Erasmus University Rotterdam
Erasmus School of Economics

Master Thesis

Memetic Algorithm for Stochastic Inventory Optimization with Seasonal Demand

Author:
Seong Ho Lee
384055

Supervisor:
Prof. Dr. Ir. Rommert Dekker
Co-reader:
Dr. A. S. Eruguz Çolak

Operations Research & Quantitative Logistics
Econometrics and Management Science

July 24, 2018

Erasmus Universiteit Rotterdam
Erasmus School of Economics

Abstract

Due to the increasing complexity of logistics networks driven by rapid globalization and technological transformation, a naive inventory management leads to serious damage to both the business and its supply chain. Therefore, achieving and maintaining the optimal balance between cost reduction and customer service has become an important strategic concern for all types of organizations.

In this thesis, we study a single item, single-echelon inventory replenishment system with stochastic seasonal demand and stochastic lead time. Our objective is to find a time-varying, periodic review (T, R_t, Q_t) policy that minimizes holding, ordering, reviewing and purchasing costs under a minimum fill rate constraint. We propose a simulation-based stochastic optimization framework based on a new Memetic Algorithm (MA), which combines a Genetic Algorithm (GA) with a gradient-free Local Search Algorithm (LSA). The performance of the MA has been validated by solving a benchmark experiment with 18 different test instances and by comparing the solution quality with a benchmark model. Furthermore, its performance and robustness have been evaluated with extensive tests and numerical analyses. Our computational experiments show that the proposed MA obtains significantly better solutions than the benchmark method and that it is robust to problem parameters and inventory structures.

Keywords: *Stochastic Inventory Control, Seasonal Demand, Memetic Algorithm, Simulation Optimization, Anticipation Inventory*

Acknowledgement

I would like to express my gratitude and respect to my supervisor Prof. Dr. Rommert Dekker for his inspiring and valuable guidance during the entire research project. Without his support or supervision, it would not have been possible to complete my thesis.

I would also like to thank Dr. A. S. Eruguz Çolak for her constructive comments during this research project. Her meaningful advices have greatly improved the quality of the thesis.

Finally, I am most grateful to my family and girlfriend for their unfailing support and warm encouragement during the entire duration of my studies. This accomplishment would not have been possible without their love and care. Thank you.

Contents

List of Tables

List of Figures

1 Introduction

Seasonal demand is common in many industries, including high-tech, toy, fashion and pharmaceutical sectors (Gardner Jr and Diaz-Saiz, 2002; Metters, 1998), but managing an inventory with seasonal demand is often far more complicated than it might first appear. Unlike with stationary demand, seasonal fluctuation requires a dynamic replenishment strategy that carefully balances the trade-off between holding large inventory during seasons with low demand (or *off-seasons*) and facing high shortage risk during seasons with high demand (or *peak seasons*). Likewise, determining an optimal dynamic replenishment strategy requires complex decisions of when and how much to adjust the order quantity across seasons. Despite this problem's high practical significance, available techniques are rather scarce in the literature (Grewal et al., 2015).

In this thesis, we consider a stochastic variant of a single item, single-echelon inventory system under seasonal customer demand and stochastic lead time. Also, the inventory operates in a lost sales system and it needs to achieve a target customer service level, which is a commonly observed situation in many retail inventories (Bijvank and Vis, 2011). For this inventory system, we investigate the problem of determining a cost minimizing replenishment strategy while achieving a sufficient level of customer service. Moreover, in order to properly manage seasonal demand we consider a time-varying periodic replenishment strategy (T, R_t, Q_t), of which the reorder point R_t and the reorder quantity Q_t may vary throughout the seasonal demand cycle. Assuming that the exact length of each demand cycle is known a priori and that the number of seasons during a demand cycle is fixed, our problem also concerns the optimal timing of the seasonal adjustments made to R_t and Q_t.

In general, many stochastic inventory problems that are recognized in practice are analytically intractable because of their complexity. To overcome our problem's complexity and modeling difficulties, we propose an effective simulation-based optimization model, which is a commonly applied framework for approximating the optimal solutions for practical inventory problems (Jalali and Nieuwenhuyse, 2015). Our solution framework combines a stochastic inventory simulation with a metaheuristic model without posing unrealistic restrictions on the problem. Also, a simulation-based approach may produce more detailed and visual information about the inner dynamics of inventory operations, providing meaningful insights for its subsequent analysis.

The objectives of our thesis are threefold. The first objective is to create an efficient approach to determine the optimal dynamic replenishment strategy under the described inventory system. Our second objective is to verify and validate the performance of our proposed approach for solving the problem under various demand, cost and inventory configurations. Finally, we aim to conduct a scientific numerical analysis to examine the method's behavior under different inventory parameters and cost structures. The remainder of the paper is structured as follows. In Section 2 we present a comprehensive overview of existing literature on related inventory problems and solution techniques. Section 3 contains a formal description of the considered problem as well as its formulation into a stochastic non-linear programming model. In Section 4 our method is described with a proper set of notations. The computational results and numerical analyses are presented in Section 5, and in Section 6 we present our conclusions together with discussion and limitations of our research. The suggestions for the future research are also provided in this final section.

2 Literature Review

2.1 Introduction

In this section, we provide a general review on inventory control problems for a single-echelon supply chain (Section 2.3). The focus of this review is on inventory replenishment problems with non-stationary demand, stochastic lead time and service level constraint due to their direct relevance to our research. Also, in Section 2.4 we provide a detailed summary of available solution techniques for the relevant problems in Section 2.3 with a focus on simulation-optimization and metaheuristic approaches.

2.2 Different Ordering Policies

Among many different ordering policies to control stochastic inventory system, (R, Q) policy and (s, S) policy are the two most commonly studied policies in the literature (Axsäter, 2015). Both R and s denote for the reorder points in each of these two policies. Q stands for the reorder quantity, also referred to as the batch quantity or the lot size, and S is the order-up-to level or the maximum inventory level. In the next two subsections, these two ordering policies and their key variants are briefly explained for our literature review.

2.2.1 (R, Q) Policy

For a (R, Q) policy, a replenishment order of a fixed size Q is generated whenever the inventory position (i.e., the number of on hand stock plus outstanding orders minus backorders) declines to or below the reorder point R. In case of continuous review and continuous demand, the inventory position increases by the size of reorder quantity Q whenever the reorder point R is hit exactly as shown in Figure 1. On the other hand, the inventory level (i.e., the number of on hand stock minus backorders) that does not consider the number of outstanding orders remains below R during the lead time.

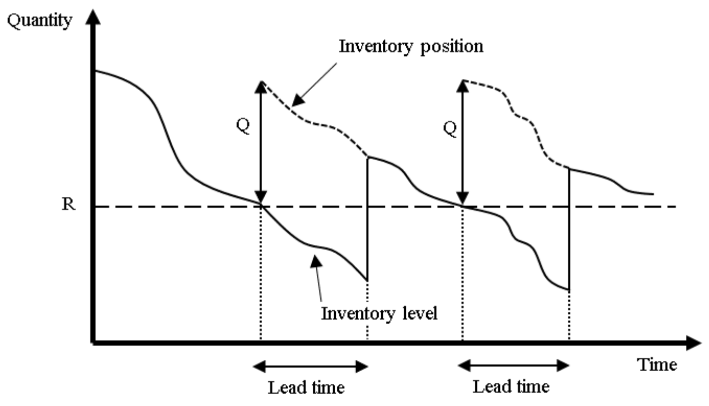


Figure 1: Continuous review (R, Q) policy with continuous demand

In case of periodic review (R, Q) with review period T, the inventory position is inspected at every T time units. Figure 2 illustrates the periodic review (T, R, Q) policy with fixed lead time. Unlike the continuous review analogue, the inventory position increases by Q only if it is below R at review period.

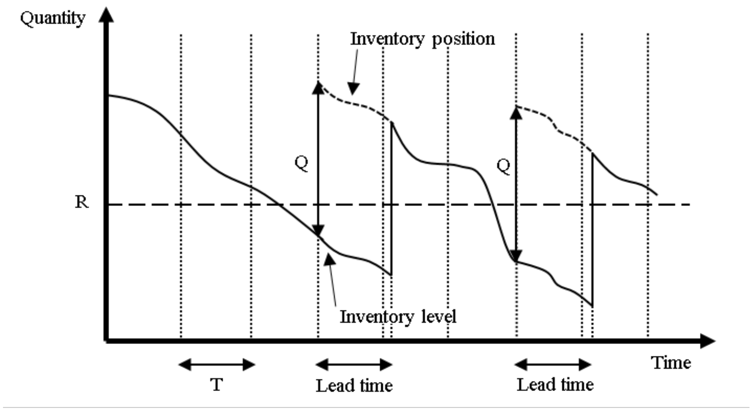


Figure 2: Periodic review (T, R, Q) policy with continuous demand

In addition to the static (R, Q) policy, some authors also consider the dynamic (R_t, Q_t) policy that applies time-varying reorder point R_t and reorder quantity Q_t at time t. Also, a (R, nQ) policy is commonly used in the literature, where it is possible to order more than one batch quantity Q until the inventory position reaches above the reorder point R.

2.2.2 (s, S) Policy

An (s, S) policy, also known as a min/max policy, sets a minimum and a maximum level of the inventory position. When the inventory position declines to or below the reorder point s, a replenishment order of size $S - s$ is issued to bring the inventory position up to the order-up-to level S. For the continuous review case, the inventory position hits the reorder point exactly and it is immediately brought up to the level S. Figure 3 illustrates the policy in a continuous review, continuous demand situation.

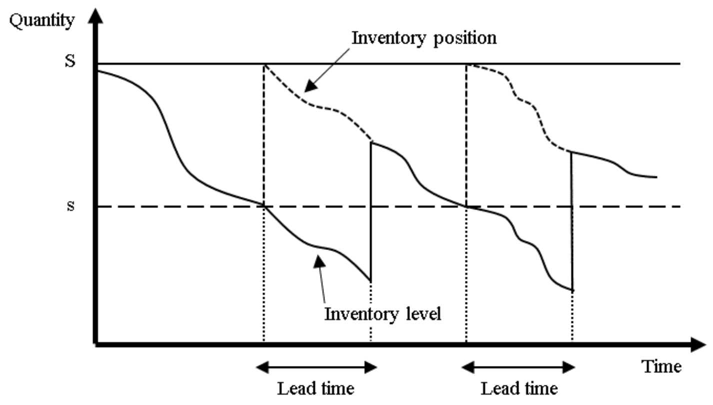


Figure 3: Continuous review (s, S) policy with continuous demand

In case of the periodic review (T, s, S) policy, the inventory position is reviewed at every T time units and the inventory position is brought up to S if it is below or equal to the reorder point s. Figure 4 demonstrates this policy. Again, for periods other than review time, the inventory position does not increase.

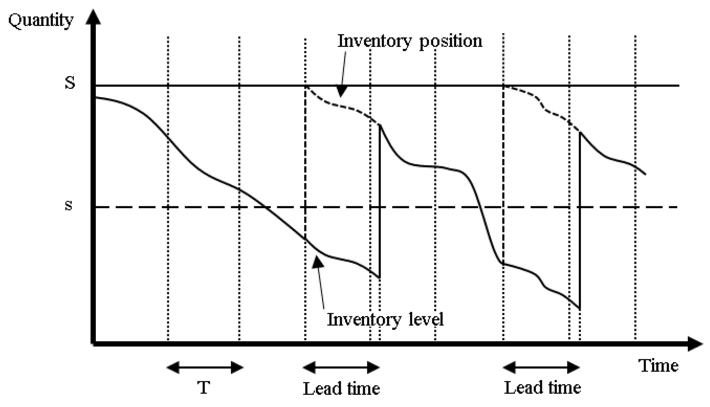


Figure 4: Periodic review (T, s, S) policy with continuous demand

Another important variant of the (s, S) policy is a *base stock* policy, or an $(S - 1, S)$ policy, which always orders up to the level S whenever the period demand is greater than zero. Therefore, in a continuous review base stock policy, every positive demand in each period leads to a replenishment order of size equal to the period demand.

2.3 Inventory Control Problems

2.3.1 Lot Sizing Problem

A *lot-sizing* problem considers the determination of the optimal batch quantity in a replenishment policy. When using a static (R, Q) replenishment policy, this corresponds to the determination of optimal level and timing of the reorder quantity Q. The typical objective of this type of problem is to balance inventory holding and ordering cost without creating shortages. General lot-sizing models impose the assumption of exactly known future demand and infinite time horizon. The classical economic order quantity (EOQ) equation by Harris (1991) is one of such models, in which the cost optimal batch quantity is determined under constant purchase, ordering and holding costs. There are many variants of this simple EOQ model that extend to problems with multiple items (Sana, 2010), quantity discounts (Benton and Park, 1996) and backorders (Mak, 1987; Pentico and Drake, 2009).

In practical inventory problems, however, time-varying customer demands are more commonly observed. For example, customer demand for medicines, winter coats and air conditioners have strong seasonal variations (Feng and Gallego, 1995). For these products, considering a class of *dynamic* lot-sizing models is more appropriate, as these models can manage time-varying demand over a finite number of discrete time periods with linear holding cost. For the uncapacitated version of the dynamic lot sizing problem, Wagner and Whitin (1958) proposed the first dynamic programming approach with the quadratic computational complexity, which means that the computation time is proportional to the square of the number of periods in the considered problem, to solve the deterministic case exactly. Later, various extensions of the Wagner-Whitin algorithm were proposed, including the work of Zangwill (1966) that studies the backordering case and the work of Heady and Zhu (1994) which integrates the concept of Economic Part-Period into the planning-horizon theorem to achieve an increased performance.

Despite the proven optimality of the exact methods, in practice it is more common to use simple heuristic methods to obtain an approximate, yet “good enough” solution. While an efficient implementation of the exact methods can greatly reduce the computation time, the alternative heuristics are still more preferred in practice because of their simplicity and adaptability (Yilmaz, 1992). One of the most popular heuristics is the Silver-Meal method, which is a simple sequential method proposed by Silver and Meal (1973). This heuristic technique determines the order timing and quantity by iteratively computing the average cost per period for orders that would cover successive periods. When the average cost increases for the first time, one should order the total demand for the periods that are covered. Other alternative procedures are Least Unit Cost heuristic by Gorham (1968), which considers the average cost per order quantity instead of cost per time like in Silver-Meal heuristic, and Part-Period Balancing heuristic by DeMatteis (1968), which orders the quantity that keeps the ordering cost as close to the holding cost as possible.

So far, we have discussed the lot-sizing problems with deterministic demand. Nevertheless, most of lot-sizing problems that are encountered in practice deals with stochastic demand. This is because the future demand is usually only available in forms of forecasts and the presence of forecast errors introduces random variation in the future demand (Tarim and Kingsman, 2004). Silver (1978) proposed a heuristic method for the stochastic lot-sizing problem with Normally distributed forecast errors. His heuristic determines when to order, the size of order and the number of periods which an order needs to cover, while keeping sufficient service level. Furthermore, Bookbinder and Tan (1988) considered the same problem in a rolling horizon environment, in which the lot size is determined for a tighter finite horizon than the original horizon, which rolls over the complete horizon. They formulated a "dynamic uncertainty" strategy to determine the lot size for the future periods based on the demand that become updated at a later point in time.

2.3.2 Dynamic Ordering Policy

Existing literature on optimization of the static ordering policy is flourishing, yet relatively little attention was given to the dynamic ordering policy. One of the earliest publications is the study of Karlin (1960), which shows the optimality of the (s, S) policy for a non-stationary demand with a special problem structure. The author showed that for periods with stochastically low demand, the order-up-to level S decreases while the opposite is not necessarily true. Nevertheless, the proposed approach requires complex computation and makes restrictive assumptions, including the integrability of the objective function.

More recently, Graves (1999) developed a myopic (i.e., short-period ahead), time-dependent ordering policy based on demand forecast made from a simple exponential smoothing model. Assuming for deterministic lead time, the reorder quantity consists of the demand for the immediate period as well as the expected change in the forecast. Despite its simplicity, however, the model does not take into account service level constraint and stochastic lead time, hence limiting its applicability in practice. Babaï and Dallery (2006) proposed a sequential approach to approximate the optimal parameters of a dynamic (R_t, Q) policy for a general class of inventory problems with a fill rate constraint. The authors derived a simple formula for the optimal safety stock based on a given series of time-dependent reorder points assuming that the demand distribution is independent and identically distributed. As their model is applicable to the system with non-stationary demand and a minimum fill rate constraint, their model is very relevant to our own. Yet, their model does not optimize the review time and applies a constant reorder quantity Q instead of dynamic quantities Q_t.

Besides the objective of minimizing the total cost, Grewal et al. (2015) focused on minimizing the work-in-progress and finished goods inventory by means of a simulation-based optimization algorithm. Their algorithm decides a dynamic (R_t, Q_t) replenishment strategy with a fixed number of adjustments of policy parameters in a seasonal cycle. Given the exact seasonal pattern of future demand during any given seasonal cycle, the optimal strategy determines the timing of seasons within a demand cycle as well as the size of the reorder point and the reorder quantity in each season. They show that effective seasonal adjustments were able to stabilize the finished goods inventory level but made no significant improvements for the work-in-progress inventory.

2.3.3 Service Level Constraint

An important criterion for the optimal reorder policy is the policy's expected customer service level. Since an inventory's service level is generally expressed in terms of size or frequency of shortages (Axsäter, 2015), a sufficient number of safety stock needs to be maintained in order to achieve the desired level of customer service. In some inventories, shortage or backordering costs are charged instead of monitoring the service level. But in practice the determination of stock out costs is considered to be very difficult (Hadley and Whitin, 1963).

Only a few case studies have been conducted for setting an appropriate level of shortage cost. Chang and Niland (1967) stated that the expected shortage cost may be modeled in terms of direct measurements and conditional costs, such as the number of past stock-outs, the cost of emergency procurement and the opportunity cost of losing customer goodwill. However, it may be impossible to accurately estimate the abstract measurements such as the loss of customer goodwill due to their subjective nature (Dion et al., 1991). An intuitive approach is to conduct direct interviews and market surveys on target customers, but such a method requires costly market investigations and is valid under very restrictive premises (Oral et al., 1972). In some situations, shortage costs may be directly available from contracts that state shortage penalties, but even in such cases monitoring the service level would be useful for understanding the performance of the inventory system (Diks et al., 1996).

There are two most commonly used definitions of the inventory service level in the literature (Ronen, 1983; Axsäter, 2015).

1. Cycle Service Level: probability of not stocking out in an order cycle
2. Fill Rate: proportion of demand that can be immediately satisfied from available stock on hand

Cycle service level can be understood as the probability of an order arriving before the stock on hand is depleted. This definition is quite simple to implement because the optimal order point satisfying the cycle service level does not get affected by the change in the lengths of replenishment cycles (Schneider, 1981). However, it does not consider the actual size of batch quantity, resulting in a significant divergence from the "real" service level that indicates how many demands are directly satisfied (Axsäter, 2015). Several indicative examples of single-echelon inventory models with cycle service level constraints include Schneider (1981), Babaï et al. (2009) and Axsäter (2015).

In practice and in a major part of literature, fill rate definition is more frequently employed as the main service measure as it considers both the possibility of stock-out situations and the size of the unmet demand (Tempelmeier, 2007). For a multi-item environment, an order-based fill rate may be used to estimate the probability of serving all items in the customer order instead of the typical item-based fill rate that considers individual item (Song, 1998). Available literature for the single-item inventory problem with a minimum fill rate constraint is extensive and they employ great varieties of optimization techniques, including constrained optimization models (Bashyam and Fu, 1998; Jha and Shanker, 2009), approximation heuristics (Axsäter, 2006; van Donselaar and Broekmeulen, 2013), and simulation-based optimization models (Fu, 2002; Kleijnen et al., 2010).

2.3.4 Stochastic Lead Time

Incorporating stochastic lead times in inventory models brings remarkable impacts on the optimal reorder policies and the system performance. The most common assumption is that the lead time is independent of demand and order size, and that the orders do not cross in time, meaning that the replenishment orders issued earlier do not arrive later than the orders issued later in time (Axsäter, 2015; Zipkin, 1986).

For finite horizon problems, Song et al. (2010) established important effects of lead time volatility in the standard (R, Q) policy with a compound Poisson demand process. They showed that a larger lead time volatility leads to a higher optimal reorder point but does not necessarily result in a higher optimal average cost. Due to the added stochasticity, earlier works often modeled the problem as a queuing system with random order arrivals (Berman and Kim, 1999; Kaplan, 1970). Kaplan (1970) successfully implemented a finite-horizon dynamic programming model and observed that the optimal policy was dependent on whether ordering cost is fixed or variable. An interesting extension was studied by Ehrhardt (1984) who included a salvage value for the left-over inventory at the end of the planning periods and showed that a myopic solution is optimal in case of zero fixed ordering cost.

2.3.5 Lost Sales System

In a lot of commercial inventories in retail sectors, majority of customers who observe a stock-out gets lost immediately (Gruen et al., 2002), and therefore a lost sales model is more appropriate to represent these cases. However, many of the inventory models studied in academics assume that excess demand is backordered, meaning that customers will wait until the next delivery of replenishment order and purchase the items when made available. The central reason for this preference is the problem's relatively simple structure that facilitates better theoretical analysis and that requires less computational effort (Bijvank and Vis, 2011). As a matter of fact, (s, S) type policies are proven to be optimal for backorder models under periodic review but an optimal policy for the lost sales model is not trivial (Scarf, 1959). Also, a greedy approximation of the lost sales system by a similar backorder model may lead up to 30% deviations of the optimal cost (Zipkin, 2008*a*).

Extensive research has been performed to optimize continuous review inventory with lost sales system, and most of these works focus either on an (R, Q) or an (s, S) type ordering policy. For static (R, Q) policies, an exact expression for the expected total cost has been derived under the Poisson demand process and with the assumption of at most one outstanding order (Hadley and Whitin, 1963). Then this model has been extended to incorporate stochastic lead time (Buchanan and Love, 1985; Johansen and Thorstenson, 1993), general demand distribution (Mohebbi and Posner, 1998) and discounted model (Johansen and Thorstenson, 1996). In addition, Aardal et al. (1989) considered the same problem with a minimum service level constraint and solved it by using Lagrangian multipliers. She modeled the relationship between the shortage cost and service level, and also addressed that the shadow price of the service level constraint corresponds to the shortage cost. In case of (s, S) policies, Archibald (1981) obtained the exact expression for the average stationary cost with the discrete compound Poisson demand process and applied a small grid search for obtaining the global optimum. He also proposed a near-optimal approximation procedure using the optimal policy in the equivalent backorder model. Kalpakam and Sapna (1994) considered the inventory problem of perishable items with general lead time distribution, where they obtained the steady-state performances and investigated the analytical properties of the expected cost.

In periodic review system, existing models can be classified by whether the fixed order costs are zero or positive (Bijvank and Vis, 2011). In case of no fixed ordering cost, Morton (1969) provided the upper and lower bounds on the optimal order quantity and the minimum expected total cost for problems with linear holding cost and variable ordering cost. It also conditions that the lead time is an integral multiple of the review period. Then Zipkin (2008*b*) extended the results of Morton (1969) to prove that the optimal cost function is a *L-natural* convex function. This means the function is convex and submodular in discrete number space, and it implies that the optimal order quantities are monotonically decreasing with respect to the inventory position. When there is a fixed ordering cost per each order, a periodic review (T, s, S) policy is shown to be optimal under zero lead time (Veinott, 1966; Cheng and Sethi, 1999). Yet, the optimal policy for positive lead time case is unknown. The numerical analysis performed by Hill and Johansen (2006) showed that both (R, Q) and (s, S) policies are not optimal for this problem, but if the parameters are selected carefully they can provide near-optimal performance. Finally, Bijvank and Vis (2012) derived lower and upper bounds on the order-up-to level for the lost sales model with a service level constraint, which could effectively narrow the parameter search space.

2.4 Solution Methods

2.4.1 Exact Method

The optimal solution technique, often referred to as an *exact method*, can identify a globally optimal solution to an optimization problem. Due to its guarantee of global optimality, numerous researchers have been keen on formulating efficient exact methods. However, exact methods often require intense computational effort and time, which is burdensome in day-to-day industrial applications. Moreover, the underlying assumptions may be in appropriate for many inventory systems in real life.

Since the research in stochastic inventory system is flourishing and continues to expand, it is hard to generalize the existing exact methods into a single global framework. Instead, we refer the readers to the work of Axsäter (2015), which provides extensive results on theoretically optimal ordering policies for both single- and multi-echelon systems. The author extensively studies and develops several different analytical models for the optimal determination of the reorder policy under various discrete and continuous demand distributions.

2.4.2 Simulation-Optimization

A computer-based simulation model can be employed as a potentially powerful alternative to the traditional methods because it can model the complex inventory operations as well as the inherent system uncertainties without making unrealistic assumptions (Glover et al., 1999). Simulation-optimization, also referred to as simulation-based optimization, is an optimization technique that enhances the system's simulation model with a suitable optimization technique to efficiently search for the optimal decision variables (Nguyen et al., 2014). Unlike the classical analytic methods that determine the optimal decision variables by algebraically minimizing the objective function, simulation-optimization models evaluate system performance by combining and processing the responses from the simulation model and therefore does not require the exact gradient information of the objective function (Mele et al., 2006).

In the field of inventory control systems, several works have integrated optimization techniques with inventory simulation models and the two most popular techniques are simple metaheuristics and hybrid metaheuristic methods (Jalali and Nieuwenhuyse, 2015).

The most common metaheuristic algorithm employed for the inventory problems is Genetic Algorithm (GA) due to its flexibility and ability to handle simulation noise (Andradóttir, 2006). GA is a population-based search heuristic that iterates through fitness assessment, parent selection and reproduction from an initial population of candidate solutions. It is generally applied to optimize discrete decision variables because the number of feasible solutions is unbounded for continuous problems. Application of GA on inventory problems spans from single-echelon (Yang et al., 2012; Pasandideh et al., 2013; Saracoglu et al., 2014) to multi-echelon (Köchel and Nieländer, 2005; Min et al., 2006; Pasandideh et al., 2011) supply chain. The algorithm's main advantage is its flexibility to handle different inventory settings, such as multi-item situation (Pasandideh et al., 2011), storage constraints (Mandal et al., 2011), and purchase discounts (Taleizadeh et al., 2010). Other popular metaheuristics are Simulated Annealing, Tabu Search and Particle Swarm Pptimization algorithms (Silva et al., 2003; Altiparmak et al., 2006) and the comprehensive survey of their applications is available in the paper of Jalali and Nieuwenhuyse (2015).

Hybrid metaheuristic algorithms combine different individual algorithms in order to improve the overall search performance. The main benefit of mixing different algorithms derives from the synergy that improves both explorative and exploitative power of the resulting algorithm. The most common hybridization approach is to augment an explorative algorithm, such as GA, with some exploitative algorithms (e.g., Local Search Algorithm) (Luke, 2009). Memetic Algorithm (MA), which was coined in Moscato et al. (1989), is a class of such hybrid algorithms which combines population-based algorithms with individual local search algorithms. There have been several works that applied MA to inventory management systems, such as the machine scheduling problem (França et al., 2001; Mendes, Alexandre S and Müller, Felipe M and França, Paulo M and Moscato, Pablo, 2002) and lot-sizing problems (Berretta and Rodrigues, 2004), but only a few of these papers consider inventory replenishment systems. The work of Pasandideh et al. (2013), for an example, finds the optimal reorder quantities for multiple items by applying an MA that combines a standard GA with a Simulated Annealing method for the initial population generation.

3 Problem Description

3.1 Introduction

In this section, we properly introduce the inventory problem considered in this thesis and state the scope of our research by presenting the research questions we would like to answer from this project. In Section 3.2, the inventory system considered in this thesis is described together with a simple supply chain representation. In Section 3.3 a formal description of our problem is presented and in Section 3.4 our problem is formulated as a stochastic non-linear programming problem. In Section 3.5 and 3.6 we address the central question of our research and provide the list of problem assumptions that we imposed for our scientific research.

3.2 System Description

Our inventory model fits in a serial supply chain with a supplier, a warehouse and customers, in which a single type of item is delivered to the customers via the warehouse. The warehouse serves the customers' orders using its available inventory and by regularly procuring the product from the supplier. As soon as the supplier receives a new purchase order from the warehouse, it processes and transports the order to the warehouse after finite periods of lead time. Under the discrete time system, all the operations take place in discrete time periods and no events are assumed to occur during the time in between. Figure 5 illustrates the main process of the described serial supply chain.

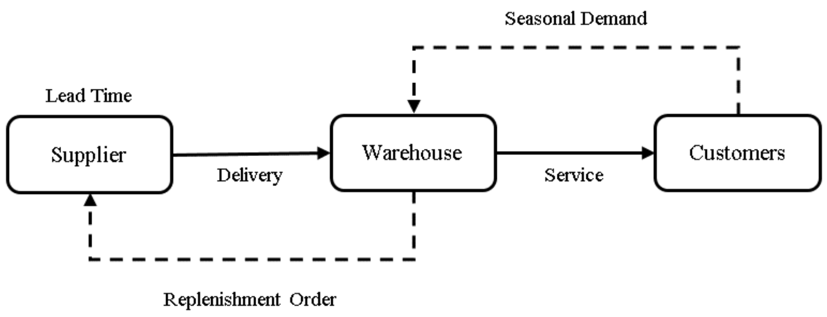


Figure 5: Demonstration of the single-echelon serial supply chain

In the considered inventory system, the uncertainty originates from the demand and lead time. The number of customer demand in each period is assumed to follow a discrete Normal distribution with a seasonal mean and a constant standard deviation. The lead time is a random variable from a Geometric distribution with a known parameter value. At first, such distributional assumptions may seem rather impractical. Yet, many researchers and practitioners support that these are suitable for items with typically high demand. Firstly, the high demand item is usually bought by multiple customers who make decisions independent of each other, and by the Central Limit Theorem the sum of these independent random demands tends toward a Normal distribution under very general conditions (Axsäter, 2015). Since in our system the product is exchanged in discrete units, we use a discrete Normal distribution instead to model customer demand. Its probability mass function is presented in Appendix A.2. Secondly, the Geometric distribution is a discrete analogue of the continuous Exponential distribution, which is commonly adapted when modeling inventory systems due to its tractability and broad applicability in real world situations (Ramasesh et al., 1991; Tijms, 2003).

3.3 Formal Problem Description

Consider the inventory that operates until a finite planning horizon N for equally spaced discrete time periods $t = 1, 2, \dots, N$. The inventory stocks a single type of product with high, seasonal customer demand d_t and it follows a discrete Normal distribution with a time-varying mean $\mu_t \geq 0$ and a constant standard deviation $\sigma \geq 0$. Also, because of the product's high demand the probability of seeing negative demand is none and having zero demand quantity is absolutely small. Thus, the demand distribution's coefficient of variation, which is the ratio σ/μ_t, is smaller than 1/4 for all t, and any negative demand is rounded up to zero. Assume that the seasonal pattern in μ_t is uniquely defined over a seasonal cycle of length Y and that the same pattern repeats until the planning horizon N for an integer number of times. Therefore, the planning horizon N is an integral multiple of the cycle length, such that $N = nY$ for $n = 1, 2, \dots$. See Figure 6 for an example of two seasonal demand cycles, each with length $Y = 52$ periods. The inventory applies lost sales system, in which the excess demand above the stock on hand gets lost immediately. The replenishment lead time is Geometrically distributed with a known rate $0 < q < 1$, and therefore the lead time for any replenishment orders takes a positive non-zero integer value. We mention that our problem's special seasonality structure is originally from the seasonality structure presented by Grewal et al. (2015).

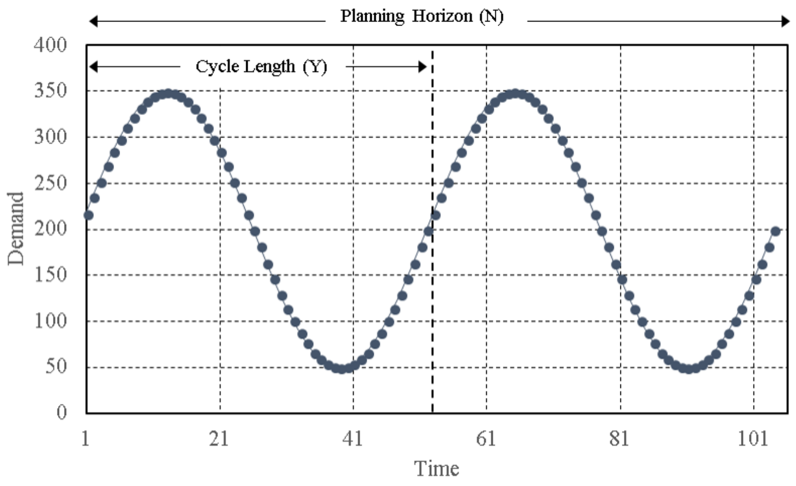


Figure 6: Two seasonal cycles with $N = 2Y$

The ordering decisions are made based on the inventory position IP_t at the end of period t. The inventory position is defined as a linear combination of three inventory measures; the physical stock on hand OH_t, the outstanding orders OO_t that are yet to be delivered from the supplier, and the number of backorders, which is the demand that has not been satisfied immediately. For our lost sales inventory system, the number of backorders is zero because there cannot be any backorders if all excess demand gets lost.

$$IP_t = OH_t + OO_t \tag{1}$$

While all replenishment decisions are made based on IP_t, the cost of carrying inventory is determined based on the inventory level IL_t. It is equal to the number of stock on hand minus the number of backorders, but under lost sales system, there is no backorder.

$$IL_t = OH_t \tag{2}$$

Therefore, in lost sales system both IP_t and IL_t can never become negative.

The inventory level is controlled by a seasonal (T, R_t, Q_t) policy for all periods $t = 1, \dots, N$, where T is a review time, R_t is a reorder point, and Q_t is a reorder quantity for time t. All three types of decision variables take strictly positive integer values and T is bounded by the interval $[1, T_{\max}]$ where $T_{\max}$ is a positive integer upper bound for T that is larger or equal 1. Under this periodic ordering system, the inventory position is reviewed at every T time periods. If IP_t declines to or below the reorder point R_t at time t, a fixed reorder quantity Q_t is ordered from the supplier. Furthermore, we assume that exactly m adjustments are made to the pair of reorder point and reorder quantity (R_t, Q_t) within a seasonal demand cycle of Y periods. See Figure 7 for an example of the seasonal policy for two cycles of length $Y = 21$, each with three seasons $(m = 3)$. The frequency m is an integer valued design parameter that is chosen from the interval $[1, Y]$. If $m = 1$ then the policy is essentially a static (T, R, Q) policy, and if $m = Y$ then the policy is a fully dynamic (T, R_t, Q_t) policy that may take different values for every time $t = 1, \dots, N$.

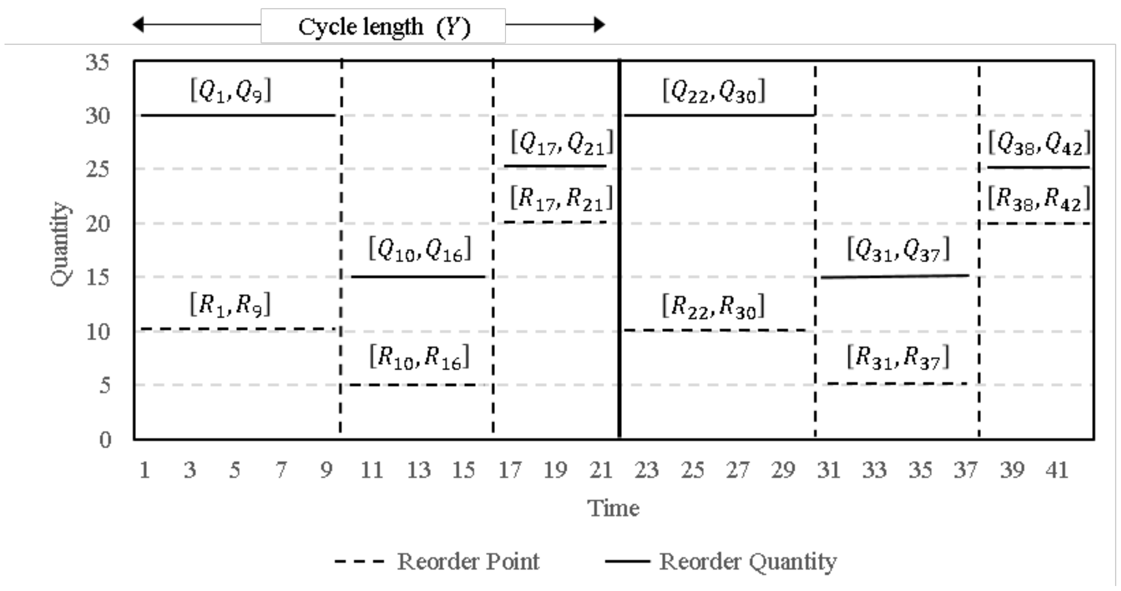


Figure 7: Example (T, R_t, Q_t) policy with $m = 3$

In every review period, a replenishment order of size Q_t is submitted to the supplier when the inventory position at the beginning of period t has dropped to or below the reorder point. We assume that at most one replenishment order can be issued in each time period and that the lead time L_t for the order can be larger than the length of review time T. Furthermore, we ensure that the replenishment orders cannot cross in time, meaning that the orders issued at later time periods cannot be delivered earlier than the orders issued earlier. This implies that the following inequality $t + L_t \leq t + x + L_{t+x}$ always holds for $x = 1, 2, \dots$. We also assume that the supplier has an unlimited supply capacity for any replenishment quantities at any point of time and that there is no restriction on the maximum number of outstanding orders.

Before serving any demand at time t, the inventory receives all past replenishment quantities that were expected to arrive at time t. Therefore, when these orders arrive at time t the inventory level increases by $EQ_t = \sum_{\{k|1\leq k\leq t, k+L_k=t\}} Q_k x_k$, where x_k is a binary variable that equals to 1 if a replenishment order was placed at time k and 0 otherwise. After receiving the replenishment orders, customer demand at t is served with available stock on hand and any excess demand above the available stock is lost without penalties. The number of lost sales in each period is obtained by $LS_t = \max\{0, d_t - IL_{t-1} - EQ_t\}$ for $t = 1, 2, \ldots, N$.

Within every time period t, three inventory operations may take place: *inventory review*, *order arrival* and *demand satisfaction*. Each of these operations has a purpose of checking inventory position for possible replenishment orders, receiving past replenishment orders and meeting customer demand, respectively. In case multiple operations take place in the same period, they are managed in the order of review, order arrival and demand satisfaction. That is, the inventory position is reviewed first if t belongs to a review period, and then all previously issued replenishment orders with scheduled arrival at t are received. After completing all reviews and order receipts the period demand is served with available inventory.

Four types of costs are considered in this problem. Setting up a replenishment order incurs a fixed order set up cost $K \geq 0$ per unit order. Carrying positive inventory incurs a per unit per time unit holding cost of $h \geq 0$. Purchasing a unit of item costs $p \geq 0$ and reviewing the inventory at any point of time incurs a review cost of $r \geq 0$ per review. Review cost typically includes the costs of administration and labor involved during the inspection process (Metzger et al., 2013). When review cost is neglected, then setting the smallest discrete review time ($T = 1$) might become optimal and the model essentially reduces to a discrete time continuous review model. In order to study the effect of different values of T, we consider a general model with a positive review cost.

Define a replenishment strategy $\mathcal{P}$ as a unique sequence of (T, R_t, Q_t) policies with m seasons within a seasonal cycle. The objective of our problem is to determine the optimal replenishment strategy $\mathcal{P}^*$ that minimizes the expected total cost while maintaining a sufficient quality of expected customer service. The expected total cost for a replenishment strategy $f(\mathcal{P})$ is a sum of expected purchasing, holding, ordering and review cost for all demand and lead time scenarios during N periods. Let $\hat{d}_t$ and $\hat{L}_t$ denote a possible realization of the period demand d_t and the lead time L_t for an order placed at time t, respectively. Then the expected total cost is expressed as in Equation (3).

$$f(\mathcal{P}) = r\left\lfloor \frac{N}{T} \right\rfloor + \sum_{\hat{d}_1} \cdots \sum_{\hat{d}_N} \sum_{\hat{L}_1} \cdots \sum_{\hat{L}_N} \left(\sum_{t=1}^{N} hIL_t + Kx_t + pQ_t x_t \right) g_1(\hat{d}_1) \cdots g_N(\hat{d}_N) v(\hat{L}_1, x_1 = 1) \cdots v(\hat{L}_N, x_N = 1) \tag{3}$$

where the floor function $\lfloor a \rfloor$ finds the largest integer smaller or equal to real number a, $g_t(\hat{d}_t)$ is the probability mass of receiving demand $\hat{d}_t$ at time t, and $v(\hat{L}_t, x_t = 1)$ obtains the joint probability of having a lead time of $\hat{L}_t$ and setting up a reorder at time t. We use joint probability for the lead time because the lead time at time t is not realized unless a replenishment order is actually issued at t. The extra subscript t for the function $g_t(\cdot)$ is used for non-stationary demand distribution at time t, unlike the joint probability for the lead time $v(\cdot)$ that is identical across all time periods.

The expected service level is expressed as the expected *item fill rate*, $0 \leq FR(\mathcal{P}) \leq 1$. Since the fraction of satisfied demand is equivalent to one minus the fraction of lost sales to the total demand, we obtain

$$FR(\mathcal{P}) = 1 - \sum_{\hat{d}_1} \cdots \sum_{\hat{d}_N} \sum_{\hat{L}_1} \cdots \sum_{\hat{L}_N} \left(\frac{\sum_{t=1}^{N} LS_t}{\sum_{t=1}^{N} \hat{d}_t} \right) g_1(\hat{d}_1) \cdots g_N(\hat{d}_N) v(\hat{L}_1, x_1 = 1) \cdots v(\hat{L}_N, x_N = 1) \quad (4)$$

A replenishment strategy is called feasible if its expected fill rate is higher than a minimum threshold. We denote a feasible replenishment strategy by adding a hat to its standard notation, such as $\hat{\mathcal{P}}$ and its expected item fill rate is above a lower bound $0 \leq FR_{\min} \leq 1$, such that

$$FR(\hat{\mathcal{P}}) \geq FR_{\min} \quad (5)$$

Therefore, our objective is to minimize Equation (3) with the minimum service level constraint in Equation (5). In next section, we formulate our problem as a stochastic non-linear programming problem for a coherent presentation of our inventory system.

3.4 Stochastic Non-Linear Programming Model

Due to stochastic problem parameters and non-linear constraints, our problem can be more accurately modeled by a Stochastic Non-linear Programming (SNLP) model. The problem consists of the decisions of the review time T, reorder point R_t and reorder quantity Q_t to minimize the expected total cost subject to demand and lead time uncertainties. The non-linear constraints consist of maximum functions and division operators that are implemented for calculating expected lost sales and for expressing the item fill rate condition. Also, the binary decision variable x_t that indicates a setup of order at time t involves a non-linear modulus operator. We present our SNLP model using the maximum function $(y)^+ = \max\{y, 0\}$ that finds the positive part of the random variable y and the modulus operator $a \bmod b$ that finds the remainder from the division of real number a by another real number b.

Stochastic Non-Linear Programming Model

$$\min \quad f(\mathcal{P}) \quad \text{(Equation (3))} \tag{6}$$

$$\text{subject to} \quad EQ_t = \sum_{\{k \mid 1 \le k \le t, k+\hat{L}_k = t\}} Q_k x_k \qquad t = 1, 2, \ldots, N \tag{7}$$

$$IP_t = (IP_{t-1} - \hat{d}_t + Q_t x_t)^+ \qquad t = 1, 2, \ldots, N \tag{8}$$

$$x_t = \begin{cases} 1 & \text{if } IP_{t-1} \le R_t \text{ and } t \bmod T = 0 \\ 0 & \text{otherwise} \end{cases} \qquad t = 1, 2, \ldots, N \tag{9}$$

$$IL_t = (IL_{t-1} + EQ_t - \hat{d}_t)^+ \qquad t = 1, 2, \ldots, N \tag{10}$$

$$LS_t = (\hat{d}_t - IL_{t-1} - EQ_t)^+ \qquad t = 1, 2, \ldots, N \tag{11}$$

$$1 - \frac{\sum_{t=1}^{N} LS_t}{\sum_{t=1}^{N} \hat{d}_t} \ge FR_{\min} \tag{12}$$

$$N \bmod Y = 0 \tag{13}$$

$$R_t = R_{t+Y} \qquad t = 1, 2, \ldots, N - Y \tag{14}$$

$$Q_t = Q_{t+Y} \qquad t = 1, 2, \ldots, N - Y \tag{15}$$

$$y_t = \begin{cases} 0 & \text{if } |R_t - R_Y| = 0 \text{ and } |Q_t - Q_Y| = 0 & t = 1 \\ 0 & \text{if } |R_t - R_{t-1}| = 0 \text{ and } |Q_t - Q_{t-1}| = 0 & t = 2, 3, \ldots, Y \\ 1 & \text{otherwise} & t = 1, 2, \ldots, Y \end{cases} \tag{16}$$

$$\max \left\{ \sum_{t=1}^{Y} y_t, 1 \right\} \le m \tag{17}$$

$$IL_0 = IP_0 = LS_0 = EQ_0 = 0 \tag{18}$$

$$Q_t, R_t, IL_t, IP_t, LS_t, EQ_t \in \mathbb{Z}^+ \qquad t = 1, 2, \ldots, N \tag{19}$$

$$x_t \in \{0, 1\} \qquad t = 1, 2, \ldots, N \tag{20}$$

$$y_t \in \{0, 1\} \qquad t = 1, 2, \ldots, Y \tag{21}$$

$$T \in \{1, 2, \ldots, T_{\max}\} \tag{22}$$

The objective function (Equation (6)) is equal to the sum of expected holding, purchasing, review and order setup cost throughout the planning horizon. The exact expression for the expected total cost $f(\mathcal{P})$ can be found in Equation (3). Equation (7) finds the number of items scheduled to arrive at time t. Equation (8) states that the inventory position at the end of time t is a function of previous inventory position at time $t-1$, demand at

t and the amount of reorder quantity placed at time t. Equation (9) defines the binary variable x_t that equals to 1 if a replenishment order is placed at time t and 0 otherwise. A replenishment order is placed when the inventory position at time $t-1$ is below or equal to the reorder point R_t and when t is an integral multiple of the review period T. Equation (10) updates the inventory level at the end of time t based on the scheduled receipt and observed demand at t. Equation (11) states that the number of lost sales at t is equal to the number of customer demand that cannot be immediately satisfied from the inventory at t. Equation (12) expresses the service level constraint in terms of lost sales compared to the total demand throughout the planning horizon. Equation (13) to (17) ensure that the optimal replenishment strategy is adjusted seasonally during each demand cycle. Equation (13) ensures that the planning horizon consists of an integer number of repetitions of a seasonal demand cycle. Equation (14) and (15) express the repetition of identical reorder policies over each seasonal cycle. Equation (16) evaluates if the reorder policy of period t has been adjusted from that of preceding period $t-1$ (or Y if $t=1$). Equation (17) ensures that the number of seasonal adjustments made to the reorder policy during each demand cycle does not exceed the number of seasons m. Since observing zero seasonal adjustment indicates that the strategy has only one season, the equation's maximum operator compares the number of seasonal adjustments with 1. Equation (18) sets the initial values for the inventory level, inventory position, lost sales and scheduled arrivals. In Equation (19) all variables are defined as non-negative integer variables from an unbounded integer set, while in Equation (20) and (21) all binary variables are defined. Finally, Equation (22) addresses that the optimal review time takes an integer value between 1 and $T_{\max}$.

To solve the proposed stochastic non-linear programming problem with existing stochastic programming techniques, the objective function and the constraints should be accurately linearized and reformulated into its deterministic analogue. This might involve many auxiliary variables and additional assumptions that can eventually make the problem analytically intractable. In our research, we present an alternative metaheuristic approach that obtains near-optimal replenishment strategy without such extensive reformulation.

3.5 Research Questions

The main contribution of our research is to provide a new simulation-optimization model for a realistic inventory replenishment system with stochastic seasonal demand and service level constraint. While inventory models for stationary demand have been extensively studied, models that handle demand seasonality with seasonal replenishment strategy are rarely found in the literature. To fill this research gap, we aim to investigate the optimal characteristics of the replenishment strategy that can protect from the high shortage risk during peak seasons and reduce excess stock in off-seasons through anticipation. The model's efficiency and robustness will be evaluated in multiple aspects with critical numerical analysis and benchmark testing.

The central research question of our thesis is formulated as follows

Central question How can we determine the cost-optimal seasonal replenishment strategy for a single item, single-echelon inventory system with stochastic seasonal demand, stochastic lead time and service level constraint in the lost sales environment?

The central question is divided into following sub-questions about the effects of demand seasonality and service level constraint, as well as the proposed model's optimization performance. Furthermore, the section numbers in brackets address the relevant sections of our paper that are devoted to answer each of these sub-questions.

- What is the effect of demand seasonality on optimal replenishment strategy and how can we determine the optimal timing of seasons? How can the amount of anticipation inventory be determined in order to protect against projected demand fluctuations? (Section 5.4.3 and 5.5.1)
- How can the inventory operations be simulated? What are the inventory events and underlying assumptions for the simulation model? How can we integrate the simulation model into the optimization process? (Section 4.2 and 4.3.2)
- How can the proposed simulation-optimization method incorporate and handle demand and lead time uncertainty? (Section 4.3.3)
- How can the proposed model handle the service level constraint and what is the effective size of penalty for constraint violation during the optimization process? (Section 4.4.4)
- How does the proposed method perform in benchmark experiments against a competing method in the literature? What are the reasons for differences in comparison to the benchmark method? (Section 5.4)
- How robust is the proposed method to different demand, cost and inventory structures? Are all operators and decisions included in the proposed method valid and effective? (Section 5.5 and Section 5.6)

Our thesis aims to identify scientific answers to these research questions and provide fruitful insights for handling demand seasonality under service level constraint and lost sales system.

3.6 Problem Assumptions

In order to conduct more focused and efficient research, we limited our research scope by imposing following list of assumptions. The assumptions are classified into three separate classes of inventory model, supply and demand depending on which part of the supply chain they would make the most direct impact.

Inventory model

- Single item, single-echelon inventory system
- Multi-period with a finite planning horizon and discrete time system
- Minimum item fill rate constraint
- Seasonal replenishment strategy with integer valued review period, reorder point and reorder quantity policy per time unit (T, R_t, Q_t)
- Fixed number of seasons for the replenishment strategy
- No returns, reservations and discounts
- Lost sales system
- Imperishable goods and no obsolescence
- Inventory operations take place in the order of inventory review - order arrival - demand satisfaction
- Unlimited storage capacity

Supply

- Geometrically distributed lead time with a known distribution parameter
- Replenishment orders cannot cross in time
- Unlimited number of outstanding orders
- Replenishment orders do not arrive in pieces over time
- Lead time can be larger than the review period T

Demand

- Stochastic seasonal demand from positive, discrete Normal distribution with known distribution parameters
- Demand is insensitive to the endogenous factors such as lost sales and lead time
- Coefficient of variation for the demand distribution in each period is less than 1/4
- Length of seasonal demand cycle is known exactly

4 Methodology

4.1 Introduction

In this section, we describe our new simulation-based metaheuristic method that determines the optimal replenishment strategy by utilizing a stochastic simulation model of the inventory replenishment system. We begin by introducing the overall structure of our two-phase simulation-optimization framework and continue with explaining all elements of each phase in more detail.

Our simulation-optimization algorithm involves two-way interactions between the discrete-event simulation and the stochastic optimization. In the simulation phase, the performance of a replenishment strategy of interest is evaluated by simulating over demand and lead time scenarios, generating numerical estimates of the expected total cost and service level. In the optimization phase, our optimization algorithm utilizes the simulation output to search for an improved replenishment policy. This simulation-optimization process repeats until certain termination criterion is met, which is manually set by the user.

In Section 4.2, a graphical overview of the simulation-optimization structure is presented. In Section 4.3, we introduce the Discrete-Event Simulation (DES) model that evaluates the performance of the considered replenishment system. In Section 4.4 we propose our new Memetic Algorithm (MA), which combines a Genetic Algorithm (GA) and a gradient-free Local Search Algorithm (LSA) in order to identify a near-optimal seasonal replenishment strategy. Finally in Section 4.5, we give a summary of the main steps involved in our simulation-optimization model.

4.2 Simulation-Optimization Structure

The structure diagram in Figure 8 illustrates how a replenishment strategy $\mathcal{P}$ from the optimization procedure is simulated in the simulation module, and the mean estimates of the total cost TC and the fill rate FR over $\tilde{S}$ simulation replications are used in the optimization module for the future search. Let TC_{best} be the expected total cost of the best replenishment strategy found by the algorithm until the considered iteration.

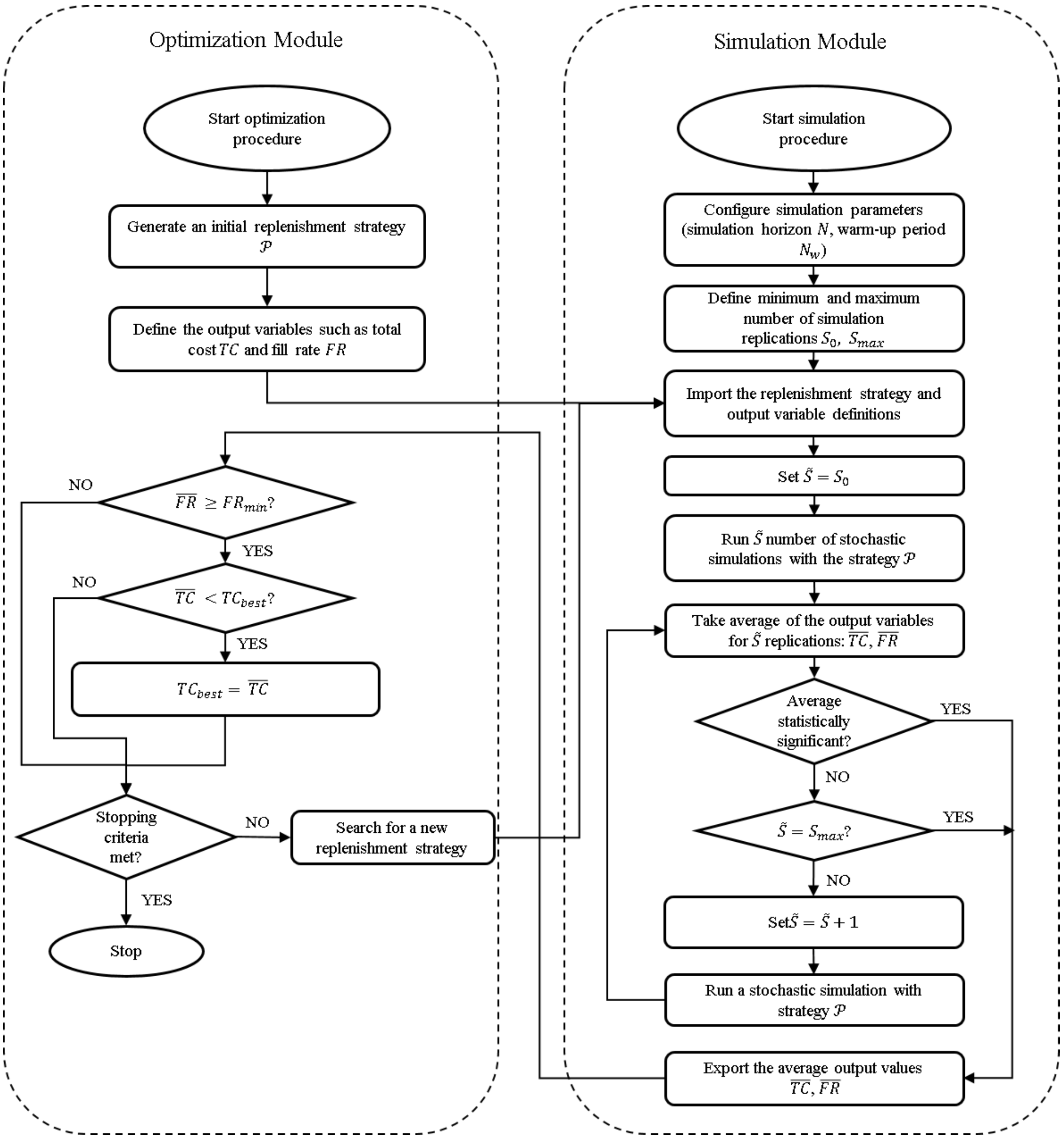

Optimization Module
Start optimization procedure
Generate an initial replenishment strategy $\mathcal{P}$
Define the output variables such as total cost TC and fill rate FR
$\overline{FR} \geq FR_{min}$?
NO
YES
$\overline{TC} < TC_{best}$?
NO
YES
$TC_{best} = \overline{TC}$
Stopping criteria met?
NO
Search for a new replenishment strategy
YES
Stop
Simulation Module
Start simulation procedure
Configure simulation parameters (simulation horizon N, warm-up period N_w)
Define minimum and maximum number of simulation replications S_0, S_{max}
Import the replenishment strategy and output variable definitions
Set $\tilde{S} = S_0$
Run $\tilde{S}$ number of stochastic simulations with the strategy $\mathcal{P}$
Take average of the output variables for $\tilde{S}$ replications: $\overline{TC}, \overline{FR}$
Average statistically significant?
YES
NO
$\tilde{S} = S_{max}$?
YES
NO
Set $\tilde{S} = \tilde{S} + 1$
Run a stochastic simulation with strategy $\mathcal{P}$
Export the average output values $\overline{TC}, \overline{FR}$

Figure 8: Simulation-Optimization structure diagram

At the beginning of the algorithm, an initial replenishment strategy is produced and definitions of the output variables are input into the optimization module. Then these data are passed to the simulation module that has been configured with initial simulation parameters such as planning horizon and simulation warm-up period. The main purpose of having an initial warm-up period is to attain steady-state conditions of the inventory level, the inventory position and the amount of replenishment orders that are being delivered. The simulation module evaluates the quality of the input replenishment strategy by simulating its performance for a finite number of times until certain stopping criterion is met. The simulation module reports the performance output of the replenishment strategy to the optimization module and the optimization module examines whether the considered strategy is an improvement or not. Until the termination criterion is satisfied, the optimization module continues to search for a new replenishment strategy and the whole simulation-optimization procedure is repeated.

4.3 Phase 1: Inventory Simulation

The performance of a replenishment strategy can be verified through a stochastic simulation model, which evaluates the strategy's expected ouput for various demand and lead time scenarios. Next two subsections are dedicated to describe the essentials of our stochastic simulation model. In Section 4.3.1, we introduce the seasonal demand generation function proposed by Grewal et al. (2015). In Section 4.3.2, the DES model for the stochastic inventory system is explained.

4.3.1 Seasonal Demand Generation

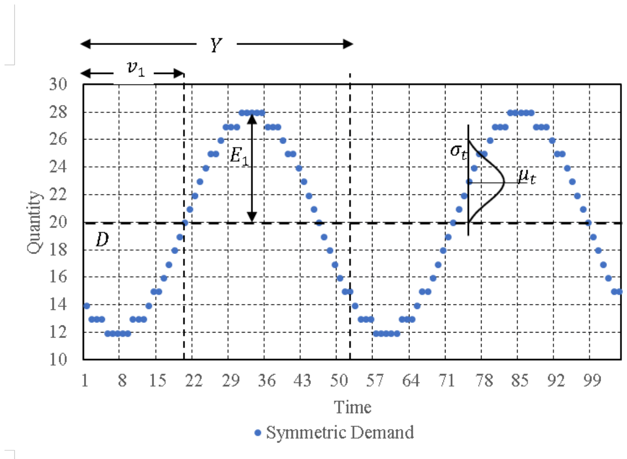
In order to generate various classes of stochastic seasonal demand, we adapted the special demand generating trigonometric function developed by Grewal et al. (2015), which can simulate both symmetric and asymmetric seasonal patterns that are commonly observed in practical applications. In addition, we generate mean μ_t instead of actually observed demand d_t to preserve demand uncertainty. Equation (23) generates a seasonal series of the expected demand rate μ_t at time t.

$$\mu_t = D + E_1 \sin\left\{\left(\frac{2\pi}{Y}(t - v_1)\right) + E_2 \cos\left(\frac{4\pi}{Y}(t - v_2)\right)\right\} \tag{23}$$

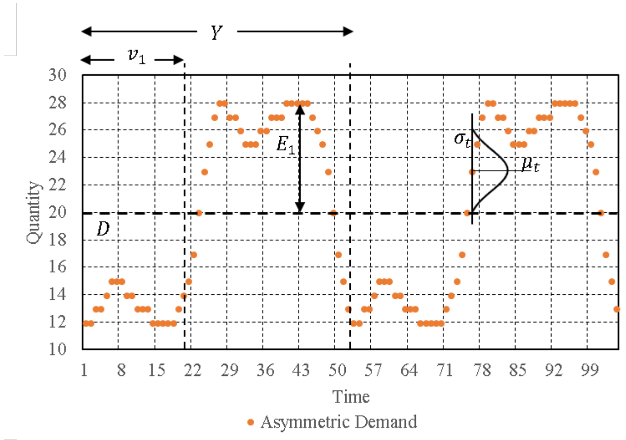
D is the mean demand level across the seasonal cycle, E_1 defines the amplitude of the demand seasonality, v_1 is the horizontal seasonality lag parameter, E_2 is the coefficient that controls the symmetry of the seasonality and v_2 is the symmetry lag parameter. Due to its cycle-based formulation, the same value of μ_t repeats in every cycle. That is, $\mu_t = \mu_{t+Y}$ for $t = 1, \ldots, N - Y + 1$.

A symmetrical seasonal pattern can be generated by setting $E_2 = 0$. For example, Figure 9a is a symmetrical seasonal pattern generated for weekly demand (i.e., $Y = 52$ weeks) by setting $D = 20$, $E_1 = 8$, $v_1 = 20$, $E_2 = 0$ and $v_2 = 0$. By setting $E_2 > 0$, we can generate an asymmetrical pattern as in Figure 9b. The figure shows a resulting pattern with $E_2 = 1$ and $v_2 = 5$ while holding all the other parameters identical.

The cosine function is contained inside the sine function to create various asymmetric patterns from the plain symmetric demand. The main purpose of considering asymmetric seasonality is to study the effects of intra-seasonal demand volatility on the identification of optimal seasons and the optimal reorder parameters. As can be seen from Figure 9b, the asymmetric seasonality contains multiple peaks and troughs within each season, and therefore the expected rate of inventory usage in a season is more variable than for the symmetric demand. In this case, it may be preferable to adjust the seasons and to alter the reorder policy for each season to effectively stabilize the inventory level.



(a) Symmetrical demand seasonality



(b) Asymmetrical demand seasonality

Figure 9: Trigonometric demand generating function

4.3.2 Discrete-Event Simulation

DES is an efficient stochastic simulation framework which models the procedure of a stochastic system as a sequence of unique time-based events. Every event occurs at its assigned time and uniquely alters the system's state. Unlike the discrete-time simulation, which simulates the system at every time period, DES simulates only at the occurrences of events and avoids unnecessary simulation during the times with no events. Since our inventory replenishment system is a discrete time system with integer valued review time T and discrete order lead time, the DES framework is used to effectively model our system.

Our replenishment system can be modeled with three unique events: REVIEW, ARRIVAL and DEMAND. In the REVIEW event, a new replenishment order is scheduled if the inventory position at certain moment is below or equal to the associated reorder point. In the ARRIVAL event, any past replenishment orders that were scheduled to arrive at the time of current simulation are received, and they increase the inventory level by the scheduled order quantities. In the DEMAND event, the period demand is served by the available stock on hand and any lost sales are recorded.

The flowchart in Figure 10 shows the DES process, which consists of four routines that are connected by bold and dotted arrows. For correct representation, the steps directed by dotted arrows should be followed first before following bold arrows. In the Initialization routine, the system state, simulation settings and counter variables are initialized. The simulation clock is initially set equal to 0 and it advances during the main routines. Also, the first REVIEW and DEMAND events are generated based on review time T and demand distribution, and they are entered in the event list. In the Timing routine, the next earliest event is invoked and the simulation clock is updated. The Event routine simulates the system operations according to the event's type and updates the system variables. It also generates a new event in the event list. Finally, when the simulation is complete by simulating all events up to the planning horizon, the Output generation routine reports the simulation results and the simulation process is terminated. For the complete DES algorithm, readers are referred to Appendix A.3.

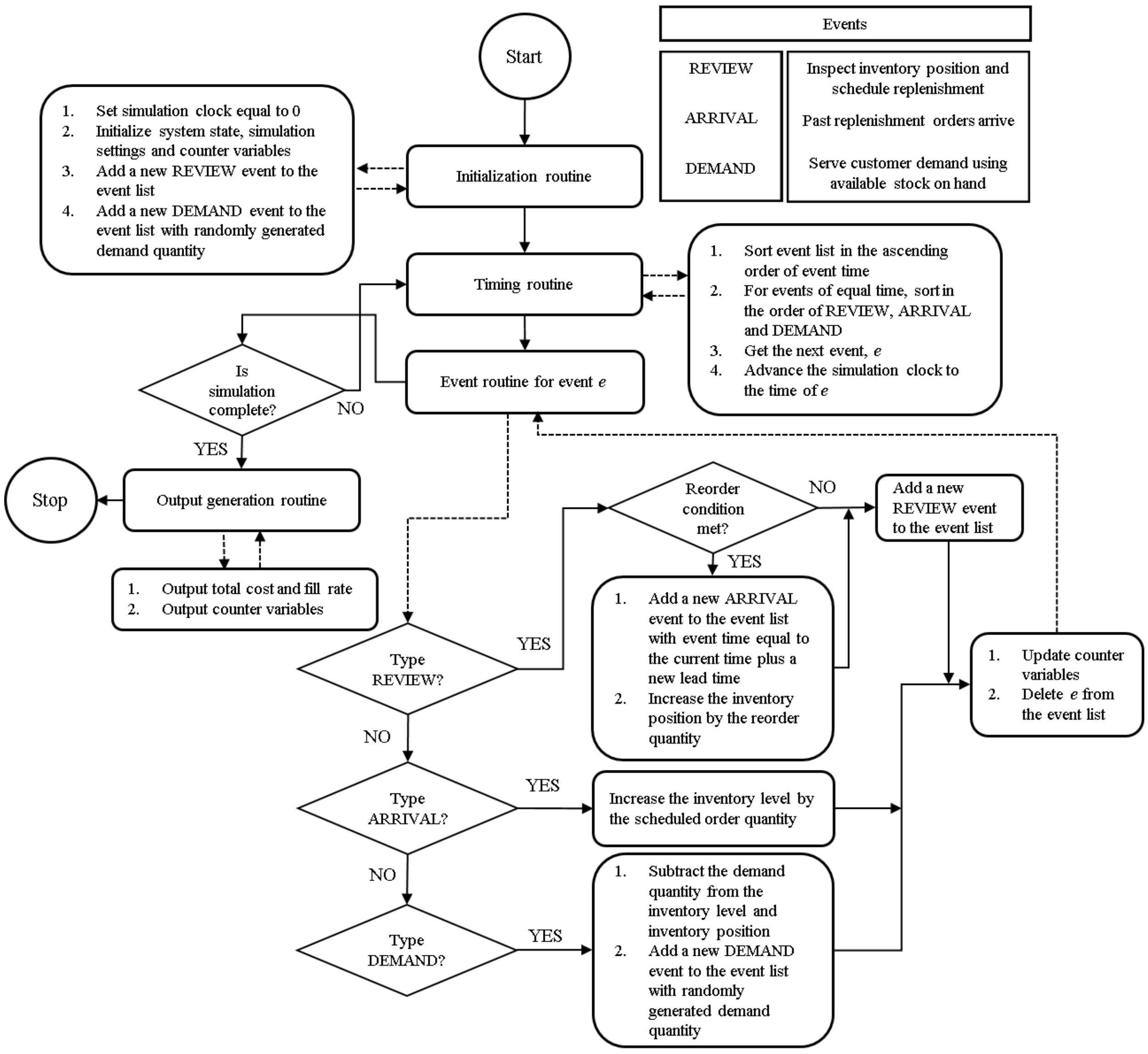

Start
Events
REVIEW
ARRIVAL
DEMAND
Inspect inventory position and schedule replenishment
Past replenishment orders arrive
Serve customer demand using available stock on hand
1. Set simulation clock equal to 0
2. Initialize system state, simulation settings and counter variables
3. Add a new REVIEW event to the event list
4. Add a new DEMAND event to the event list with randomly generated demand quantity
Initialization routine
Timing routine
1. Sort event list in the ascending order of event time
2. For events of equal time, sort in the order of REVIEW, ARRIVAL and DEMAND
3. Get the next event, e
4. Advance the simulation clock to the time of e
Is simulation complete?
NO
YES
Event routine for event e
Stop
Output generation routine
1. Output total cost and fill rate
2. Output counter variables
Reorder condition met?
NO
YES
Add a new REVIEW event to the event list
1. Add a new ARRIVAL event to the event list with event time equal to the current time plus a new lead time
2. Increase the inventory position by the reorder quantity
1. Update counter variables
2. Delete e from the event list
Type REVIEW?
YES
NO
Type ARRIVAL?
YES
Increase the inventory level by the scheduled order quantity
NO
Type DEMAND?
YES
1. Subtract the demand quantity from the inventory level and inventory position
2. Add a new DEMAND event to the event list with randomly generated demand quantity

Figure 10: DES process flowchart (follow dotted arrows first before bold arrows)

An important remark is that if an order crossing occurs, the DES resamples a new lead time from the Geometric distribution until all orders would arrive in a sequential order. In this manner, our simulation model is able to implement the non-order crossing condition. However, such a repetitive strategy may lead to a serious accumulation of long lead times once after an exceptionally long lead time has been simulated. For example, if the lead time of a replenishment order placed at $t = 1$ is excessively long then the lead time of an order at $t = 2$ should be long enough to prevent order crossing, which continues to apply in all subsequent time periods $t = 3, 4, \ldots, N$. As the lead times tend to become longer than the theoretical mean $1/q$, the results from the DES may not match the real-world performance. However, we emphasize that this was rarely observed in our computational experiments because of the Geometric distribution's positive skewness. The distribution's longer tail on higher values (i.e., right side of the distribution) keeps the likelihood of repetitively drawing long lead times very low.

4.3.3 Dynamic Resampling

Due to the inherent uncertainty in demand and lead time, the inventory simulation model outputs performance estimates that are also random variables themselves. Using the noisy estimates for optimization can lead to a longer search time and inaccurate identification of the optimal strategy (Branke et al., 2001; Jin and Branke, 2005). In order to counteract performance deterioration due to output noise, many researchers have successfully implemented explicit averaging techniques, in which the noiseless output variables are estimated by calculating the average of the output variables after simulating many different demand and lead time scenarios (Jin and Branke, 2005). While averaging over many samples are required to minimize the estimation error, it would significantly increase the computational time and effort. Therefore, it is essential to determine the minimum number of samples that can sufficiently compensate output noise for a fixed computation budget.

The inventory simulation model outputs two key performance indices, namely, expected total inventory cost and expected fill rate, and the noise intensity inherent in their sample values may be different for different replenishment strategies. To handle estimates with different variance, it appears more reasonable to sample more for high-noise solutions to obtain more reliable estimates, while sampling less for low-noise solutions to save computational budget. In the work of Di Pietro et al. (2004), this type of dynamic technique is referred to as *dynamic resampling technique* because it determines the appropriate number of samples for each individual solution with different noise levels.

In the paper of Di Pietro et al. (2004), the authors developed an intuitive resampling technique based on the standard error of the sample mean. The idea is straightforward: simulate performance of a given replenishment strategy and sample total cost and fill rate until the standard error of their sample mean is below a pre-specified threshold. By setting an identical threshold for all replenishment strategies, the resulting noise level is expected to be similar for all strategies. This is desirable for both estimation accuracy and performance comparison between different strategies.

In order to calculate standard error of a replenishment strategy $\mathcal{P}$, it is initially simulated for $S_0 \geq 2$ number of times. A larger size of S_0 would generate more reliable estimation of its true mean performance but would also require more computational effort. For every simulation instance $i = 1, \ldots, S_0$, we obtain a sample value of total cost $\widehat{f}_i(\mathcal{P})$ and fill rate $\widehat{FR}_i(\mathcal{P})$. Our goal is to estimate noise-free true value of expected total cost $f(\mathcal{P})$ and expected fill rate $FR(\mathcal{P})$ by averaging out noise. Thus, the resampling technique is applied to each of these two measures separately. Since the identical resampling procedure applies to estimate both measures, we proposed the dynamic resampling algorithm in terms of an auxiliary variable z that may be replaced by either of $\widehat{f}_i(\mathcal{P})$ or $\widehat{FR}_i(\mathcal{P})$. To ensure that the resulting mean estimates of $f(\mathcal{P})$ and $FR(\mathcal{P})$ have sufficiently low variance, the final sample size is set equal to the maximum of the two sample sizes, such that $S^* = \max\{S^*_{FR(\mathcal{P})}, S^*_{f(\mathcal{P})}\}$. Figure 11 shows the steps for the proposed dynamic resampling method.

Standard Error Dynamic Resampling by Di Pietro et al. (2004)

(1) Sample the value of z for S_0 number of times and set $S_z^* = S_0$

(2) Calculate its sample mean: $\bar{z} = \frac{\sum_{i=1}^{S_z^*} z_i}{S_z^*}$ where z_i is the i^{th} sample value of z

(3) Calculate its sample standard deviation: $\hat{s}_z = \sqrt{\frac{\sum_{i=1}^{S_z^*}(z_i^2 - \bar{z}^2)}{S_z^* - 1}}$

(4) Calculate the standard error of the sample mean: $\hat{se}_z = \frac{\hat{s}_z}{\sqrt{S_z^*}}$

(5) Stop if the ratio of the standard error to the sample mean is below the threshold: $\frac{\hat{se}_z}{\bar{z}} < n_{thr}$

(6) Stop if the number of samples is equal to the maximum sample size: $S_z^* = S_{\max}$

(7) Otherwise, sample one more measurement of z and update the sample size: $S_z^* = S_z^* + 1$. Return to Step (2)

Figure 11: Standard error dynamic resampling method by Di Pietro et al. (2004)

The proposed resampling method in Figure 11 is slightly modified from the original model by Di Pietro et al. (2004) by Step (5). This modified version applies the threshold n_{thr} to the ratio of the standard error to the sample mean, unlike how the original model applies it to the standard error only. Since the magnitude of the standard error depends on the scale of units, we applied the threshold condition to the standardized measure of variance instead of the absolute value of the standard error.

There are only two parameters to set for the proposed method: the threshold $0 < n_{thr}$ and the maximum sample size $0 < S_{\max}$. The threshold should be set to balance the trade-off between the number of samples and the accuracy of the mean estimate. Furthermore, small $S_{\max}$ will generate an unreliable estimation of the true mean while large $S_{\max}$ may overload computation for high-noise replenishment strategies. In our research, we determined their values by conducting numerical experiments and it is explained in more detail in Section 5.2.

4.4 Phase 2: Memetic Algorithm

In order to solve the proposed stochastic replenishment problem, we developed an MA that augments an evolutionary GA with a gradient-free LSA. In Section 4.4.1, we provide a comprehensive description of the general MA process. The details of the proposed MA are explained in Sections 4.4.2 - 4.4.10. Finally in Section 4.5, the summary of our proposed MA is provided.

4.4.1 Algorithm Framework

MA is a population-based evolutionary algorithm that hybridizes GA with a problem specific LSA in order to incorporate individual learning procedures. In this section, we first explain the general framework of a simple GA and elaborate on how hybridization can be achieved.

GA is one of the most successful evolutionary computation algorithms that works with a population of candidate solutions by iterating through fitness assessment, selection and reproduction (crossover and mutation) for multiple generations. Because of its capability to explore the search space in large-scale problems and handle complex nonlinear constraints, GA has been successfully implemented for various inventory management problems (Altiparmak et al., 2006).

A typical GA starts with a set of randomly generated candidate solutions, referred to as a *population*. Each individual solution in the population is called a *chromosome*, or an *individual*, and it is a string of a set of solution parameters, where each parameter is referred to as a *gene*. Therefore, every solution of the optimization problem needs to be *encoded* as a chromosome in order to apply GA. Each chromosome has its own *fitness value* evaluated from the *fitness function*, which represents the quality of the associated solution. Typically, the fitness function is identical to the objective function of the considered problem, but it is not always the case (Beasley et al., 1993). Also, depending on how the function is defined, a larger or a smaller fitness value may be regarded better.

In each iteration of the GA, which is referred to as a *generation*, the fitness value of every chromosome in the population is evaluated. The individual with a better fitness value is selected for *reproduction* (i.e., modification of its gene to form a new population) and *survival by elitism* (i.e., best chromosomes remain in the new population) with a higher probability. In a simple GA, three genetic operators are typically used for the population reproduction: *selection*, *crossover* and *mutation*. To generate a new chromosome, a pair of "parent" chromosomes are randomly selected from the population with a higher chance of selecting an individual with a better fitness value. Then the crossover operator randomly recombines the genes of two parents to generate one or more "child" chromosomes (i.e., chromosomes in the next generation). After crossover, one or more randomly selected child chromosomes are mutated by randomly changing its one or more gene values. The algorithm repeats this process until a termination criterion is satisfied, which is commonly represented by the maximum number of generations.

MA extends the GA operators by one or more LSA to utilize problem-specific knowledge in the search. Commonly used LSA for continuous GA are Hill-Climbing, Simplex method and Newton/Quasi-Newton method (Tang et al., 2009). The advantage of MA comes from hybridizing an explorative global optimization algorithm (GA) with an exploitative LSA, which creates a good balance between the algorithm's generality and problem specificity (Tang et al., 2009). We conclude this section with a flowchart of the process in a typical Memetic Algorithm shown in Figure 12.

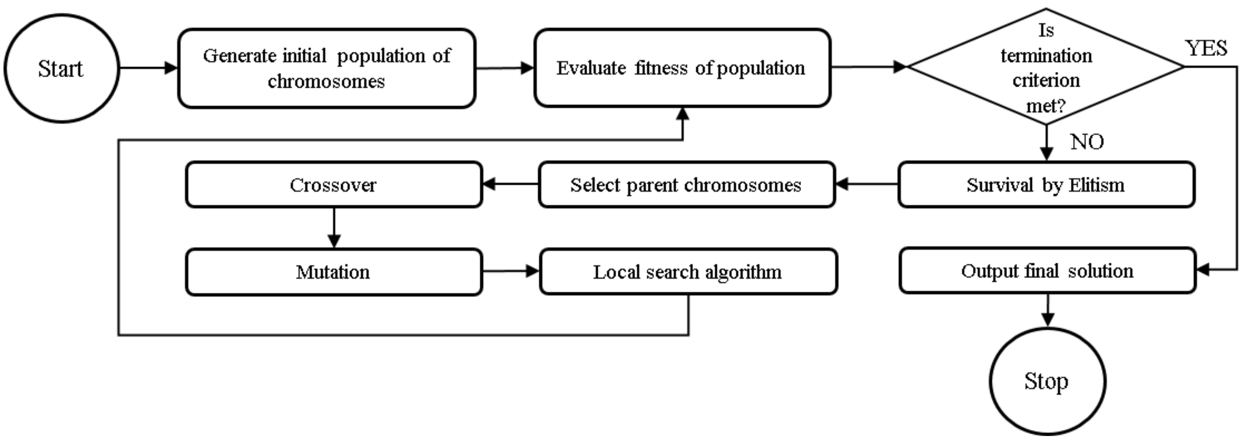

Start
Generate initial population of chromosomes
Evaluate fitness of population
Is termination criterion met?
YES
NO
Survival by Elitism
Select parent chromosomes
Crossover
Mutation
Local search algorithm
Output final solution
Stop

Figure 12: Flowchart of the proposed MA

4.4.2 Algorithm Notations

The parameters used in the GA part of the proposed MA are population size P, the maximum number of generations M, the probability of selecting a fitter individual in the binary tournament (tournament probability) λ_t, the probability of crossover (crossover rate) λ_c, the probability of mutation (mutation rate) λ_m and the number of elite solutions n_{elite}. In the LSA of the MA, the number of chromosomes improved by the LSA in each generation is denoted by n_{LS}.

4.4.3 Chromosome Representation

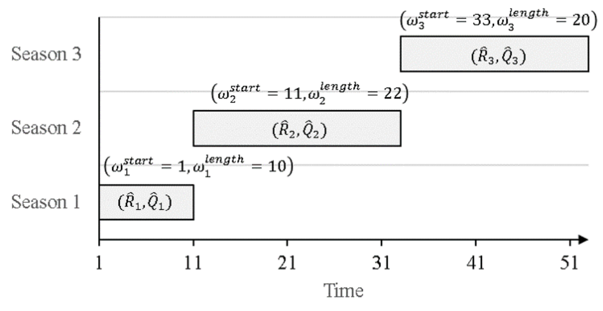
As discussed in Section 3.3, a replenishment strategy $\mathcal{P}$ consists of a review time T and a pair of reorder point and reorder quantity throughout the planning horizon $(R_t, Q_t), t = 1, 2, \ldots, N$. However, due to the cycle-based definition of our seasonal replenishment strategy, the pair (R_t, Q_t) for two consecutive time periods cannot be different for more than m number of times during each seasonal cycle of Y periods. Therefore, we can alternatively define $\mathcal{P}$ in terms of m seasons.

A *season* is a set of consecutive time periods that has the same (R_t, Q_t) values within a seasonal demand cycle. The start time of each season ω_i^{start} for $i = 1, \ldots, m$ lies between the interval $[1, Y]$ and does not overlap with other seasons. Also, the season's length ω_i^{length} is at least one time period long and the sum of all seasons' lengths must equal to the cycle length to cover the entire demand cycle: $\sum_{i=1}^{m} \omega_i^{\text{length}} = Y$. Therefore, below list of conditions must be met for each season set $\{(\omega_i^{\text{start}}, \omega_i^{\text{length}}) | i = 1, 2, \ldots, m\}$ to be valid for a replenishment strategy.

1. Seasons are sorted in an increasing order of the start time: $\omega_i^{\text{start}} < \omega_j^{\text{start}} \quad \forall j > i$ where $i \in \{1, 2, \ldots, m-1\}$ and $j \in \{2, 3, \ldots, m\}$
2. Season exceeding the cycle length: if the end time of season i exceeds the demand cycle, the season still continues in the next demand cycle. Hence, the season is split into two smaller sub-seasons $(i, 1)$ and $(i, 2)$, where the second sub-season starts at the beginning of the demand cycle
   - $\omega_{i,1}^{\text{start}} = \omega_i^{\text{start}}$ and $\omega_{i,1}^{\text{length}} = Y - \omega_i^{\text{start}} + 1$
   - $\omega_{i,2}^{\text{start}} = 1$ and $\omega_{i,2}^{\text{length}} = \omega_i^{\text{length}} - \omega_{i,1}^{\text{length}}$
   - Keep the original season set: $\omega_i^{\text{start}} = \omega_{i,1}^{\text{start}}$ and $\omega_i^{\text{length}} = \omega_{i,1}^{\text{length}} + \omega_{i,2}^{\text{length}}$
3. No overlap between seasons: $\omega_i^{\text{start}} + \omega_i^{\text{length}} - 1 < \omega_j^{\text{start}} \quad \forall j > i$ where $i \in \{1, 2, \ldots, m-1\}$ and $j \in \{2, 3, \ldots, m\}$
4. Covers the whole seasonal cycle: $\sum_{i=1}^{m} \omega_i^{\text{length}} = Y$
5. Possible start time: $\omega_i^{\text{start}} \in \{1, 2, \ldots, Y\}$ for $i = 1, 2, \ldots, m$
6. Possible length: $\omega_i^{\text{length}} \in \{1, 2, \ldots, Y\}$ for $i = 1, 2, \ldots, m$

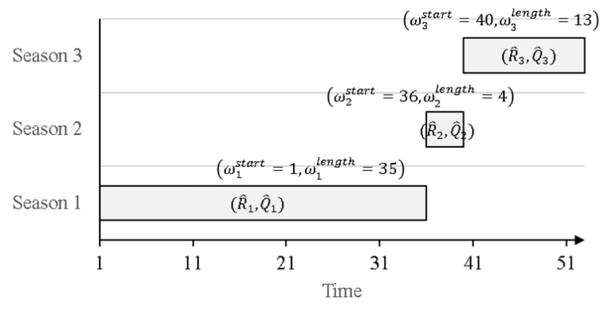
Figure 13: Conditions for seasons in a replenishment strategy

Finally, the pair of reorder point and reorder quantity for season i is denoted with a hat, as $(\hat{R}_i, \hat{Q}_i)$. Figure 14 illustrates four different examples of possible replenishment strategies with $m = 3$ and $Y = 52$.



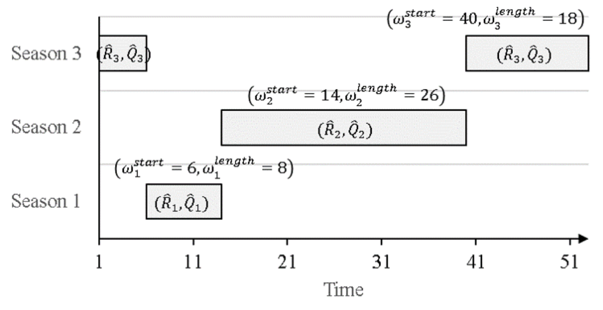
Season 3
Season 2
Season 1
$(\omega_3^{start} = 33, \omega_3^{length} = 20)$
$(\hat{R}_3, \hat{Q}_3)$
$(\omega_2^{start} = 11, \omega_2^{length} = 22)$
$(\hat{R}_2, \hat{Q}_2)$
$(\omega_1^{start} = 1, \omega_1^{length} = 10)$
$(\hat{R}_1, \hat{Q}_1)$
1 11 21 31 41 51
Time

(a) Seasons of similar lengths with no cycle surplus



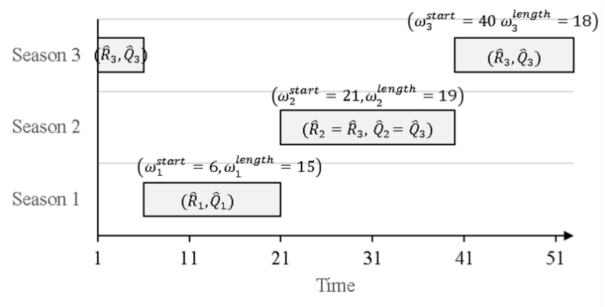
Season 3
Season 2
Season 1
$(\omega_3^{start} = 40, \omega_3^{length} = 13)$
$(\hat{R}_3, \hat{Q}_3)$
$(\omega_2^{start} = 36, \omega_2^{length} = 4)$
$(\hat{R}_2, \hat{Q}_2)$
$(\omega_1^{start} = 1, \omega_1^{length} = 35)$
$(\hat{R}_1, \hat{Q}_1)$
1 11 21 31 41 51
Time

(b) Seasons of different lengths with no cycle surplus



Season 3
Season 2
Season 1
$(\omega_3^{start} = 40, \omega_3^{length} = 18)$
$(\hat{R}_3, \hat{Q}_3)$
$(\hat{R}_3, \hat{Q}_3)$
$(\omega_2^{start} = 14, \omega_2^{length} = 26)$
$(\hat{R}_2, \hat{Q}_2)$
$(\omega_1^{start} = 6, \omega_1^{length} = 8)$
$(\hat{R}_1, \hat{Q}_1)$
1 11 21 31 41 51
Time

(c) Seasons of different lengths with cycle surplus



Season 3
Season 2
Season 1
$(\omega_3^{start} = 40\ \omega_3^{length} = 18)$
$(\hat{R}_3, \hat{Q}_3)$
$(\hat{R}_3, \hat{Q}_3)$
$(\omega_2^{start} = 21, \omega_2^{length} = 19)$
$(\hat{R}_2 = \hat{R}_3,\ \hat{Q}_2 = \hat{Q}_3)$
$(\omega_1^{start} = 6, \omega_1^{length} = 15)$
$(\hat{R}_1, \hat{Q}_1)$
1 11 21 31 41 51
Time

(d) 2 seasons of different lengths with cycle surplus

Figure 14: Examples of possible replenishment strategies with 3 seasons $(m = 3)$

Figure 14a shows a typical example of a replenishment strategy with 3 seasons, where each season has a similar length and does not overlap with one another. It is still possible that some particular seasons are exceptionally longer than the other, as demonstrated in Figure 14b where season 1 is noticeably longer than the other two seasons. Figure 14c shows an example of a strategy with a season that ends after the cycle length. Season 3, which starts before the end of cycle time ($\omega_3^{\text{start}} = 40 < 52$) ends after the cycle length ($\omega_3^{\text{start}} + \omega_3^{\text{length}} - 1 > 52$). Since the seasons are defined with respect to the seasonal cycle Y and not to the actual planning horizon N, the excess season over Y are passed to the beginning of the next cycle. Notice how season 3 continues to time 1 until 5 before ω_1^{start}. Figure 14d shows an example of a strategy with season 2 having the same reorder parameters as season 3. This is essentially a 2-season strategy.

Every chromosome used in the proposed MA represents a replenishment strategy defined in terms of seasons. As described above, every season $i = 1, 2, \ldots, m$ contains information about its start time and length ($\omega_i^{\text{start}}, \omega_i^{\text{length}}$) as well as its associated reordering policy ($\hat{R}_i, \hat{Q}_i$). Therefore, we can encode the seasonal replenishment in a $[1 \times (4m + 1)]$ matrix. Each column of the matrix represents a gene, which could be review time (T), a season's start time (ω_i^{start}), length (ω_i^{length}), reorder point ($\hat{R}_i$) or reorder quantity ($\hat{Q}_i$). Figure 15 illustrates a chromosome with $m = 3$.

T	$\boldsymbol{\omega_1^{\text{start}}}$	$\boldsymbol{\omega_1^{\text{length}}}$	$\boldsymbol{\hat{R}_1}$	$\boldsymbol{\hat{Q}_1}$	$\boldsymbol{\omega_2^{\text{start}}}$	$\boldsymbol{\omega_2^{\text{length}}}$	$\boldsymbol{\hat{R}_2}$	$\boldsymbol{\hat{Q}_2}$	$\boldsymbol{\omega_3^{\text{start}}}$	$\boldsymbol{\omega_3^{\text{length}}}$	$\boldsymbol{\hat{R}_3}$	$\boldsymbol{\hat{Q}_3}$

Figure 15: Example chromosome with $m = 3$

4.4.4 Fitness Function

Due to the target service level condition, applying an ordinary GA that neglects the service level constraint will naturally prefer the cheapest strategy that may be seriously infeasible. To avoid such situation, many researchers have developed various constraint-handling techniques, and the penalty function method has been one of the mostly studied and applied technique for solving constrained optimization problems. Penalty function methods prevent infeasible solutions to attain the same or higher priorities to feasible solutions by penalizing their fitness values in proportion to the magnitude of constraint violation. Since our objective is to minimize the expected total cost, we have a "smaller the better" type of fitness function and therefore a positive penalty term is added to the total fitness function if an infeasible solution is evaluated.

$$\text{Fitness function} = \text{expected total cost} + \text{penalty term}$$

Many sophisticated penalty function methods have been applied to various constrained combinatorial problems (Yeniay, 2005), and we implement an adaptive penalty method proposed by Coit et al. (1996) that utilizes the search length and severity of constraint violation. The term Near-Feasibility Threshold (NFT) stands for the threshold of infeasible search space where the GA is willing explore. The penalty function based on the NFT criteria encourages the GA to explore within the feasible region and its NFT-neighborhood solutions, while discouraging search beyond the threshold.

For a chromosome that encodes a replenishment strategy $\mathcal{P}$, its fitness value in generation e depends on the severity of constraint violation and the expected total cost of best known strategy until generation e. The fitness function $g(\mathcal{P}, e)$ is then expressed as Equation (24)

$$g(\mathcal{P}, e) = f(\mathcal{P}) + (f^e_{\text{feas}} - f^e_{\text{all}})\Big(\frac{\delta(\mathcal{P})}{\widehat{NFT}}\Big)^{\kappa} \tag{24}$$

where $f(\mathcal{P})$ is the expected total cost for the chromosome $\mathcal{P}$, f^e_{feas} denotes the expected total cost of the best feasible solution found until generation e, f^e_{all} denotes the expected total cost of the best solution until generation e, $\delta(\mathcal{P})$ is the distance of the expected fill rate of chromosome $\mathcal{P}$ from the target fill rate $FR_{\min}$, $\widehat{NFT}$ is the near-feasibility threshold value, and κ is the user-specified severity parameter for constraint violation. Let f^e_{infeas} be the expected total cost of the best infeasible solution found until generation e. Then we can express $f^e_{\text{all}} = \min\{f^e_{\text{feas}}, f^e_{\text{infeas}}\}$.

The NFT function is adaptive to the complexity of the service level constraint. If the constraint is difficult to meet, the gap between the best feasible solution and the best solution found so far ($f^e_{\text{feas}} - f^e_{\text{all}}$) will be wide because it would require more frequent replenishments with larger ordering quantities, which would incur a greater cost. In contrary, a very narrow gap suggests that the constraint is rather easy to satisfy.

The degree of constraint violation $\delta(\mathcal{P})$ is expressed by the positive difference between the target service level and the chromosome's expected fill rate.

$$\delta(\mathcal{P}) = (FR_{\min} - FR(\mathcal{P}))^+ \tag{25}$$

The dynamic version of the NFT fitness function defines the constant $\widehat{NFT}$ in a unique form

$$\widehat{NFT} = \frac{NFT_0}{1 + ce} \tag{26}$$

where NFT_0 is some upper bound for NFT and $0 \leq c \leq 1$ is a positive multiplier to multiply the penalty term as the generation evolves. Thus, the threshold value is a function of the generation number e, which would decrease monotonically as the generation continues. This forces the total penalty term for infeasible solutions to grow and make the GA to search toward the feasible region as the generation grows.

Two potential problems may arise when using the NFT model. Firstly, when f^e_{feas} is lower than f^e_{infeas}, the difference $f^e_{\text{feas}} - f^e_{\text{all}}$ becomes zero because $f^e_{\text{all}} = \min\{f^e_{\text{feas}}, f^e_{\text{infeas}}\} = f^e_{\text{feas}}$. As a consequence, any infeasible solutions with higher expected total costs than f^e_{feas} in generation e and in later generations will have zero penalties until a better (i.e., lower cost) infeasible solution is produced. Having zero penalty term for these infeasible solutions may influence the GA's selection and reproduction procedures, and possibly deteriorate the algorithm's overall performance. However, the zero penalty situation can be quite beneficial to the algorithm's explorative capabilities in later generations as it encourages to search beyond the established search region. Furthermore, the risk of losing the best feasible solution due to unpenalized infeasible solutions is extremely low because even if the difference $f^e_{\text{feas}} - f^e_{\text{all}}$ is small, the final penalty term will dynamically increase at an exponential rate as the number of generations increases. The second potential problem may occur when f^e_{all} is much higher than f^e_{feas}, imposing extremely severe penalty to infeasible solutions. This can be critical especially in early generations when the algorithm needs to explore the search space without restriction. However, a good initial population generated by our population initialization method (Section 4.4.5) prevents from having infeasible solutions that are unacceptably far from the feasible region.

4.4.5 Initial Population Generation Method

With the population-based structure, we need to build an initial population of replenishment strategies and encode them as chromosomes for further steps. Since the choice of initial population has a substantial impact in the algorithm's exploration strength and convergence rate in early generations, we designed and developed a heuristic method to construct a set of reasonable replenishment strategies instead of simple randomization.

Season start time and length

An intuitive approach to define seasons is to cluster together the periods of similar demand level. That is, average demand of a peak season should be higher than average demand of an off-season. Also, the periods in which the transitions between seasons take place (i.e., from high to low demand season or from low to high demand season) should be captured as separate transition seasons. A simple random search can be used to identify such seasons by minimizing the total mean squared error (MSE) between the seasonal mean and the period demand during the seasons.

The MSE of the seasonal mean for season i measures the distance between the season's mean demand $\hat{\mu_i} = (1/\omega_i^{\text{length}}) \sum_{k=\omega_i^{\text{start}}}^{\omega_i^{\text{start}}+\omega_i^{\text{length}}-1} \mu_k$ and the average period demand μ_t for time t that belongs to the season. Equation (27) defines the MSE for season i.

$$\text{MSE}_i = \frac{1}{\omega_i^{\text{length}}} \sum_{k=\omega_i^{\text{start}}}^{\omega_i^{\text{start}}+\omega_i^{\text{length}}-1} (\mu_k - \hat{\mu_i})^2 \tag{27}$$

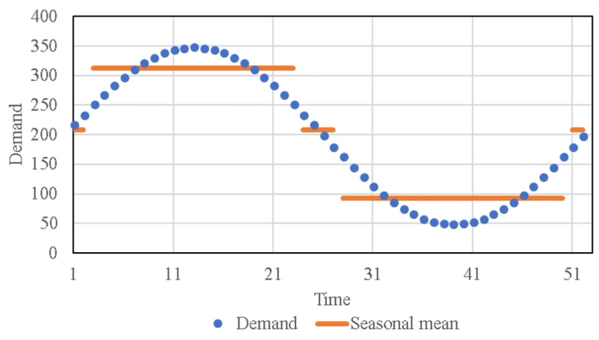
In case the end time of season i exceeds the cycle length Y, we split the summation $\sum_{k=\omega_i^{\text{start}}}^{\omega_i^{\text{start}}+\omega_i^{\text{length}}-1}(\cdot)$ into two parts: $\sum_{k=\omega_i^{\text{start}}}^{Y}(\cdot)$ and $\sum_{k=1}^{\omega_i^{\text{start}}+\omega_i^{\text{length}}-Y}(\cdot)$ to obtain the demand that is passed onto the next cycle.

A simple random search algorithm can randomly generate n_{rs} number of season sets, where each season set $\{(\omega_i^{\text{start}}, \omega_i^{\text{length}})|i = 1, \ldots, m\}$ contains a unique definition of m seasons according to the condition in Figure 13. Then, we select a set among these n_{rs} season sets that minimize the total MSE (denoted by MSE*) throughout the seasonal cycle.

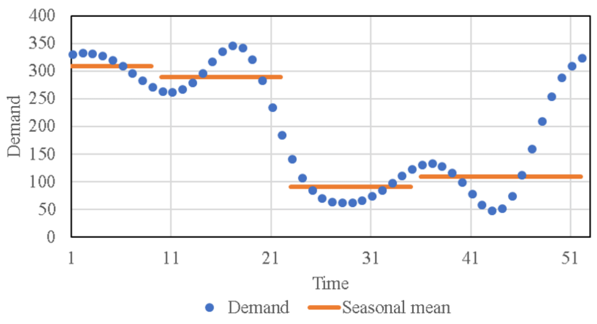
$$\text{MSE}^* = \sum_{i=1}^{m} \text{MSE}_i \tag{28}$$

One important drawback of using a random search is that one needs to set a very large value for n_{rs} to find the correct seasons that minimize the total MSE. Indeed, enumerating over all possible season definitions would require at least $\frac{Y!}{(Y-m)!}$ number of iterations to find the exact minimum. However, setting a relatively low value for n_{rs} could be more desirable for the purpose of population diversification since it would enable some randomization of seasons while strictly preferring the season set with the least MSE*.

The example seasons identified by minimizing the total MSE are demonstrated in Figure 16. Four seasons were defined for both symmetric and asymmetric seasonal demand by setting n_{rs} equal to $1,000,000$.



(a) Seasons for symmetric demand



(b) Seasons for asymmetric demand

Figure 16: Seasons for symmetric and asymmetric seasonal demand

Ordering policy

The review time T for the replenishment strategy is selected randomly with a uniform probability from the discrete interval $[1, T_{\max}]$. By pure randomization, the initial population can consist of a more diverse set of individuals.

For each season, we determine the corresponding reorder quantity $\hat{Q}_i$ by a simple economic order quantity (EOQ) formula. We note that the EOQ model is not necessarily optimal for our problem as it assumes for deterministic, stationary demand and zero lead time. But this may still provide a fairly good initial guess if the demand and lead time variations are reasonably low. The reorder quantity for each season $i = 1, 2, \ldots, m$ then equals to

$$\hat{Q}_i = \sqrt{\frac{2K\hat{\mu}_i}{h}} \tag{29}$$

where $\hat{\mu}_i$ is the seasonal mean.

The initial reorder point for each season $\hat{R}_i$ are generated following the results on Axsäter (2015). He proposed a novel approach to determine the reorder point under periodic review, continuous Normally distributed demand and constant lead time. To apply his approach in our problem, we replace continuous distribution by a discrete Normal demand distribution for each season with mean $\hat{\mu}_i$ and standard deviation $\hat{s}_i = \sigma/\sqrt{\omega_i^{\text{length}}}$. Also, constant lead time is assumed using the mean lead time, such that $L = 1/q$.

Let the demand during lead time L have a mean $\mu_i' = L\hat{\mu}_i$ and a standard deviation $s_i' = \sqrt{L}\hat{s}_i$. Also the demand during the time $L + T$, which would be the next possible delivery time, have a mean $\mu_i'' = (L+T)\hat{\mu}_i$ and a standard deviation $s_i'' = \sqrt{L+T}\hat{s}_i$. Then we obtain following inventory level distribution just after a possible delivery has arrived

$$P(IL_i' \leq x) = \frac{s_i'}{\hat{Q}_i}\Big[G\Big(\frac{\hat{R}_i - x - \mu_i'}{s_i'}\Big) - G\Big(\frac{\hat{R}_i + \hat{Q}_i - x - \mu_i'}{s_i'}\Big)\Big] \tag{30}$$

$G(x)$ is a discrete Normal loss function

$$G(x) = \tilde{\varphi}(x) - x(1 - \tilde{\Phi}(x)) \tag{31}$$

where $\tilde{\varphi}(x)$ and $\tilde{\Phi}(x)$ are the probability mass function and the cumulative distribution function of the discrete analogue of standard Normal distribution at x, respectively. The precise definitions of these two functions are available in Appendix A.2

Then the expected fill rate during season i is expressed as

$$\widehat{FR}_i = 1 - \frac{E(IL''_i)^- - E(IL'_i)^-}{\hat{\mu}_i T} \tag{32}$$

where $E(IL'_i)^-$ and $E(IL''_i)^-$ are the expected negative inventory level just after a replenishment arrival and just before the next possible delivery respectively. The exact expressions for $E(IL'_i)^-$ and $E(IL''_i)^-$ can be found in Appendix A.1. Therefore, the optimal reorder point $\hat{R}_i$ in season i for a given $\hat{Q}_i$ is determined by calculating the smallest $\hat{R}_i$ value that satisfies $\widehat{FR}_i \geq FR_{\min}$ where $\widehat{FR}_i$ is defined according to Equation (32).

Summary

Figure 17 summarizes the initial population generation algorithm.

Initial Population Generation Algorithm

(1) Randomly generate a review time: $T \in [1, T_{\max}]$

(2) Generate m seasons by applying a random search with n_{rs} number of solutions to minimize the mean squared error (MSE) of seasonal mean $\hat{\mu}_i$ subject to the mean period demand μ_t covered by the season

(3) For every season $i = 1, 2, \ldots, m$:

- (3.1) Set the season's reorder quantity by EOQ model: $\hat{Q}_i = \sqrt{\frac{2K\hat{\mu}_i}{h}}$
- (3.2) Round $\hat{Q}_i$ to the nearest integer
- (3.3) Determine the season's reorder point $\hat{R}_i$ by numerically calculating the smallest integer $\hat{R}_i$ value that satisfies $FR_i \geq FR_{\min}$, where FR_i is defined as in Equation (32)
- (3.4) Check for the bound: $\hat{R}_i = \max\{\hat{R}_i, 0\}$

Figure 17: Initial population generation algorithm

When implementing the above initialization method, each season can have at most $T_{\max}$ number of unique reorder policies $(\hat{R}_i, \hat{Q}_i)$, assuming that the seasons are correctly identified to minimize the total MSE. This implies that the proposed method can generate at most $(T_{\max})^m$ unique chromosomes, which may be too small for the MA with a larger population size. In order to generate more unique chromosomes, the target fill rate $0 \leq FR_{\min} \leq 1$ in Step (3.3) can be replaced by a Normally distributed random variable $\Delta_{FR} \sim \mathcal{N}(FR_{\min}, \sigma(FR_{\min}))$ with a mean $FR_{\min}$ and a standard deviation $\sigma(FR_{\min})$. In case the new target fill rate Δ_{FR} is outside of the bounds $[0, 1]$, a new value of Δ_{FR} is drawn from the Normal distribution. Setting a high value for the standard deviation $\sigma(FR_{\min})$ would allow for a more diverse set of chromosomes in the initial population while more infeasible solutions will be generated as well.

4.4.6 Selection and Elitism

In our proposed MA, the parent chromosomes are selected by a simple binary tournament method with a winning probability λ_t. A tournament selection rule is a stochastic parent selection approach that randomly selects two or more chromosomes from the population with equal probability and conducts "tournaments" between the selected chromosomes. A chromosome with a better fitness value has a higher chance of winning each tournament, and the winner of all tournaments is selected for reproduction.

In case of our binary tournament, only two chromosomes are randomly selected, with replacement, from the population with equal chance, and a chromosome with a lower (hence better) fitness value is selected for reproduction with probability λ_t. To make sure that the chromosome with a lower fitness value has a higher chance to reproduce, λ_t is set above 0.5. We applied the binary tournament instead of larger tournaments in order to preserve diversity of the population and ultimately enhance the algorithm's explorative strength since our objective function is strongly multi-modal. See Figure 18 for the diagram of binary tournament.

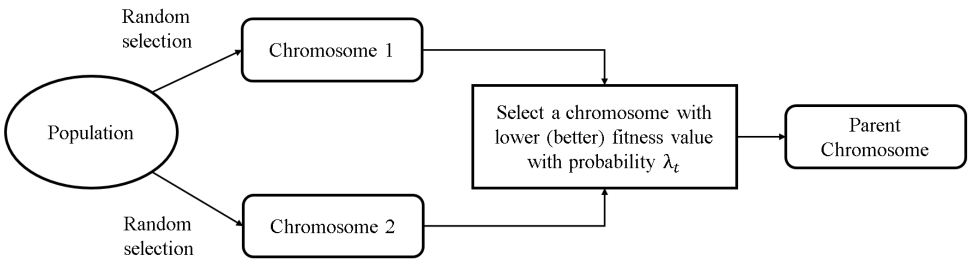


Figure 18: Illustration of binary tournament

Moreover, n_{elite} number of elite chromosomes with the best fitness values in current generation are copied to the next generation in order to preserve the best found solution so far. This is known as the elitism strategy that prevents from losing the best chromosomes throughout the MA iterations while performing crossover or mutation operation. The elite chromosomes can still be selected as parent chromosomes for breeding a new child, but they are copied to the next generation beforehand.

4.4.7 Crossover

The proposed MA breeds new child chromosomes by performing uniform crossover where two parent chromosomes exchange their genes randomly at the rate of λ_c. If a uniform random number that is drawn from the continuous interval $[0, 1]$ takes a value smaller than or equal to λ_c, two parent chromosomes selected from the binary tournament swap their genes according to the following crossover operator. We emphasize that the probability λ_c determines whether a crossover would happen or not at all. If it does not happen then the two parent chromosomes survive to the next generation.

In the uniform crossover, every gene of a child chromosome is randomly chosen from either of two parents with equal probability. Figure 19 illustrates our uniform crossover between two parents for their review time T and seasonal reorder policies $(\hat{R}_i, \hat{Q}_i)$. Because the crossover operator generates two new child chromosomes, if the population size P is odd valued then only one of the two child chromosomes is entered into the next generation.

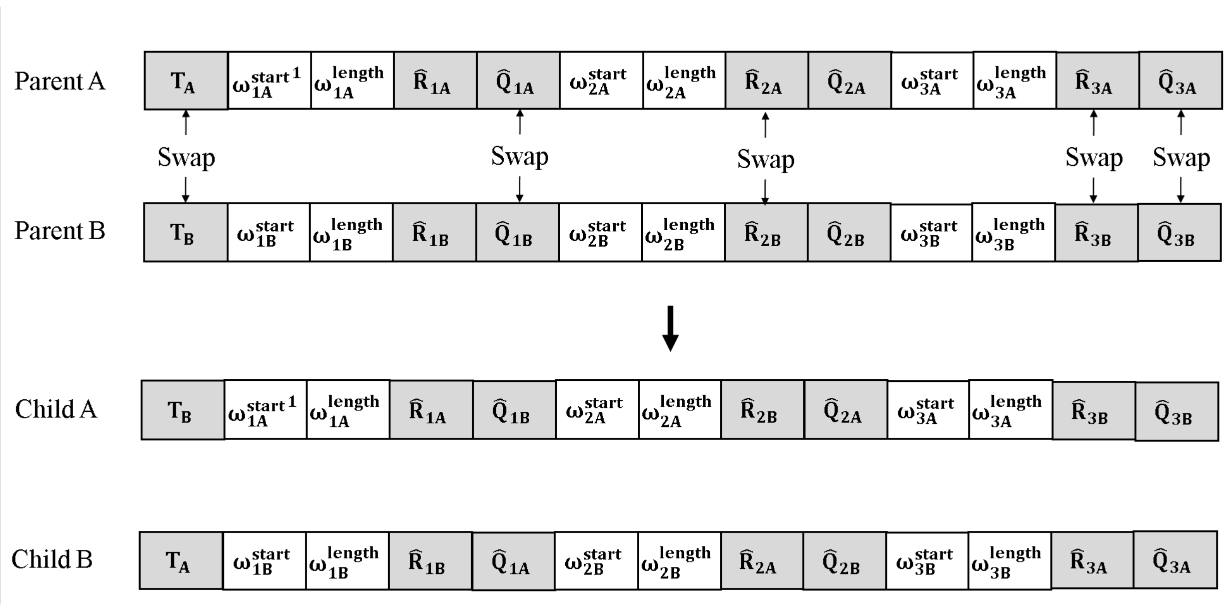

Parent A
T_A $\omega_{1A}^{start\,1}$ ω_{1A}^{length} $\hat{R}_{1A}$ $\hat{Q}_{1A}$ ω_{2A}^{start} ω_{2A}^{length} $\hat{R}_{2A}$ $\hat{Q}_{2A}$ ω_{3A}^{start} ω_{3A}^{length} $\hat{R}_{3A}$ $\hat{Q}_{3A}$
Swap
Swap
Swap
Swap
Swap
Parent B
T_B ω_{1B}^{start} ω_{1B}^{length} $\hat{R}_{1B}$ $\hat{Q}_{1B}$ ω_{2B}^{start} ω_{2B}^{length} $\hat{R}_{2B}$ $\hat{Q}_{2B}$ ω_{3B}^{start} ω_{3B}^{length} $\hat{R}_{3B}$ $\hat{Q}_{3B}$
Child A
T_B $\omega_{1A}^{start\,1}$ ω_{1A}^{length} $\hat{R}_{1A}$ $\hat{Q}_{1B}$ ω_{2A}^{start} ω_{2A}^{length} $\hat{R}_{2B}$ $\hat{Q}_{2A}$ ω_{3A}^{start} ω_{3A}^{length} $\hat{R}_{3B}$ $\hat{Q}_{3B}$
Child B
T_A ω_{1B}^{start} ω_{1B}^{length} $\hat{R}_{1B}$ $\hat{Q}_{1A}$ ω_{2B}^{start} ω_{2B}^{length} $\hat{R}_{2A}$ $\hat{Q}_{2B}$ ω_{3B}^{start} ω_{3B}^{length} $\hat{R}_{3A}$ $\hat{Q}_{3A}$

Figure 19: Uniform crossover for reorder policies

On the other hand, the crossover for all seasons' start times and lengths $(\omega_i^{\text{start}}, \omega_i^{\text{length}})$ must either happen altogether or not at all due to the special structure of our chromosome. As described in Figure 13, the seasons become invalid for a replenishment strategy if the seasons overlap with each other or if the demand cycle is not completely covered. Accordingly, swapping seasons individually instead of as a whole may create invalid seasons for the child chromosomes. Also, repairing invalid seasons may destroy the original gene structure that should have been preserved from the parent's chromosome. Hence, we treat all seasons' timing genes $(\omega_i^{\text{start}}, \omega_i^{\text{length}})$ as a whole and swap them altogether during crossover. Figure 20 illustrates how they are swapped altogether for chromosomes with $m = 3$.

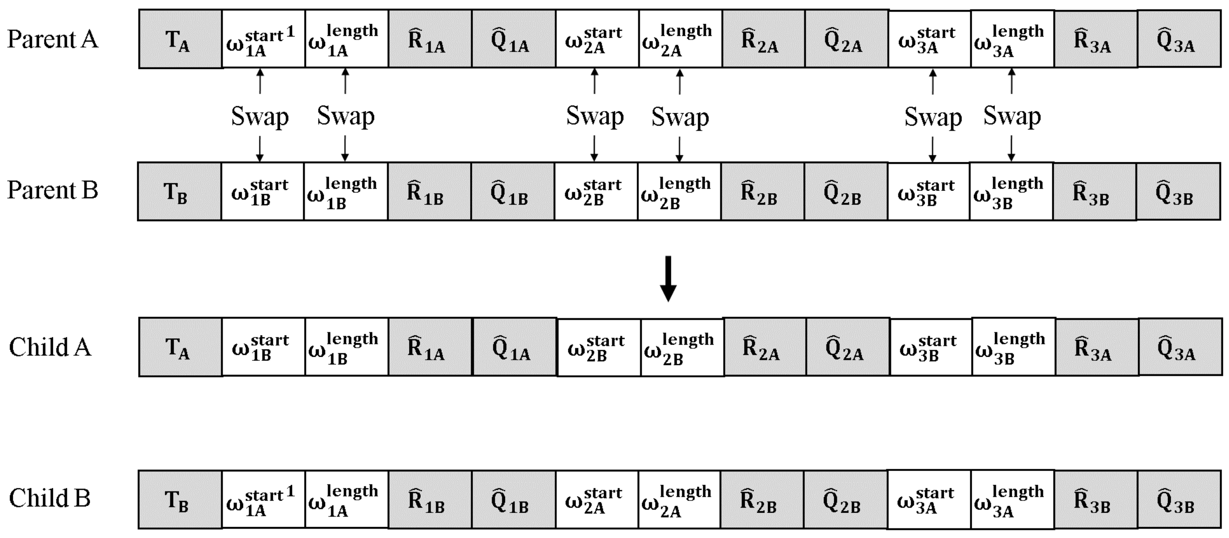

Parent A
T_A $\omega_{1A}^{start\,1}$ ω_{1A}^{length} $\hat{R}_{1A}$ $\hat{Q}_{1A}$ ω_{2A}^{start} ω_{2A}^{length} $\hat{R}_{2A}$ $\hat{Q}_{2A}$ ω_{3A}^{start} ω_{3A}^{length} $\hat{R}_{3A}$ $\hat{Q}_{3A}$
Swap
Swap
Swap
Swap
Swap
Swap
Parent B
T_B ω_{1B}^{start} ω_{1B}^{length} $\hat{R}_{1B}$ $\hat{Q}_{1B}$ ω_{2B}^{start} ω_{2B}^{length} $\hat{R}_{2B}$ $\hat{Q}_{2B}$ ω_{3B}^{start} ω_{3B}^{length} $\hat{R}_{3B}$ $\hat{Q}_{3B}$
Child A
T_A ω_{1B}^{start} ω_{1B}^{length} $\hat{R}_{1A}$ $\hat{Q}_{1A}$ ω_{2B}^{start} ω_{2B}^{length} $\hat{R}_{2A}$ $\hat{Q}_{2A}$ ω_{3B}^{start} ω_{3B}^{length} $\hat{R}_{3A}$ $\hat{Q}_{3A}$
Child B
T_B $\omega_{1A}^{start\,1}$ ω_{1A}^{length} $\hat{R}_{1B}$ $\hat{Q}_{1B}$ ω_{2A}^{start} ω_{2A}^{length} $\hat{R}_{2B}$ $\hat{Q}_{2B}$ ω_{3A}^{start} ω_{3A}^{length} $\hat{R}_{3B}$ $\hat{Q}_{3B}$

Figure 20: Uniform crossover for seasons' start times and lengths

4.4.8 Mutation

The main purpose of mutation operator is to create diversity of chromosomes in the population by randomly altering the genes of chromosomes with a very small probability $\lambda_m \in [0, 1]$. For every chromosome in the population, we iterate over each gene and draw a random number at uniform probability from the continuous interval $[0, 1]$. If the random number is below or equal to the mutation rate λ_m, the gene's value is replaced by a randomly drawn number from its bounds at uniform probability. When the gene is either T, $\hat{R}_i$ or $\hat{Q}_i$ of certain season i, a new value is randomly drawn from their respective bounds $[1, T_{\max}]$, $[0, R_{\max}]$, and $[0, Q_{\max}]$. Since our inventory model is uncapacitated, the values for $R_{\max}$ and $Q_{\max}$ are not available in advance. Yet, based on our computational experiments, we suggest setting $R_{\max}$ and $Q_{\max}$ equal to three times the average size of $\hat{R}_i$ and $\hat{Q}_i$ for all chromosomes in the current generation. The suggested bounds are wide enough to capture feasible regions without expanding the search space too much, which can slow down convergence of the MA. Also, the bounds are adaptive to the search region that is being explored in each generation. See Figure 21 for an example of review time and reorder quantity mutation.

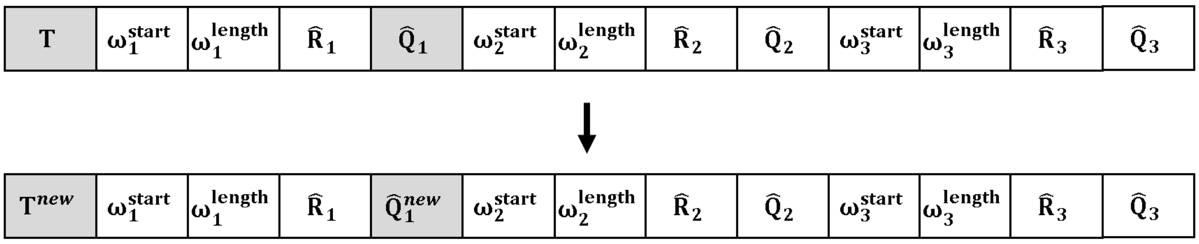


Figure 21: Mutation for review time and reorder policy

When the selected gene for mutation is either ω_i^{start} or ω_i^{length} of certain season i, the entire set of seasons $(\omega_i^{\text{start}}, \omega_i^{\text{length}})$ must be generated for all $i = 1, 2, \ldots, m$ from their integer bounds $[1, Y]$ while satisfying the conditions in Figure 13. See Figure 22 for an example mutation of seasons' start times and lengths.

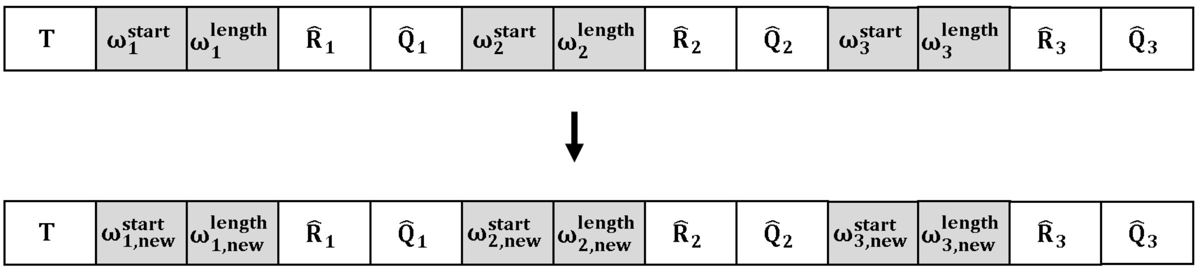


Figure 22: Mutation for seasons' start times and lengths

4.4.9 Local Search Algorithm

The simple GA is great for exploring the search space and exploiting the superior genetics of elite individuals. However its convergence to the optimal solution in practical settings is not stable and it lacks the overall sharpness to refine an individual in more detail (Tang et al., 2007).

In order to boost convergence and to better locate the feasible region during the search, we implemented a Local Search Algorithm (LSA) that looks for the best improvement in the neighborhood. Our LSA is a sequential heuristic method, which first adjusts seasons' orders, start times and lengths for the selected chromosome and later applies a Discrete Simultaneous Perturbation Stochastic Approximation (DSPSA) proposed by Wang and Spall (2011) to adjust seasonal reorder policy $(\hat{R}_i, \hat{Q}_i)$ for $i = 1, 2, \dots, m$. We refer to the first part as Neighborhood Search and the second part as DSPSA.

Neighborhood Search

The neighborhood of a chromosome is found by altering one of five main parameters of the chromosome, which are review time (T), season start time $(\omega_i^{\text{start}})$, length $(\omega_i^{\text{length}})$, reorder point $(\hat{R}_i)$ and reorder quantity $(\hat{Q}_i)$. Next five basic neighborhood operators are applied to a properly chosen chromosome to change their parameters until positive improvement in its fitness value is achieved.

- **ChangeReviewTime**: Replace the review time T with a neighbor $T + 1$ or $T - 1$, whichever that obtains the largest improvement in the fitness value. Continue replacing T until no more improvements are realized. Stop if the most-improving neighbor lies outside of the bounds $[1, T_{\max}]$.
- **CopySeasons**: Apply *Copy forward* and *Copy backward* to the season with the earliest start time $(\omega_i^{\text{start}})$. Iterate over the remaining seasons in the increasing order of ω_i^{start}. Even if a season is overwritten by another season's copy, still apply *Copy forward* and *Copy backward* to the original season.

  Copy forward: Copy the season's reorder policy $(\hat{R}_i, \hat{Q}_i)$ to its immediately preceding season $i - 1$ (or m in case $i = 1$) and check for improvement in fitness value. Continue until there is no more improvement.

  Copy backward: Copy the season's ordering policy $(\hat{R}_i, \hat{Q}_i)$ to the immediately succeeding season $i + 1$ (or 1 in case $i = m$) and check for improvement in fitness value. Continue unless there is no more improvement.

- **RearrangeSeasons**: For every time period in demand cycle $t = 1, \dots, Y$, calculate the average of expected demand μ_t during the protection interval $[t, t + 1/q + T]$ and denote this by $\mu_{t,\text{pro}}$. Then for every season $i = 1, 2, \dots, m$, calculate the season's total protection demand $\hat{\mu}_{i,\text{pro}} = \sum_{t=\omega_i^{\text{start}}}^{\omega_i^{\text{start}}+\omega_i^{\text{length}}-1} \mu_{t,\text{pro}}$. Now apply *Rearrange by reorder point* and *Rearrange by reorder quantity* until there is no more improvement.

  Rearrange by reorder point: Rank every season's reorder policy $(\hat{R}_i, \hat{Q}_i)$ in a decreasing order of $\hat{R}_i$. Also rank every season set $(\omega_i^{\text{start}}, \omega_i^{\text{length}})$ in a decreasing order of $\hat{\mu}_{i,\text{pro}}$. Assign every reorder policy to the season set of the same rank. Thus, higher $\hat{R}_i$ is assigned to manage higher $\hat{\mu}_{i,\text{pro}}$.

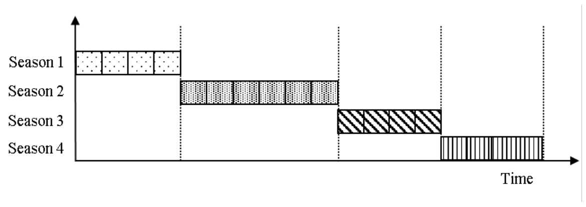
  Rearrange by reorder quantity: Rank every season's reorder policy $(\hat{R}_i, \hat{Q}_i)$ in a decreasing order of $\hat{Q}_i$. Also rank every season set $(\omega_i^{\text{start}}, \omega_i^{\text{length}})$ in a decreasing order of $\hat{\mu}_{i,\text{pro}}$. Assign every reorder policy to the season set of the same rank. Thus, higher $\hat{Q}_i$ is assigned to manage higher $\hat{\mu}_{i,\text{pro}}$.

- **DragSeasons**: While keeping all season lengths fixed, move all seasons' start times by one unit forward ($\omega_i^{\text{start}} - 1$) or backward ($\omega_i^{\text{start}} + 1$), whichever that yields the highest fitness improvement. Continue until no more improvements are realized.
- **StretchSeasons**: Apply *Stretch forward* and *Stretch backward* to the season with the earliest start time (ω_i^{start}). Iterate over the remaining seasons in the increasing order of ω_i^{start}.

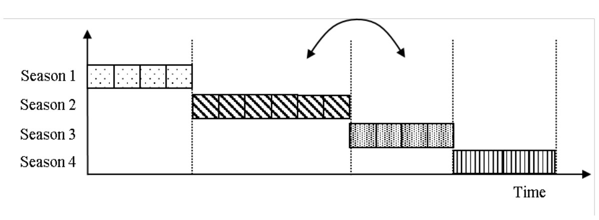
  Stretch forward: Decrease the season's start time and increase its length by one unit ($\omega_i^{\text{start}} - 1, \omega_i^{\text{length}} + 1$) as long as the preceding season can be shortened by one time unit (i.e., $\omega_{i-1}^{\text{length}} \geq 2$, or $\omega_m^{\text{length}} \geq 2$ if $i = 1$). Continue until no more improvements are realized or the preceding season cannot be shortened.

  Stretch backward: Fix the season's start time and increase its length by one unit ($\omega_i^{\text{start}}, \omega_i^{\text{length}} + 1$) as long as the succeeding season can be shortened by one time unit (i.e., $\omega_{i+1}^{\text{length}} \geq 2$, or $\omega_1^{\text{length}} \geq 2$ if $i = m$). Continue until no more improvements are realized or the succeeding season cannot be shortened.

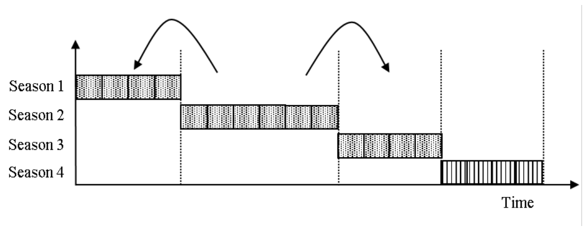
Figure 23 illustrates how each neighborhood operator alters the replenishment strategy with 4 different seasons. Each unique pattern that fills the the bar indicates a unique pair of seasonal reorder policy ($\hat{R}_i, \hat{Q}_i$).



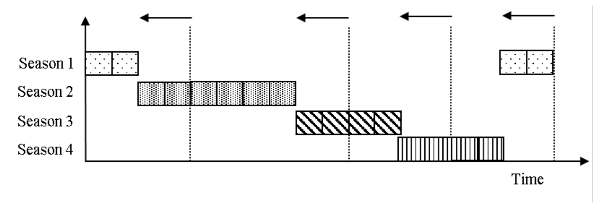
(a) Original strategy



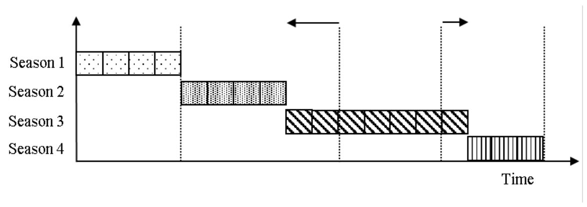
(b) Rearrange season 2 and 3 by swapping



(c) Copy season 2 to season 1 and 3



(d) Drag forward by 2 time periods



(e) Stretch season 3 by 2 periods forward and 1 period backward

Figure 23: Local search operators

The operators are applied in the order of **ChangeReviewTime**, **CopySeasons**, **RearrangeSeasons**, **DragSeasons** and **StretchSeasons**. The first three operators are applied with priority as they can overwrite the existing policies with entirely new policies of different underlying structure. More specifically, **ChangeReviewTime** changes the review time, which affects the reorder policies of all seasons, and therefore an entirely new replenishment strategy may be optimal for the new review time. **CopySeasons** and **RearrangeSeasons** can entirely overwrite some seasons and change the order of seasons, respectively. This creates more radical changes to seasons' correlation structure. The other two operators are applied later in order to fine-tune the timing of the seasons while maintaining the strategy's underlying correlation structure.

Discrete SPSA

After performing all the neighborhood operators, we apply the gradient-free DSPSA algorithm to refine $(\hat{R}_i, \hat{Q}_i)$. Wang and Spall (2011) proposed a middle point DSPSA algorithm, which is shown to have almost sure convergence to the global optimum. There are three main benefits of incorporating their DSPSA algorithm into our LSA. Firstly, the DSPSA algorithm has a unique ability to utilize the search space structure based on approximate gradient information. Secondly, the algorithm produces gradient approximation with only two fitness measurements in each generation, regardless of the number of decision variables involved in a chromosome. Finally, Wang and Spall (2011) have shown that the algorithm attains a tight upper bound for its finite sample convergence and optimization performance.

The underlying idea of the continuous version SPSA is to explore the search space based on an approximation of the gradient from the difference between two noisy fitness value measurements. It is a gradient-free steepest descent type of method for global optimization that does not require direct measurements of the true fitness value. Also, it only requires two measurements regardless of the variable's dimension to estimate the gradient (Spall, 2005). The DSPSA is a discrete analogue of the continuous SPSA without function derivatives.

Let U be the maximum number of DSPSA iterations and θ_k be the chromosome at k^{th} iteration of the algorithm. θ_0 is therefore the original chromosome selected for the DSPSA, which is expressed as $\theta_0 = [\hat{R}_1, \hat{Q}_1, \ldots, \hat{R}_m, \hat{Q}_m]$. The basic form of the recursive model for the $2m$-dimensional chromosome θ_k is

$$\theta_{k+1} = \theta_k - a_k \zeta_k(\theta_k) \tag{33}$$

where $\zeta_k(\theta_k)$ is the gradient estimate obtained from simultaneously perturbing the elements of the variable θ_k at k^{th} iteration and a_k is a non-negative gain coefficient.

In the work of Wang and Spall (2011), the gradient approximation $\zeta_k(\theta_k)$ is obtained by simultaneously perturbing all $2m$ components of θ_k by a $2m$-dimensional vector of independent Bernoulli random variables $\Delta_k = [\Delta_{k,1}, \Delta_{k,2}, \ldots, \Delta_{k,2m}]^T$, where each element $\Delta_{k,i}$ takes a value either -1 or $+1$ with an equal probability. The component-wise middle point between the ceiling value $\lceil \theta_k \rceil$ and the floor value $\lfloor \theta_k \rfloor$ is expressed as $\pi(\theta_k) = (2\lfloor \theta_k \rfloor + 1_{2m})/2$ with a $2m$-dimensional vector of ones 1_{2m}. Then, the gradient approximation is obtained in terms of two fitness measurements for $\pi(\theta_k) + \Delta_k/2$ and $\pi(\theta_k) - \Delta_k/2$

$$\zeta_k(\theta_k) = \left[g\left(\pi(\theta_k) + \frac{\Delta_k}{2}, e \right) - g\left(\pi(\theta_k) - \frac{\Delta_k}{2}, e \right) \right] \Delta_k^{-1} \tag{34}$$

where $\Delta_k^{-1} = [\Delta_{k,1}^{-1}, \Delta_{k,2}^{-1}, \ldots, \Delta_{k,2m}^{-1}]^T$ and $g(x, e)$ is the dynamic fitness function defined for solution x in generation e (see Section 4.4.4).

The gain sequence a_k in Equation (33) is expressed in terms of three non-negative coefficients α, A and a

$$a_k = \frac{a}{(k+1+A)^\alpha} \tag{35}$$

The choice of these coefficients are very important to the algorithm's finite sample performance and Spall (2005) suggests that setting $\alpha = 0.602$ and $A = 0.1U$. Based on our own computational experiments, however, we have chosen $A = 0.2$ instead as it attained the best fitness value with $\alpha = 0.602$. The value of a is chosen by solving the equation

$$a = \frac{u}{\zeta_0(\theta_0)}(A+1)^\alpha \tag{36}$$

where u is the desired magnitude of changes for the elements of θ_k in the early iterations. In our method, we replace u by a percentage of average size of θ_k elements to reflect the average magnitude of the reorder parameters $\hat{R}_i$ and $\hat{Q}_i$. Thus, u is replaced by a new coefficient $\tilde{u}\frac{1}{2m}\sum_{j=1}^{2m}\theta_{kj}$ where $0 \le \tilde{u} \le 1$ is a percentage term.

Figure 24 shows the DSPSA algorithm procedure for the $2m$-dimensional input vector θ_k.

DSPSA Algorithm Procedure by Wang and Spall (2011)

(0) Set the maximum number of iterations $U \ge 1$ and set $k = 0$

(1) Pick non-negative coefficients α, a and A according to the guidelines suggested by Spall (2005):

  (1.1) Set $\alpha = 0.602$ and $A = 0.2U$ based on the guideline by Spall (2005) and by conducting numerical experiments

  (1.2) Determine a according to Equation (36)

(2) Increase $k = k + 1$

(3) Generate a $2m$-dimensional vector of Bernoulli random variables: $\Delta_k = [\Delta_{k,1}, \Delta_{k,2}, \ldots, \Delta_{k,2m}]^T$ where each $\Delta_{k,i}$ takes value either -1 or $+1$ with 0.5 probability

(4) Calculate the middle point of θ_k: $\pi(\theta_k) = (2\lfloor\theta_k\rfloor + 1_{2m})/2$ where 1_{2m} is a $2m$-dimensional vector of ones and $g(x, e)$ is the dynamic fitness function defined for solution x in generation e (Section 4.4.4)

(5) Calculate the gradient approximation by simultaneous perturbation: $\zeta_k(\theta_k) = \left[g\left(\pi(\theta_k)+\frac{\Delta_k}{2}, e\right) - g\left(\pi(\theta_k)-\frac{\Delta_k}{2}, e\right)\right]\Delta_k^{-1}$ where $\Delta_k^{-1} = [\Delta_{k,1}^{-1}, \Delta_{k,2}^{-1}, \ldots, \Delta_{k,2m}^{-1}]^T$

(6) Update the reorder policies: $\theta_{k+1} = \theta_k - a_k\zeta_k(\theta_k)$

(7) Stop if $k = U$. Else return to Step (2)

Figure 24: DSPSA algorithm procedure by Wang and Spall (2011)

4.4.10 Parameter Optimization

As with every parametric algorithm, optimizing the control parameters for the MA is the key for attaining the best performance with the least computational effort. The most important parameters in the GA part of the proposed MA are population size (P), maximum generation (M), tournament probability (λ_t), crossover rate (λ_c) and mutation rate (λ_m). For tuning these parameters with a small number of test experiments, we apply Taguchi method suggested by Phadke (1989). In this section, we discuss the central idea of Taguchi method and its main steps for tuning the GA parameters. We discuss other MA parameters in Section 5.2 and focus only on the GA parameters in this section because they have the largest influence on navigating the search during the optimization procedure.

Taguchi method has been widely used for choosing the optimal parameter combinations of various optimization algorithms. For the class of inventory problems, the parameters of common metaheuristics including Genetic Algorithm, Simulated Annealing and Particle Swarm Optimization algorithm were often optimized by Taguchi method and several researchers reported improved computational efficiency and solution quality (Pasandideh et al., 2013; Mousavi et al., 2013; Sadeghi et al., 2014; Saracoglu et al., 2014). In our paper, we also apply Taguchi method to optimize the GA parameters instead of the full factorial design that experiments on all possible parameter combinations.

Taguchi method uses special orthogonal arrays to quickly identify the optimal parameter settings with only a small number of experiments. To optimize the parameters in the GA, we consider 3 possible levels for each of these parameters based on common settings in the inventory optimization literature (Daniel and Rajendran, 2005; Min et al., 2006; Pasandideh et al., 2013) whereby population size ranges from 20 to 200, crossover rate lies between 0.5 and 0.9, and mutation rate lies between 0.05 and 0.3. We suggest slightly different parameter levels as shown in Table 1. Then a standard L_{18} orthogonal array is used to conduct only 18 experiments instead of performing $3^5 = 243$ experiments that would have been required for full factorial design. The standard L_{18} array has eight columns and eighteen rows, each row representing an experiment and it is commonly chosen for tuning from 5 to 7 parameters with 3 levels (Phadke, 1989). Except for its first column, which is for parameter with only 2 levels, other 7 columns have entries of levels (1, 2 or 3) for 3 level parameters. Due to its special orthogonal structure, each pair of columns in the array has all combinations of levels and they appear for an equal number of times. See Appendix B.1 for the L_{18} orthogonal array.

Table 1: GA parameters to optimize and their three levels

Parameters	Maximum generation (M)	Population size (P)	Tournament probability (λ_t)	Crossover rate (λ_c)	Mutation rate (λ_m)
Level 1	100	20	0.5	0.25	0.05
Level 2	200	50	0.7	0.5	0.1
Level 3	400	100	1	0.7	0.3

Taguchi method measures the degree of variation in the fitness function by calculating the signal-to-noise (S/N) ratio. Here, signal and noise each represents the mean and the variance of the fitness function (Phadke, 1989). Taguchi method considers three types of ratio functions, namely the "smaller the better", the "larger the better", and the "nominal is the best". As we aim to minimize the fitness function, our objective function belongs to the "smaller the better" type. The corresponding S/N ratio is

$$S/N = -10\log_{10}\frac{1}{n}\sum_{i=1}^{n}(\textit{fitness function}_i)^2 \tag{37}$$

where n is the number of algorithm executions and *fitness function*$_i$ is the fitness value obtained from the i^{th} replication of the MA. In our experiment, we conducted the GA for 10 times and estimated the average S/N values with $n = 10$. The parameter values that have attained the best average performance in the experiment are the ones with the highest S/N ratios.

For Taguchi experiments, we considered a representative problem instance with seasonal demand. The complete experiment settings and the table of average S/N ratio for each parameter level are available in Appendix B.1. Also, we conduct the experiments only with the GA part of the complete MA by skipping the LSA in order to isolate the effects of GA parameters from the influence of exogenous factors. Figure 25 graphically shows the average S/N ratios for each level of the GA parameters over 10 independent executions.

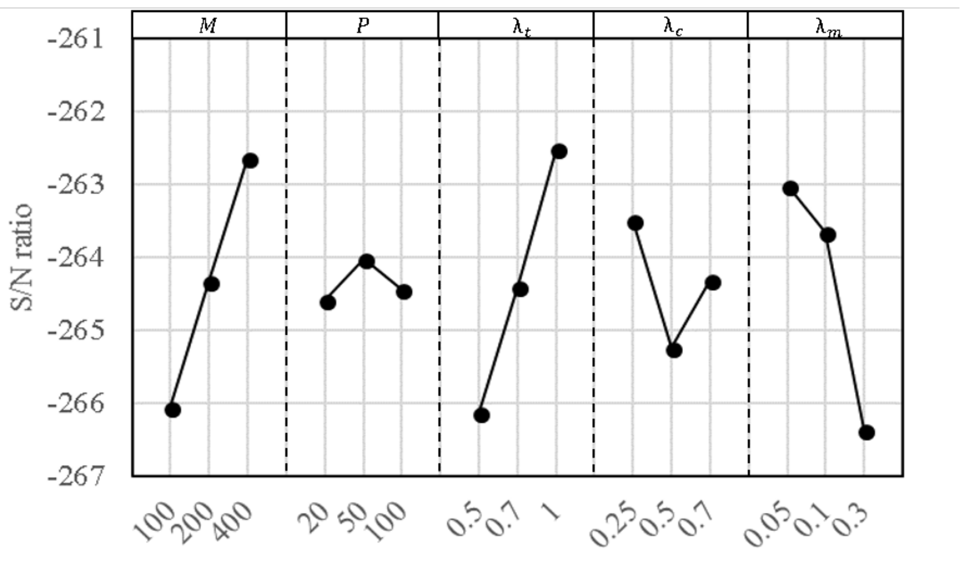


Figure 25: Average S/N ratios for each level of GA parameters over 10 executions of MA (larger S/N ratio is better)

According to Figure 25, Taguchi method suggests to use the following parameter values: $\{M = 400, P = 50, \lambda_t = 1, \lambda_c = 0.25, \lambda_m = 0.05\}$ as these parameter levels have attained the highest S/N ratios. In addition, it is clear that higher population size and tournament probability result in higher S/N ratios because of increased explorative and exploitative power. On the other hand, a higher mutation rate deteriorates the search ability and results in a premature convergence. This is clear from the sharp decline in S/N value for increasing level of mutation rate. The relationship with the other two parameters (i.e., population size and crossover rate) is not so straightforward and appears to be relatively insignificant.

In order to determine the relative size of contribution that each parameter has on the

S/N ratios and to approximate the magnitude of variance caused from noise, we conduct analysis of variance (ANOVA) on computed S/N ratios. ANOVA summarizes the variance of the S/N ratios explained by each parameter and tests if its contribution is statistically significant compared to the variance due to random error. A common approach to test this hypothesis is to apply F-test on ANOVA results (Phadke, 1989). To conduct F-test with one or more independent parameters, their degrees of freedom are determined based on the number of experiments and the number of parameters involved in ANOVA.

ANOVA results in Table 2 show that three parameters $(M, \lambda_t, \lambda_m)$ have the largest effects on the GA's performance. All these three parameters have their p values below 0.01, implying that they have significant effects on the results at 99% confidence level. Also, according to the *Percentage* column, which contains the ratio between each parameter's sum of squares and the total sum of squares, these three parameters constitute more than 80% of total effects on the algorithm's performance. On the other hand, the remaining two parameters did not produce significant effects on the results. Still, care should be taken when selecting their values as their interactions with the other three parameters may have caused significant improvements in the results.

Table 2: ANOVA results for the GA parameters

Parameters	Degrees of freedom	Sum of squares	Mean squares	F value	p value	Percentage
Maximum generation (M)	2	35.32	17.66	11.11	0.00673	26.34%
Population size (P)	2	1.08	0.54	0.34	0.723	0.81%
Tournament rate (λ_t)	2	39.22	19.61	12.33	0.0051	29.25%
Crossover rate (λ_c)	2	9.29	4.64	2.92	0.12	6.93%
Mutation rate (λ_m)	2	38.03	19.01	11.96	0.00552	28.37%
Error	7	11.13	1.59			8.30%
Total	17	134.06				

4.5 Algorithm Summary

Our simulation-optimization algorithm combines both Phase 1 (DES model) and Phase 2 (MA) in order to attain the optimal replenishment strategy that is effective and robust to demand and lead time uncertainty. In Figure, 26, we first summarize the main steps of our MA.

Memetic Algorithm Procedure

(1) Generate initial population according to Section 4.4.5.

(2) Evaluate current population in terms of fitness function formulated in Section 4.4.4.

(3) Copy the best n_{elite} number of chromosomes to the next population without modifying their genes.

(4) Select $P - n_{\text{elite}}$ number of parent chromosomes using binary tournament selection rule with selection rate λ_t, as described in Section 4.4.6.

(5) For each pair of parent chromosomes selected in step (4), perform uniform crossover proposed in Section 4.4.7 to create two new child chromosomes at the rate of λ_c. If no crossover takes place, then both parents survive to the next generation.

(6) Except for n_{elite} best chromosomes, mutate the genes of child chromosomes at the rate of λ_m, as described in Section 4.4.8.

(7) Apply the LSA proposed in Section 4.4.9 to n_{LS} number of chromosomes in each generation and update their genes.

(8) Terminate the algorithm if the maximum number of generations has reached. Otherwise return to Step (2).

Figure 26: Memetic Algorithm procedure

Figure 27 summarizes how the inventory simulation model (Phase 1) and the Memetic Algorithm (Phase 2) are integrated.

Simulation-Optimization Algorithm Summary

(1) Simulate seasonal demand throughout the planning horizon using the trigonometric demand generating function in Section 4.3.1.

(2) Optimize the GA parameters in the MA according to Section 4.4.10.

(3) Follow the main steps of the MA as presented in Figure 26 using the simulation outputs.

  (3.1) When evaluating the fitness value of a solution (i.e., replenishment strategy), determine the minimum number of simulation replications to obtain robust estimates of the expected total cost and expected fill rate based on the dynamic resampling technique described in Section 4.3.3.

  (3.2) Estimate the solution's expected total cost and expected fill rate by simulating for the number of times determined in Step (3.1) using the inventory DES model described in Section 4.3.2.

(4) Stop if the MA has reached the maximum number of generations. Otherwise return to Step (2).

Figure 27: Summary of proposed algorithm that combines Phase 1 Simulation and Phase 2 Memetic Algorithm

5 Computational Study

5.1 Introduction

In this section, the proposed MA is validated on various problem instances with different inventory configurations. Different problem instances were generated by altering the cost parameters and seasonal patterns of the problem described in Section 3, and the performance of the optimal solution produced by the MA was compared with a benchmark solution obtained from an algorithm implemented by Babaï and Dallery (2006). The main goal of this section is to provide a simulation-based analysis of the MA's performance and to gain fruitful insights for handling seasonal demand.

In Section 5.2, we describe how the MA parameters were set before carrying out the experiments. In Section 5.3, the benchmark algorithm implemented by Babaï and Dallery (2006) is presented. Then in Section 5.4, we conduct a comparative analysis with the benchmark algorithm and validate our approach in terms of the solution's expected total cost, computational effort and robustness. We also present a numerical sensitivity analysis in Section 5.5 to study the impact of demand seasonality, cost and inventory structure on the optimal solution. This chapter concludes by providing a numerical evidence for effectiveness of the MA's crucial components in Section 5.6.

5.2 Algorithm Parameters

For all experiments, we used the identical MA parameters chosen according to the performance attained from several test problems described as follows.

- The GA parameters within the MA were determined by the Taguchi experiment introduced in Section 4.4.10. The size of random search n_{rs} for generating initial seasons was chosen from the set $\{100, 500, 1,000, 5,000, 10,000, 20,000, 50,000\}$ to ensure that about 5% of generated season sets are different from the exact MSE-minimizing season set for diversification.
- For $100,000$ randomly populated replenishment strategies, the resampling parameters were selected to obtain less than 1% deviation of the sample mean to the true mean. The true expected total cost and fill rate of a replenishment strategy were estimated by averaging its simulation performance for $10,000$ demand and lead time scenarios. The minimum sample size was set by balancing computational effort and the accuracy of sample estimate through simple sub-experiments.
- The parameters for the NFT fitness function were tuned in order to impose a growing penalty term to infeasible solutions as the generation increases. Specifically, the penalty for 1% deviation from the target service level $FR_{\min}$ was set to increase at a quadratic rate and to become approximately 5 times larger than the average magnitude of the expected total cost at the final generation.
- According to our computational experiments in Section 5.6, applying the LSA to 10 randomly selected chromosomes in each generation brought more robust performance improvement under reasonable computational time.
- The parameters for the DSPSA algorithm were chosen to maximize the algorithm's optimization performance for $10,000$ randomly generated replenishment strategies. The details are described in Section 4.4.9.

The parameters used for the MA are summarized in Table 3. 3.

Table 3: Table of parameters used for the Memetic Algorithm

Dynamic Resampling			
Minimum sample size (S_0)	10	Maximum sample size ($S_{\max}$)	200
Standard error threshold (n_{thr})	1%		
Genetic Algorithm			
Maximum generation (M)	400	Population size (P)	50
Tournament probability (λ_t)	100%	Crossover probability (λ_c)	25%
Mutation probability (λ_m)	5%	Number of elites (n_{elite})	1
Number of random search (n_{rs})	10,000		
Fitness Function			
NFT upper bound (NFT_0)	0.8	Generation multiplier (c)	2
Penalty severity (κ)	2		
Local Search Algorithm			
Number of chromosomes improved by LSA (n_{LS})	10	Type of chromosomes improved by LSA	Random
Coefficient alpha (α)	0.602	Maximum iterations (U)	100
Percentage step size coefficient ($\tilde{u}$)	10%	Budget coefficient (A)	20

We carried out the MA 10 times for every problem instance and reported the best, worst, median and average results produced among those 10 executions. In addition, we reported the average computation time per problem instance to provide an estimation of total computational requirement. The MA has been implemented and executed in Java SE 8 on an Intel(R) Core(TM) workstation with i7-7700HQ CPU at 2.80GHz and 16.0 GB RAM.

In order to visualize the algorithm's performance, we computed the percentage deviation from the benchmark result using Equation (38)

$$\text{Gap} = \frac{f_{\text{MA}} - f_{\text{benchmark}}}{f_{\text{benchmark}}} \tag{38}$$

where f_{MA} is the expected total cost of the optimal solution produced by the MA and $f_{\text{benchmark}}$ is that of the benchmark algorithm. The gap was calculated for the best, worst, median and average MA solutions against the optimal benchmark solution.

5.3 Algorithm by Babaï and Dallery (2006)

In the work of Babaï and Dallery (2006), a similar inventory problem to our problem has been studied, but without considering lead time uncertainty, review and purchase costs. In their paper, the authors propose a simple sequential approach for approximating the optimal safety stock in the single-echelon inventory system with non-stationary demand and the fill rate constraint. The algorithm applies periodic review policy with a time-varying reorder point and a constant reorder quantity (T, R_t, Q). This is a special case of our dynamic (T, R_t, Q_t) strategy that does not consider the dynamic adjustment of the reorder quantity as well as the restriction on the number of seasons for the reorder policies. For convenience, we refer to their algorithm as **BD** method.

In our comparative study, we implemented a slightly modified version of the BD method in order to handle the lead time uncertainty using the Law of Total Probability. Recall that N is the planning horizon, and let SS denote the number of safety stock. Following BD algorithm determines the optimal level of safety stock for a specified target service level and an initial reorder quantity, which is obtained using the simple EOQ model. Then the reorder point R_t for each period is determined by the expected demand during the protection interval (i.e., lead time plus review time) and the number of safety stock.

Approximation algorithm by Babaï and Dallery (2006)

(1) Compute initial reorder quantity according to the EOQ model $Q = \sqrt{\frac{2K\sum_{i=1}^{N}\mu_t}{hN}}$ and set the review time $T = 1$.

(2) For a given lead time L, define demand's standard deviation during lead time and review time $\sigma_L = \sqrt{L+T+1}\sigma$ where σ is the standard deviation of homoscedastic demand d_t (i.e., constant demand variance).

(3) Determine the optimal safety stock SS^* based on the minimum service level constraint and the expected number of lost sales during a reorder cycle. Let $\tilde{\Phi}(\cdot)$ and $\tilde{\varphi}(\cdot)$ be the discrete analogues of standard Normal cumulative distribution and probability density functions respectively (their definitions are available in Appendix A.2).

$$\begin{aligned} Q(1 - FR_{\min}) &= \sum_{k=1}^{\infty} P(L=k)\Big[m_{LS}(SS^*, k)\Big] \\ &= \sum_{k=1}^{\infty} P(L=k)\left[\sigma_k \tilde{\varphi}\left(\frac{SS^*}{\sigma_k}\right) - SS^*\left(1 - \tilde{\Phi}\left(\frac{SS^*}{\sigma_k}\right)\right)\right] \end{aligned}$$

The left-hand side of the equation expresses the maximum number of lost sales that are acceptable under the target service level in a reorder cycle. In order to approximate the average number of demand during a reorder cycle, Babaï and Dallery (2006) use the EOQ model's Q obtained in Step (1). The right-hand side of the equation uses the function $m_{LS}(SS, L)$ to find the expected number of lost sales for a given safety stock SS and a constant lead time L. Since the lead time is a Geometrically distributed discrete random variable, we apply the Law of Total Probability to approximate $m_{LS}(SS, L)$ for stochastic lead time.

(4) Decide the optimal reorder point for all periods $t = 1, 2, \ldots, N$ based on the average demand during the protection interval ahead and the optimal safety stock determined in Step (3)

$$R_t^* = \sum_{k=1}^{\infty} P(L=k)\left[\sum_{j=1}^{k+T+1} \mu_{t+j-1} + SS^*\right]$$

(5) Apply a simple bisection method to find the optimal reorder quantity Q^*, which remains identical throughout the whole planning periods $t = 1, , \dots, N$. Q^* should minimize the objective function (Equation 3) subject to the service level constraint (Equation 5) for the given sequence of optimal reorder points R_t^* with $t = 1, 2, \dots, N$. Set the upper bound for Q^* relative to the size of observable demand throughout the planning horizon. In our problem, we use the EOQ model for the upper bound: $Q^* \in \left\{0, 3\sqrt{\frac{2K\sum_{i=1}^{N}\mu_t}{hN}}\right\}$.

The expansion of the function $m_{LS}(SS^*, k)$ in Step (3), which involves σ_k, $\tilde{\varphi}(\cdot)$ and $\tilde{\Phi}(\cdot)$, has been derived in Babai (2005) by reformulating Equation (39). Equation (39) calculates the expected number of demand in the protection interval that exceeds the period's reorder point $R_t = \sum_{j=1}^{L+T+1} \mu_{t+j-1} + SS$, which eventually gets lost.

$$m_{LS}(SS, L) = \int_{x=\sum_{j=1}^{L+T+1}\mu_{t+j-1}+SS}^{+\infty} \left(x - \sum_{j=1}^{L+T+1} \mu_{t+j-1} + SS\right) \tilde{\varphi}_{L+T+1}(x)dx \tag{39}$$

where the function $\tilde{\varphi}_{L+T+1}(x)$ gives the standard Normal probability mass of observing x number of customer demand over the period of $L + T + 1$.

We make the following three remarks concerning the BD algorithm. Firstly, because the problem is in discrete times, the BD method adds 1 to the protection interval $L + T + 1$ to ensure the completion of possible deliveries and reviews. Secondly, applying the Law of Total Probability introduces the summations with infinite numbers in Step (3) and (4). Since the lead time is a discrete random variable, we can approximate these summations by iterating over all positive integer lead times until its probability equals to 0 to an arbitrary level of precision. In our thesis, we used 10^{-5} as the probability precision level. Lastly, the optimal safety stock SS^* in Step (3) can be determined numerically by using an algorithm of dichotomy (Babaï and Dallery, 2006). In our experiment, we applied a simple binary search method.

Step (5) is an extension we made to the original BD method to improve the quality of the optimal reorder quantity Q^*. Since the original BD method reduces the problem into a one-dimensional optimization problem where the R_t^* is the only decision variable, we can first determine an appropriate level of R_t^* for an arbitrary reorder quantity Q. In this research we use the EOQ model as addressed in the original algorithm. Once R_t^* has been determined, Q can be further improved by numerically searching for the optimal reorder quantity Q^*. We emphasize that Step (5) is especially important for our benchmark experiment because the traditional EOQ model does not include per unit purchasing cost p. Through numerical optimization, we can determine Q^* that incorporates p and conduct fair comparative study with our own MA.

5.4 Benchmark Results

5.4.1 Problem Setting

Our MA has been applied to determine the optimal replenishment strategy for the next 5 years of weekly inventory operations with a 98% target service level. The parameters for the weekly demand, inventory settings and costs have been set according to Table 4. Both our MA and the benchmark BD method were used to solve this problem under identical conditions and the results were compared in terms of expected total inventory cost, robustness and computational time. The values in the curly brackets in Table 4 were selected individually depending on the problem instance. We emphasize that the review cost r is set equal to 0 since the BD method does not incorporate r.

Table 4: Design parameters for the experiment

Inventory plan			
Planning horizon (N)	5 years	Time unit	Weeks
Seasonal cycle length (Y)	52	Target service level ($FR_{\min}$)	98%
Maximum review time ($T_{\max}$)	8		
Demand characteristics			
Number of seasons (m)	4	Mean demand level (D)	200
Demand standard deviation (σ)	20	Seasonality amplitude 1 (E_1)	150
Seasonality amplitude 2 (E_2)	$\{0, 1\}$	Seasonality lag 1 (v_1)	0
Seasonality lag 2 (v_2)	0		
Lead time and cost			
Lead time rate (q)	0.5	Set up cost per order (K)	$\{20, 50, 80\}$
Holding cost per unit per time (h)	$\{0.2, 0.4, 0.6\}$	Review cost per review (r)	0
Purchase cost per unit (p)	1		

As one can observe, only three problem parameters were adjusted to generate 18 unique problem instances.

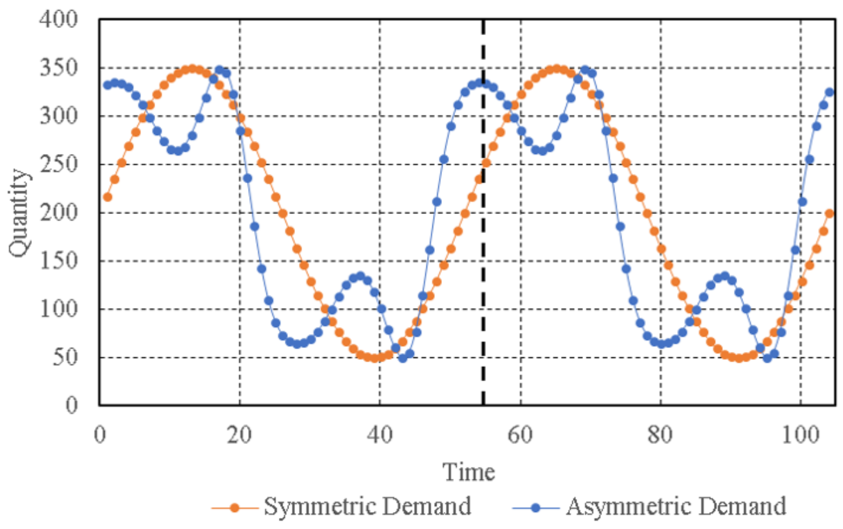


Figure 28: Seasonal demand with symmetric ($E_2 = 0$) and asymmetric ($E_2 = 1$) amplitudes

We tested our algorithm on all 18 combinations of $K \in \{20, 50, 80\}$, $h \in \{0.2, 0.4, 0.6\}$ and $E_2 \in \{0, 1\}$ while other design parameters were set fixed according to Table 4. We refer to a problem instance with a unique combination of these three parameters as a **scenario**.

Setting the seasonality amplitude E_2 equal to 0 generates a symmetrical seasonal demand pattern with a single peak and a trough during a demand cycle. But when its value is set equal to 1, the demand fluctuates more intensely with multiple peaks and troughs within a cycle. Figure 28 demonstrates both symmetric and asymmetric seasonal demand patterns generated by setting $E_2 = 0$ and 1. The complete list of scenarios is available in Appendix B.

For clarity, we briefly describe the characteristics of each problem scenario. The scenarios with odd-valued indices $(1, 3, \ldots, 17)$ consider symmetric demand seasonality with $E_2 = 0$ while the scenarios with even-valued indices $(2, 4, \ldots, 18)$ have asymmetric demand seasonality with $E_2 = 1$. Also, each of three equally sized subsets of 18 scenarios ($\{1, 2, \ldots, 6\}$, $\{7, 8, \ldots, 12\}$, $\{13, 14, \ldots, 18\}$) assigns a unique setup cost from $K = 20, 50, 80$, respectively. Finally, within each of these subsets, every successive pair of scenarios applies the same holding cost from the set $h \in \{0.2, 0.4, 0.6\}$ (e.g., for the first subset $\{1, 2, \ldots, 6\}$, each pair of scenarios $\{1, 2\}$, $\{3, 4\}$ and $\{5, 6\}$ applies $h = 0.2, 0.4, 0.6$ respectively).

5.4.2 Performance Comparison

Figure 29 graphically summarizes the performance of the solutions determined by the MA and the BD method. Each box plot illustrates the distribution of the expected total costs for the optimal solutions found by 10 independent MA runs in respective problem scenario. The band inside the box represents the median value while the bottom and the top of the box represents the first and the third quartile of the distribution. The upper and lower tails represent the maximum and the minimum value. The length of each box plot is inspected to assess the robustness and convergence of the algorithm, since the MA terminates as soon as the maximum number of generations M is reached. Accordingly, a long box may imply that the MA was not able to produce consistently good solution and that its performance is sensitive to inventory setting. On the contrary, the BD method was executed only once because the algorithm does not involve random variables in its procedure unlike the MA, which incorporates random variables in the parent selection, the reproduction and the LSA.

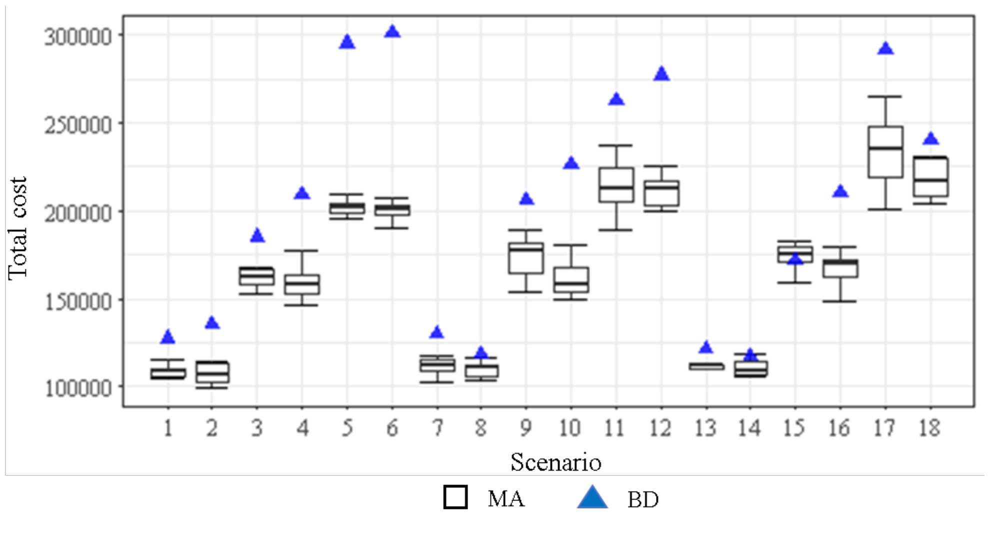


Figure 29: Box plot of expected total costs of 10 optimal solutions found by the MA and the optimal solution by the BD algorithm

Based on the results in Figure 29, it is clear that the MA is outperforming the BD method for most of the scenarios. We observe that for all of the scenarios the optimal solutions by the MA and the BD method satisfied the 98% minimum target service level. The increasing pattern of the expected total cost within each of three scenario subsets $\{1, 2, \ldots, 6\}$, $\{7, 8, \ldots, 12\}$ and $\{13, 14, \ldots, 18\}$ is due to the increasing unit holding cost h by 0.2 units. Indeed, as we explained earlier in Section 5.4.1, each of these scenario subsets applies a different unit holding cost: $h = 0.2, 0.4$, and 0.6 respectively. Besides the increase in the average cost level, the size of the cost difference between the MA and the BD method also increases for increasing h, which implies that the optimal solution from the BD method is holding excessive inventory compared to the solution by our MA. Furthermore, the BD method is inefficient at handling irregular, asymmetric seasonality since the average costs for asymmetric scenarios with even number indices are on average 3.22% higher than for the symmetric scenarios with odd number indices. On the other hand, the MA showed superior performance with irregular seasonality: the optimal expected costs for asymmetric scenarios were on average 2.89% lower than for symmetric scenarios, which suggests that the MA is more robust to irregular seasonal patterns. Finally, the increasing setup cost K appears to complicate the MA's convergence as the average length of the box plots for higher k tends to increase. For instance, the average length of box plots is larger for scenarios $13, 14, \ldots, 18$ than for scenarios $7, 8, \ldots, 12$. But this is not so evident from Figure 29 and therefore it is more closely examined in Section 5.5.2.

The cost gap between the MA and the BD method is illustrated in Figure 30, plotting one-to-one comparison of each of 10 MA solutions to the corresponding BD solution in every scenario. By our definition of benchmark gap in Equation (38), negative-valued gap indicates that the considered MA solution achieves a lower cost (hence a better quality) than the associated BD solution.

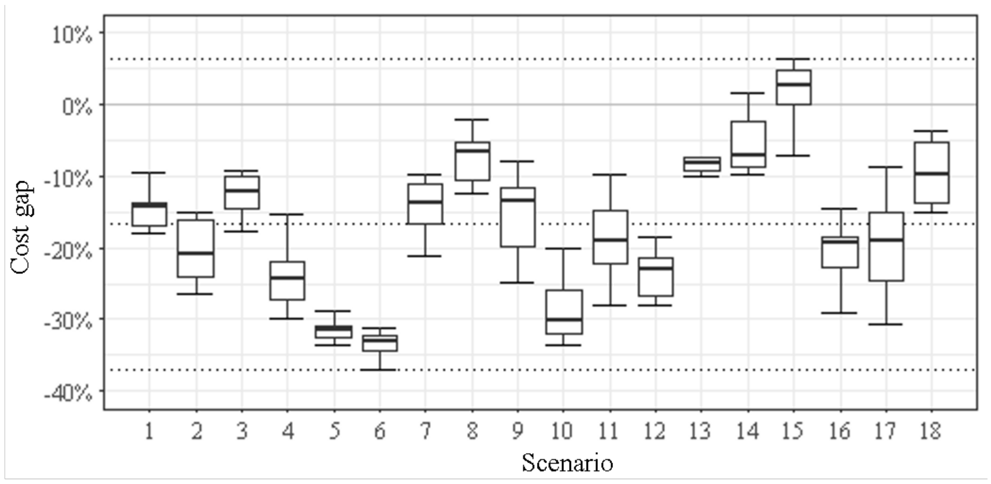


Figure 30: Box plot of cost gaps of 10 optimal solutions by the MA against the solution produced by BD method

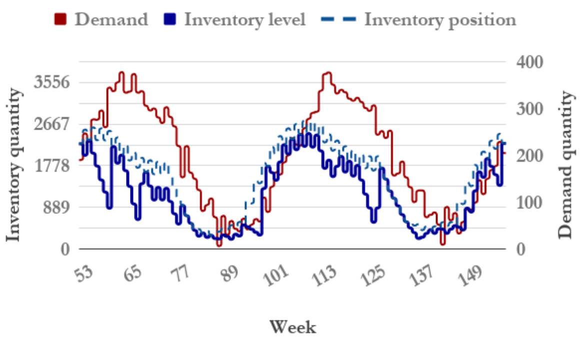
According to Figure 30 and Table 12, the MA saves average 16% of expected total cost compared to the BD method. Moreover, the average of worst-case cost gap, which compares each scenario's most costly solution among the 10 MA solutions to the optimal BD solution, is -8.56%. Therefore, these gaps strongly suggest that the quality of the solutions produced by the MA are superior on average to that of the solutions made by the BD method. However, the size of each individual gap tends to vary widely for different problem scenarios. It is mainly due to the lack of flexibility in the BD method, which determines its reorder points based on the simple EOQ model and a constant number of safety stock. For scenarios that require smaller safety stock for the BD method (e.g., scenario 14 and 15), the optimal BD solutions show competitive results to the MA solutions, while for the opposite types of problems the solution qualities are quite poor (e.g., scenario 5 and 6). Overall, the performance of the proposed MA can be seen more superior to the BD method, but the magnitude of cost savings can vary per problem instance.

We confirmed that the average computational time required by a single MA execution is approximately 268 seconds for all scenarios. This is significantly longer than the BD method, which terminates almost instantly. Requiring more computational requirements may be a significant drawback of the MA when the problem is considered in the context of online optimization, which would require real-time updates of the reorder policies. However, in practice the demand forecasts and replenishment schedules for fast-moving products are typically planned on a weekly or a monthly basis (Bilgen and Günther, 2010; Thomassey, 2014). Considering that the solutions from the MA attain better overall quality than the BD solutions, one should not simply understate the practicality of applying the MA in real-life inventory systems. In Section 5.5, the impact of different inventory settings on the total computational time are further investigated.

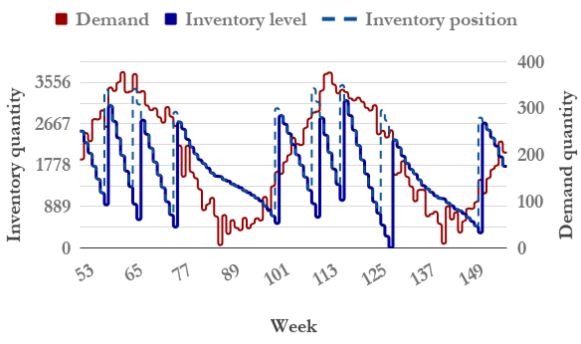
5.4.3 Strategy Comparison

Symmetric seasonality: Single simulation performance

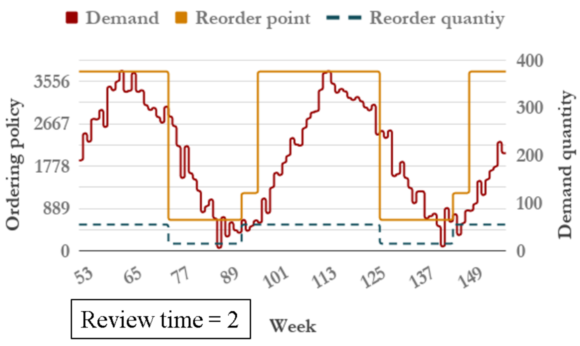
The three largest factors that cause the MA to achieve lower average costs than the BD method are its dynamic safety stock, dynamic reorder quantity and operation of anticipation inventory. Despite its simplicity, the BD method assumes for a constant safety stock and a static reorder quantity, which sacrifices flexibility and responsiveness to demand fluctuations. In this section, we consider the optimal BD and MA strategies found for scenario 1 to understand the optimal structure and characteristics for handling symmetric demand. Consider Figure 31, which shows an instance of the stochastic simulation of the optimal replenishment strategies found by both methods in scenario 1. Identical demand and lead time realizations were used for both methods to make these graphs. For clear illustrations, the graphs present the results for the first two seasonal cycles (excluding the initial cycle for simulation warm-up) and leave out three remaining cycles. Also, we selected MA solution that has achieved the best performance among 10 MA solutions from 10 MA runs.



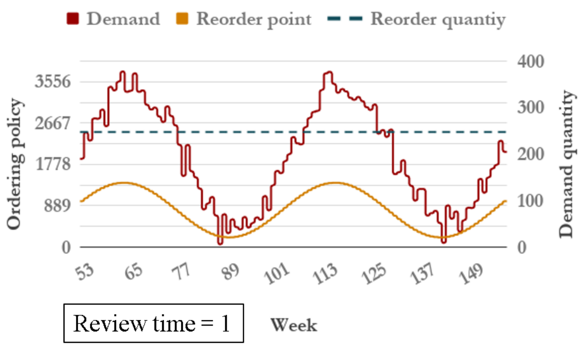
(a) Inventory situation of the optimal solution by the MA



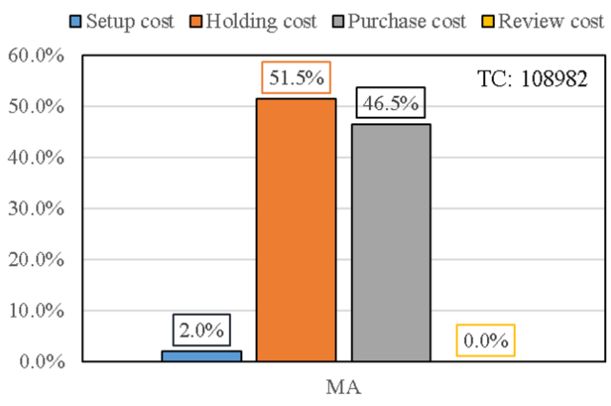
(b) Inventory situation of the optimal solution by the BD method



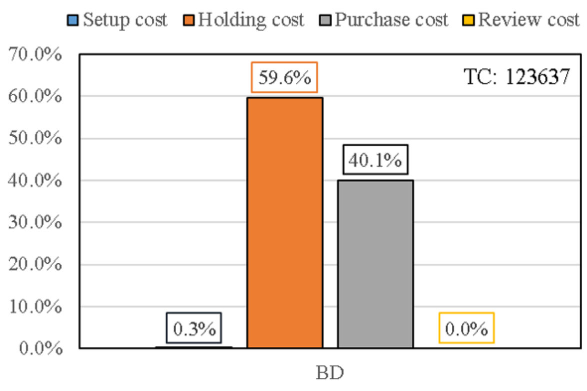
(c) Reorder point and quantity of the optimal solution by the MA



(d) Reorder point and quantity of the optimal solution by the BD method



(e) Cost profile of the optimal solution by the MA



(f) Cost profile of the optimal solution by the BD method

Figure 31: Stochastic simulation instance of the optimal replenishment strategies found by the MA and the BD method in scenario 1

Although optimal reorder points in the BD strategy are dynamically adjusted to handle demand's seasonal fluctuation, it uses a constant reorder quantity throughout the planning horizon. A direct consequence of this is the excess stock during off-seasons and an increased risk of stock-out during peak seasons. As can be seen from Figure 31b, the inventory level changes more rapidly with the BD solution than with the MA solution in Figure 31a. Since the BD solution sets a high reorder quantity with relatively low reorder points, every new order arrival sharply increases the inventory level. However, large number of leftover batches form high inventory level during off-seasons (weeks $[79, 98]$ and $[131, 149]$). This has driven up the holding cost substantially as shown in Figure 31f. Also, because of its low reorder points by the end of the peak seasons (weeks $[70, 77]$ and $[122, 129]$ in Figure 31d) the risk of stock-out is high when the lead time is unexpectedly long. Observe that in week 129 the inventory level hits zero due to unexpectedly long lead time.

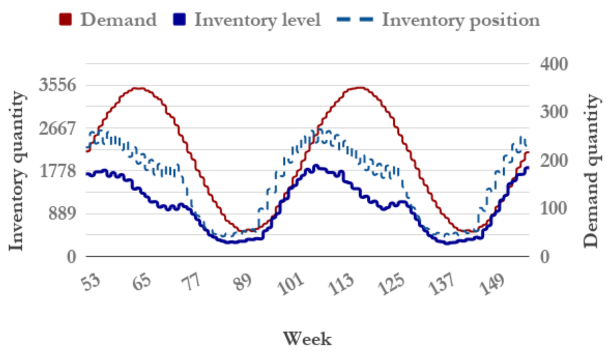
The results for the optimal MA solution illustrated in Figure 31a and 31c show clear differences to the optimal BD solution. Firstly, the inventory level depicted in Figure 31a changes more smoothly and in accordance with the seasonal fluctuations. The inventory level is high for peak seasons to reduce stock-out risks, while it is kept low for off-seasons to minimize the holding cost. The primary driver of such a smooth behavior is the solution's extremely seasonal reorder policy. The policy shown in Figure 31c sets a very high reorder point during peak seasons (equal to 3766) while keeping the reorder quantity low (equal to 552). This results in more frequent replenishments with small batch sizes, improving its overall responsiveness to demand volatility. The review time is set equal to 2 units instead of 1 unit for this reason: even more frequent replenishments will needlessly increase both order setup and holding cost. During off-seasons, both reorder point and reorder quantity are kept low to reduce excess inventory.

Figure 31e and 31f demonstrate the expected proportion of each cost component relative to the total expected cost for MA and BD optimal solutions. For the BD method, the expected holding cost is the most dominating component of the expected total cost because of the excess inventory during off-seasons. For the MA, on the other hand, the size of total holding cost and purchasing cost are more balanced. In fact, the expected total reordering cost (i.e., sum of expected total setup cost and purchasing cost) makes up to 48.5% of the expected total cost, which is almost equal to the size of expected holding cost (51.5%). This is a clear indication that the optimal MA solution is more efficient in managing excess inventory and making balanced replenishment decisions.

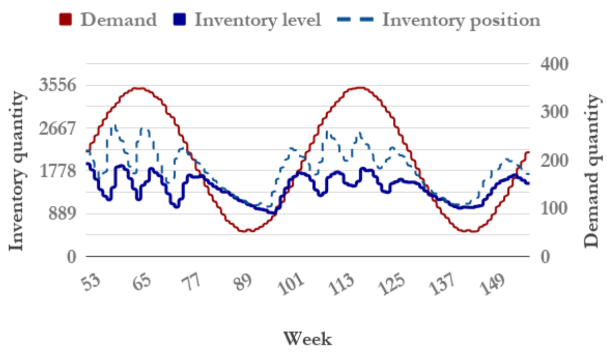
An important aspect of the optimal MA solution is the operation of anticipation inventory in order to minimize the impact of seasonal changes. Consider week 75 and 127 in Figure 31c, where transitions from the highest to the lowest reorder policy takes place. Besides other elements, the anticipatory timing of these transitions make them effective to prepare for the upcoming demand fall. Observe that these transitions take place during peak seasons and not by the start of subsequent off-seasons. As a consequence, the average inventory level in Figure 31a starts to drop already during peak season and reaches to a low point by the start of off-seasons (week 85 and 137). This is a type of anticipatory strategy that uses up any accumulated inventory during peak season in order to limit redundant inventory during subsequent off-seasons. An opposite type of anticipatory move takes place in week 92 and 144 to prevent stock-out situations for the upcoming demand peak. In Figure 31c, the reorder policy shifts to a higher level before the end of off-seasons (week 92 and 144). Then, because of finite lead time, the inventory level starts to increase only by the end of the off-seasons (week 99 and 147 in Figure 31a). By keeping the reorder policy at its high level during the peak season, the inventory level continues to rise along with the demand and further reduces the risk of running out of stock. Therefore, preparing for the peak season early during the off-season could secure a sufficient service level against the sharp demand increase.

Symmetric seasonality: Average simulation performance

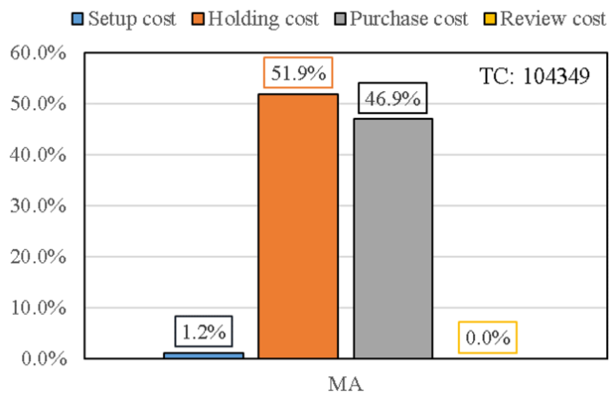
Above, we have looked into the performance of the optimal MA and BD solutions in a single instance of inventory simulation. To understand their average performance for different realizations of stochastic demand and lead time, we summarized their results for 100 different randomly generated simulation instances. Figure 32 shows the average performance of the optimal MA and BD strategies for 100 different demand and lead time scenarios.



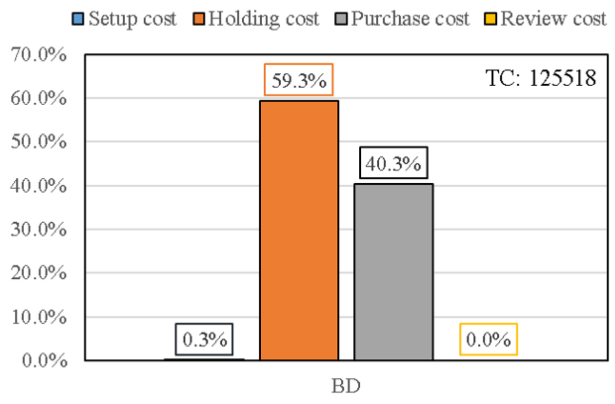
(a) Average inventory situation of the optimal solution produced by the MA method



(b) Average inventory situation of the optimal solution produced by the BD method



(c) Cost profile of the optimal solution produced by the MA method



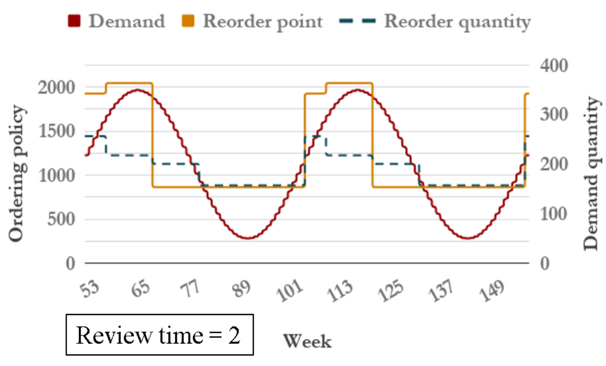
(d) Cost profile of the optimal solution produced by the BD method

Figure 32: Average simulation results of the optimal replenishment strategies found by the MA and the BD method in scenario 1

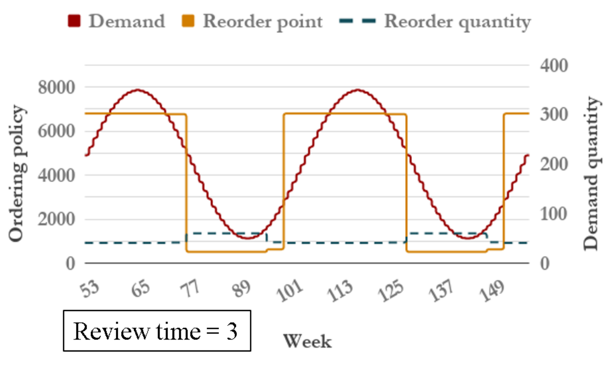
The average demand, inventory level and inventory position over 100 random simulation instances for the optimal MA and BD solutions are shown in Figure 32a and 32b. As discussed earlier, the optimal MA strategy operates anticipation stock to protect from seasonal shocks. This is observed from the average inventory level in Figure 32a, which rises along with climbing demand (week 90 and 147) and starts to fall before the end of the peak season (week 65 and 113). Furthermore, the gap between inventory position and inventory level remains wide during peak seasons because of the solution's high reorder point, but it narrows quickly before the end of the peak season, indicating that replenishment decisions are responsive to anticipated seasonal fluctuations. On the other hand, the average inventory level for the optimal BD strategy in Figure 32b is somewhat bumpy and insensitive to seasonal demand fluctuations. Since replenishments are made less frequently and in larger batches, both inventory position and inventory level tend to fluctuate during peak seasons. Also because this strategy does not employ anticipation inventory the inventory level is kept very high even during off-seasons (weeks [80, 95] and [130, 149]). Finally, as we saw earlier the holding cost is still the most dominating cost component of the BD strategy, unlike the balanced cost profile of the MA strategy according to Figure 32c and 32d.

Symmetric seasonality: Average replenishment strategy

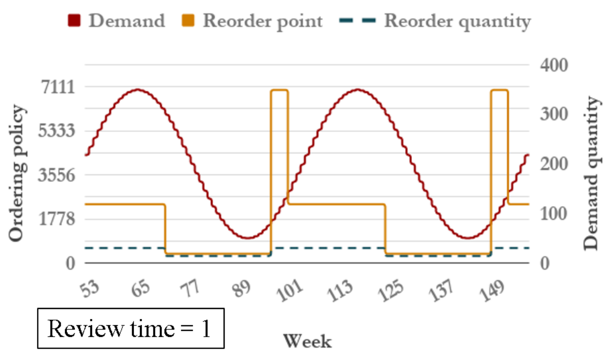
Recall that for our benchmark experiment the MA was executed for 10 times, and we compared its best, worst, median and average performance to the BD method under different problem settings. During this procedure, each of 10 individual MA executions produced a unique optimal replenishment strategy that composed of radically different reorder parameters due to the special multi-modality of our objective function. This implies that two very different strategies may have similar objective value, while two very similar strategies may achieve largely different performance. Figure 33 presents first four optimal strategies found from four different MA executions in scenario 1 that attained similar level of expected total cost. These four strategies all fulfilled the 98% target service level and the difference between their expected total costs was very small; the largest difference was less than 1% of the average of their expected total costs.



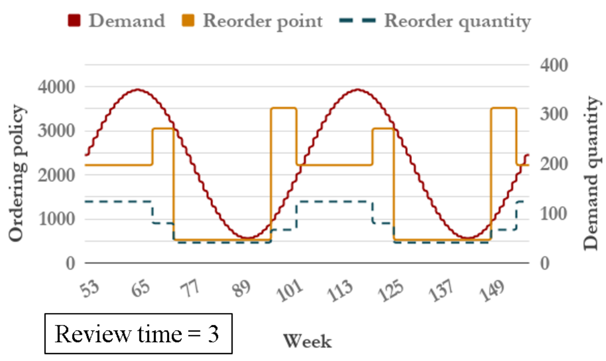
(a) Optimal strategy from the first MA run



(b) Optimal strategy from the second MA run



(c) Optimal strategy from the third MA run



(d) Optimal strategy from the fourth MA run

Figure 33: Four optimal replenishment strategies produced by the MA in scenario 1

All four strategies commonly have higher reorder points than the associated reorder quantities and the transitions between seasons take place prior to the actual transitions in demand. However, the magnitude of the reorder points and reorder quantities is clearly different between strategies, and the seasonal transitions happen at slightly different times for different reorder policies. Such differences may indicate that the MA has not converged to the global optimum and that it would require more iterations. However, for a limited computational budget, making iterations until perfect convergence is not realistic and may even be unnecessary. Indeed, we can already identify common characteristics among these four strategies and gain insights of the optimal features that are essential for managing seasonal demand. Firstly, considering that an unreasonably high (low) reorder point is mainly to encourage (avoid) issuing a new replenishment order when possible, their precise values might be less important than how they first appear. Secondly, the absolute value of reorder quantities has more significant impact on the amount of inventory throughout the cycle. This is why even minor adjustments of reorder quantities across seasons could bring such dramatic changes in the inventory level. Lastly, one can observe that the reorder quantities reflect the length of review time. For a shorter (longer) review time, the average reorder quantities are lower (higher) since replenishment decisions can be made more (less) frequently. Therefore, the dramatic difference in parameters' precise values may at first appear to be random and difficult to generalize, but in fact they share important features that can explain their performance similarity.

To better visualize the common features, an average of 10 optimal replenishments found by the MA can be plotted. We mention that this is nothing more than the simple arithmetic mean of 10 optimal replenishments found by the MA, which defines the average reorder point and reorder quantity at time t as $\bar{R}_t = \frac{1}{10}\sum_{i=1}^{10} R_{t,i}$ and $\bar{Q}_t = \frac{1}{10}\sum_{i=1}^{10} Q_{t,i}$ with $R_{t,i}$ and $Q_{t,i}$ being the optimal reorder point and reorder quantity at time t determined from the i^{th} MA run, respectively. The average review time $\bar{T}$ is also determined in a similar fashion. Since taking a simple average implies overlooking the correlation between seasons, using this type of averaged strategy incurred an average of 46.62% higher expected total cost for all scenarios than the average performance of an optimal replenishment strategy produced by a single execution of the MA (full results are available in Table 15 of Appendix B.2.3). Hence, we only use this graph with an intention to illustrate the common features among the optimal solutions found by different MA executions, and not for actual implementation. Figure 34 shows the average of 10 optimal strategies from 10 MA runs. The graph clearly reflects above-mentioned common characteristics in the optimal replenishment strategies: (1) anticipatory timing, (2) wide gap between reorder point and reorder quantity, and (3) stable and low reorder quantity with more dramatic and high reorder point.

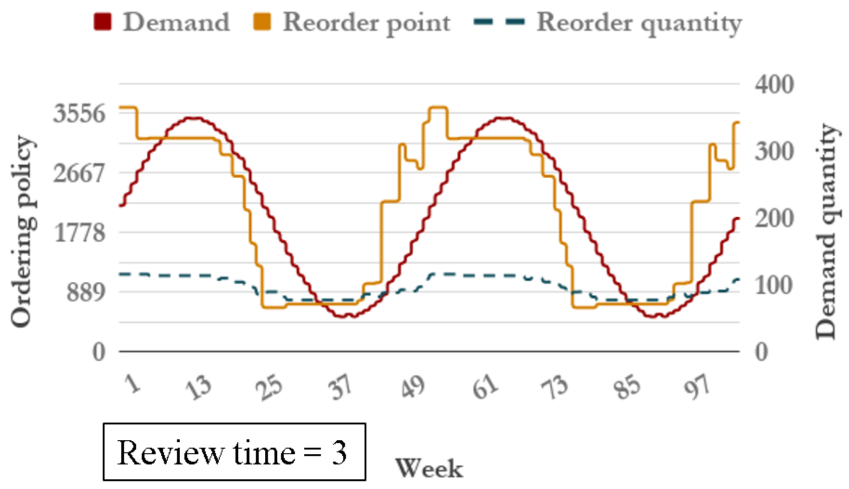
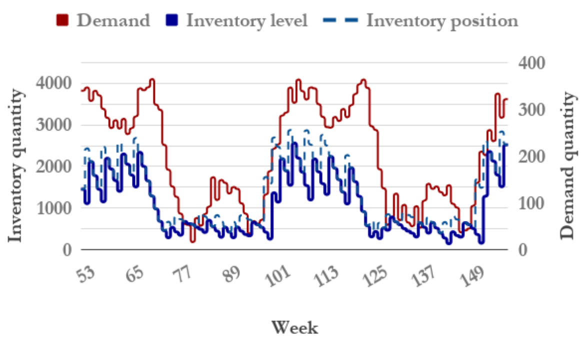


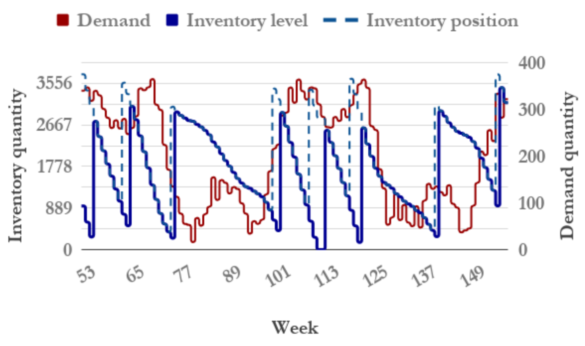
Figure 34: Average cost of 10 optimal solutions produced by the MA in scenario 1

Asymmetric seasonality: Single simulation performance

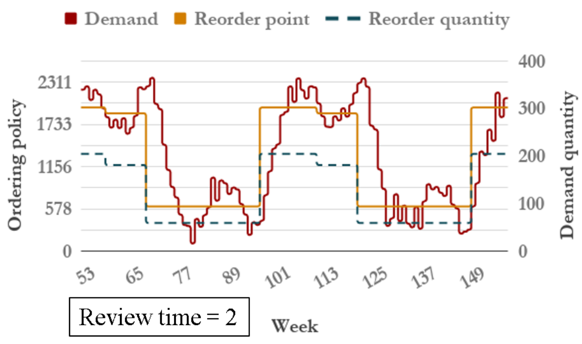
An asymmetric demand series has multiple peaks and troughs within a seasonal cycle and changes more rapidly over time. Consequently, the optimal replenishment strategy for symmetric seasonal demand may be inefficient to manage asymmetric seasonality. To understand how the MA and the BD method handle asymmetric demand, we investigated their optimal strategies in scenario 2, which generates asymmetric demand with $E_2 = 1$. Figure 31 presents a simulation instance of the optimal replenishment strategy found by the MA and the BD method. For a better illustration, we have selected the MA solution that has achieved the best performance among 10 MA solutions.



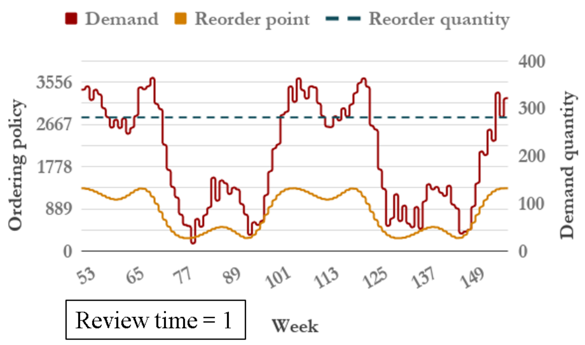
(a) Inventory situation of the optimal solution by the MA



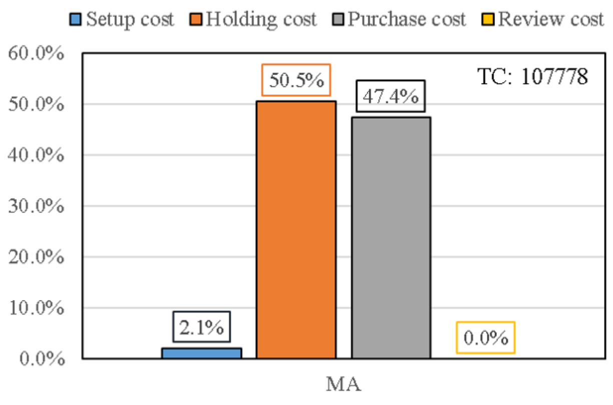
(b) Inventory situation of the optimal solution by the BD method



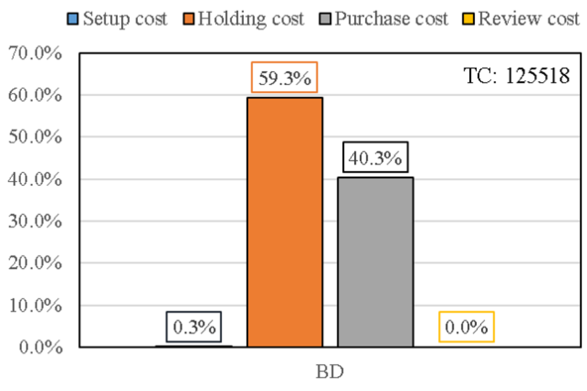
(c) Reorder point and quantity of the optimal solution by the MA



(d) Reorder point and quantity of the optimal solution by the BD method



(e) Cost profile of the optimal solution by the MA



(f) Cost profile of the optimal solution by the BD method

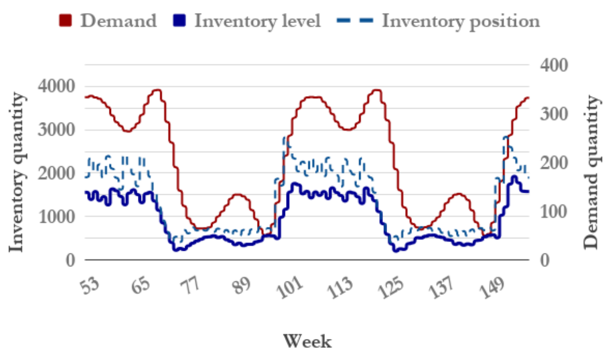
Figure 35: Stochastic simulation instance of the optimal replenishment strategies found by the MA and the BD method in scenario 2

The optimal MA strategy for asymmetric demand seems similar to the strategy for symmetric demand. As shown in Figure 35c, both reorder point and reorder quantity shift together with demand seasonality, and the reorder point is higher than the reorder quantity. Also, their transition from peak season to off-season occurs earlier in anticipation of the immediate decline in demand to reduce excess inventory in off-season. As a result, the inventory level demonstrated in Figure 35a develops seasonally in accordance to the demand level (week 67 and 119). However, there is a remarkable difference from the symmetric seasonality case, which is caused from setting higher reorder quantity for peak season. By doing so, the inventory level could grow very quickly for rising demand unlike the symmetric strategy that raises the inventory level more gradually (week 101 and 151 in Figure 35a). Also, reorder quantity is kept at a reasonably high level during off-season to protect from the secondary demand peak in week 89 and 141.

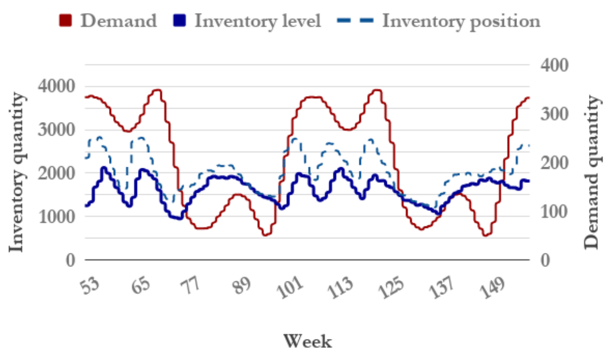
The optimal strategy produced by the BD algorithm creates more extreme variations of the inventory level because of excessively high reorder quantity. In Figure 35b, the inventory level strongly fluctuates between 3,300 and 0. Accordingly, the amount of excess stock during off-season remains high as the strategy does not make anticipatory decisions. Finally, the strategy's chance of facing stock-out during peak season is very high because of stochastic lead time. In weeks 75, 113 and 123 of Figure 35b, the inventory level approaches to zero due to unexpectedly long lead time. Identical to the symmetric seasonality case, the excess inventory added significant weight to the expected total holding cost and created an unbalanced cost profile in Figure 35f. This is very opposite to the balanced profile of the MA solution in Figure 35e that weighs both expected holding cost and the expected reordering cost about equally.

Asymmetric seasonality: Average simulation performance

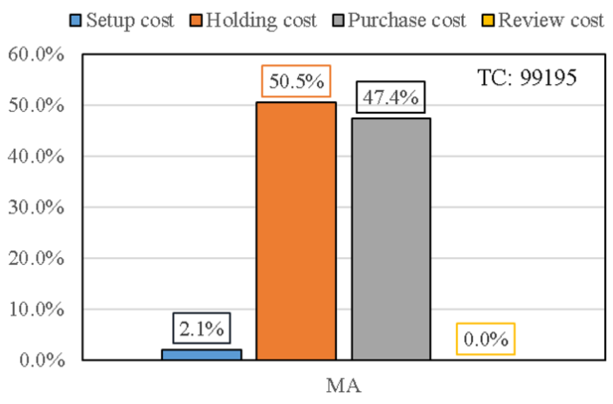
The optimal strategies' average performance over 100 different simulation instances is shown in Figure 36.



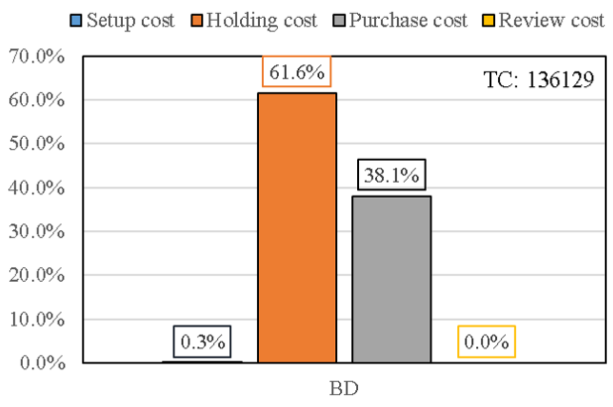
(a) Average inventory situation of the optimal solution produced by the MA



(b) Average inventory situation of the optimal solution produced by the BD method



(c) Cost profile of the optimal solution produced by the MA



(d) Cost profile of the optimal solution produced by the BD method

Figure 36: Average simulation results of the optimal replenishment strategies found by the MA and the BD method in scenario 2

In case of the optimal BD solution the size of redundant inventory in off-season is substantial (weeks [75, 100] and [125, 149] in Figure 36b) compared to the optimal MA solution (same weeks in Figure 36a). Comparing the average cost charts in Figure 36c and 36d, it is more evident that the optimal BD solution is carrying excess inventory, which has driven up the expected total holding cost, while the optimal MA solution is carefully balancing the expected costs of holding and reordering decisions. Hence, more frequent replenishments in smaller batch sizes can be very effective in reducing the excess inventory during off-seasons and handling irregular seasonality.

Asymmetric seasonality: Average replenishment strategy

The simple average of 10 optimal replenishment strategies identified by 10 independent MA runs in scenario 2 is plotted in Figure 37. Again, this result is presented only to demonstrate the common properties among those 10 different MA solutions and not for the actual implementation (Table 15 in Appendix B.2.3 demonstrates the performance loss by implementing the average strategy). We can confirm that the gap between the reorder point and the reorder quantity is noticeably smaller and that the reorder quantities are more sensitive to the demand seasonality than for the symmetric demand example in Figure 34. One can also observe that the reorder point is adjusted about 6 weeks prior to the actual demand in peak seasons. See week 57, for an example, where the reorder point dips during the peak season in response to the intra-peak decline of demand in week 63.

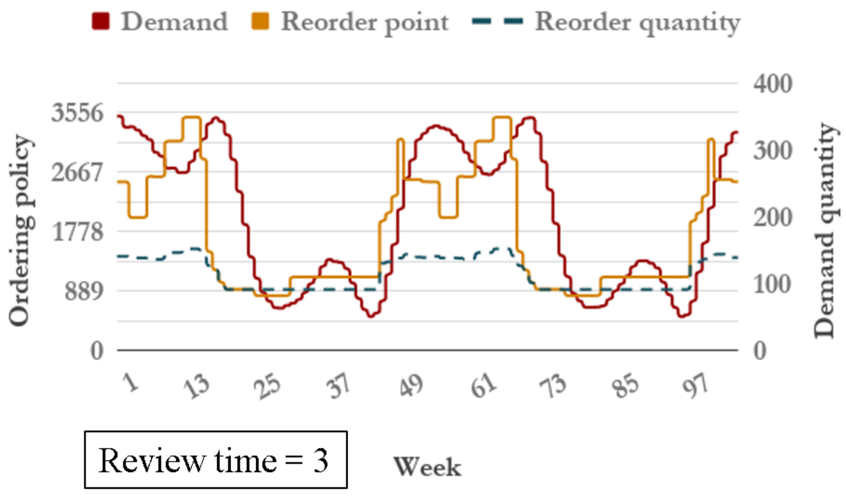


Figure 37: Average of 10 optimal solutions produced by the MA in scenario 2

To summarize, the optimal replenishment strategy found by the proposed MA employs anticipation stock to manage the inventory level throughout the seasons and to reduce the expected total inventory cost for both symmetric and asymmetric seasonal demand. Also, the optimal MA strategy keeps the reorder quantity low while setting the reorder point high in order to maintain flexibility and agility against seasonal demand fluctuations. The solution by the BD method, on the other hand, is not very capable of preventing excess inventory during off-seasons and of circumventing the stock-out risk when the lead time is unexpectedly longer than usual. Therefore, our MA could deliver substantial cost savings and improvement in service stability, which are consequential enough to offset its computational burden for common practical inventory problems.

5.5 Numerical Analysis

In this section, a more detailed simulation-based numerical analysis is presented to assess the influence of problem parameters on the optimal solutions of the proposed MA. Earlier in Section 5.4 the overall performance of our MA has been discussed in comparison to the benchmark BD method. In this section, we focus on identifying the key factors that have caused the MA's superior or inferior performance in the benchmark experiment. The central purpose of our analysis is to provide empirical evidence for the algorithm's robustness to experiment conditions including demand seasonality, cost structure and more. Therefore, through this chapter we aim to provide a valuable insight of the optimal replenishment strategy's practical applicability.

For a concise investigation within limited time constraint, the analysis was performed in two representative problem instances: scenario 1 and 10. These two scenarios were most appropriate to examine the algorithm's sensitivity to symmetry or asymmetry of demand seasonality as well as to different cost structures. Additionally, we observed the impact of different problem parameters by altering their values one at a time while keeping all the other parameters fixed at their original values. In Section 5.5.1, we investigate the effect of demand seasonality in the performance of the optimal replenishment strategy. The MA's sensitivity to different cost parameters (Section 5.5.2), target service level (Section 5.5.3), planning horizon (Section 5.5.4) and the number of seasons (Section 5.5.5) will be further examined.

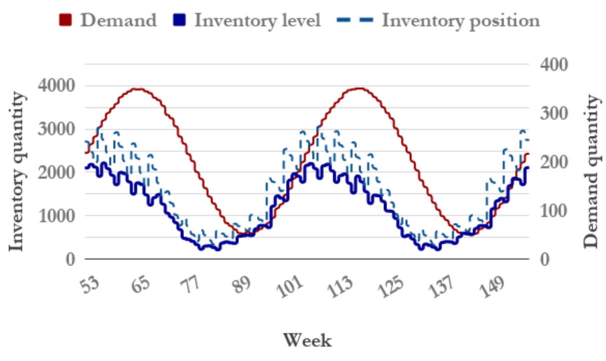
5.5.1 Effect of Demand Seasonality

The most influential factor to the performance of the proposed algorithm is the seasonality in demand. Incorrect treatment of demand seasonality can result in superfluous amount of inventory during off-seasons and serious risk of losing demand during peak seasons. Similarly, applying a seasonal replenishment strategy to manage stationary demand may bring costly consequences. In this section, we provide the results from three separate sub-experiments that demonstrate the impact of incorrect seasonality management. The purpose of first and second sub-experiment is to observe the impact of incorrectly applying seasonal (stationary) replenishment strategy to stationary (seasonal) demand. The last sub-experiment explicitly illustrates the benefit of applying anticipation inventory.

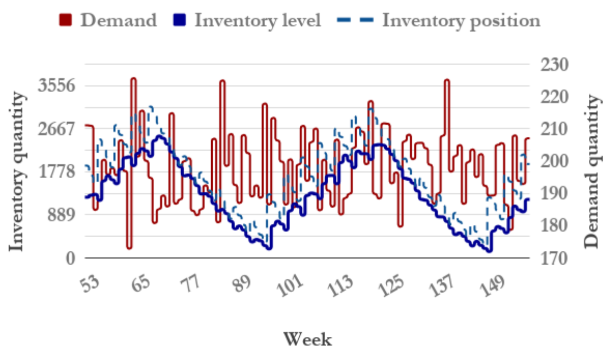
Seasonal policy for stationary demand

In many practical retail inventories, the reorder policy is calculated for the inventory's mid-term operation spanning several weeks or months (Bilgen and Günther, 2010; Thomassey, 2014). But in most real-life retail environment, the strength of seasonality is volatile throughout the time. For example, once very seasonal demand for a popular fashion item may become more stationary when the item loses popularity over weeks and only regular customers insist to purchase the item. Hence, the cost of managing (nearly) stationary demand with the old seasonal replenishment strategy may be significant. To test this premise, we have designed following simple experiment: first, the optimal seasonal replenishment strategy is determined by executing the MA for scenario 1 and 10 with seasonal demand. Second, a series of stationary demand that is analogous to the seasonal demand is generated by randomly resampling the seasonal demand for each time period without replacement. Finally, the seasonal replenishment strategy is applied to the resampled stationary demand and its performance is evaluated.

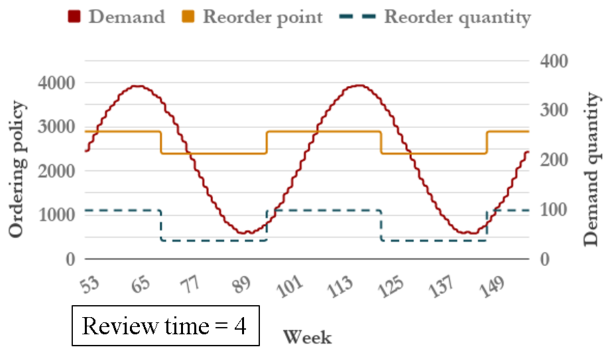
Figure 38 illustrates the average simulation results of applying the optimal seasonal policy in scenario 1 to simulated stationary demand. These graphs were produced by performing 100 simulation replications and averaging their demand, cost, inventory position and inventory level.



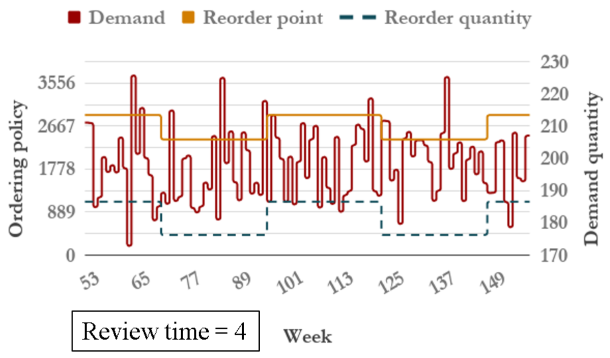
(a) Average inventory situation in seasonal demand



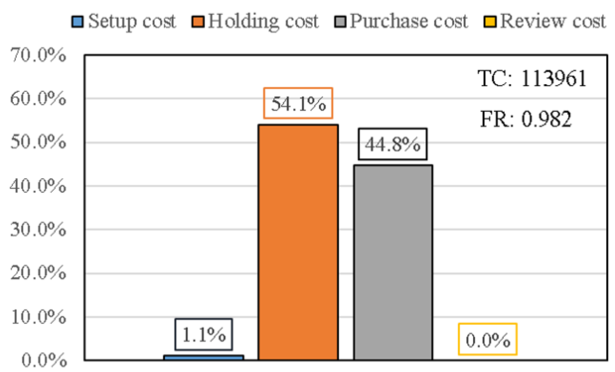
(b) Average inventory situation in stationary demand



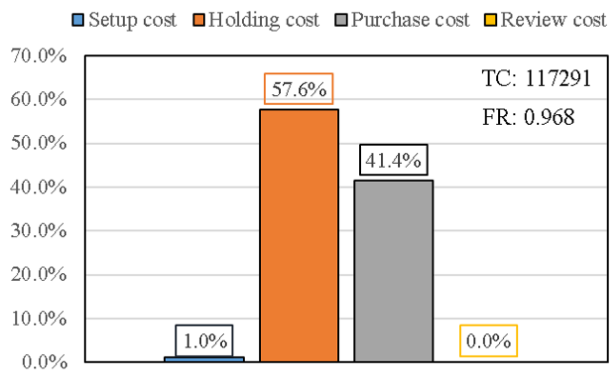
(c) Optimal seasonal policy in seasonal demand



(d) Optimal seasonal policy in stationary demand



(e) Cost profile of seasonal policy in seasonal demand



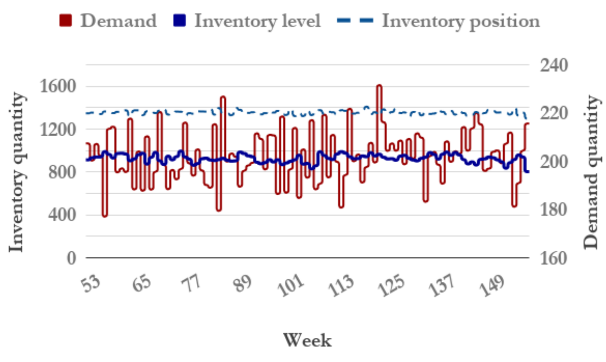
(f) Cost profile of seasonal policy in stationary demand

Figure 38: Average results of seasonal policy for seasonal and stationary demand in scenario 1

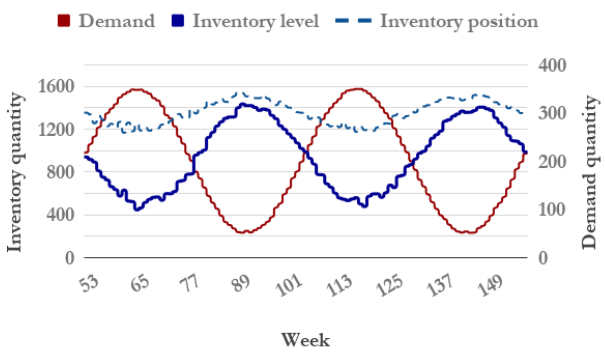
Managing stationary demand with a seasonal replenishment strategy does not affect the expected total cost significantly (TC in Figure 38f), but the expected fill rate denoted by FR declines by 1.2% below the 98% target level. It is natural to observe a lower service level because of the seasonal policy's low reorder quantity during the periods that originally belonged to off-season (e.g., from week 80 to 95). Also, the excess inventory during the weeks that originally saw high demand causes a significant increase in the total holding cost relative to other replenishment costs. Incorrect application of seasonal strategy to stationary demand incurs a 2.92% increase of the expected total cost. This suggests that erroneously using a seasonal policy to handle stationary demand may lead to serious consequences of observing an insufficient service level as well as experiencing a noticeable increase in the expected total cost. We refer the readers to Appendix B.3.1 for the complete table of results for scenario 1 and 10.

Stationary policy for seasonal demand

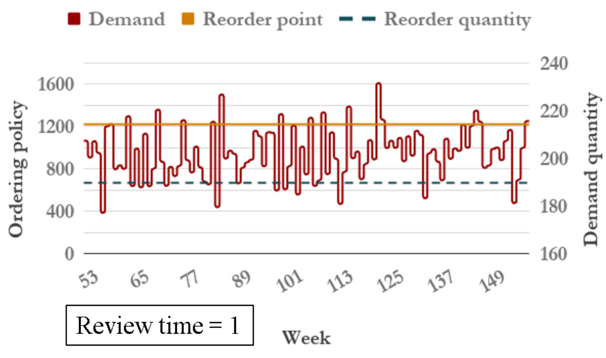
Given the optimal stationary (i.e., single-season) policy determined under stationary demand, experimenting its performance under non-stationary seasonal demand may provide valuable insights for practical applications. To conduct an unbiased experiment, we had to generate a time series of seasonal demand that is analogous to the original stationary demand series. Yet, unlike the stationary series that does not have autocorrelation structure (i.e., correlation between demand in different periods), demand in the seasonal series is naturally autocorrelated. Therefore, we applied the same resampling procedure as before to build a stationary demand series that was originally a seasonal series. Firstly, we generated $S_{\max} = 200$ random instances of seasonal demand, where each instance spans from week 1 until the planning horizon of $N = 5$ years. Secondly, every instance of seasonal demand was randomly re-sampled without replacement to generate 200 unique instances of stationary demand. Finally, the optimal stationary policy was determined based on these stationary demand series using the proposed MA. When evaluating the optimal stationary policy under seasonal demand, we used the original seasonal demand that was used to create the stationary series.



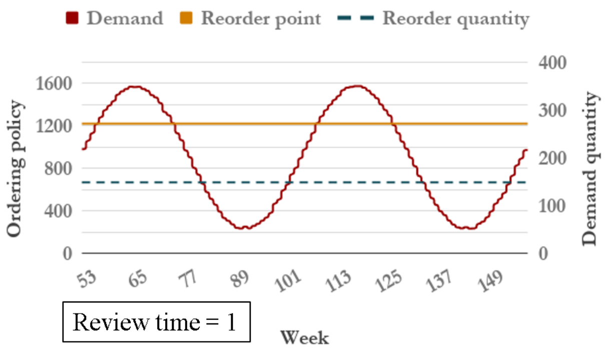
(a) Average inventory situation in stationary demand



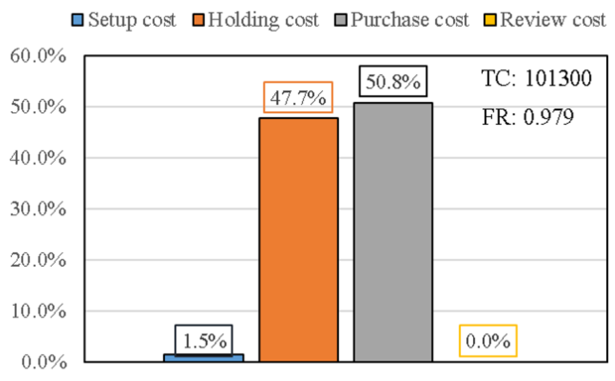
(b) Average inventory situation in seasonal demand



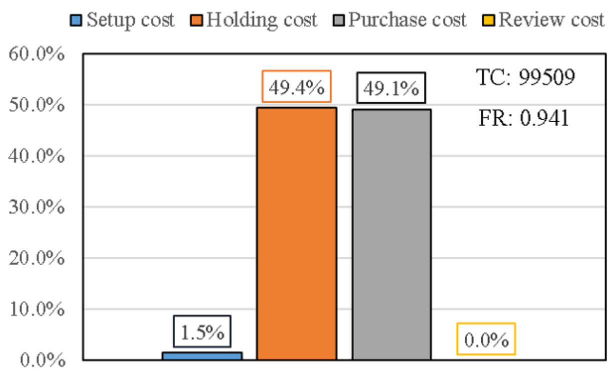
(c) Optimal stationary policy in stationary demand



(d) Optimal stationary policy in seasonal demand



(e) Cost profile of stationary policy in stationary demand



(f) Cost profile of stationary policy in seasonal demand

Figure 39: Average results of stationary policy for seasonal and stationary demand in scenario 1

Figure 39 illustrates the performance of the optimal stationary policy under seasonal demand in scenario 1. As shown in Figure 39b the inventory level oscillates in an opposite direction to the demand series due to constant reorder policy. Since the stationary policy uses a fixed number of safety stock that is determined by averaging the expected demand throughout the seasonal cycle, it is sensitive to unexpectedly high demand during peak season (e.g., low inventory level in week 65 and 117). Accordingly, the average fill rate (FR) stays at 94.1%, which is much below the 98% minimum service level. On the other hand, the expected total cost (TC) under stationary demand is slightly lower than the expected total cost under seasonal demand (by 1.77%), but this is because the policy does not adhere to the minimum service level. Considering the policy's unreasonably high excess stock during off-season in Figure 39b (e.g., excessively high inventory level in week 89 and 141), applying stationary policy to seasonal demand cannot be seen adequate. The complete table of results for scenario 1 and 10 is available in Appendix B.3.1.

Anticipation inventory
Anticipation inventory is an important part of the total inventory when dealing with seasonality in product demand. By maintaining an adequate level of anticipation inventory, one can reduce the risk of shortage during peak seasons and avoid building excess inventory during off-seasons. In the context of our replenishment system, the anticipation inventory can be built by placing new replenishment orders before the actual start of demand shift. Therefore, the key decision variables for optimizing the size of anticipation inventory are the size of ordering parameters R_t and Q_t as well as the timing of seasonal adjustments.

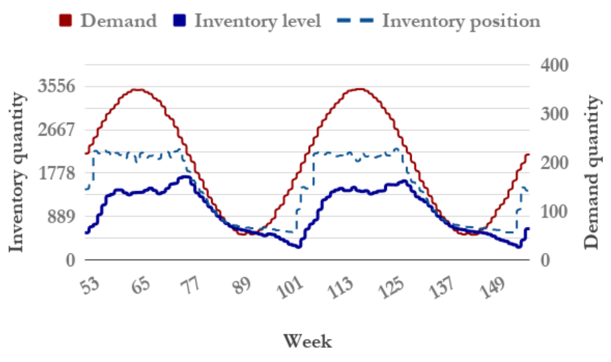
For comparison, we produced a replenishment strategy that does not operate anticipation inventory by optimizing for each season in a piece-wise manner. First, $m = 4$ unique seasons were identified by minimizing the mean squared error (MSE) between the i^{th} season's mean demand $\hat{\mu}_i$ and the average period demand μ_t for time t that belongs to the season. We mention that the seasons are defined based on the mean demand during a seasonal cycle of length Y (i.e., the MSE is computed for μ_t with $t = 1, 2, \ldots, Y$), and not for the planning horizon N because the same seasonal cycle repeats for an integral number of times to form the planning horizon. Then we fitted a discrete Normal distribution to each of these seasons by approximating the mean $\hat{\mu}_i$ and standard deviation $\hat{\sigma}_i$ as follows

$$\hat{\mu}_i = \sum_{t=\omega_i^{\text{start}}}^{\omega_i^{\text{start}}+\omega_i^{\text{length}}-1} \frac{\mu_t}{\omega_i^{\text{length}}} \tag{40}$$

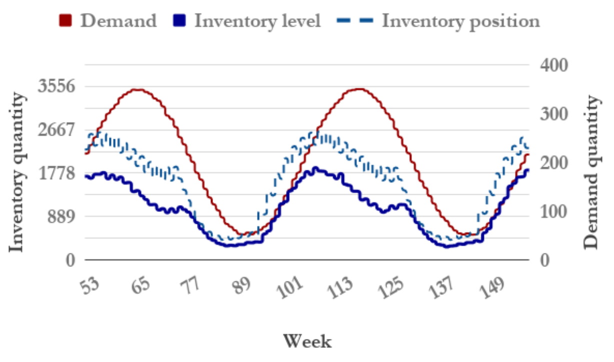
$$\hat{\sigma}_i = \frac{\sigma}{\omega_i^{\text{length}}} \tag{41}$$

where ω_i^{start} and ω_i^{length} are i^{th} season's start time and length defined within a seasonal cycle.

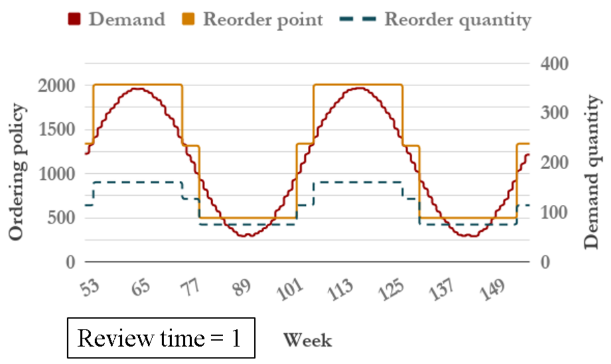
Treating each season separately, we optimized $m = 4$ different problems with respective stationary demand that has mean $\hat{\mu}_i$ and standard deviation $\hat{\sigma}_i$. Figure 40 demonstrates the performance of the optimal piece-wise strategy that does not employ anticipation inventory. We mention that the following piece-wise strategy was determined by setting 99% target service level for each of 4 piece-wise problems instead of the original 98% level. This was to ensure that the resulting piece-wise policy is feasible under 98% target service level. By confirming that both non-anticipatory strategy and the optimal MA solution are feasible at 98% minimum level, we could make a more fair and meaningful comparison.



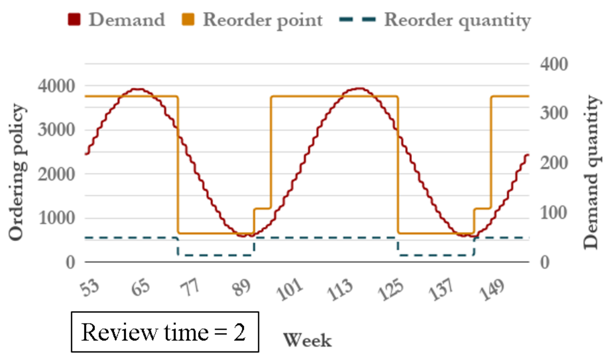
(a) Average inventory situation for piece-wise strategy



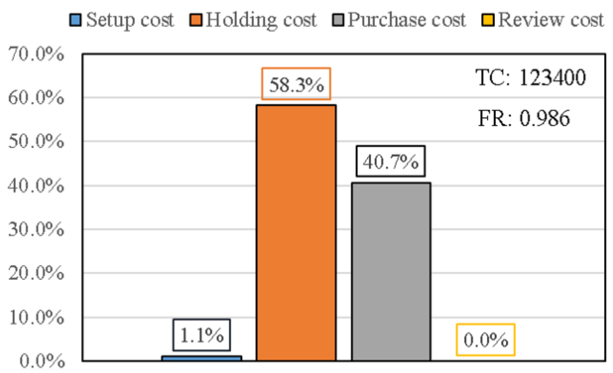
(b) Average inventory situation for optimal MA solution



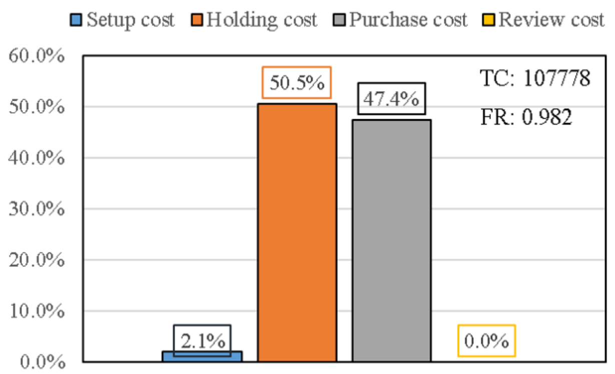
(c) Optimal stationary policy for piece-wise strategy



(d) Optimal stationary policy for optimal MA solution



(e) Cost profile of stationary policy for piece-wise strategy



(f) Cost profile of stationary policy for optimal MA solution

Figure 40: Average simulation results of non-anticipatory piece-wise strategy and optimal MA solution in scenario 1

Notice that the optimal piece-wise strategy illustrated in Figure 40c switches its reorder policy to next season at the exact moment of demand shift. This is an example of a non-anticipatory strategy because its reorder policy in each season does not reflect the expected demand in the subsequent season. If lead time was zero, such strategy could have been effective at handling seasonality without having to build costly anticipation stock. But because of stochastic non-zero lead time the piece-wise strategy bears a high shortage risk when demand starts to pick up. See week 102 in Figure 40a where the inventory level reaches its lowest point due to delayed receipts. Moreover, unlike the optimal MA solution in Figure 40b, which has gradually declining inventory level until the end of peak season, the inventory level for the piece-wise policy stays high until the end of peak season. Consequently, the high leftover stock from the peak season is directly passed onto the off-season and remains unconsumed (week 89 and 145). This has driven up the expected total holding cost substantially, which explains the large 14.49% cost gap between the piece-wise strategy and the optimal MA solution (Figure 40f and 40e). The complete table of results for scenario 1 and 10 is available in Appendix B.3.1.

In conclusion, our analysis suggests that the impact of inaccurate treatment of demand seasonality puts both service level and inventory cost at a high risk. A naive application of seasonal policy to stationary demand or stationary policy to seasonal demand can create a large number of lost sales during periods where high demand is expected, while too much inventory remains redundant during periods of low expected demand. Finally, we observed that the benefit of anticipation inventory is significant at maintaining a sufficient service level from sudden demand shocks while preventing any excess inventory during off-seasons.

5.5.2 Effect of Cost Structure

The problem's cost structure is an important consideration when determining the optimal replenishment strategy for seasonal demand. Each of four cost parameters: order setup cost K, holding cost h, purchase cost p and review cost r, makes different impact on the performance of the optimal solution. If K is too high (low) compared to h, holding more (less) inventory and procuring less (more) may be more profitable for the expected total cost. Also, since unit purchase cost p can be classified as a variable setup cost, its value has a direct impact in determination of the reorder quantity. Lastly, the review cost counteracts the benefit of frequent inventory review and necessitates a careful decision of when to review the inventory position. In this section, we examine the impact of each cost parameter on the optimal solution quality.

To experiment the algorithm's sensitivity to different cost parameters, the MA was executed 10 times under 37 different cost configurations. While holding all the other parameters fixed, each cost parameter was iteratively selected from the following sets:

$$K \in \{10, 20, 30, 50, 100, 200, 300, 400, 500\}$$

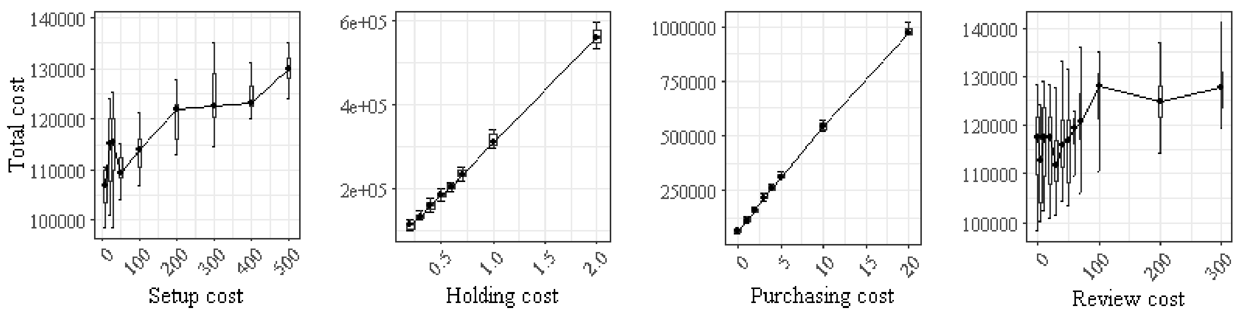
$$h \in \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 1, 2, 3, 4\}$$

$$p \in \{0, 1, 2, 3, 4, 5, 10, 20, 30\}$$

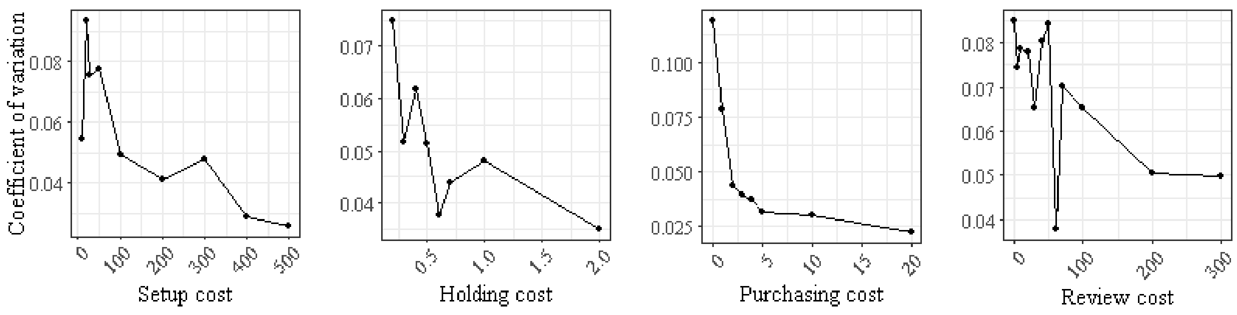
$$r \in \{0, 5, 10, 20, 30, 40, 50, 60, 70, 100, 200, 300\}$$

In order to assess the MA's degree of convergence in each problem instance, we also examined the coefficient of variation, which is the ratio of the standard deviation to the mean. For each problem instance, we calculated the mean and the standard deviation of the expected total costs of 10 optimal MA solutions. A lower coefficient of variation implies that the algorithm was able to converge and produce consistent outcomes, hence attaining a smaller variance among different MA solutions.

Figure 41 shows the effects of different cost parameters on the expected total cost. To help visualize the trend, we connected the median value of each box plot by a straight line. From the diagrams, one can observe that the expected total cost increases approximately linearly with all cost parameters except for the review cost.



(a) Total cost for different cost parameters



(b) Coefficient of variation for different cost parameters

Figure 41: Results for different cost structure

In Figure 41b the coefficients of variation for each cost component generally decrease as per unit value of each cost parameter increases. This is mainly because of the increase in the magnitude of the expected total cost and hence does not necessarily indicate that the MA converges better for higher cost parameters.

Consider the expected holding cost in Figure 41a. Starting from the initial holding cost of $h = 0.2$ per unit per time unit, an increase in h by 0.1 unit increases the optimal expected total cost by 11.86% on average. At first glance, it appears as if the performance of the MA is extremely sensitive to the changes in h for such a dramatic increase of the expected total cost to happen from a minor increase of h. However, one should remember that h is defined in terms of the number of items held per time unit. Considering that the average size of demand per week is set equal to 200 in our experiment, the average size of inventory level per time unit is usually at around 800 units (see Figure 32a). This leads to an average 208,000 units of total inventory level during 5 years. Therefore, the dramatic change in the total holding cost is mainly due to the large magnitude of product units involved in the experiment, and not necessarily because the MA is particularly sensitive to h. Moreover, the coefficient of variation for 10 optimal expected total costs is below 10% for all h, indicating that the MA maintains consistent solution quality for different values of h.

A unit change in the order setup cost K changes the optimal expected total cost by 0.08% on average, which is much smaller than the effect of change in h by 0.1 unit. In addition, the coefficient of variation remains below 10% for all K values, indicating that the MA is robust to K. A similar conclusion can be drawn for the unit purchasing cost p, which increases the expected total cost by 18.78% on average for a unit increase of its value.

Finally, the relationship between the optimal expected total cost and the review cost per review r can be modeled by a piece-wise linear model with a breakpoint at $r = 100$. A unit decrease of r from 100 reduced the expected total cost by an average 0.0839%, and a unit increase from 100 increased the cost at a lower average rate of 0.0119%. An interesting observation is that with stochastic lead time and seasonal demand, zero or almost zero r does not always imply a smaller value for the optimal review time. Figure 42 demonstrates the optimal review times for different values of r. One can observe that the optimal review time becomes longer for higher r, but it is not very clear whether a lower r leads to a shorter review interval. One of the main reasons for this result is to avoid triggering too many replenishment orders during peak seasons where both the reorder point and the reorder quantity are high. In particular, more frequent review together with high reorder point issues new replenishment orders more frequently, resulting in significantly higher setup cost as well as more exposure to lead time volatility.

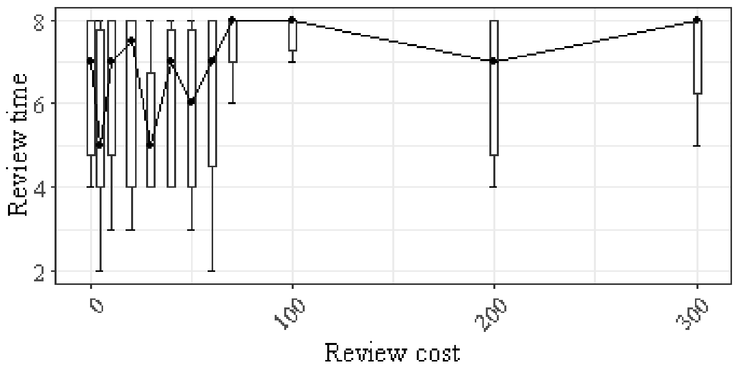


Figure 42: Effect of review cost on the optimal review time in scenario 1

Overall, all four cost parameters led to an approximately linear increase in the optimal expected total cost. Nevertheless, the performance of the MA remained consistent regardless of the changes in the cost parameters. The results for scenario 10 can be found in Appendix B.3.2.

5.5.3 Effect of Target Service Level

So far, we have considered the replenishment system with a 98% target service level. However, different target service levels could create different impacts on the solution quality. In this section, we investigate the impact of different target service levels on the optimal expected total cost by evaluating the optimal solutions under 6 different target service levels. The following set contains the target service levels that we have experimented with

$$FR_{\min} \in \{90\%, 92.5\%, 95\%, 98\%, 99\%, 100\%\}$$

Figure 43 illustrates the effect of minimum target service level on the optimal expected total cost.

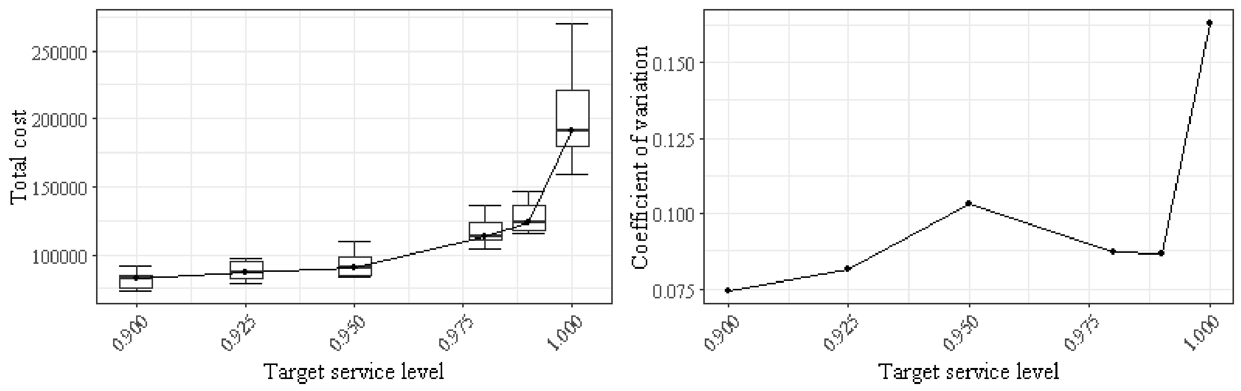


Figure 43: Sensitivity results for target service levels in scenario 1

The graph clearly shows that a higher target service level increases the optimal expected total cost. When service requirement is tight, the optimal replenishment strategy puts much more weight on minimizing the number of lost sales in spite of carrying more inventory. Accordingly, the average holding cost becomes higher and increases the expected total cost. Moreover, an interesting observation is the exponential rate of increase in the optimal expected total cost. Starting from the 99% target level, 1% decrease in the target level decreases the total cost by an average 5.65%, while the increase to 100% target level shoots up the total cost by 58.59%. This result implies that allowing for 1% room for lost sales may be much more cost-effective than enforcing a perfect 100% service level.

At 100% target service level, the coefficient of variation of 10 optimal MA solutions escalates to 20% from the average of 8.7% for lower levels. It is because a very tight constraint can seriously complicate the problem for solving. While the MA was able to converge to a reasonable degree of precision for lower target levels, requiring 100% made the MA quite unstable.

5.5.4 Effect of Planning Horizon

When determining an optimal replenishment strategy, setting an appropriate value for the planning horizon N is crucial. A short-term strategy that achieves the best performance for a short planning horizon may be ineffective for long-term inventory systems because of potential over-fitting error. Conversely, a long-term strategy may jeopardize short-term performance to achieve stable long-term performance. In this experiment, we investigated the significance of different planning horizons on the solution quality. Considering a seasonal cycle of 52 weeks, we defined a set of 5 candidate planning horizons as below. We mention that these lengths exclude the warm-up period of one seasonal cycle (1×52 weeks).

$$N \in \{104 \text{ (2 cycles)}, 156 \text{ (3 cycles)}, 208 \text{ (4 cycles)}, 312 \text{ (6 cycles)}, 468 \text{ (9 cycles)}\}$$

Figure 44 shows the total expected costs of 10 optimal replenishment strategies identified by 10 independent MA executions for each different length of planning horizon.

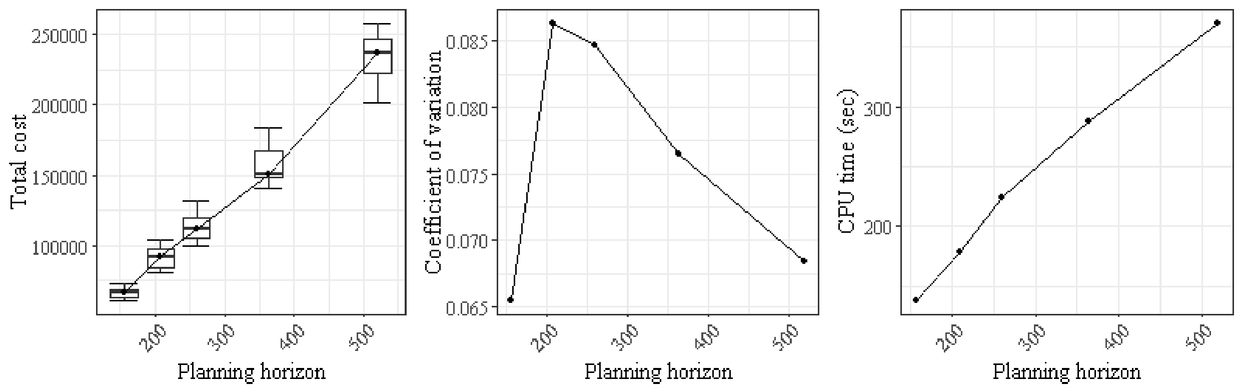


Figure 44: Sensitivity results for planning horizon in scenario 1

As the planning horizon increases by one time unit, the expected total cost increases linearly by 0.457% on average. It is a reasonable outcome considering that an extension of planning horizon also expands the number of periods with positive inventory level, causing the total holding cost to increase proportionally. In addition, the amount of computation time increases linearly as the horizon increases. A unit increase of planning horizon increases the CPU time by 0.63 seconds on average, implying an average computation time of 370.4 seconds (6.17 minutes) for a long-term problem spanning 10 years ahead. This suggests that our algorithm is computationally efficient and stable even for large-scale problems. Finally, the coefficient of variation graph demonstrates that the MA is able to produce consistent solutions to both short-term and long-term problems.

5.5.5 Effect of Number of Seasons

In this experiment, we examine the impact of having different number of seasons in a replenishment strategy m on the optimal cost and the MA's overall performance. The value of m was adjusted for 6 times according to the set

$$m \in \{1, 2, 3, 4, 5, 6\}$$

Figure 45 shows the optimal expected total costs, coefficients of variation and average computation times identified by 10 different MA runs for above-specified number of seasons.

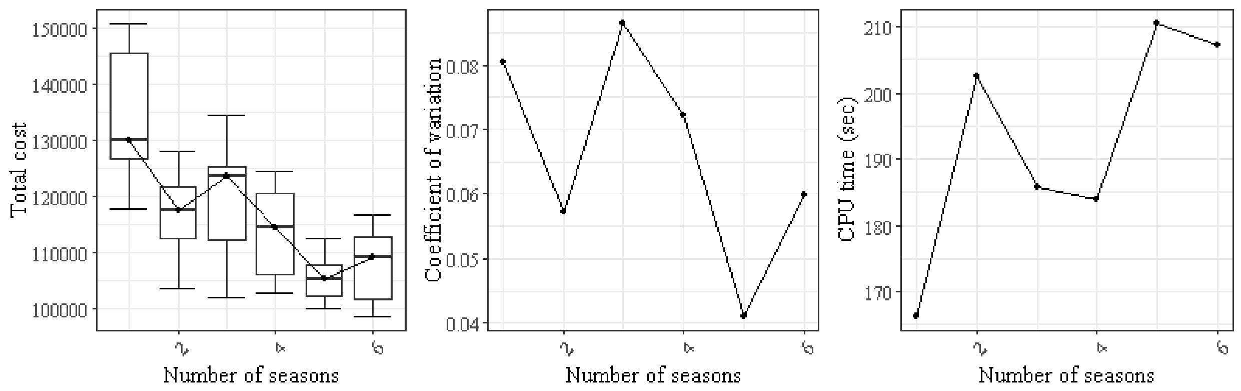


Figure 45: Sensitivity results for number of seasons in scenario 1

According to Figure 45, the reduction of the expected total cost due to increased number of seasons is very evident. Extending m by one more season adds extra flexibility to handle demand non-stationarity, reducing the expected total cost by an average 4.13%. However, the computation time increases by 6.49 seconds on average by adding one more season. Therefore, the average time required for determining a fully dynamic model with $m = Y = 52$ is projected as 508.4 seconds. Lastly, the coefficient of variation for different number of seasons remains below 10% and it approximately decreases as more seasons are considered. This suggests that allowing for more seasons can facilitate more reliable performance and convergence of the MA.

5.6 Algorithm Testing

In our simulation-based optimization algorithm four unique components have been integrated: dynamic resampling (Section 4.3.3), adaptive fitness function (Section 4.4.4), population initialization heuristic (Section 4.4.5) and LSA (Section 4.4.9) for effective management of the problem's stochasticity as well as enhancement of the algorithm's search performance. In order to demonstrate the effectiveness and the contribution of these components to the algorithm's optimization performance, we have conducted several computational experiments to each of these components one at a time. To keep our report compact, in this section we only provide a summary of all experiments' main procedures and their key results instead. The complete results of our tests can be found in Appendix B.4 together with more detailed explanation.

Table 5 and 6 provide the summary of our computational experiments for each of four algorithm components. The column *Index* contains the index of each sub-experiment conducted to individual component while *Experiment description* and *Key results* columns contain the settings and the key takeaways of each sub-experiment. We emphasize that when conducting each of the sub-experiments for each component, all the other operators and components in the algorithm were held fixed identical to the setting described in Section 5.2. Also, all sub-experiments were conducted both in scenario 1 and 10 to examine if the effect of each component is systematically dependent on different demand and cost structure. Finally, the MA was executed 10 times for every sub-experiment in case the sub-experiment requires an execution of the MA, and the remarks in the *Key results* are based on the average results from these 10 MA runs.

Table 5: Summary (1) of experiment settings and key results from the algorithm testing

Component	Index	Experiment description	Key results
Dynamic resampling technique	1	**MA with single-sample estimates:** We have executed the MA based on noisy estimates of the expected total cost and expected fill rate that resulted from a single simulation replication for each solution, instead of performing multiple simulation replications. Then the quality of the optimal solution based on single-sample estimates was verified against its true quality that is based on noiseless estimates.	• Significant overestimation of the solution's true performance: in scenario 1, the true performance of the solution obtained from single-sample estimates is actually 7.15% more expensive and achieves 1.34% lower service level on average • In scenario 10, the problem of overestimation is even more severe, attaining 20.01% of bias compared to the solution's true expected total cost and 1.03% overestimation of the true service level on average • The estimation bias seems to get larger for a higher per unit per time unit holding cost h
	2	**MA with deterministic simulation:** We have investigated the impact of optimizing the replenishment strategy based on deterministic simulation (i.e., deterministic demand and lead time) instead of stochastic simulation. Therefore, demand d_t and lead time L_t in each time period were replaced by their mean values μ_t and $1/q$, and we have evaluated the optimal solution from the MA in comparison to its true performance.	• Significant overestimation of the solution's true performance: in scenario 1, the true performance of the solution obtained from deterministic simulation is actually 13.9% more expensive and achieves 6.74% lower expected service level on average • In scenario 10, the problem of overestimation is even more severe, attaining 15.27% of bias compared to the solution's true expected cost and 6.5% overestimation of the true expected service level on average • The estimation bias in the expected service level seems more substantial than in the expected total cost • Compared to the single-sample estimates, the consequence of applying the deterministic simulation a lot more severe
	3	**Sample size efficiency:** We have estimated the expected total cost and expected fill rate of $10,000$ randomly generated replenishment strategies using the proposed dynamic resampling technique. Then we have evaluated the efficiency of its estimation: we evaluated the extra number of samples required by the proposed technique compared to the ideal sample size, which has been derived from the true value of the expected total cost and expected fill rate.	• Approximately 5 times higher sample size was required than the ideal size • Sample size required by the proposed technique remained consistent in different scenarios

Table 6: Summary (2) of experiment settings and key results from the algorithm testing

Component	Index	Experiment description	Key results
Fitness function	1	**Benefit of adaptive fitness function compared to static fitness function:** The performance of the MA with the adaptive NFT fitness function has been compared to the performance with static fitness function using a static penalty coefficient. The static penalty coefficient was set at a very high level to prevent producing infeasible solutions.	• The comparative experiment brought mixed results. While utilizing the NFT function obtained significant improvement in scenario 10, the benefit in scenario 1 was not significant at 95% confidence level • The MA's convergence rate with the NFT function was substantially higher than with the static fitness function
Population initialization method	1	**Population quality compared to a randomly generated population:** In this experiment the quality of the population generated by our initialization method has been compared to a randomly generated population. The average expected cost and expected fill rate of all individuals in the population have been considered as well as the number of feasible solutions identified in the population.	• Substantial improvement from a pure randomization: average 30% cost improvement in scenario 1 and 39% improvement in scenario 10 • Noticeably more feasible solutions were generated using the proposed method than through pure randomization
LSA	1	**Effect of LSA:** The effectiveness of the LSA has been examined by measuring the improvement in the optimal expected total cost by integrating the LSA into the MA. Furthermore, we have analyzed the impact of applying the LSA to different type and number of individuals in each generation: application to 5, 10, 15 or 20 individuals selected based on their fitness value (best or worst) or selected at random.	• Observed an average 17% cost improvement by integrating the LSA • Applying the LSA to 5 more individuals in each generation brought approximately 5.56% cost reduction • Applying the heuristic to randomly selected individuals instead of the best (fittest) or the worst (least fit) individuals in each generation led to more robust outcomes (i.e., smaller cost variance)
	2	**Relative contribution of each operator in the LSA:** The average percentage contribution of each LSA operator to the fitness improvement has been measured by performing the MA 10 times in scenario 1 and 10. The LSA was applied to 20 randomly selected individuals in each generation.	• The neighborhood operators **CopySeasons** and **ChangeReviewTime** attained the largest improvement of the fitness value • All operators showed positive contribution to the improvement of fitness value

6 Conclusion

6.1 Introduction

In this final section, we conclude our paper with a comprehensive overview of the main findings from our research. In Section 6.2 we reflect on the results from the benchmark experiments, numerical analysis and algorithm testing, and discuss their implications for both theoretical and practical applications. In Section 6.4 all the limitations of our method and analysis are discussed with their potential impacts on the algorithm's performance. Finally in Section 6.5 we draw an overall conclusion and propose directions for further research.

6.2 Discussion

The proposed MA obtained remarkable results in both the benchmark experiment and the numerical analysis, which have important implications for both academics and practitioners. In this section we discuss these implications in detail.

Our research has made several important contributions to the field of inventory optimization with seasonal demand. First of all, our problem is unique in that it considers demand that is both stochastic and seasonal unlike many other problems in the literature that either consider stochastic stationary demand or deterministic seasonal demand. Moreover, our problem defines the optimal solution as a seasonal strategy with a finite number of seasons, which is a generalization of strategies with static and completely dynamic reorder policies.

We have formulated our problem as a stochastic non-linear programming problem and identified the difficulty of solving this problem using conventional stochastic programming algorithms due to extensive recurrence of non-linear constraints. Alternatively, we designed and developed a discrete-event simulation model that could make reliable estimation of the expected total cost and expected fill rate without needing to reformulate the original problem into a deterministic linear problem.

Additionally, our unique Memetic Algorithm has been validated against a benchmark model by Babaï and Dallery (2006) and has shown to achieve superior performance in terms of the solution quality and robustness. The optimal solution identified by the MA incurred significantly lower expected total cost than the optimal solution from the benchmark algorithm. Also, the MA's performance was consistent for all problem instances regardless of cost structure and asymmetry in seasonality. For the best performance, we have designed and incorporated a unique LSA that fully leverages problem characteristics and enhances the overall solution quality. On the other hand, the proposed MA has an important drawback of requiring significantly more computation time. However, considering that typical retail inventories in practice update their replenishment strategies on a weekly or a monthly basis (Bilgen and Günther, 2010; Thomassey, 2014), the MA is more than fast enough for solving the large-scale problems in commercial inventories.

Through numerical analysis, we have observed that overlooking demand seasonality can lead to serious loss in both the steady-state service level and the cost efficiency. Especially when a static reorder policy is applied to manage highly seasonal demand, the inventory can face enormous shortages during peak season and dangerous instability of the overall inventory level. Furthermore, we have found that the operation of anticipation inventory is absolutely crucial for protecting inventory from the sharp increase in demand under stochastic lead time and from having costly overstock during off-season by gradually reducing the inventory level in anticipation. The MA could consistently provide good solutions for different problem parameters including cost structure, planning horizon, target service level and number of seasons. Among these parameters, the effect of the latter two were of particular interest. Firstly, setting a higher target service level caused the optimal cost to rise at an exponential rate and increasingly complicated the convergence of the algorithm. Secondly, applying more seasonal and more dynamic policy was clearly more beneficial to the optimal total cost and the algorithm's convergence.

The performance of the unique operators that we have designed (or adapted from the existing literature) for improvement of our Memetic Algorithm was also evaluated. Through comparative experiments we have observed that the population initialization method and the LSA provided substantial improvements of the solution quality. We have also found that applying the dynamic resampling technique provided much better estimation of the expected total cost and fill rate than applying the single-sample technique or approximating the performance with deterministic simulation. Yet, using the adaptive Near-Feasiblity Threshold (NFT) fitness function instead of a static-penalty fitness function caused only a minor improvement, delivering an average of 4.7% cost saving compared to using the static function. However, the algorithm's convergence was significantly faster and more stable using the NFT function, encouraging its application in many practical situations where the maximum amount of computational time and resource is limited.

6.3 Recommendations

As the results of our experiments suggest, the potential benefit of applying the proposed simulation-based optimization model is significant compared to the existing method in the literature. Besides producing better optimal solutions than the benchmark algorithm in terms of the expected total cost, our approach is noticeably more flexible at handling a variety of inventory systems that cannot be incorporated in the benchmark model. More specifically, our inventory simulation model can be easily modified to simulate new inventory events as well as different demand and lead time distributions. Due to the independence between the simulation and the optimization model, the performance of the operators in the MA is not influenced from the adjustments made to the simulation model. Therefore, the proposed simulation-based optimization algorithm can be easily tailored to various inventory systems that manage different types of products with different demand and lead time characteristics.

We mention that special care should be taken when applying our algorithm in practice because the algorithm makes certain assumptions that might not be suitable for certain inventory systems. For example, the simulation model assumes that in each period at most one replenishment order can be placed. However, if the inventory applies a continuous review replenishment system, multiple reorders may have to be arranged within the same period in order to reach sufficient level of inventory position. In Section 6.4 we elaborate on the algorithm's assumptions and their potential limitations for its practical implementation.

Additionally, despite the algorithm's robust performance, which is observed from fairly small coefficient of variation between multiple MA runs, we recommend executing the MA for a sufficient number of times (e.g., 5 to 10 times) to ensure that no "bad luck" solution is accidentally chosen. Due to the algorithm's reasonably small computation time, this must be viable in most real-life situations. Furthermore, performing multiple executions allow the inventory manager to examine and compare each individual solution, besides the quantitative measures (i.e., expected total cost and service level), in terms of other qualitative aspects of the solution. As a result, the inventory manager is enabled to correctly choose the replenishment strategy that is more suitable for the considered inventory system, which might be slightly more expensive or that might have service level moderately below the target service level.

Finally, it is important that accurate data are entered to the proposed algorithm. Although this may appear obvious, many retail inventories do not keep accurate record of the information that is utilized in the algorithm. More specifically, if inaccurate cost parameters are input into the algorithm, the algorithm may produce an impractical replenishment strategy. Consider an example where the holding cost per unit per time unit is set abnormally high. In this case the algorithm is most likely to produce a Just-In-Time strategy that keeps the inventory level close to zero by avoiding reordering before the customer orders are actually received. However, this type of strategy may be ineffective for products with highly seasonal demand and unstable delivery lead time. Therefore, providing incorrect input data is most likely to make heavy impact on the performance of the algorithm.

6.4 Limitations

The proposed algorithm is capable of determining an efficient replenishment strategy for a product with stochastic seasonal demand. Also, the computational results are remarkable with significant practical and theoretical implications. However, the algorithm bears important limitations that need to be taken into account when interpreting these results. Firstly, our inventory simulation model, which was developed to estimate the expected total cost and the fill rate of a candidate replenishment strategy, makes certain assumptions that may be unrealistic in practice. These assumptions are:

- Exact value of the mean demand quantity for the future is known in advance
- Demand quantity in each time period is discrete Normally distributed with the identical standard deviation
- Demand and reorder policy are discrete valued
- Inventory dynamics follow a discrete time system with discrete lead time
- Inventory has three unique events that happen in the sequence of REVIEW, ARRIVAL and DEMAND.

Obviously, these assumptions may deceive the applicability of the produced solutions in practice because several of these assumptions are simply not achievable in real-life. The first assumption that the mean demand quantity in the future is known exactly in advance is perhaps the most unrealistic condition that no real-life inventory managers have access to. However, most of commercial inventories nowadays operate based on demand forecasts that are generated using extensive volume of historical customer orders as well as real-time customer information. Accordingly, these forecasts can be quite accurate estimates of the true mean of the future demand, suggesting that the first assumption is replicable in practice using the forecast values. Furthermore, the uncertainty in the forecast is typically modeled by a standard probability distribution, such as Normal or Gamma distribution, in order to provide a prediction interval. We highlight that despite having restricted our attention to discrete Normally distributed demand with a constant standard deviation in our research, we can still apply our algorithm to a general class of demand distribution as long as they can be numerically simulated. Thus, one can apply our algorithm with any available forecast distributions.

The third and fourth assumptions of discrete-valued problem parameters may be unsuitable to some real-life situations, although these conditions can largely simplify our simulation model and optimization procedure. Considering the third assumption, we emphasize that except for the industry that deals with continuous units of commodities (e.g., oil, gas and electricity), consumer products are commonly sold in discrete units (Axsäter, 2015). Thus, it is reasonable to assume that both demand and reorder policy will be expressed in discrete units. On the other hand, the fourth assumption of discrete time system is based on the observation that in practice the inventory's key performance indices as well as its transaction data are recorded in discrete time units (e.g., dates, weeks or months) (Axsäter, 2015). The last assumption was imposed as a bare minimum to develop a functional simulation model of a typical inventory replenishment system. In case of adapting the simulation model to a more complex real-life inventory system, it is rather straightforward to make adjustments to the event set due to the model's event-driven structure.

Besides the simulation model, the Memetic Algorithm has been developed under certain assumptions that can pose limitations in practice. The first assumption is that the inventory manages only a single type of item. Many commercial inventories stock and manage thousands of different products, each with different demands, lead time and cost parameters. Hence, applying our algorithm to every single product at a time will be inefficient in practice. Instead, the algorithm can be applied in a group level; the products can be categorized into a number of groups that have similar demands, lead times and cost characteristics and the algorithm can determine the replenishment strategy that achieves the best average performance for each group. Also, we assumed that the inventory has an unlimited storage capacity, which is debatable in many real-life situations. Every physical inventory is bounded to a limited storage space and therefore an unlimited stocking capacity might be illogical. Yet, unlimited capacity situation may as well occur in practice when extra storage space is acquired from a rented warehouse. Hence, all the replenishment orders that exceed the storage capacity of the original warehouse are stocked in the rented warehouse instead (Chung et al., 2009). However, we emphasize that because of our simulation model's flexibility, the capacity restriction can always be incorporated without affecting the operators of the MA.

Finally, although we have attempted to test and analyze our algorithm extensively on all possible problem instances, we had to limit our numerical analysis to scenario 1 and 10 because of the small time frame of our research. Since our algorithm is mainly intended for practical implementation, it would be more desirable to extend the computational study with more problem instances.

6.5 Conclusions and Further Research

The first objective of our research was to develop an effective methodology to optimize replenishment strategy under stochastic seasonal demand. Therefore, we have developed a new simulation-based optimization method that could produce a cost-effective replenishment strategy with periodic review and seasonal reorder policy under demand and lead time uncertainty. Our method was unique in that it utilizes the problem's seasonality characteristics for optimization without violating problem constraints. We also implemented an effective framework for selecting the algorithm's parameters for its robust performance.

Our second objective was to verify the performance and robustness of the proposed algorithm under various problem instances. Based on the extensive benchmark experiments with a model by Babaï and Dallery (2006), we have observed that the optimal solutions from the proposed method achieve an average 16% reduction in the expected total cost and are more robust to demand and cost structure. We further illustrated common features of the optimal replenishment strategy under symmetric and asymmetric demand seasonality, where reorder policies are adjusted earlier in anticipation to the upcoming demand shifts.

The last objective was to analyze the impact of different problem parameters on the quality of the optimal solution. Our numerical analysis demonstrated the significance of applying a seasonal policy to manage seasonal demand for adequate service level. The benefit of utilizing anticipation inventory was clear for protecting service level from demand fluctuations while keeping the inventory level low whenever possible. Also, our algorithm showed robust performance to different cost structure, target service level and planning horizon, which was observed from its stable performance variance regardless of these parameters' values. The results further suggested that under seasonal demand, applying more dynamic replenishment strategy with more seasons could significantly reduce the expected total cost by 15% on average. Finally, we have confirmed that the algorithm's four special components: resampling method, fitness function, population initialization method and LSA achieved an average 17% reduction on the solution's expected total cost.

Our research can be further generalized in several different directions. Firstly, one can apply our method in a multi-item inventory system instead of a single-item inventory system. However, this would require additional analysis for the optimal for replenishment decisions since different combination of products can be ordered jointly to achieve economies of scale through product discount or through synchronized replenishment. Secondly, the algorithm can be implemented to minimize the total supply chain cost in the multi-echelon logistics network, in which the effect of interaction between different inventories in different echelons should be considered. Thirdly, the algorithm's performance in a distribution-free environment can be examined. Our problem assumes that demand and lead time follow parametric discrete distribution, while in reality they do not easily fit into available standard parametric distributions. Indeed, for relatively new or slow-moving products, modeling their demand and lead time with empirical methods may be more effective due to limited historical data or intermittent values of demand and lead time. Lastly, the performance of the proposed algorithm with other types of dynamic reorder policies, such as (T, s_t, S_t) and (T, R_t, nQ_t) can be further investigated.

References

Aardal, K., Jonsson, Ö. and Jönsson, H. (1989), 'Optimal inventory policies with service-level constraints', *Journal of the operational research society* **40**(1), 65–73. Springer, Verlang, Berlin.

Altiparmak, F., Gen, M., Lin, L. and Paksoy, T. (2006), 'A genetic algorithm approach for multi-objective optimization of supply chain networks', *Computers & industrial engineering* **51**(1), 196–215. Elsevier, London, UK.

Andradóttir, S. (2006), 'An overview of simulation optimization via random search', *Handbooks in operations research and management science* **13**, 617–631. Elsevier, London, UK.

Archibald, B. C. (1981), 'Continuous review (s, S) policies with lost sales', *Management Science* **27**(10), 1171–1177. INFORMS, Catonsville, Maryland.

Axsäter, S. (2006), 'A simple procedure for determining order quantities under a fill rate constraint and normally distributed lead-time demand', *European journal of operational research* **174**(1), 480–491. Elsevier, London, UK.

Axsäter, S. (2015), *Inventory control*, Vol. 225, Springer, Verlang, Berlin.

Babai, M. Z. (2005), Politiques de pilotage de flux dans les chaînes logistiques: impact de l'utilisation des prévisions sur la gestion de stocks, PhD thesis, Ecole Centrale Paris, Gif-sur-Yvette, France.

Babaï, M. Z. and Dallery, Y. (2006), 'A dynamic inventory control policy under demand, yield and lead time uncertainties', **2**, 1026–1031.

Babaï, M. Z., Syntetos, A. A., Dallery, Y. and Nikolopoulos, K. (2009), 'Dynamic re-order point inventory control with lead-time uncertainty: analysis and empirical investigation', *International Journal of Production Research* **47**(9), 2461–2483. Taylor & Francis, Abingdon, UK.

Bashyam, S. and Fu, M. C. (1998), 'Optimization of (s, S) inventory systems with random lead times and a service level constraint', *Management Science* **44**(12-part-2), S243–S256. INFORMS, Catonsville, Maryland.

Beasley, D., Bull, D. R. and Martin, R. R. (1993), 'An overview of genetic algorithms: Part 1, fundamentals', *University computing* **15**(2), 56–69.

Benton, W. and Park, S. (1996), 'A classification of literature on determining the lot size under quantity discounts', *European Journal of Operational Research* **92**(2), 219–238. Elsevier, London, UK.

Berman, O. and Kim, E. (1999), 'Stochastic models for inventory management at service facilities', *Stochastic Models* **15**(4), 695–718. Taylor & Francis, Abingdon, UK.

Berretta, R. and Rodrigues, L. F. (2004), 'A memetic algorithm for a multistage capacitated lot-sizing problem', *International Journal of Production Economics* **87**(1), 67–81. Elsevier, London, UK.

Bijvank, M. and Vis, I. F. (2011), 'Lost-sales inventory theory: A review', *European Journal of Operational Research* **215**(1), 1–13. Elsevier, London, UK.

Bijvank, M. and Vis, I. F. (2012), 'Lost-sales inventory systems with a service level criterion', *European Journal of Operational Research* **220**(3), 610–618. Elsevier, London, UK.

Bilgen, B. and Günther, H.-O. (2010), 'Integrated production and distribution planning in the fast moving consumer goods industry: a block planning application', *Or Spectrum* **32**(4), 927–955. Springer, Verlang, Berlin.

Bookbinder, J. H. and Tan, J.-Y. (1988), 'Strategies for the probabilistic lot-sizing problem with service-level constraints', *Management Science* **34**(9), 1096–1108. INFORMS, Catonsville, Maryland.

Branke, J., Schmidt, C. and Schmeck, H. (2001), 'Efficient fitness estimation in noisy environments', pp. 243–250. Morgan Kaufmann Publishers Inc., San Francisco, CA.

Buchanan, D. J. and Love, R. F. (1985), 'A (Q, R) inventory model with lost sales and erlang-distributed lead times', *Naval Research Logistics (NRL)* **32**(4), 605–611. Wiley Online Library, New York, USA.

Chang, Y. S. and Niland, P. (1967), 'A model for measuring stock depletion costs', *Operations Research* **15**(3), 427–447. INFORMS, Catonsville, Maryland.

Cheng, F. and Sethi, S. P. (1999), 'Optimality of state-dependent (s, S) policies in inventory models with markov-modulated demand and lost sales', *Production and operations management* **8**(2), 183–192. Wiley Online Library, New York, USA.

Chung, K.-J., Her, C.-C. and Lin, S.-D. (2009), 'A two-warehouse inventory model with imperfect quality production processes', *Computers & Industrial Engineering* **56**(1), 193–197. Elsevier, London, UK.

Coit, D. W., Smith, A. E. and Tate, D. M. (1996), 'Adaptive penalty methods for genetic optimization of constrained combinatorial problems', *INFORMS Journal on Computing* **8**(2), 173–182. INFORMS, Catonsville, Maryland.

Daniel, J. S. R. and Rajendran, C. (2005), 'A simulation-based genetic algorithm for inventory optimization in a serial supply chain', *International Transactions in Operational Research* **12**(1), 101–127. Wiley Online Library, New York, USA.

Deb, K. (2000), 'An efficient constraint handling method for genetic algorithms', *Computer methods in applied mechanics and engineering* **186**(2-4), 311–338. Elsevier, London, UK.

DeMatteis, J. J. (1968), 'An economic lot-sizing technique, i: The part-period algorithm', *IBM systems Journal* **7**(1), 30–38. IBM, New York, USA.

Di Pietro, A., While, L. and Barone, L. (2004), 'Applying evolutionary algorithms to problems with noisy, time-consuming fitness functions', **2**, 1254–1261. IEEE, Piscataway, NJ.

Diks, E., De Kok, A. and Lagodimos, A. (1996), 'Multi-echelon systems: A service measure perspective', *European Journal of Operational Research* **95**(2), 241–263. Elsevier, London, UK.

Dion, P. A., Hasey, L. M., Dorin, P. C. and Lundin, J. (1991), 'Consequences of inventory stockouts', *Industrial marketing management* **20**(1), 23–27. Elsevier, London, UK.

Ehrhardt, R. (1984), '(s, S) policies for a dynamic inventory model with stochastic lead times', *Operations Research* **32**(1), 121–132. INFORMS, Catonsville, Maryland.

Feng, Y. and Gallego, G. (1995), 'Optimal starting times for end-of-season sales and optimal stopping times for promotional fares', *Management Science* **41**(8), 1371–1391. INFORMS, Catonsville, Maryland.

França, P. M., Mendes, A. and Moscato, P. (2001), 'A memetic algorithm for the total tardiness single machine scheduling problem', *European Journal of Operational Research* **132**(1), 224–242. Elsevier, London, UK.

Fu, M. C. (2002), 'Optimization for simulation: Theory vs. practice', *INFORMS Journal on Computing* **14**(3), 192–215. INFORMS, Catonsville, Maryland.

Gardner Jr, E. S. and Diaz-Saiz, J. (2002), 'Seasonal adjustment of inventory demand series: a case study', *International Journal of Forecasting* **18**(1), 117–123. Elsevier, London, UK.

Glover, F., Kelly, J. P. and Laguna, M. (1999), 'New advances for wedding optimization and simulation', **1**, 255–260.

Gorham, T. (1968), 'Dynamic order quantities', *Production and Inventory Management* **9**(1), 75–81.

Graves, S. C. (1999), 'A single-item inventory model for a nonstationary demand process', *Manufacturing & Service Operations Management* **1**(1), 50–61. INFORMS, Catonsville, Maryland.

Grewal, C. S., Enns, S. T. and Rogers, P. (2015), 'Dynamic reorder point replenishment strategies for a capacitated supply chain with seasonal demand', *Computers & Industrial Engineering* **80**, 97–110. Elsevier, London, UK.

Gruen, T. W., Corsten, D. S. and Bharadwaj, S. (2002), 'Retail out of stocks: A worldwide examination of extent, causes, and consumer responses'.

Hadley, G. and Whitin, T. M. (1963), Analysis of inventory systems, Technical report.

Harris, F. W. (1991), 'How many parts to make at once', *Operations Research* **38**(6), 947–950. INFORMS, Catonsville, Maryland.

Heady, R. B. and Zhu, Z. (1994), 'An improved implementation of the wagner-whitin algorithm', *Production and operations management* **3**(1), 55–63. Wiley Online Library, New York, USA.

Hill, R. M. and Johansen, S. G. (2006), 'Optimal and near-optimal policies for lost sales inventory models with at most one replenishment order outstanding', *European journal of operational research* **169**(1), 111–132. Elsevier, London, UK.

Jalali, H. and Nieuwenhuyse, I. V. (2015), 'Simulation optimization in inventory replenishment: a classification', *IIE Transactions* **47**(11), 1217–1235. Taylor & Francis, Abingdon, UK.

Janecek, A. and Tan, Y. (2011), *Using population based algorithms for initializing nonnegative matrix factorization*, Springer, Verlang, Berlin.

Jha, J. and Shanker, K. (2009), 'Two-echelon supply chain inventory model with controllable lead time and service level constraint', *Computers & Industrial Engineering* **57**(3), 1096–1104. Elsevier, London, UK.

Jin, Y. and Branke, J. (2005), 'Evolutionary optimization in uncertain environments-a survey', *IEEE Transactions on evolutionary computation* **9**(3), 303–317. IEEE, Piscataway, NJ.

Johansen, S. G. and Thorstenson, A. (1993), 'Optimal and approximate (Q, r) inventory policies with lost sales and gamma-distributed lead time', *International Journal of Production Economics* **30**, 179–194. Elsevier, London, UK.

Johansen, S. G. and Thorstenson, A. (1996), 'Optimal (r, Q) inventory policies with poisson demands and lost sales: discounted and undiscounted cases', *International Journal of Production Economics* **46**, 359–371. Elsevier, London, UK.

Kalpakam, S. and Sapna, K. (1994), 'Continuous review (s, S) inventory system with random lifetimes and positive leadtimes', *Operations Research Letters* **16**(2), 115–119. Elsevier, London, UK.

Kaplan, R. S. (1970), 'A dynamic inventory model with stochastic lead times', *Management Science* **16**(7), 491–507. INFORMS, Catonsville, Maryland.

Karlin, S. (1960), 'Dynamic inventory policy with varying stochastic demands', *Management Science* **6**(3), 231–258. INFORMS, Catonsville, Maryland.

Kleijnen, J. P., Van Beers, W. and Van Nieuwenhuyse, I. (2010), 'Constrained optimization in expensive simulation: Novel approach', *European journal of operational research* **202**(1), 164–174. Elsevier, London, UK.

Köchel, P. and Nieländer, U. (2005), 'Simulation-based optimisation of multi-echelon inventory systems', *International Journal of Production Economics* **93**, 505–513. Elsevier, London, UK.

Luke, S. (2009), *Essentials of metaheuristics*, Vol. 113, Raleigh, UK. Lulu.

Mak, K.-L. (1987), 'Determining optimal production-inventory control policies for an inventory system with partial backlogging', *Computers & Operations Research* **14**(4), 299–304. Elsevier, London, UK.

Mandal, S., Maity, A., Maity, K., Mondal, S. and Maiti, M. (2011), 'Multi-item multi-period optimal production problem with variable preparation time in fuzzy stochastic environment', *Applied Mathematical Modelling* **35**(9), 4341–4353. Elsevier, London, UK.

Mele, F. D., Guillen, G., Espuna, A. and Puigjaner, L. (2006), 'A simulation-based optimization framework for parameter optimization of supply-chain networks', *Industrial & Engineering Chemistry Research* **45**(9), 3133–3148. ACS Publications, Washington, DC.

Mendes, Alexandre S and Müller, Felipe M and França, Paulo M and Moscato, Pablo (2002), 'Comparing meta-heuristic approaches for parallel machine scheduling problems', *Production Planning & Control* **13**(2), 143–154. Taylor & Francis, Abingdon, UK.

Metters, R. (1998), ‘General rules for production planning with seasonal demand’, *International Journal of Production Research* **36**(5), 1387–1399. Taylor & Francis, Abingdon, UK.

Metzger, C., Thiesse, F., Gershwin, S. and Fleisch, E. (2013), ‘The impact of false-negative reads on the performance of RFID-based shelf inventory control policies’, *Computers & Operations Research* **40**(7), 1864–1873. Elsevier, London, UK.

Min, H., Ko, H. J. and Ko, C. S. (2006), ‘A genetic algorithm approach to developing the multi-echelon reverse logistics network for product returns’, *Omega* **34**(1), 56–69. Elsevier, London, UK.

Mohebbi, E. and Posner, M. J. (1998), ‘A continuous-review inventory system with lost sales and variable lead time’, *Naval Research Logistics (NRL)* **45**(3), 259–278. Wiley Online Library, New York, USA.

Morton, T. E. (1969), ‘Bounds on the solution of the lagged optimal inventory equation with no demand backlogging and proportional costs’, *SIAM review* **11**(4), 572–596. SIAM, Philadelphia, USA .

Moscato, P. et al. (1989), ‘On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms’, *Caltech concurrent computation program, C3P Report* **826**, 1989.

Mousavi, S. M., Hajipour, V., Niaki, S. T. A. and Alikar, N. (2013), ‘Optimizing multi-item multi-period inventory control system with discounted cash flow and inflation: two calibrated meta-heuristic algorithms’, *Applied Mathematical Modelling* **37**(4), 2241–2256. Elsevier, London, UK.

Nguyen, A.-T., Reiter, S. and Rigo, P. (2014), ‘A review on simulation-based optimization methods applied to building performance analysis’, *Applied Energy* **113**, 1043–1058. Elsevier, London, UK.

Oral, M., Salvador, M. S., Reisman, A. and Dean, B. V. (1972), ‘On the evaluation of shortage costs for inventory control of finished goods’, *Management Science* **18**(6), B–344. INFORMS, Catonsville, Maryland.

Pasandideh, S. H. R., Niaki, S. T. A. and Mousavi, S. M. (2013), ‘Two metaheuristics to solve a multi-item multiperiod inventory control problem under storage constraint and discounts’, *The International Journal of Advanced Manufacturing Technology* **69**(5-8), 1671–1684. Springer, Verlang, Berlin.

Pasandideh, S. H. R., Niaki, S. T. A. and Nia, A. R. (2011), ‘A genetic algorithm for vendor managed inventory control system of multi-product multi-constraint economic order quantity model’, *Expert Systems with Applications* **38**(3), 2708–2716. Elsevier, London, UK.

Pentico, D. W. and Drake, M. J. (2009), ‘The deterministic eoq with partial backordering: a new approach’, *European Journal of Operational Research* **194**(1), 102–113. Elsevier, London, UK.

Phadke, M. S. (1989), ‘Quality engineering using design of experiments’, pp. 31–50. Springer, Verlang, Berlin.

Rahnamayan, S., Tizhoosh, H. R. and Salama, M. M. (2007), 'A novel population initialization method for accelerating evolutionary algorithms', *Computers & Mathematics with Applications* **53**(10), 1605–1614. Elsevier, London, UK.

Ramasesh, R. V., Ord, J. K., Hayya, J. C. and Pan, A. (1991), 'Sole versus dual sourcing in stochastic lead-time (s, Q) inventory models', *Management science* **37**(4), 428–443. INFORMS, Catonsville, Maryland.

Ronen, D. (1983), 'Inventory service levelscomparison of measures', *International Journal of Operations & Production Management* **3**(2), 37–45. MCB UP Ltd, Bingley, UK.

Roy, D. (2003), 'The discrete normal distribution', *Communications in Statistics-theory and Methods* **32**(10), 1871–1883. Taylor & Francis, Abingdon, UK.

Sadeghi, J., Sadeghi, S. and Niaki, S. T. A. (2014), 'Optimizing a hybrid vendor-managed inventory and transportation problem with fuzzy demand: an improved particle swarm optimization algorithm', *Information Sciences* **272**, 126–144. Elsevier, London, UK.

Sana, S. S. (2010), 'Demand influenced by enterprises initiativesa multi-item EOQ model of deteriorating and ameliorating items', *Mathematical and Computer Modelling* **52**(1-2), 284–302. Elsevier, London, UK.

Saracoglu, I., Topaloglu, S. and Keskinturk, T. (2014), 'A genetic algorithm approach for multi-product multi-period continuous review inventory models', *Expert Systems with Applications* **41**(18), 8189–8202. Elsevier, London, UK.

Scarf, H. (1959), 'The optimality of (s, S) policies in the dynamic inventory problem'.

Schneider, H. (1981), 'Effect of service-levels on order-points or order-levels in inventory models', *The International Journal of Production Research* **19**(6), 615–631. Taylor & Francis, Abingdon, UK.

Silva, C. A., Runkler, T. A., Sousa, J. M. and da Costa, J. S. (2003), *Optimization of logistic processes in supply-chains using meta-heuristics*, Springer, Verlang, Berlin.

Silver, E. (1978), 'Inventory control under a probabilistic time-varying, demand pattern', *Aiie Transactions* **10**(4), 371–379. Taylor & Francis, Abingdon, UK.

Silver, E. A. and Meal, H. C. (1973), 'A heuristic for selecting lot size quantities for the case of a deterministic time-varying demand rate and discrete opportunities for replenishment', *Prod. Inventory Manage.* **2**, 64–74. APICS, Washington,DC.

Song, J.-S. (1998), 'On the order fill rate in a multi-item, base-stock inventory system', *Operations Research* **46**(6), 831–845. INFORMS, Catonsville, Maryland.

Song, J.-S., Zhang, H., Hou, Y. and Wang, M. (2010), 'The effect of lead time and demand uncertainties in (r, q) inventory systems', *Operations Research* **58**(1), 68–80. INFORMS, Catonsville, Maryland.

Spall, J. C. (2005), *Introduction to stochastic search and optimization: estimation, simulation, and control*, Vol. 65, John Wiley & Sons, New York, USA.

Taleizadeh, A. A., Niaki, S. T. A., Aryanezhad, M.-B. and Tafti, A. F. (2010), ‘A genetic algorithm to optimize multiproduct multiconstraint inventory control systems with stochastic replenishment intervals and discount’, *The International Journal of Advanced Manufacturing Technology* **51**(1-4), 311–323. Springer, Verlang, Berlin.

Tang, J., Lim, M. H. and Ong, Y. S. (2007), ‘Diversity-adaptive parallel memetic algorithm for solving large scale combinatorial optimization problems’, *Soft Computing* **11**(9), 873–888. Springer, Verlang, Berlin.

Tang, K., Mei, Y. and Yao, X. (2009), ‘Memetic algorithm with extended neighborhood search for capacitated arc routing problems’, *IEEE Transactions on Evolutionary Computation* **13**(5), 1151–1166. IEEE, Piscataway, NJ.

Tarim, S. A. and Kingsman, B. G. (2004), ‘The stochastic dynamic production/inventory lot-sizing problem with service-level constraints’, *International Journal of Production Economics* **88**(1), 105–119. Elsevier, London, UK.

Tempelmeier, H. (2007), ‘On the stochastic uncapacitated dynamic single-item lotsizing problem with service level constraints’, *European Journal of Operational Research* **181**(1), 184–194. Elsevier, London, UK.

Thomassey, S. (2014), Sales forecasting in apparel and fashion industry: A review, *in* ‘Intelligent fashion forecasting systems: Models and applications’, Springer, Verlang, Berlin, pp. 9–27.

Tijms, H. C. (2003), *A first course in stochastic models*, New York, USA. John Wiley and sons.

van Donselaar, K. H. and Broekmeulen, R. A. (2013), ‘Determination of safety stocks in a lost sales inventory system with periodic review, positive lead-time, lot-sizing and a target fill rate’, *International Journal of Production Economics* **143**(2), 440–448. Elsevier, London, UK.

Veinott, Jr, A. F. (1966), ‘On the opimality of (s,S) inventory policies: New conditions and a new proof’, *SIAM Journal on Applied Mathematics* **14**(5), 1067–1083. SIAM, Philadelphia, USA .

Wagner, H. M. and Whitin, T. M. (1958), ‘Dynamic version of the economic lot size model’, *Management science* **5**(1), 89–96. INFORMS, Catonsville, Maryland.

Wang, Q. and Spall, J. C. (2011), *Discrete simultaneous perturbation stochastic approximation on loss function with noisy measurements*, IEEE, Piscataway, NJ.

Yang, W., Chan, F. T. and Kumar, V. (2012), ‘Optimizing replenishment polices using genetic algorithm for single-warehouse multi-retailer system’, *Expert Systems with Applications* **39**(3), 3081–3086. Elsevier, London, UK.

Yeniay, Ö. (2005), ‘Penalty function methods for constrained optimization with genetic algorithms’, *Mathematical and computational Applications* **10**(1), 45–56. Multidisciplinary Digital Publishing Institute, Basel, Switzerland.

Yilmaz, C. (1992), ‘Incremental order quantity for the case of very lumpy demand’, *International Journal of Production Economics* **26**(1-3), 367–371. Elsevier, London, UK.

Zangwill, W. I. (1966), ‘A deterministic multi-period production scheduling model with backlogging’, *Management Science* **13**(1), 105–119. INFORMS, Catonsville, Maryland.

Zipkin, P. (1986), ‘Stochastic leadtimes in continuous-time inventory models’, *Naval research logistics quarterly* **33**(4), 763–774. Office of Naval Research, Arlington, VA.

Zipkin, P. (2008*a*), ‘Old and new methods for lost-sales inventory systems’, *Operations Research* **56**(5), 1256–1263. INFORMS, Catonsville, Maryland.

Zipkin, P. (2008*b*), ‘On the structure of lost-sales inventory models’, *Operations research* **56**(4), 937–944. INFORMS, Catonsville, Maryland.

Appendix A Supplementary Materials

A.1 Axäter's Expected Fill Rate

Axsäter (2015) formulated the expected negative inventory levels in terms of integral loss function $H(x)$

$$H(x) = \int_x^\infty G(v)dv = \frac{1}{2}[(x^2+1)(1-\tilde{\Phi}(x)) - x\tilde{\varphi}(x)] \tag{42}$$

The expected lost sales right after a possible delivery is

$$E(IL_i')^- = \int_{-\infty}^0 F(x)dx = \frac{(s_i')^2}{\hat{Q}_i}\Big[H\Big(\frac{\hat{R}_i - \mu_i'}{s_i'}\Big) - H\Big(\frac{\hat{R}_i + \hat{Q}_i - \mu_i'}{s_i'}\Big)\Big] \tag{43}$$

Equivalently, the expected lost sales before the next possible delivery is

$$E(IL_i'')^- = \int_{-\infty}^0 F(x)dx = \frac{(s_i'')^2}{\hat{Q}_i}\Big[H\Big(\frac{\hat{R}_i - \mu_i''}{s_i''}\Big) - H\Big(\frac{\hat{R}_i + \hat{Q}_i - \mu_i''}{s_i''}\Big)\Big] \tag{44}$$

A.2 Discrete Normal Distribution

Roy (2003) presented a discrete Normal distribution that follows a continuous Normal distribution $\mathcal{N}(\mu, \sigma)$ where μ and σ are its mean and standard deviation respectively. The probability mass function of the discrete Normal distribution is

$$P(Y = k) = \Phi\bigg(\frac{k+1-\mu}{\sigma}\bigg) - \Phi\bigg(\frac{k-\mu}{\sigma}\bigg), k = 0, \pm1, \pm2, \ldots; \sigma > 0; -\infty < \mu < +\infty$$

where $\Phi(x)$ is the cumulative distribution function of the continuous standard Normal distribution. If $\mu = 0$ and $\sigma = 1$, then the discrete version of the standard Normal distribution can be obtained as

$$\tilde{\varphi}(k) = \Phi(k+1) - \Phi(k), k = 0, \pm1, \pm2, \ldots; \sigma = 1; \mu = 0$$

The cumulative distribution of the discrete Normal distribution is obtained as

$$P(Y \le k) = \sum_{\hat{k} \le k} P(Y = \hat{k})$$

The cumulative distribution function of the standard Normal mass function is denoted as $\tilde{\Phi}(k)$.

We chose for this definition of discrete Normal distribution by Roy (2003) as it attains the same survival function (and therefore the same cumulative probability distribution) as the continuous analogue.

Figure 46 shows an example of the presented discrete Normal distribution

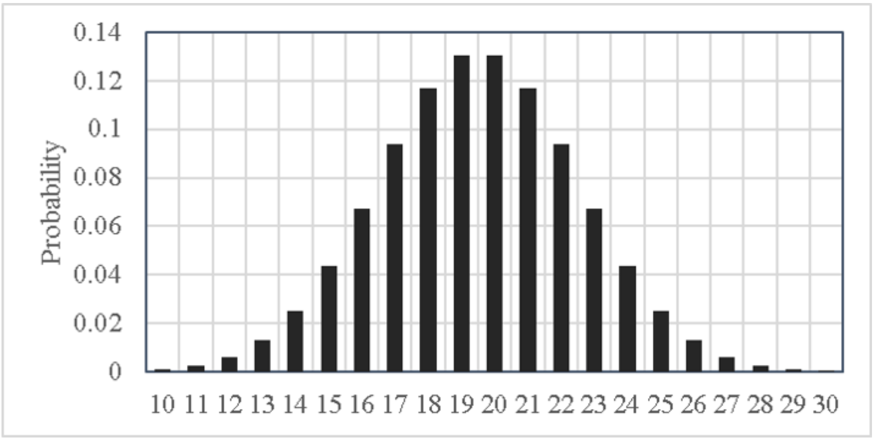


Figure 46: An example discrete normal distribution

A.3 Discrete-Event Simulation Algorithm

The DES model for inventory replenishment system consists of three events and a list of state and counter variables. In this section, we introduce the relevant notation and the simulation algorithm.

Events

- REVIEW: reviewing inventory position
- ARRIVAL: replenishment arrival
- DEMAND: serving customer demand

Event list

- t_A: time for the next order arrival
- t_D: time for the next demand observation
- t_R: time for the next inventory review
- t_A^{list}: list of arrival times of scheduled orders
- t_D^{list}: list of times with positive demand
- t_R^{list}: list of scheduled review times

Variables

- $\tilde{t}$: simulation clock
- $\tilde{t}'$: time of the last event update
- IL_t: inventory level at the end of time t
- IP_t: inventory position at the end of time t
- LS_t: lost sales at the end of time t
- $A(t)$: scheduled order arrival at time t
- b_O: total number of replenishment orders placed
- b_D: total number of demands observed
- b_A: total number of replenishment orders arrived

- b_R: total number of reviews
- HC: total holding cost
- SC: total order setup cost
- PC: total purchase cost
- RC: total review cost
- TC: total inventory cost

Parameters

- $\tilde{S}$: total number of simulation runs
- $\tilde{N}_w$: number of warm-up periods ($< N$)
- $\mathcal{P} = \{(T, R_t, Q_t)|t = 1, \ldots, N)\}$: replenishment strategy

In order to reach the steady-state conditions, a finite length of warm-up periods $\tilde{N}_w$ are required. This is especially important if the starting conditions were set arbitrarily and can lead to under(over) estimation of system performance.

Discrete Event Simulation Algorithm

```
(0) Initialize
  (0.1) Set starting values for simulation clock and last event time:
        $\tilde{t} = \tilde{t}' = 0$
  (0.2) Set initial values for inventory situation:
        $IL_t = IP_t = LS_t = A(t) = 0$ for $t = 1, 2, \ldots, N$
  (0.3) Initialize state variables:
        $TC = HC = SC = PC = RC = b_A = b_D = b_R = b_O = 0$
  (0.4) Set simulation warm-up time and total planning horizon such that $Y \leq \tilde{N}_w < N$, and set the total number of simulation runs $\tilde{S} \geq 1$
  (0.5) Sample demand $d_t$ from its probability distribution for $t = 1, 2, \ldots, N$
  (0.6) Initialize lists of time variables:
    (0.6.1) $t_D^{list}$ = list of times $t \in \{1, 2, \ldots, N\}$ with positive demand $d_t > 0$. Sort $t_D^{list}$ in an increasing order of demand time
    (0.6.2) $t_R^{list} = \{T, 2T \ldots, \lfloor \frac{N}{T} \rfloor T\}$. Sort $t_R^{list}$ in an increasing order of review time
    (0.6.3) $t_A^{list} = \emptyset$
    (0.6.4) $t_D$ = first entry in list $t_D^{list}$, $t_R$ = first entry in list $t_R^{list}$, $t_A = \infty$
(1) REVIEW
    Event condition: If $t_R \leq \min\{t_A, t_D\}$ and $t_R \leq N$
  (1.1) $\tilde{t} = t_R$
  (1.2) If $IP_{\tilde{t}} \leq R_{\tilde{t}}$
    (1.2.1) Schedule a new ARRIVAL event:
      (1.2.1.1) Sample $L_{\tilde{t}}$ from the Geometric distribution
      (1.2.1.2) Prevent order crossing: If $\tilde{t} + L_{\tilde{t}}$ is smaller than any arrival times in $t_A^{list}$, re-sample $L_{\tilde{t}}$ by returning to Step (1.2.1.1)
      (1.2.1.3) Add $\tilde{t} + L_{\tilde{t}}$ at the end of the list $t_A^{list}$ and set $t_A$ = first entry in $t_A^{list}$
      (1.2.1.4) If $\tilde{t} + L_{\tilde{t}} \leq N$: $A(\tilde{t} + L_{\tilde{t}}) = A(\tilde{t} + L_{\tilde{t}}) + Q_{\tilde{t}}$
    (1.2.2) $IP_{\tilde{t}} = IP_{\tilde{t}} + Q_{\tilde{t}}$
    (1.2.3) Warm-up periods: If $\tilde{t} > \tilde{N}_w$
      (1.2.3.1) $b_O = b_O + 1$
      (1.2.3.2) $SC = SC + K$
      (1.2.3.4) $PC = PC + pQ_{\tilde{t}}$
  (1.3) Warm-up periods: If $\tilde{t} > \tilde{N}_w$
    (1.3.1) $b_R = b_R + 1$
    (1.3.2) $HC = HC + (\tilde{t} - \tilde{t}')hIL_{\tilde{t}}$
    (1.3.2) $RC = RC + r$
```

```
(1.4) $\tilde{t}' = \tilde{t}$
(1.5) Update $t_R$ and $t_R^{list}$:
    (1.5.1) Delete the first entry (which equals to $t_R$) in $t_R^{list}$
    (1.5.2) If $t_R^{list} \neq \emptyset$: Set $t_R$ equal to the new first entry in $t_R^{list}$.
            If $t_R^{list} = \emptyset$: Set $t_R = \infty$
(2) ARRIVAL
    Event condition: If $t_A \leq \min\{t_D, t_R\}$ and $t_A \leq N$:
    (2.1) $\tilde{t} = t_A$
    (2.2) $IL_{\tilde{t}} = IL_{\tilde{t}'} + A(\tilde{t})$
    (2.3) Warm-up periods: If $\tilde{t} > \tilde{N}_w$:
        (2.3.1) $HC = HC + (\tilde{t} - \tilde{t}')hIL_{\tilde{t}}$
        (2.3.2) $b_A = b_A + 1$
    (2.4) $\tilde{t}' = \tilde{t}$
    (2.5) Update $t_A$ and $t_A^{list}$:
        (2.5.1) Delete the first entry (which equals to $t_A$) in $t_A^{list}$
        (2.5.2) If $t_A^{list} \neq \emptyset$ and the new first entry in $t_A^{list} \leq N$: Set $t_A =$ the new first
                entry in $t_A^{list}$
        (2.5.3) Else: Set $t_A = \infty$
(3) DEMAND
    Event condition: If $t_D \leq \min\{t_A, t_R\}$ and $t_D \leq N$:
    (3.1) $\tilde{t} = t_D$
    (3.2) $LS_{\tilde{t}} = \max\{d_{\tilde{t}} - IL_{\tilde{t}'}, 0\}$
    (3.2) $IL_{\tilde{t}} = \max\{IL_{\tilde{t}'} - d_t, 0\}$
    (3.3) $IP_{\tilde{t}} = IP_{\tilde{t}'} + LS_{\tilde{t}}$
    (3.4) Warm-up periods: If $\tilde{t} > \tilde{N}_w$
        (3.4.1) $HC = HC + (\tilde{t} - \tilde{t}')hIL_{\tilde{t}}$
        (3.4.2) $b_D = b_D + d_{\tilde{t}}$
    (3.5) $\tilde{t}' = \tilde{t}$
    (3.6) Update $t_D$ and $t_D^{list}$:
        (3.6.1) Delete the first entry (which equals to $t_D$) in $t_D^{list}$
        (3.6.2) If $t_D^{list} \neq \emptyset$: Set $t_D =$ the new first entry in $t_D^{list}$
        (3.6.3) Else: Set $t_D = \infty$
```

(4) Repeat Steps (1)-(3) until $\min\{t_A, t_D, t_R\} > N$

(5) Save simulation results for $\tilde{t} = \tilde{N_w} + 1, \tilde{N_w} + 2, \dots, N$

(5.1) $FR(\mathcal{P}) = 1 - \frac{\sum_{t=\tilde{N_w}+1}^{N} LS_t}{b_D}$

(5.1) $TC = HC + SC + PC + RC$

(6) Repeat Steps (0)-(5) for $\tilde{S}$ number of times and return the average value of output variables over $\tilde{S}$ replications:
$\overline{TC}_{\tilde{S}}, \overline{FR}_{\tilde{S}}, \overline{SC}_{\tilde{S}}, \overline{HC}_{\tilde{S}}, \overline{RC}_{\tilde{S}}, \overline{PC}_{\tilde{S}}, \overline{b_O}_{\tilde{S}}, \overline{b_D}_{\tilde{S}}, \overline{b_A}_{\tilde{S}}, \overline{b_R}_{\tilde{S}}$

Appendix B Computational Results

B.1 Taguchi Experiment

B.1.1 Orthogonal Array

We use the standard L_{18} orthogonal array for the algorithm with a parameter of 2 levels and 7 parameters of 3 levels, which is shown by Table 7. The first column lists the level for the 2-level parameter for each experiment, and the next seven columns assign a unique level for the 3-level parameters to each experiment.

Table 7: L_{18} orthogonal array for a parameter with 2 levels and 7 parameters with 3 levels

Experiment Number	Parameter							
	1	2	3	4	5	6	7	8
1	1	1	1	1	1	1	1	1
2	1	1	2	2	2	2	2	2
3	1	1	3	3	3	3	3	3
4	1	2	1	1	2	2	3	3
5	1	2	2	2	3	3	1	1
6	1	2	3	3	1	1	2	2
7	1	3	1	2	1	3	2	3
8	1	3	2	3	2	1	3	1
9	1	3	3	1	3	2	1	2
10	2	1	1	3	3	2	2	1
11	2	1	2	1	1	3	3	2
12	2	1	3	2	2	1	1	3
13	2	2	1	2	3	1	3	2
14	2	2	2	3	1	2	1	3
15	2	2	3	1	2	3	2	1
16	2	3	1	3	2	3	1	2
17	2	3	2	1	3	1	2	3
18	2	3	3	2	1	2	3	1

For the given set of GA parameters and their three levels as in Table 1, we translated the orthogonal array in Table 7 using the level definitions. The first column and the last two columns are removed from the orthogonal array because our experiment only has 5 parameters, each with 3 levels of value.

Table 8: L_{18} orthogonal array with level definitions given in Table 1

Experiment Number	Maximum generation (M)	Population size (P)	Parameter Tournament probability (λ_t)	Crossover rate (λ_c)	Mutation rate (λ_m)
1	100	20	0.5	0.25	0.05
2	100	50	0.7	0.5	0.1
3	100	100	1	0.7	0.3
4	200	20	0.5	0.5	0.1
5	200	50	0.7	0.7	0.3
6	200	100	1	0.25	0.05
7	400	20	0.7	0.25	0.3
8	400	50	1	0.5	0.05
9	400	100	0.5	0.75	0.1
10	100	20	1	0.75	0.1
11	100	50	0.5	0.25	0.3
12	100	100	0.7	0.5	0.05
13	200	20	0.7	0.75	0.05
14	200	50	1	0.25	0.1
15	200	100	0.5	0.5	0.3
16	400	20	1	0.5	0.3
17	400	50	0.5	0.75	0.05
18	400	100	0.7	0.25	0.1

B.1.2 Experiment Setting

Experiment parameters for Taguchi experiments are shown in Table 9.

Table 9: Experiment parameters for Taguchi method

Inventory plan			
Planning horizon (N)	5 years	Time unit	Weeks
Seasonal cycle length (Y)	52	Target service level ($FR_{\min}$)	98%
Maximum review time ($T_{\max}$)	8		
Demand characteristics			
Number of seasons (m)	4	Mean demand level (D)	200
Demand standard deviation (σ)	20	Seasonality amplitude 1 (E_1)	150
Seasonality amplitude 2 (E_2)	0	Seasonality lag 1 (v_1)	0
Seasonality lag 2 (v_2)	0		
Lead time and cost			
Lead time rate (q)	0.5	Set up cost per order (K)	20
Holding cost per unit per time (h)	0.2	Review cost per review (r)	0
Purchase cost per unit (p)	1		

B.1.3 Signal-to-Noise Results

The average S/N ratios for each of their three levels from the five GA parameters are presented in Table 10. The reported values are the average S/N ratios from 10 independent GA executions (i.e., MA without the LSA) under the experiment setting described in Appendix B.1.2.

Table 10: Average S/N ratios for all parameters, where larger value is better

Parameters	Maximum generation (M)	Population size (P)	Tournament probability (λ_t)	Crossover rate (λ_c)	Mutation rate (λ_m)
Level 1	-266.06	-264.59	-266.12	-263.48	-263.01
Level 2	-264.33	-264.01	-264.40	-265.24	-263.65
Level 3	-262.63	-264.43	-262.51	-264.30	-266.36

B.2 Benchmark Experiment

B.2.1 Experiment Scenarios

Table 11 contains 18 different experiment scenarios used for our benchmark experiments. 18 unique combinations of setup cost K, holding cost h and symmetry coefficient E_2 are considered.

Table 11: Complete list of experiment scenarios

Scenario	Setup cost (K)	Holding cost (h)	Symmetry coefficient (E_2)
1	20	0.2	0
2	20	0.2	1
3	20	0.4	0
4	20	0.4	1
5	20	0.6	0
6	20	0.6	1
7	50	0.2	0
8	50	0.2	1
9	50	0.4	0
10	50	0.4	1
11	50	0.6	0
12	50	0.6	1
13	80	0.2	0
14	80	0.2	1
15	80	0.4	0
16	80	0.4	1
17	80	0.6	0
18	80	0.6	1

B.2.2 Benchmark Results

Table 12 and 13 contain the expected total cost and fill rate of the optimal MA and the BD solutions for all 18 different problem scenarios. In *Method* column, we report which method has been used. In *Type* column, the type of result contained in the corresponding row is stated. The *Average* type row contains the arithmetic mean of 10 optimal solutions' performances. Meanwhile each of the rows with type *Median*, *Best* and *Worst* contains the performance of a solution that attained the median, minimum and maximum total cost out of 10 optimal solutions. Since the BD method has been executed only once while the MA has been executed 10 times, *Type* for the BD method is fixed as *Average*. In *Scenario* column, the problem scenario is reported. *TC* and *FR* column respectively contains the expected total cost and the fill rate of associated optimal solution. The columns *TC Gap* and *FR Gap* contain the percentage difference between the results of the MA executions and of the BD method. Negative-valued percentage gap indicates that the MA solution attains a lower value than the BD solution. Finally, *CPU time* contains the average computational time required for optimizing with the considered method under the corresponding scenario.

Table 12: Complete results obtained by the MA and the BD method (1)

Method	Type	Scenario	TC	FR	TC Gap	FR Gap	CPU time (sec)
BD	Average	1	127,222	0.9871	-	-	
MA	Average		108,190	0.9815	-14.96%	-0.5673%	282.3
	Median		108,882	0.9858	-14.42%	-0.1317%	
	Best		104,349	0.9805	-17.98%	-0.6686%	
	Worst		115,003	0.9804	-9.60%	-0.6788%	
BD	Average	2	134,968	0.9766	-	-	
MA	Average		107,441	0.9823	-20.40%	0.5837%	228.8
	Median		107,555	0.9809	-20.31%	0.4403%	
	Best		99,195	0.9805	-26.50%	0.3993%	
	Worst		114,606	0.9825	-15.09%	0.6041%	
BD	Average	3	185,097	0.9760	-	-	
MA	Average		161,293	0.9831	-12.86%	0.7275%	279.3
	Median		162,454	0.9830	-12.23%	0.7172%	
	Best		152,566	0.9806	-17.58%	0.4713%	
	Worst		168,244	0.9846	-9.10%	0.8811%	
BD	Average	4	209,419	0.9846	-	-	
MA	Average		158,928	0.9831	-24.11 %	-0.1523%	285.6
	Median		154,752	0.9814	-26.10%	-0.3250%	
	Best		146,685	0.9886	-29.96%	0.4063%	
	Worst		177,213	0.9810	-15.38%	-0.3656%	
BD	Average	5	295,127	0.9875	-	-	
MA	Average		201,746	0.9818	-31.64%	-0.5772%	276.8
	Median		202,671	0.9812	-31.33%	-0.6380%	
	Best		196,070	0.9824	-33.56%	-0.5165%	
	Worst		209,755	0.9807	-28.93%	-0.6886%	
BD	Average	6	300,920	0.9785	-	-	
MA	Average		199,866	0.9819	-33.58%	0.3475%	264.8
	Median		200,755	0.9867	-33.29%	0.8380%	
	Best		189,694	0.9803	-36.96%	0.1840%	
	Worst		207,085	0.9827	-31.18%	0.4292%	
BD	Average	7	130,332	0.9738	-	-	
MA	Average		111,808	0.9835	-14.21%	0.9961%	275.7
	Median		111,599	0.9834	-14.37%	0.9858%	
	Best		102,757	0.9811	-21.16%	0.7496%	
	Worst		117,678	0.9814	-9.71%	0.7804%	
BD	Average	8	118,518	0.9763	-	-	
MA	Average		109,838	0.9824	-7.32%	0.6248%	242.4
	Median		109,666	0.9801	-7.47%	0.3892%	
	Best		103,660	0.9822	-12.54%	0.6043%	
	Worst		116,169	0.9834	-1.98%	0.7272%	
BD	Average	9	205,509	0.9842	-	-	
MA	Average		174,179	0.9823	-15.25%	-0.1931%	267.2
	Median		146,458	0.9870	-14.14%	0.2845%	
	Best		154,200	0.9825	-24.97%	-0.1727%	
	Worst		189,359	0.9801	-7.86%	-0.4166%	

Table 13: Complete results obtained by the MA and the BD method (2)

Method	Type	Scenario	TC	FR	TC Gap	FR Gap	CPU time (sec)
BD	Average	10	226,370	0.9805	-	-	
MA	Average		161,616	0.9822	-28.61%	0.1734%	287.1
	Median		159,510	0.9832	-29.54%	0.2754%	
	Best		150,111	0.9820	-33.69%	0.1530%	
	Worst		180,614	0.9828	-20.21%	0.2346%	
BD	Average	11	262,774	0.9751	-	-	
MA	Average		214,681	0.9827	-18.30%	0.7794%	269.8
	Median		210,176	0.9900	-20.02%	1.5280%	
	Best		189,140	0.9817	-28.02%	0.6769%	
	Worst		237,365	0.9821	-9.67%	0.7179%	
BD	Average	12	276,785	0.9737	-	-	
MA	Average		210,183	0.9819	-24.06%	0.8421%	266.6
	Median		211,090	0.9802	-23.74%	0.6676%	
	Best		199,503	0.9800	-27.92%	0.6470%	
	Worst		225,338	0.9801	-18.59%	0.6873%	
BD	Average	13	121,711	0.9801	-	-	
MA	Average		111,242	0.9846	-8.60%	0.4891%	249.9
	Median		111,182	0.9850	-8.65%	0.4999%	
	Best		109,600	0.9878	-9.95%	0.7856%	
	Worst		112,702	0.9810	-7.40%	0.0918%	
BD	Average	14	117,112	0.9863	-	-	
MA	Average		110,620	0.9830	-5.54%	-0.3346%	258.6
	Median		107,905	0.9824	-7.86%	-0.3954%	
	Best		105,685	0.9838	-9.76%	-0.2535%	
	Worst		118,882	0.9808	1.51%	-0.5576%	
BD	Average	15	171,283	0.9716	-	-	
MA	Average		173,632	0.9819	1.37%	1.0601%	275.3
	Median		176,029	0.9819	2.77%	1.0601%	
	Best		158,982	0.9862	-7.18%	1.5027%	
	Worst		182,319	0.9803	6.44%	0.8954%	
BD	Average	16	210,286	0.9864	-	-	
MA	Average		166,390	0.9833	-20.87%	-0.3143%	289.0
	Median		170,219	0.9813	-19.05%	-0.5170%	
	Best		149,026	0.9807	-29.13%	-0.5779%	
	Worst		179,880	0.9805	-14.46%	-0.5981%	
BD	Average	17	291,033	0.9826	-	-	
MA	Average		233,382	0.9833	-19.81%	-0.3143%	258.4
	Median		240,318	0.9813	-17.43%	-0.5170%	
	Best		201,413	0.9807	-30.79%	-0.5779%	
	Worst		265,405	0.9805	-8.81%	-0.5981%	
BD	Average	18	239,824	0.9772	-	-	
MA	Average		216,761	0.9822	-9.62%	0.5117%	272.4
	Median		216,326	0.9830	-9.80%	0.5935%	
	Best		203,917	0.9811	-14.97%	0.3991%	
	Worst		231,129	0.9822	-3.63%	0.5117%	

In Table 14 the optimal solutions found by the MA with $m = 4$ for all 18 different scenarios are presented. The *Review Time* column contains the optimal review time T of the associated optimal solution. For each season $i = 1, 2, 3, 4$ we report the season's replenishment strategy in brackets: $[\hat{R}_i;\ \hat{Q}_i;\ \omega_i^{\text{start}};\ \omega_i^{\text{end}}]$. $\hat{R}_i$ and $\hat{Q}_i$ are the season i's reorder point and reorder quantity. ω_i^{start} and ω_i^{end} are the season's start time and end time, defined within the seasonal cycle of length $Y = 52$. Therefore, $\omega_i^{\text{start}} \in [1, 52]$ and $\omega_i^{\text{end}} \in [1, 104]$ where $\omega_i^{\text{end}} \geq \omega_i^{\text{start}}$. We remark that the upper bound for ω_i^{end} is 104 and not 52 because if season i's end time ω_i^{end} exceeds the cycle length, then the season continues to the next cycle as long as it does not overlap with other seasons in the next cycle. Therefore, if a season ends at time $\omega_i^{\text{end}} > 52$, we split the season into two smaller seasons where the first season starts and ends at period ω_i^{start} and 52, while the second season starts and ends at period 1 and $\omega_i^{\text{end}} - 52$.

Table 14: Optimal solutions obtained by the MA

Method	Type	Scenario	Review Time (T)	Season 1	Season 2	Season 3	Season 4
MA	Median	1	4	[2,684; 1,148; 42; 62]	[2,685; 1,148; 11; 22]	[807; 1,149; 23; 28]	[178; 667; 29; 41]
	Best		2	[3,766; 1,552; 1; 40]	[650; 151; 23; 40]	[1213; 552; 41; 44]	[3,766; 552; 45; 52]
	Worst		3	[2,838; 1,062; 1; 17]	[697; 1,274; 18; 25]	[699; 1,305; 26; 48]	[2,838; 1,093; 49; 52]
MA	Median	2	2	[1,946; 1,037; 1; 11]	[1,983; 1,216; 12; 16]	[521; 1,437; 17; 46]	[1,970; 862; 47; 52]
	Best		2	[1,967; 1,332; 1; 6]	[1,886; 1,178; 7; 16]	[611; 386; 17; 44]	[1,967; 1,332; 45; 52]
	Worst		4	[3,065; 1,414; 48; 57]	[9,264; 1,344; 6; 11]	[2,861; 1,344; 12; 15]	[882; 1,344; 16; 47]
MA	Median	3	2	[5,383; 728; 47; 70]	[584; 157; 19; 30]	[584; 157; 31; 39]	[584; 728; 40; 46]
	Best		3	[2,028; 1,260; 52; 70]	[1,390; 405; 19; 33]	[955; 556; 34; 47]	[955; 1,138; 48; 51]
	Worst		4	[9,785; 1,436; 49; 53]	[9,785; 1,436; 2; 16]	[915; 910; 17; 27]	[915; 910; 28; 48]
MA	Median	4	2	[1,737; 1,162; 45; 57]	[1,737; 1,162; 6; 19]	[414; 1,162; 20; 32]	[414; 1,162; 33; 44]]
	Best		2	[1,907; 713; 52; 68]	[777; 460; 17; 35]	[777; 460; 36; 42]	[1,907; 713; 43; 51]
	Worst		4	[4,342; 1,161; 50; 53]	[6,733; 1,640; 2; 9]	[2,639; 585; 10; 38]	[4,272; 585; 39; 49]
MA	Median	5	4	[2,462; 1,342; 51; 69]	[9,250; 661; 18; 21]	[4; 1,303; 435; 22; 42]	[1,257; 435; 43; 50]
	Best		3	[2,346; 913; 47; 58]	[2,346; 913; 7; 22]	[788; 373; 23; 35]	[788; 373; 36; 46]
	Worst		4	[4,783; 16,43; 1; 17]	[582; 961; 18; 26]	[780; 470; 27; 45]	[5,140; 918; 46; 52]
MA	Median	6	4	[2,764; 1,037; 1; 9]	[5,324; 1,037; 10; 16]	[4,019; 506; 17; 45]	[5,873; 1,591; 46; 52]
	Best		2	[2,214; 632; 43; 68]	[668; 464; 17; 27]	[668; 404; 28; 31]	[625; 404; 32; 42]
	Worst		4	[8,761; 1,405; 45; 53]	[2,494; 1,335; 2; 13]	[1,784; 665; 14; 17]	[1,224; 490; 18; 44]
MA	Median	7	4	[9,403; 1,157; 50; 54]	[2,246; 1,280; 3; 23]	[1,279; 673; 24; 28]	[1,279; 673; 29; 49]
	Best		1	[1,501; 1,256; 1; 20]	[784; 569; 21; 26]	[722; 569; 27; 48]	[7,446; 569; 49; 52]
	Worst		5	[5,789; 1,587; 52; 55]	[4,633; 1,587; 4; 21]	[5; 1,146; 788; 22; 32]	[1,584; 788; 33; 51]
MA	Median	8	4	[2,144; 1,350; 48; 54]	[2,144; 1,350; 3; 13]	[2,090; 491; 14; 25]	[2,090; 491; 26; 47]
	Best		2	[1,714; 1,349; 46; 55]	[1,714; 654; 4; 10]	[1,714; 1,924; 11; 17]	[532; 951; 18; 45]
	Worst		5	[3,752; 1,786; 1; 10]	[2,394; 718; 11; 29]	[2,394; 320; 30; 41]	[8,985; 1,551; 42; 52]
MA	Median	9	3	[4,019; 998; 48; 69]	[5,629; 1,217; 18; 21]	[802; 858; 22; 25]	[698; 801; 26; 47]
	Best		3	[2,387; 818; 44; 56]	[2,044; 1,083; 5; 16]	[2,387; 699; 17; 23]	[1,106; 221; 24; 43]
	Worst		6	[4,046; 1,929; 47; 58]	[3,169; 2,418; 7; 16]	[2,067; 1,280; 17; 23]	[901; 658; 24; 46]
MA	Median	10	3	[1,873; 1,376; 45; 69]	[502; 1,149; 18; 31]	[502; 1,149; 32; 38]	[337; 1,314; 39; 44]
	Best		2	[2,031; 890; 51; 68]	[516; 862; 17; 38]	[516; 1,009; 39; 44]	[2,136; 743; 45; 50]
	Worst		2	[1,745; 1,625; 50; 59]	[1,997; 1,636; 8; 13]	[1,045; 927; 14; 19]	[1,045; 927; 20; 49]
MA	Median	11	4	[3,037; 1,470; 51; 71]	[1,063; 453; 20; 24]	[1,063; 453; 25; 46]	[1,172; 468; 47; 50]
	Best		4	[2,393; 1,236; 45; 72]	[810; 212; 21; 31]	[360; 212; 32; 38]	[8,143; 528; 39; 44]
	Worst		6	[9,553; 1,937; 52; 72]	[1,301; 596; 21; 40]	[5,029; 966; 41; 47]	[5,029; 966; 48; 51]
MA	Median	12	1	[1,912; 1,062; 48; 61]	[1,912; 1,062; 10; 15]	[320; 527; 16; 36]	[1,936; 358; 37; 47]
	Best		3	[2,292; 895; 49; 67]	[7,310; 353; 16; 22]	[782; 353; 23; 44]	[8,207; 1,560; 45; 48]
	Worst		5	[6,681; 1,606; 42; 56]	[6,159; 603; 5; 8]	[9,921; 2,029; 9; 13]	[2,673; 513; 14; 41]
MA	Median	13	4	[6,653; 1,687; 52; 55]	[2,350; 1,771; 4; 19]	[6,305; 267; 20; 41]	[1,379; 826; 42; 51]
	Best		4	[2,555; 1,279; 51; 56]	[2,555; 1,279; 5; 23]	[954; 365; 24; 29]	[4,259; 465; 30; 50]
	Worst		6	[4,784; 1,919; 47; 54]	[6,471; 1,919; 3; 18]	[2,213; 575; 19; 29]	[1,273; 547; 30; 46]
MA	Median	14	1	[1,583; 1,204; 1; 11]	[1,567; 859; 12; 18]	[766; 851; 19; 48]	[1,622; 1,110; 49; 52]
	Best		4	[3,492; 1,179; 41; 60]	[4,844; 1,179; 9; 17]	[846; 447; 18; 22]	[846; 447; 23; 40]
	Worst		4	[2,075; 2,227; 51; 68]	[522; 5,537; 17; 20]	[56; 965; 21; 27]	[1,437; 719; 28; 50]
MA	Median	15	4	[7,103; 1,138; 1; 22]	[1,657; 626; 23; 26]	[1,657; 627; 27; 48]	[3,635; 1,000; 49; 52]
	Best		2	[1,773; 861; 49; 64]	[2,041; 861; 13; 19]	[1,023; 382; 20; 44]	[1,023; 382; 45; 48]
	Worst		8	[8,939; 2,256; 47; 72]	[1,168; 776; 21; 35]	[870; 776; 36; 41]	[8,474; 1,439; 42; 46]
MA	Median	16	6	[2,472; 1,853; 50; 70]	[1,216; 517; 19; 38]	[1,216; 517; 39; 42]	[2,472; 1,854; 43; 49]
	Best		2	[1,976; 984; 45; 63]	[1,976; 417; 12; 18]	[817; 287; 19; 39]	[797; 417; 40; 44]
	Worst		2	[5,348; 767; 49; 55]	[2,188; 846; 4; 8]	[2,050; 1,560; 9; 13]	[1,379; 485; 14; 48]
MA	Median	17	5	[6,390; 1,620; 50; 53]	[6,390; 1,620; 2; 12]	[6,390; 1,620; 13; 17]	[6,742; 587; 18; 49]
	Best		2	[2,017; 846; 51; 73]	[474; 612; 22; 31]	[474; 268; 32; 42]	[724; 1,585; 43; 50]
	Worst		2	[2,492; 400; 47; 53]	[2,732; 1,180; 2; 19]	[3,240; 232; 20; 30]	[590; 759; 31; 46]
MA	Median	18	4	[6,670; 1,551; 47; 56]	[1,418; 2,309; 5; 15]	[1,401; 2,359; 16; 19]	[1,153; 513; 20; 46]
	Best		2	[2,469; 662; 44; 66]	[799; 747; 15; 21]	[742; 269; 22; 28]	[742; 269; 29; 43]
	Worst		3	[1,935; 2,493; 48; 54]	[1,784; 1,756; 3; 14]	[3,567; 376; 15; 32]	[3,567; 376; 33; 47]

B.2.3 Average Replenishment Strategy Results

In Table 15, the performance of the average replenishment strategy, which results from averaging 10 optimal replenishment strategies produced by 10 MA executions with $m = 4$ for all 18 different scenarios. Each of the columns *TC Gap* and *FR Gap* contains the percentage deviation of the average strategy's expected total cost (TC) and expected fill rate (FR) from the average TC and FR of 10 optimal MA solutions in each scenario (i.e., TC and FR of *Average* type solutions in Table 12 and 13).

Table 15: Performance of average replenishment strategies that result from averaging 10 optimal MA solutions in each scenario

Scenario	TC	FR	TC Gap	FR Gap
1	139,208	0.9974	28.67%	1.62%
2	138,073	0.9964	28.51%	1.44%
3	255,173	0.9984	58.20%	1.56%
4	234,695	0.9967	47.68%	1.38%
5	259,674	0.9922	28.71%	1.06%
6	362,356	0.9990	81.30%	1.74%
7	135,963	0.9953	21.60%	1.20%
8	145,924	0.9982	32.85%	1.61%
9	261,173	0.9990	49.95%	1.70%
10	216,343	0.9961	33.86%	1.42%
11	392,325	0.9996	82.75%	1.72%
12	296,895	0.9960	41.26%	1.44%
13	168,147	0.9994	51.15%	1.50%
14	157,443	0.9980	42.33%	1.53%
15	302,902	0.9999	74.45%	1.83%
16	193,580	0.9928	16.34%	0.97%
17	404,138	0.9990	38.86%	1.60%
18	391,522	0.9987	80.62%	1.68%
		Average	46.62%	1.50%

B.3 Numerical Analysis Results

B.3.1 Demand Seasonality Results

Table 16 contains the result of applying an optimal seasonal replenishment strategy to compatible stationary demand. The experiment was conducted in both scenario 1 and 10, and the MA was executed for 10 times. The optimal seasonal strategy has been determined by optimizing the inventory under seasonal demand and the *Seasonal Demand* column contains the optimization results. *Stationary Demand* column contains the seasonal strategy's performance under stationary demand. Columns *TC* and *FR* contain optimal expected total cost and fill rate respectively. Columns *SC*,*HC*, *PC*, and *RC* include the percentage contribution of the setup cost, holding cost, purchase cost and review cost to the expected total cost *TC*. In *Gap* column, the seasonal strategy's optimal performance under seasonal demand is contrasted with its performance under stationary demand. The gaps for *TC* and *FR* are computed by $(TC_{\text{stationary}} - TC_{\text{seasonal}})/TC_{\text{seasonal}} \times 100$ and $(FR_{\text{stationary}} - FR_{\text{seasonal}})/FR_{\text{seasonal}} \times 100$, where TC_{seasonal} and FR_{seasonal} are the strategy's optimal total cost and fill rate under seasonal demand and $TC_{\text{stationary}}$ and $FR_{\text{stationary}}$ are the strategy's simulated total cost and fill rate under stationary demand.

Table 16: Results of applying seasonal policy to stationary demand

Scenario	Type	Seasonal Demand						Stationary Demand						Gap	
		TC	SC	HC	PC	RC	FR	TC	SC	HC	PC	RC	FR	TC	FR
1	Average	110,073	1.13%	54.12%	44.75%	0%	0.9822	119,001	0.632%	66.28%	33.10%	0%	0.9018	8.11%	8.19%
	Median	108,061	1.12%	51.55%	47.25%	0%	0.9848	119,258	0.830%	56.80%	42.37%	0%	0.9153	10.36%	-7.06%
	Best	103,794	1.19%	51.88%	46.93%	0%	0.9806	107,532	0.850%	56.48%	42.67%	0%	0.8921	3.66%	-9.02%
	Worst	123,313	1.13%	54.51%	44.36%	0%	0.9855	134,743	0.709%	62.15%	37.14%	0%	0.9244	9.27%	-6.20%
10	Average	166,595	1.80%	66.32%	31.87%	0%	0.9839	197,825	0.903%	79.19%	19.90%	0%	0.9079	18.75%	-7.73%
	Median	156,451	1.83%	66.03%	32.13%	0%	0.9830	211,709	1.003%	77.15%	21.84%	0%	0.9542	35.32%	-2.92%
	Best	148,526	1.94%	64.56%	33.50%	0%	0.9853	155,989	0.999%	76.99%	22.01%	0%	0.8860	5.03%	-10.08%
	Worst	207,898	1.80%	66.33%	31.87%	0%	0.9859	237,433	0.857%	79.76%	19.38%	0%	0.9634	14.21%	-1.78%

Table 17 contains the result of applying an optimal stationary replenishment strategy to compatible seasonal demand. The experiment was conducted in both scenario 1 and 10, and the MA was executed for 10 times. The optimal stationary strategy has been determined by optimizing the inventory under stationary demand and the *Stationary Demand* column contains the optimization results. *Seasonal Demand* column contains the stationary strategy's performance under seasonal demand. In *Gap* column, the stationary strategy's optimal performance under stationary demand is contrasted with its performance under seasonal demand. The gaps for *TC* and *FR* are computed by $(TC_{\text{seasonal}} - TC_{\text{stationary}})/TC_{\text{stationary}} \times 100$ and $(FR_{\text{seasonal}} - FR_{\text{stationary}})/FR_{\text{stationary}} \times 100$.

Table 17: Results of applying stationary policy to seasonal demand

Scenario	Type	Stationary Demand						Seasonal Demand						Gap	
		TC	SC	HC	PC	RC	FR	TC	SC	HC	PC	RC	FR	TC	FR
1	Average	100,992	0.10%	77.15%	21.84%	0%	0.9834	98,955	0.98%	47.51%	51.50%	0%	0.9409	-2.02%	-4.32%
	Median	101,088	0.13%	46.12%	52.55%	0%	0.9825	99,207	0.96%	48.56%	50.47%	0%	0.9417	-1.86%	-4.15%
	Best	99,418	0.13%	47.18%	51.53%	0%	0.9816	95,983	0.95%	48.96%	50.08%	0%	0.9346	-3.46%	-4.79%
	Worst	102,509	0.12%	48.44%	50.29%	0%	0.9806	101,776	0.94%	49.57%	49.49%	0%	0.9466	-0.72%	-3.47%
10	Average	150,359	2.18%	64.57%	33.24%	0%	0.9815	150,358	2.12%	65.64%	32.24%	0%	0.9434	0%	-3.87%
	Median	150,411	2.30%	62.72%	34.98%	0%	0.9812	150,327	2.19%	64.43%	33.37%	0%	0.9464	-0.06%	-3.55%
	Best	145,570	2.68%	63.38%	33.94%	0%	0.9830	144,837	1.95%	65.91%	32.14%	0%	0.9369	-0.50%	-4.69%
	Worst	155,272	2.55%	65.72%	31.73%	0%	0.9815	157,687	2.53%	65.64%	31.41%	0%	0.9492	1.55%	-3.29%

Table 18 contains the result of applying a piece-wise strategy to handle seasonal demand. The experiment was conducted in both scenario 1 and 10, and the MA was executed for

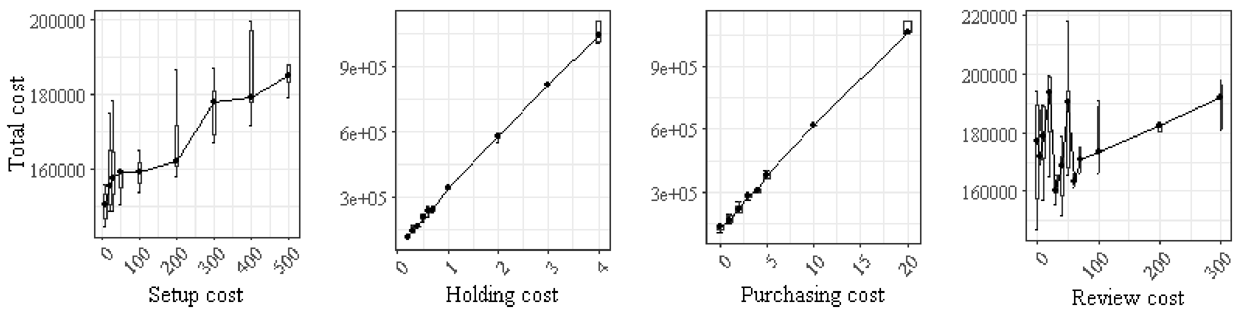
10 times. The optimal piece-wise strategy has been determined by optimizing the reorder policy for each season individually instead of jointly. In *Non-anticipatory policy* column the piece-wise strategy's performance in seasonal demand is contained. *Optimal policy by MA* column contains the optimal results of the anticipatory seasonal strategies determined by the MA. In *Gap* column, the pice-wise strategy's performance is contrasted with the performance of the optimal replenishment strategy by MA. The gaps for TC and FR are computed by $(TC_{\text{piece-wise}} - TC_{\text{MA}})/TC_{\text{MA}} \times 100$ and $(FR_{\text{piece-wise}} - FR_{\text{MA}})/FR_{\text{MA}} \times 100$, where $TC_{\text{piece-wise}}$ and $FR_{\text{piece-wise}}$ are the expected total cost and the expected fill rate of the piece-wise strategy, while TC_{MA} and FR_{MA} are the optimal total cost and the fill rate of the optimal strategy found by the MA.

Table 18: Results of piece-wise non-anticipatory policy

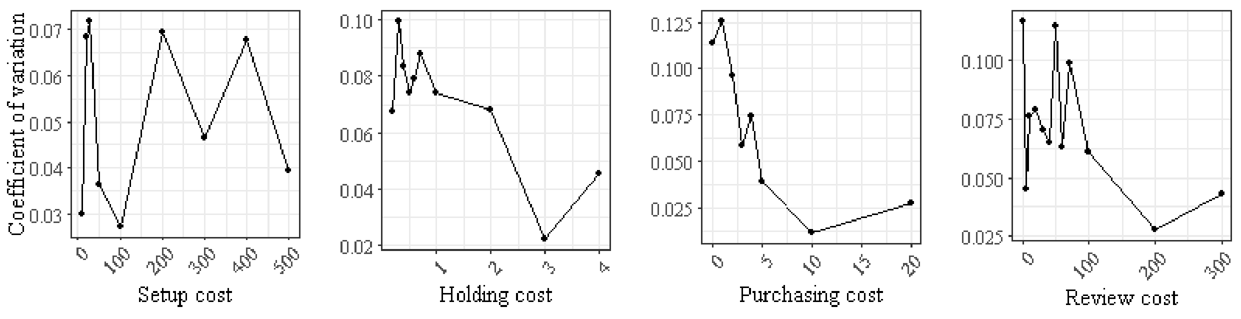
Scenario	Type	Non-anticipatory policy						Optimal policy by MA						Gap	
		TC	SC	HC	PC	RC	FR	TC	SC	HC	PC	RC	FR	TC	FR
1	Average	123,134	1.02%	58.92%	40.05%	0%	0.9882	110,073	1.13%	54.12%	44.75%	0%	0.9822	11.87%	0.61%
	Median	123,082	1.16%	59.26%	39.56%	0%	0.9892	108,061	1.12%	51.55%	47.25%	0%	0.9848	13.90%	0.45%
	Best	120,601	1.16%	59.27%	39.57%	0%	0.9887	103,794	1.19%	51.88%	46.93%	0%	0.9806	16.19%	0.83%
	Worst	125,575	1.13%	60.87%	39.00%	0%	0.9841	123,313	1.13%	54.51%	44.36%	0%	0.9855	1.83%	-0.14%
10	Average	187,726	1.61%	72.51%	25.88%	0%	0.9855	166,595	1.80%	66.32%	31.87%	0%	0.9839	12.69%	0.16%
	Median	187,313	1.56%	72.18%	26.26%	0%	0.9859	156,451	1.83%	66.03%	32.13%	0%	0.9830	19.73%	0.30%
	Best	180,048	1.58%	71.63%	26.78%	0%	0.9845	148,526	1.94%	64.56%	33.50%	0%	0.9853	21.22%	-0.08%
	Worst	192,356	1.80%	69.15%	29.05%	0%	0.9822	207,898	1.80%	66.33%	31.87%	0%	0.9859	-7.48%	-0.38%

B.3.2 Cost Structure Results

Figure 47 shows the effects of different cost parameters on the total cost in scenario 10. To help visualize the trend, we connected the median value of each box plot by a straight line.



(a) Total cost for different cost parameters



(b) Coefficient of variation for different cost parameters

Figure 47: Results for different cost structure in scenario 10

Figure 48 illustrates the optimal review time for different review costs.

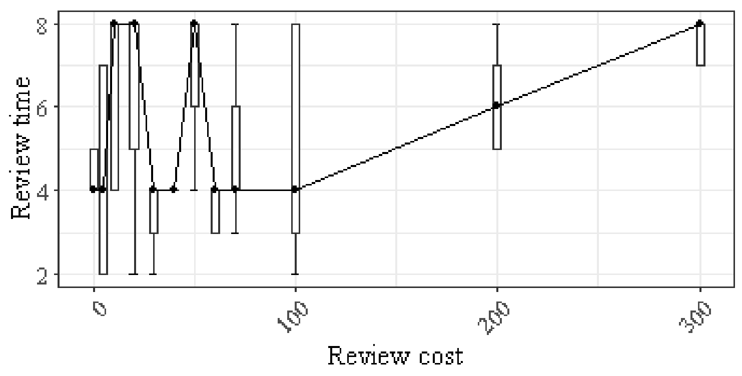


Figure 48: Effect of review cost on optimal review time in scenario 10

B.3.3 Target Service Level Results

Figure 49 illustrates the effect of minimum target service level on the optimal expected total cost in scenario 10.

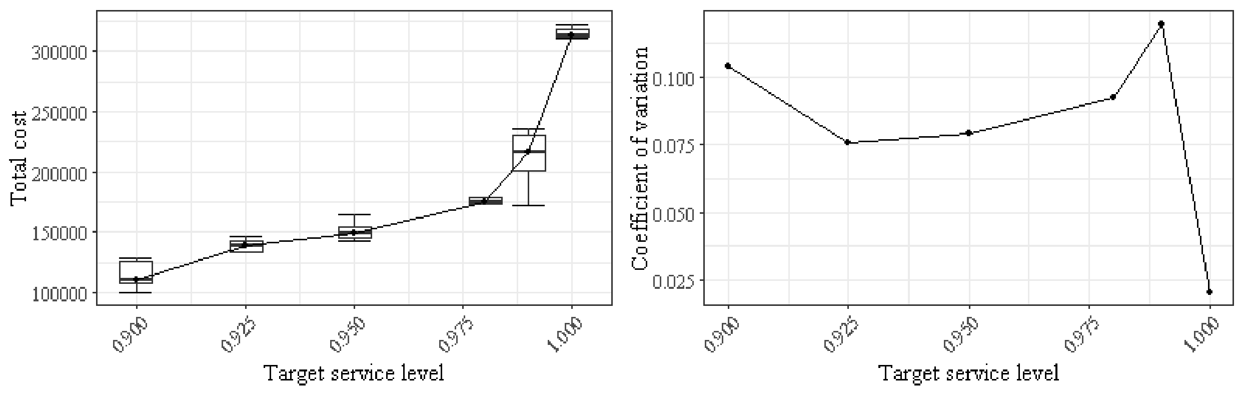


Figure 49: Sensitivity results for target service levels in scenario 10

B.3.4 Planning Horizon Results

Figure 50 shows the expected total costs of 10 optimal replenishment strategies identified by 10 independent MA executions for each different level of planning horizon in scenario 10.

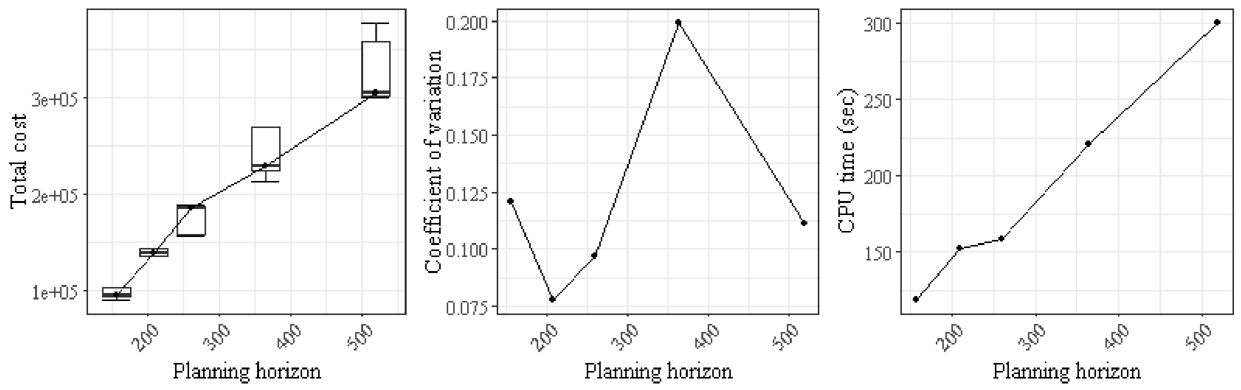


Figure 50: Sensitivity results for planning horizon in scenario 10

B.3.5 Number of Seasons Results

Figure 51 shows the optimal expected total costs, coefficients of variation and computation times identified by 10 different MA runs for different number of seasons in scenario

10.

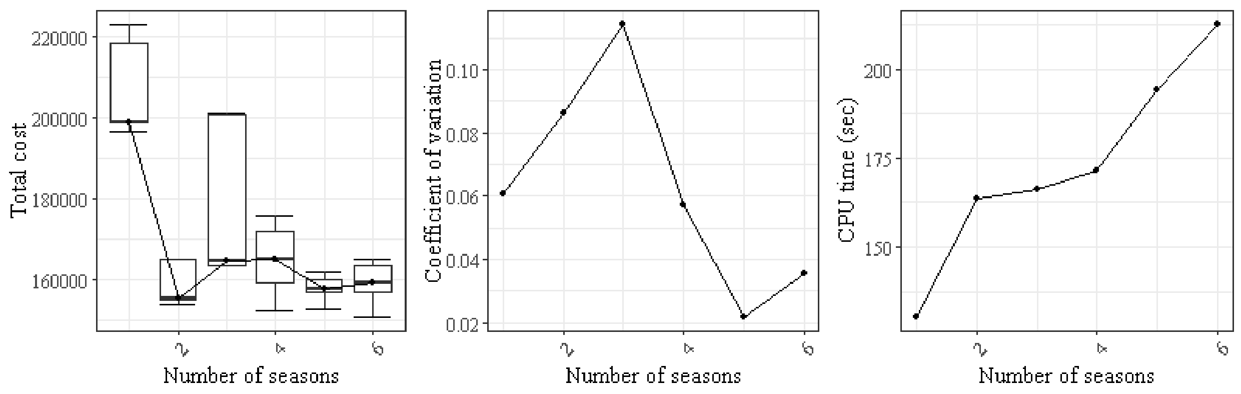


Figure 51: Sensitivity results for number of seasons in scenario 10

B.4 Algorithm Testing Results

In this section, we evaluate the effectiveness of the proposed MA by analyzing the impact of each operator. The testing experiments were conducted by isolating effect of the interested operator from all other operators. As explained earlier in Section 5.5, we conducted the experiments in scenario 1 and 10.

B.4.1 Resampling Method

To handle stochasticity of the considered inventory system, our MA uses a dynamic resampling technique that dynamically determines the appropriate number of demand and lead time scenarios for reducing the measurement noise to below a threshold standard error. For a limited computational budget, the size of simulation sample (i.e., number of simulated demand and lead time scenarios) is an important factor to the algorithm's accuracy and computational performance. In this section, we examine the accuracy of the proposed resampling method by comparing with different types of noise handling techniques.

Single-sample simulation

The first comparison involves the MA performance without any kinds of resampling. In this sub-experiment, we simulate only one randomly selected demand and lead time scenario (i.e., single unique sequence of d_t and L_t for $t = 1, \dots, N$) to estimate the expected total cost and the fill rate of a replenishment strategy $\mathcal{P}$.

Table 19 presents the bias of the optimal solution produced by the MA that estimates the performance measures (i.e., expected total cost and fill rate) based on a single instance of demand and lead time. For each of scenario 1 and 10, we executed the single-sample MA 10 times and identified the average, best and worst optimal solutions from those 10 solutions. In the *Single sample* column, the estimated performance measures are presented from the single-sample MA. The *Dynamic sample* column contains the performance estimation of those single-sample optimal solutions using the proposed dynamic resampling technique. The *True value* column contains the unbiased estimation of the solutions' true performance by simulating their performance $10,000$ different demand and lead time instances. The *Gap* column compares the solution's true value and the single-sample estimates of the expected total cost and the fill rate. Any negative percentage gap indicates that the performance measure (i.e., TC or FR) from the single-sample MA is higher than the true performance measure.

Table 19: Simulation comparison of replenishment strategies identified by the single-sample MA

		Single sample			Dynamic sample			True value				
Scenario	Strategy	Sample	TC	FR	Sample	TC	FR	Sample	TC	FR	TC gap	FR gap
1	Average	1	101893	0.983	133	110064	0.969	10000	109739	0.97	-7.15%	1.34%
	Best	1	90157	0.982	114	89573	0.960	10000	89440	0.958	-0.801%	2.51%
	Worst	1	116633	0.983	199	137468	0.985	10000	137083	0.984	-14.9%	-0.102%
10	Average	1	151972	0.986	214	188763	0.974	10000	190005	0.976	-20.01%	-1.03%
	Best	1	133780	0.980	124	139685	0.957	10000	139230	0.962	-23.70%	-1.87%
	Worst	1	172234	0.985	215	220966	0.98	10000	221315	0.98	-39.55%	-0.51%

The estimation bias of optimizing for a single instance of demand and lead time is significant for both symmetric (scenario 1) and asymmetric (scenario 10) seasonality cases. The average size of the gap between the true performance value and the single-sample estimation is large for both the expected total cost and the expected fill rate. In general, the optimal strategy produced from the single-sample simulation is in fact more costly than the estimation in single-sample (i.e., negative gap from its true TC) and its service level is significantly worse than the simulated instance (i.e., positive gap from its true FR).

Deterministic simulation

A deterministic counterpart of the stochastic simulation model replaces any random variables with fixed deterministic values. Thus, the demand and lead time scenarios are not randomly generated from their distribution. Instead, their mean value μ_t and $1/q$ is used. Accordingly, resampling techniques are not necessary. However, estimating the performance in the stochastic system with an analogous deterministic model may draw biased conclusions. In Table 20, the optimal solutions found by the deterministic simulation are compared to their true performances. The column *True value* contains the true performance measured estimated by simulating the solution for $10,000$ different demand and lead time scenarios. The columns *TC gap* and *FR gap* each contains the percentage deviation of the deterministic estimates to the true estimates. The *Dynamic sample* column contains the estimated stochastic performance using the proposed resampling technique.

Table 20: Simulation comparison of replenishment strategies identified by the deterministic MA

		Deterministic simulation			Dynamic sample			True value				
Scenario	Strategy	Sample	TC	FR	Sample	TC	FR	Sample	TC	FR	TC gap	FR gap
1	Average	1	81665	0.981	175	94910	0.921	10000	94882	0.919	-13.9%	6.74%
	Best	1	63205	0.980	130	66893	0.81	10000	67802	0.801	-6.78%	22.4%
	Worst	1	102936	0.980	122	110600	0.942	10000	110763	0.943	-7.07%	4.26%
10	Average	1	142388	0.983	272	178219	0.923	10000	168046	0.923	-15.27%	6.5%
	Best	1	85716	0.980	114	116434	0.886	10000	115642	0.881	-58.54%	11.2%
	Worst	1	183337	0.982	369	231874	0.973	10000	233125	0.975	-63.23%	7.18%

Approximating the results of the stochastic simulation with the deterministic simulation model generates an average 14% under-estimation of its true expected total cost and a 6.6% over-estimation of its true fill rate.

Efficiency of proposed resampling method

Assuming that the true noiseless fitness value of a replenishment strategy can be obtained by calculating its average performance under $10,000$ randomly generated demand and lead time scenarios, we can approximate the gap between the ideal sample size and the sample size of the proposed method. The ideal sample size was calculated by iteratively increasing its sample size until the sample fitness value was within a 0.1% tolerance level from the true fitness value. Our method, on the other hand, did not know the true fitness value and determined the sample size until the standard error of the fitness value reached below the threshold $n_{thr} = 1\%$. In Table 21 the average sample size required by the proposed dynamic resampling method is compared to the ideal sample size for $1,000$ randomly generated replenishment strategies. In *Sample size standard deviation* column, the standard deviation of the sample size calculated by each method for $1,000$ random replenishment strategies are presented. It shows that the proposed resampling requires approximately 5.6 times more samples than the ideal sample size.

Table 21: Accuracy comparison of the proposed resampling technique

	Ideal sample		Dynamic resampling		Comparison
Scenario	Sample size mean	Sample size standard deviation	Sample size mean	Sample size standard deviation	Mean sample size gap
1	82.88	348.87	571.08	167.53	589.04%
10	84.78	338.74	532.64	162.10	528.26%

B.4.2 Fitness Function

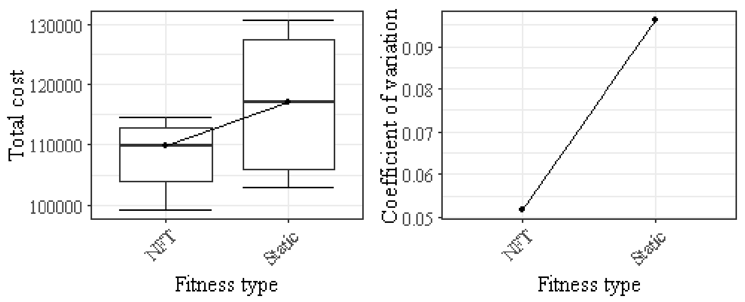
In our MA, we adapted a Near-Feasibility Threshold (NFT) fitness function proposed by Coit et al. (1996). The proposed form of fitness function is both dynamic and adaptive to the optimization procedure. In this section, the benefit of using this particular fitness function will be evaluated by comparing with another possible form of fitness function.

The simplest method of penalizing violation of the target service level is to use a constant penalty coefficient (Deb, 2000). We refer to this type of function as *static fitness function* because of the static value of penalty coefficient. While fixing the penalty coefficient constant, a common approach to set the total penalty term in the literature is to reflect the distance to feasibility (Deb, 2000). In our experiment, we use a simple difference between the target level and the observed fill rate as a distance metric of the feasibility. Therefore, the static fitness value of a replenishment strategy $\mathcal{P}$ takes the form

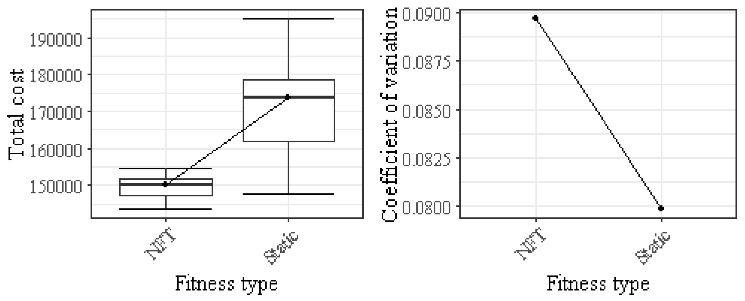
$$g_{\text{STATIC}}(\mathcal{P}) = f(\mathcal{P}) + C_{\text{STATIC}}\delta(\mathcal{P}) \tag{45}$$

where C_{STATIC} is a positive penalty coefficient imposed for constraint violation and $0 \leq \delta(\mathcal{P}) \leq 1$ is the positive distance to feasibility, as defined in Equation (25). In our experiment, we set a high C_{STATIC} value to heavily penalize infeasible solutions. C_{STATIC} is set equal to 10 times the average total cost of the individuals in the initial population.

Figure 52 shows the optimization results by using the static fitness function and the adaptive NFT function in scenario 1 and 10.



(a) Optimal costs in scenario 1



(b) Optimal costs in scenario 10

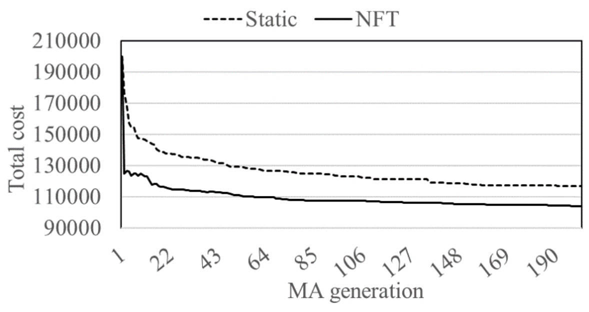
Figure 52: Optimization results with the NFT and the static fitness functions

We conducted a standard t-test to determine if the difference between the optimal expected total costs from the NFT and the static fitness functions is significant. Table 22 shows the results from the t-test. While the difference in scenario 10 is significant at 99% confidence level (i.e., p-value $0.0087 < 0.01$), it is not true in scenario 1 (i.e., p-value $0.0545 > 0.01$). Therefore, the adaptive NFT function makes mixed impacts on the final quality of the optimal solution.

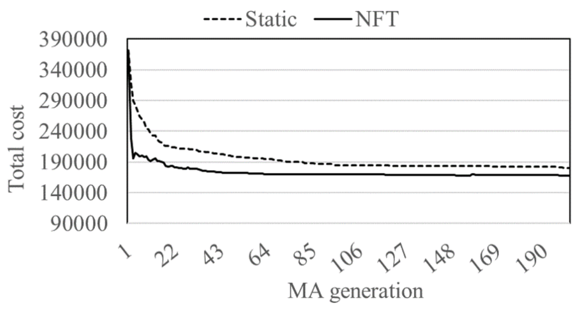
Table 22: Results of t-test for the NFT and the static fitness function

Scenario	Function	Samples	Mean	t	p
1	NFT	20	108,369	-2.057	0.0545
	Static	20	116,511		
10	NFT	20	153,565	-2.943	0.0087
	Static	20	116,511		

Figure 53 show the average convergence of the optimal expected total cost identified by the NFT and the static function during 10 MA executions in scenario 1 and 10. In order to make a fair comparison we plotted the evolution of optimal expected total cost instead of the fitness value, since in the latter case the results can be misinterpreted due to different definition of the fitness function.



(a) Average cost development in scenario 1



(b) Average cost development in scenario 10

Figure 53: Average convergence rate of the optimal expected total costs using the NFT and the static fitness functions

The NFT function achieves significantly faster convergence than the static function, especially during the first three generations. This is due to the dynamic penalty function that allowed the algorithm to effectively explore the search space without boundaries in early generations.

B.4.3 Initial Population Generation Method

An effective population initialization approach can remarkably accelerate convergence of the optimization process and improve overall solution quality (Rahnamayan et al., 2007). In our research, we produced initial population by developing a unique heuristic method that incorporates the analytical model proposed by Axsäter (2015). In this section, we examined the benefit of our initialization approach by comparing the quality of the initial population generated by our approach to a random initialization procedure, which is the most commonly selected approach in the literature (Rahnamayan et al., 2007; Janecek and Tan, 2011).

Table 23 contains the experiment results for two different methods of population initialization. For each of two scenarios, we generated 5 independent populations of 50 individuals each. Two columns *Randomization* and *Proposed method* show the mean (μ) and the standard deviation (σ) of expected total cost (TC) and expected fill rate (FR) of the individuals contained in the population, as well as the number of feasible individuals ($n_{feasible}$) out of 50 individuals.

Table 23: Simulation experiment for the performance of the population initialization method

		Randomization					Proposed method						
Scenario	Population	μ_{TC}	σ_{TC}	μ_{FR}	σ_{FR}	$n_{feasible}$	μ_{TC}	σ_{TC}	μ_{FR}	σ_{FR}	$n_{feasible}$	μ_{TC} Gap	μ_{FR} Gap
1	1	407334	96654	0.994	0.0197	46	292294	34641	0.999	0.000825	50	28%	0%
	2	384235	116016	0.975	0.0722	41	285758	37971	0.999	0.000931	50	26%	-2%
	3	403138	95475	0.993	0.0196	46	288330	41004	0.999	0.000858	50	28%	-1%
	4	423218	110440	0.989	0.0355	45	295519	33970	0.999	0.00428	49	30%	-1%
	5	390440	130682	0.985	0.0311	41	296205	29301	0.999	0.000603	50	24%	0%
10	1	777655	197187	0.987	0.0598	46	439818	20330	0.985	0.00569	46	43%	0%
	2	699083	197242	0.983	0.0424	41	437426	20802	0.981	0.034	45	37%	0%
	3	687568	189517	0.98	0.0768	45	438664	18661	0.986	0.00605	46	36%	-1%
	4	756121	225309	0.991	0.0314	47	440101	22809	0.984	0.0053	47	42%	1%
	5	708086	198846	0.988	0.0536	47	436116	27221	0.981	0.0336	48	38%	1%

One can observe that the proposed initialization method produces a better quality population than the pure random guess. For both scenarios, the average cost savings are significant (33.2%) and the proportion of feasible solutions is reasonably high: 90.6% of total population were feasible.

We mention that the proposed experiment is intended to observe the direct impact only, and not necessarily the indirect consequences of the proposed initialization method. Obviously, generating a good quality initial population can make the search process converge faster. Yet ironically, this may also increase the risk of premature convergence by neglecting a lower quality solution in a better search space while appreciating the solutions that are actually local-optima trapped in a highly multimodal search space. Nevertheless, our algorithm applies a high number of generations (P) and a powerful LSA to identify a more desirable search space as the search progresses.

B.4.4 Local Search Algorithm

An important factor of the proposed MA is the LSA. The effectiveness of the LSA was directly examined by comparing the performance of the MA with the LSA to the MA without the LSA. As we applied the LSA to a subset of best individuals in each generation, different sizes of the subset are examined.

In Figure 54, the optimal expected total costs for 10 different MA replications in Scenario 1 are presented for different number of individuals n_{LS} that are improved by the LSA in each generation: $n_{\mathrm{LS}} \in \{5, 10, 15, 20\}$. To visualize the benefit of the LSA, we also produced the solutions from the MA without the LSA and plotted their expected total cost in "OFF" category. The individuals for the LSA were selected randomly from the generation.

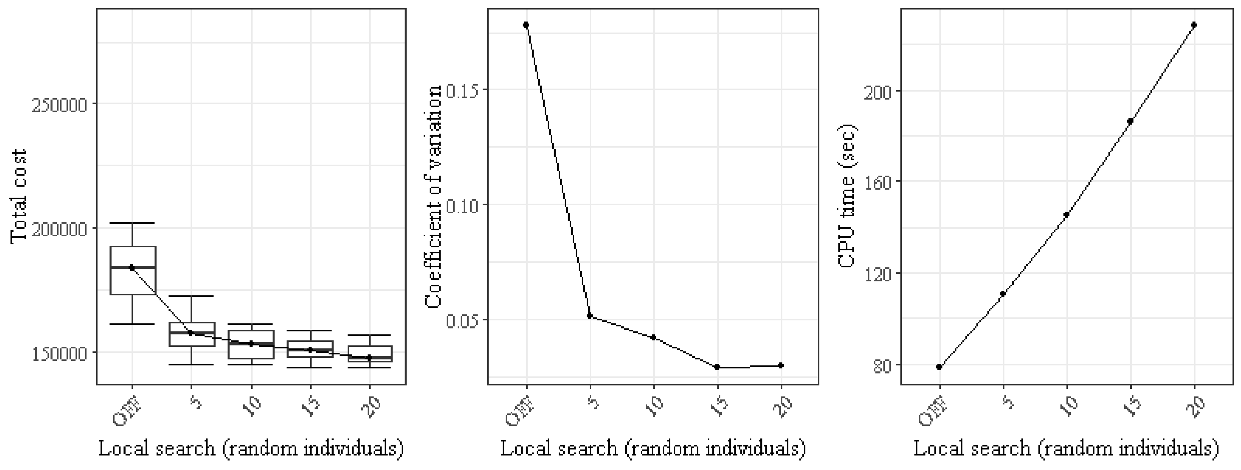


Figure 54: Test results for the LSA

Including the LSA reduced the expected total cost by more than 17%. Also, there is a clear benefit of applying the LSA to more individuals in the population since applying to 5 more individuals saved 5.56% of the expected total cost. However, the computation time also increases significantly for an increasing number of LSA applications, increasing approximately 2.5% of total CPU time for one more individual.

To achieve the best optimization performance, an appropriate subset of individuals need to be selected for the LSA in each generation. We considered three types of individual subsets with different sizes. The subset may consist of best (with the lowest fitness value), worst (with the highest fitness value) or randomly selected n_{LS} number of individuals, where n_{LS}

varies in the set $\{5, 10, 15, 20\}$. Figure 55 demonstrates the effects on algorithm performance for each of these subsets.

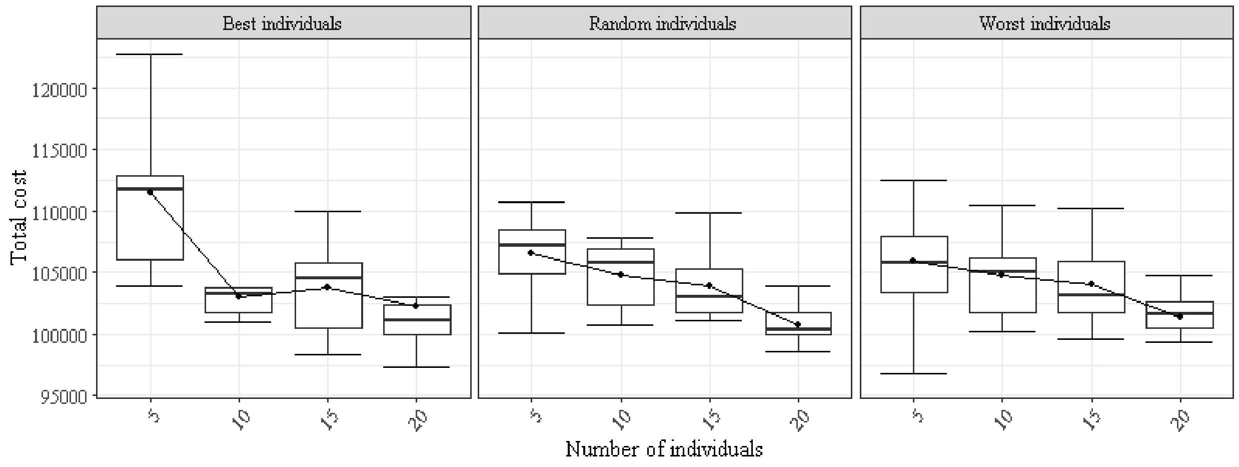


Figure 55: Effect of the LSA with different number and type of individuals in each generation

It is not very evident for which class of individuals the LSA achieves the maximum performance. However, applying the LSA to randomly selected individuals achieves the smallest average cost variance, irrespective of how many individuals have been selected.

There are six different operators involved in the LSA and each of these operators has different magnitude of impact on the optimal solution. To better understand which operators obtain the most and the least improvement during the search, the percentage contribution of each local operator has been graphically shown in Figure 56. The size of contribution was estimated by the amount of total reduction in the fitness value of all solutions that have been improved through the LSA. The graph shows the average improvement result of scenario 1 and 10.

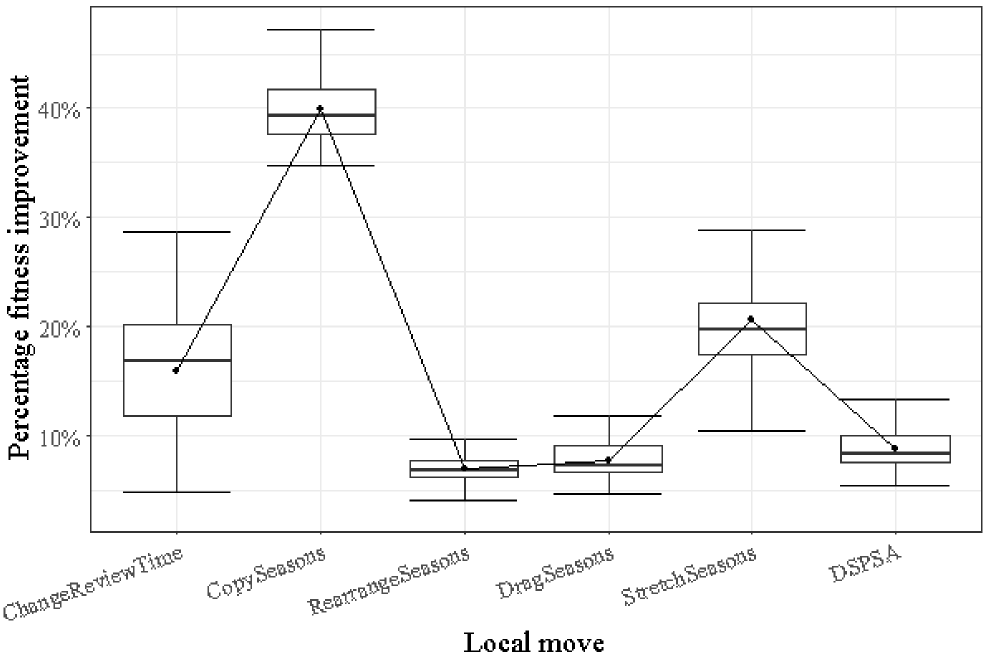


Figure 56: Average improvement of fitness value by each LSA operator

It is clear that three neighborhood operators: **ChangeReviewTime**, **CopySeasons** and **StretchSeasons** obtain the largest improvement in the fitness function, each contributing an average 16%, 39% and 19% reduction of the fitness value during the LSA. The effects of remaining three operators **RearrangeSeasons**, **DragSeasons** and **DSPSA** are relatively small, attaining less than 10% average improve in the fitness value. However, all six operators still make positive contributions to the performance of the algorithm. Furthermore, considering that the performance of each operator is dependent on the adjustments made by its preceding operators, carelessly removing the operators that showed little contributions may substantially deteriorate the algorithm's final performance.