehensive analysis of the numerical experiments conducted to evaluate the performance of our proposed solutions. Through systematic testing, we assess the efficacy of multiple heuristic optimization algorithms, comparing their computational efficiency, scalability, and solution quality under varying problem conditions. Based on these empirical findings, we identify the most suitable algorithm for our framework and subsequently refine its parameters through fine-tuning to enhance performance.

Following this optimization phase, we conduct a comparative analysis between centralized and decentralized implementations of our model. This comparison not only highlights the trade-offs between the two approaches—such as computational overhead, coordination requirements, and robustness—but also yields actionable managerial insights. These insights are particularly valuable for decision-makers seeking to balance operational efficiency with organizational flexibility in real-world applications.

By integrating theoretical rigor with empirical validation, this chapter provides a robust foundation for understanding the practical implications of our methodology.

4.1 Comparison between CMA-ES and PSO

In this computational experiment, unless otherwise specified, we generate demand values from a normal distribution $\mathcal{N}(300, 50)$ and evaluate the performance of a vanilla CMA-ES (Covariance Matrix Adaptation Evolution Strategy) and PSO (Particle Swarm Optimization) across 100 scenarios. The CMA-ES algorithm is initialized at the cost vector of the components, while

	CMA-ES	PSO
population	16	16
number of iterations	20	20
parameters	$\sigma = 0.1$	$w = 0.9$ $c_1 = 2$ $c_2 = 2$

PSO randomly samples its initial population within the search space; all variables are normalized using Lemma 2.1, with lower bounds set to the original component costs and upper bounds determined by the same lemma. As demonstrated in Figure 4.1, CMA-ES significantly outperforms PSO, which is expected given CMA-ES's inherent advantages in evaluation-expensive optimization problems. Specifically, CMA-ES automatically adapts its step size and search distribution—unlike PSO, which requires manual tuning of parameters like inertia weight and acceleration coefficients—and leverages covariance matrix learning to efficiently scale and rotate the search space, a feature absent in PSO that often struggles with local optima. Furthermore, CMA-ES is grounded in probability theory and maintains invariance under order-preserving transformations, offering theoretical robustness compared to PSO's heuristic approach. While PSO may excel in extremely high-dimensional spaces or parallel evaluation contexts, CMA-ES generally achieves higher precision with fewer function evaluations and minimal parameter tuning, making it a more reliable choice for most black-box optimization problems.

4.2 Heuristic Profit Optimization

In this section, we integrate the heuristic algorithm (Algorithm 3) with Covariance Matrix Adaptation Evolution Strategy (CMA-ES) to demonstrate its superior performance compared to a standalone implementation of vanilla CMA-ES. First, we examine the rationale behind the proposed heuristic (Algorithm 3). Theoretical observations indicate that an isolated increase in the cost of a single component may not necessarily result in a loss 2.1, whereas maintaining the original production cost could lead to procurement failure because the supplier would reject to deliver any amount of components. However, the behavior of the objective function of the assembler remains unclear when the prices of all components are adjusted simultaneously. To address this, we seek an ascent direction with an appropriate step size to identify a near-optimal solution.

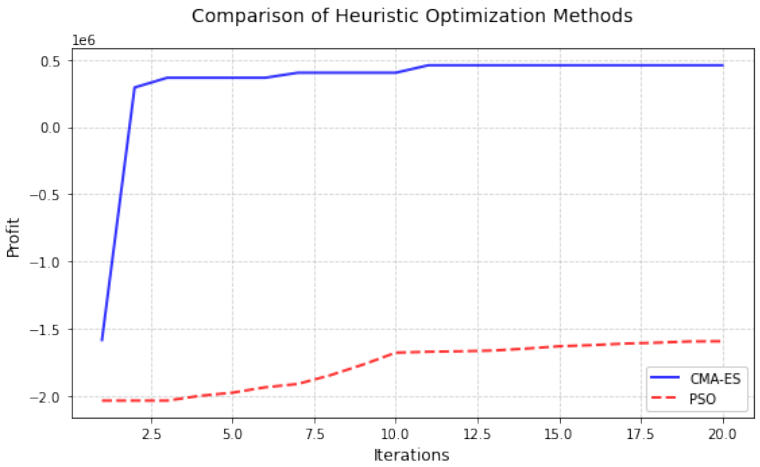

Figure 4.1: Comparing performance of CMA-ES vs PSO

Given that the impact of price adjustments varies across components, a uniform increase is not uniformly advantageous. Consequently, further refinement of the solution is necessary through optimization techniques. In this study, we employ CMA-ES to fine-tune the solution, with particular attention to the adaptation of the step size parameter (σ)

The behavior of the assembler's objective function, as illustrated in the problem landscape depicted in Figure 4.2, can be examined under two distinct directional perturbations. The first direction, denoted as $_1$, is analytically defined as the scaled difference between the upper bound vector UB, derived from Lemma 2.1, and the lower bound vector LB, which comprises the production costs of individual components in a vector, such that

$$\Delta_1 = (UB - LB)/100$$

The second direction, Δ_2 is characterized as a uniform step size across all dimensions, expressed as

$$\Delta_2 = 0.01 \cdot 1_{|\mathcal{I}|}$$

,where represents a unit vector multiplied by a constant whose dimensionality corresponds to the number of components under consideration. Empirical observations over 200 iterations reveal a consistent and interpretable pattern in the objective function's response to these directional changes. Specifically, the objective exhibits an initial phase of rapid ascent, followed by a distinct peak beyond which the function experiences a gradual decline in value. This phenomenon aligns with economic intuition: while incremental

increases in supply initially enhance the assembler's capacity to fulfill market demand—thereby improving profitability—exceeding a critical threshold leads to diminishing returns, as excessive pricing adversely affects overall profit margins. This behavior underscores the existence of an optimal operational range within which the assembler can maximize profitability before encountering the detrimental effects of overpricing.

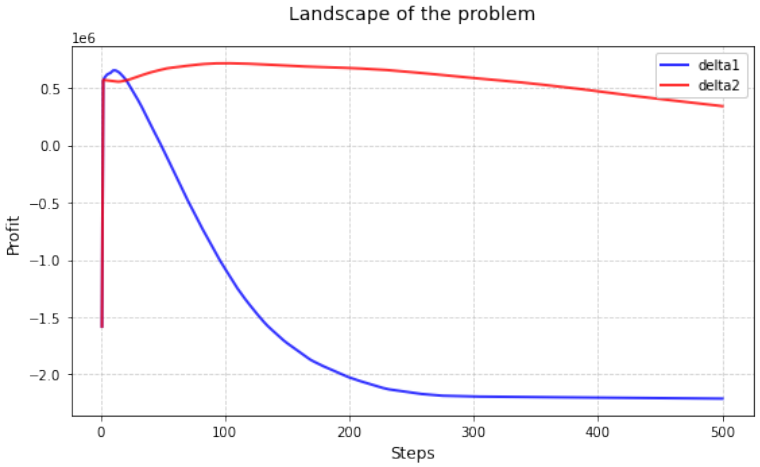

Figure 4.2: Illustration of the behaviour of the objective

Once a high-quality initial solution was obtained through the application of our proposed heuristic, the CMA-ES algorithm was executed with varying values of the step-size parameter σ, as illustrated in Figure 4.3. The results demonstrate that retaining the same σ value used in prior stages proves suboptimal once an effective initial solution has been identified. In contrast, empirical evidence indicates that reducing σ by an order of magnitude yields significantly improved convergence behavior. Consequently, the primary remaining optimization task involves the precise calibration of σ to determine an optimal value that balances exploration and exploitation, thereby maximizing algorithmic performance.

4.3 Centralized vs Decentralized

This section presents a systematic comparison between the performance outcomes of centralized and decentralized operational models under varying demand conditions. Specifically, we examine how each model responds to demand patterns generated from different probability distributions, which

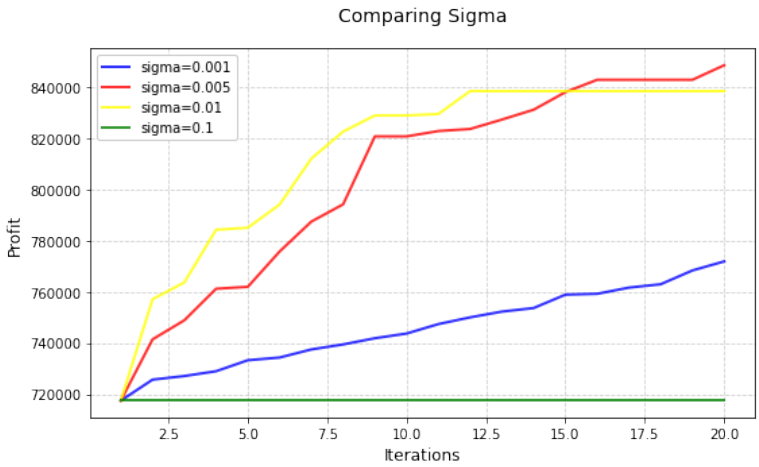

Figure 4.3: Comparing different sigmas

allows us to assess their robustness across diverse market scenarios.

Through rigorous empirical analysis of the results, we derive several critical managerial insights that address the fundamental question of optimal model selection.

A preliminary examination of the results reveals that both operational models fail to generate positive profitability when confronted with left-skewed demand distributions ($\beta(1, 4)$). This phenomenon proves particularly pronounced in the decentralized model, where the observed outcome can be attributed to suppliers' rational response to anticipated heavy-tailed demand characteristics.

The suppliers' strategic behavior manifests as a requirement for substantially increased component prices to justify expanded production capacity. This pricing strategy emerges as a direct consequence of the Newsvendor problem's closed-form solution, which governs profit-maximization behavior at the individual supplier level. Specifically, when facing demand distributions with significant left-skewness, the critical fractile solution compels suppliers to adopt more conservative production strategies unless compensated through higher marginal returns. The mathematical foundation for this behavior can be expressed through the Newsvendor optimization framework:

$$q^* = F^{-1}\left(\frac{p-c}{p-s}\right) \tag{4.1}$$

where F^{-1} represents the inverse CDF of the demand distribution, p denotes

the selling price, c the production cost, and s the salvage value. For $\beta(1, 4)$ distributed demand, the inverse CDF yields particularly conservative order quantities unless $(p - c)$ demonstrates substantial margin improvement.

In the case of high uncertainty which is described by a multi-model distribution (BiGaussian), the centralized model proved to be much more effective 4.5

When examining less challenging demand distributions, as illustrated in Figures 4.4 and 4.7, our analysis reveals that the decentralized model does not consistently generate inferior profitability compared to its centralized counterpart. Specifically, the centralized model demonstrates suboptimal performance relative to the decentralized approach in scenarios where the assembler's internal production capacity proves insufficient to meet market demand. This capacity constraint creates a critical inflection point in the make-or-buy decision framework, wherein outsourcing component production becomes the economically superior strategy.

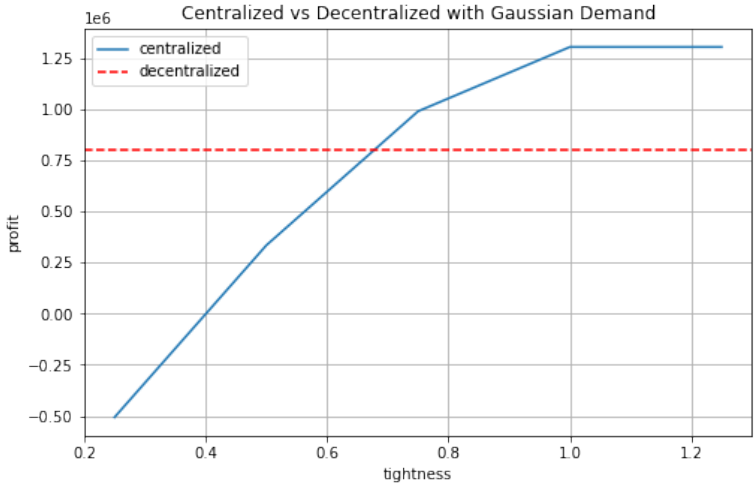

Figure 4.4: Centralized model in comparison decentralized model with gaussian demand

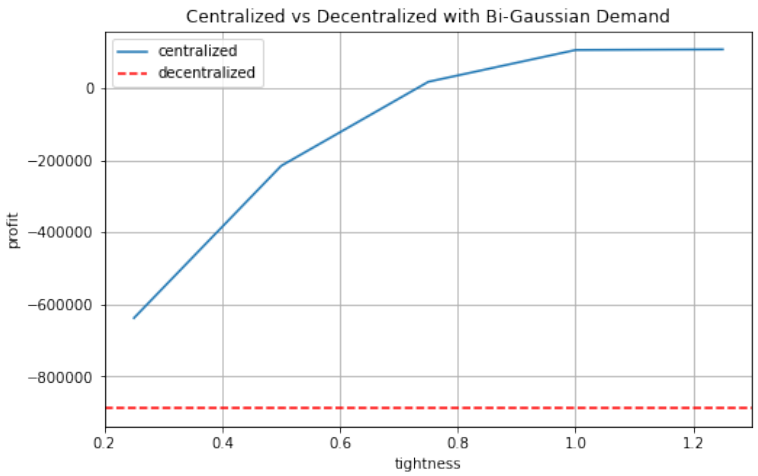

Figure 4.5: Centralized model in comparison decentralized model with gaussian demand

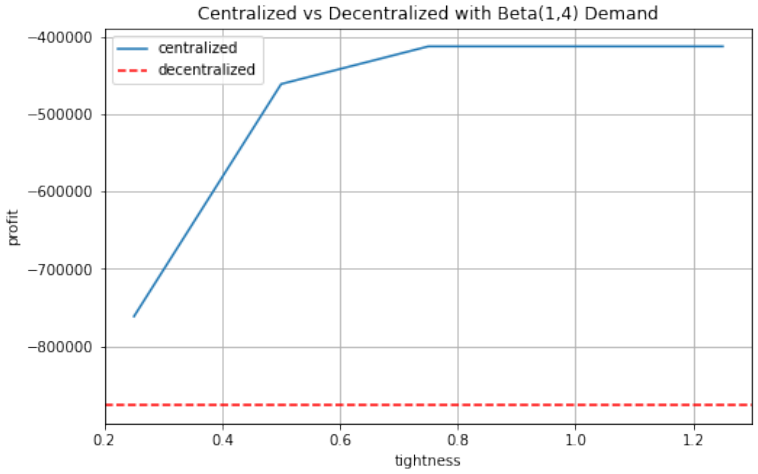

Figure 4.6: Centralized model in comparison decentralized model with gaussian demand

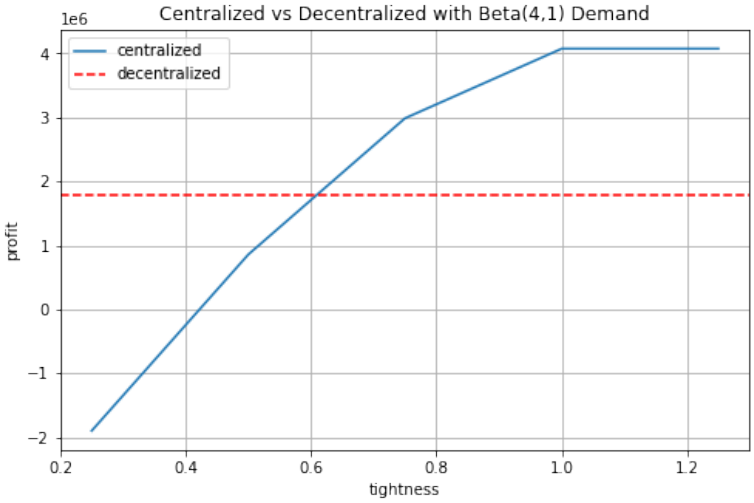

Figure 4.7: Centralized model in comparison decentralized model with gaussian demand

Chapter 5

Conclusions

In this work, we present a novel formulation for the decentralized assemble-to-order (ATO) system. The proposed model addresses the optimal pricing of individual components to ensure that the assembler maintains an efficient inventory level in anticipation of future demand. Additionally, we introduce a heuristic algorithm designed to yield high-quality solutions.

The heuristic ascent direction algorithm (Algorithm 3) operates in two key phases. First, it identifies a heuristic ascent direction and iteratively progresses along this direction until profit begins to decline, at which point the best solution encountered during the search is selected. Subsequently, the second phase employs supplementary optimization techniques to further refine the obtained solution.

Additionally, we analyze the performance of the decentralized model across different demand distributions in comparison with the centralized benchmark, offering actionable managerial implications. However, in the analysis of both centralized and decentralized model, the present study is limited to the procurement phase. Although our model generates component allocation plans for various scenarios, it does not prescribe an execution strategy once actual demand materializes.

A promising direction for future research involves developing a methodology capable of deriving optimal solutions—or at least high-quality approximations—for the decentralized model. Additionally, we intend to extend this framework by formulating the problem within a multi-stage stochastic optimization setting.

Bibliography

N. Agrawal and M. A. Cohen. Optimal material control in an assembly system with component commonality. *Naval Research Logistics (NRL)*, 48(5):409–429, June 2001. ISSN 1520-6750. doi: 10.1002/nav.1026. URL http://dx.doi.org/10.1002/nav.1026.

Z. Atan, T. Ahmadi, C. Stegehuis, T. d. Kok, and I. Adan. Assemble-to-order systems: A review. *Eur. J. Oper. Res.*, 261(3):866–879, Sept. 2017.

F. Bernstein, G. A. DeCroix, and Y. Wang. Incentives and commonality in a decentralized multiproduct assembly system. *Operations Research*, 55 (4):630–646, 2007. doi: 10.1287/opre.1070.0402. URL https://doi.org/10.1287/opre.1070.0402.

P. Brandimarte, E. Fadda, and A. Gennaro. *The Value of the Stochastic Solution in a Two-Stage Assembly-to-Order Problem*, page 105–116. Springer International Publishing, 2021. ISBN 9783030868413. doi: 10.1007/978-3-030-86841-3_9. URL http://dx.doi.org/10.1007/978-3-030-86841-3_9.

E. Fadda, D. G. Gioia, and P. Brandimarte. *Risk-averse Approaches for a Two-Stage Assembly-to-Order Problem*, page 147–156. Springer Nature Switzerland, 2023. ISBN 9783031288630. doi: 10.1007/978-3-031-28863-0_13. URL http://dx.doi.org/10.1007/978-3-031-28863-0_13.

Y. Gerchak and M. Henig. Component commonality in assemble-to-order systems: Models and properties. *Nav. Res. Logist.*, 36(1):61–68, Feb. 1989.

Y. Gerchak and Y. Wang. Revenue-sharing vs. wholesale-price contracts in assembly systems with random demand. *Production and Operations Management*, 13(1):23–33, 2004. doi: 10.1111/j.1937-5956.2004.tb00142.x. URL https://doi.org/10.1111/j.1937-5956.2004.tb00142.x.

D. G. Gioia, E. Fadda, and P. Brandimarte. Rolling horizon policies for multi-stage stochastic assemble-to-order problems. *Int. J. Prod. Res.*, 62 (14):5108–5126, July 2024.

A. Kok, de and S. Graves, editors. *Supply chain management : design, coordination and operation*. Handbooks in operations research and management science. Elsevier, Netherlands, 2003. ISBN 978-0-444-51328-1.

K. Rosling. Optimal inventory policies for assembly systems under random demands. *Operations Research*, 37(4):565–579, Aug. 1989. ISSN 1526-5463. doi: 10.1287/opre.37.4.565. URL `http://dx.doi.org/10.1287/opre.37.4.565`.

J.-S. Song and P. Zipkin. Supply chain operations: Assemble-to-order systems. In *Supply Chain Management: Design, Coordination and Operation*, Handbooks in operations research and management science, pages 561–596. Elsevier, 2003.

List of Algorithms

List of Tables

List of Figures