Expert Systems With Applications 202 (2022) 117304

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Review

A robust optimization approach for disassembly assembly routing problem under uncertain yields

Sana Frifita [a,b,*], Hasan Murat Afsar [a], Faicel Hnaien [a]

[a] LIST3N, University of Technology of Troyes, France
[b] CESIT, Kedge Business School, France

ARTICLE INFO

Keywords:
Assembly and disassembly operations
Routing problem
Yields uncertainty
Discrete scenarios
Integer programming
Matheuristic

ABSTRACT

This study examines a stochastic disassembly assembly routing problem with returns (SDARP-R), in which disassembly yields are considered uncertain due to the unknown status of the returned products. We introduce a new formulation for the robust SDARP-R defined by discrete scenarios. We address five hypotheses, where different levels of decision flexibility are given. Analyses indicate that allowing flexibility in the procurement process improves the total cost but increases the difficulty of solving the problem, as well. However, allowing flexibility only to purchasing decisions is the most practical to the industry. An efficient matheuristic method based on integer programming and the Variable Neighborhood Search algorithm is also developed. The performance of the matheuristic method is evaluated through numerical tests, under three most promising hypothesis.

1. Introduction

Over the last years, reverse logistics has received increasing attention of industry practitioners and academic researchers (Govindan et al., 2015; Sheu et al., 2005). This is for environmental as well as economic reasons. One of the main environmental concerns is the increasing rate of waste production, which is leading to the saturation of landfills. For example, in the European Union, 3 billion tons of waste are thrown away each year, including 90 million tons of hazardous products. Large quantities of waste can cause significant public health and environmental problems such as the accelerate of pollution, global warming and the depletion of natural resources. To address these problems, governments are imposing new and stricter environmental regulations that require manufacturers to recover their end-of-life products through a reverse logistics network. Companies can not only comply with legal regulations but also utilize the remaining economic value contained in end-of-life products through various product recovery options, including reuse, recycling, repackaging, disposal, etc. Many companies are interested in recycling any unwanted items. The management of product recovery involves the management of all products, components and materials that are under the responsibility of a manufacturing company. The objectives are to recover as much economic and ecological value as is reasonably possible, thus reducing the amount of ultimate waste. The disassembly of returned products is the first step in product recovery. It allows the recovery of products at the end of their life. Gupta and Taleb (1994) define this process as a systematic method for separating the product into modules, components, sub-assemblies or other groupings. In this process, a set of components of different qualities are obtained that can be reused or discarded. The reuse of used components helps to reduce pollution emissions, consumption of natural resources and to make production processes more environmentally. For this reason, several kinds of research studied the problems of reusable disassembled parts in different contexts, such as the automotive industry (Mathiyazhagan et al., 2018), cell phones (Sawanishi et al., 2015), food industry (Accorsi et al., 2020), etc. Despite the motivations and advantages of this reverse logistic, its implementation is still complicated. Indeed, there are still certain levels of uncertainty, particularly regarding the availability and the quality and quantity of returned products. Such uncertainty is influenced by many factors such as the bad returned product quality or the damage incurred during the disassembly operation. To reflect the variations in the quality of used components, the yield of the disassembly process, i.e. the quantities of components is assumed uncertain.

Since 1970, researchers prove that the supply chain made more effective and profitable through coordinating its activities via information sharing (Hein & Almeder, 2016), with particular emphasis on production and transport as two main supply chain activities. The lot-sizing problem (LSP) one of the famous production process problems, combined with the vehicle routing problem (VRP), represents a computational challenge for many researchers (Camargo et al., 2014; Fachini

* Correspondence to: 64 Rue emile combes 33400 Talence, France.
E-mail addresses: sana.frifita@kedgebs.com (S. Frifita), hasan_murat.afsar@utt.fr (H.M. Afsar), faicel.hnaien@utt.fr (F. Hnaien).

https://doi.org/10.1016/j.eswa.2022.117304
Received 28 May 2021; Received in revised form 4 March 2022; Accepted 22 April 2022
Available online 30 April 2022

et al., 2017). The LSP allows the determination of the periods and the quantities to be produced to meet the demand for each period over a finite planning horizon, minimizing the sum of all the costs involved. The VRP involves designing optimal pick-up routes from geographically dispersed suppliers, subject to multiple constraints.

This study examines a multi-period dynamic disassembly assembly routing problem with returns (SDARP-R), in which the yields (r_i) are considered uncertain due to the unknown status of the returned products. To the best of our knowledge, no existing research has addressed this problem. From the real case, many factories produce items with new and used components disassembled from returned products (Guide, 2000). This kind of production system includes the disassembly and assembly operations that present complex issues regarding the structure of production costs. Taking uncertainty into account in the integrated decisions can make more sense but also increases the difficulty of the operational problems to be solved. This uncertainty may affect the production and procurement process. For this reason, different levels of flexibility are given to the different decisions related to these two supply chain activities that may or not be impacted by the scenarios. However, the system might not be able to satisfy the demand on time due to uncertainty on the quantities of the components. In this situation, the corresponding demand is lost incurring a high penalty cost.

The contributions of the current research are threefold:

1. We use the minimax robust optimization methodology introduced by Ben-Tal and Nemirovski (1998) to formulate a new problem, the robust SDARP-R. Each scenario corresponds to a vector of components' yield.
2. We propose a matheuristic algorithm to solve the robust SDARP-R under flexibility hypothesis.
3. We analyze impacts of different flexibility levels on the total cost.

The paper is organized as follows. Section 2 provides a literature review of related research. The optimization problem is formulated in Section 3. Section 4 presents the solution approach. The obtained results are provided and analyzed in Section 5. Section 6 gives concluding remarks.

2. Literature review

In the first part of the literature review, we briefly discuss the stochastic lot sizing problems. In the second part, we focus on the vehicle routing problem with stochastic demands. Finally, we give an overview of the integrated production management and vehicle routing problems in which uncertainties are taken into account.

2.1. Stochastic lot-sizing problems

The lot-sizing problem (LSP) introduced by Wagner and Whitin (1958) consists of making production and inventory decisions over a given planning horizon. Most of the literature considers the deterministic case of LSP, which can result in some cases wrong and costly decisions. Studying the stochastic case of the LSP (SLSP) is perhaps the most correct way to be attuned to reality. The SLSP papers mostly focus on demand as the source of uncertainty and few considered stochasticity in other elements such as cost, yield, lead-time and capacity. For the problem with stochastic demands, the service level constraint is one of the famous methods used by Bookbinder and Tan (1988). The authors proposed a chance constraint to model a service level constraint, and developed three strategies to handle the resulting setting, which are the static uncertainty, dynamic uncertainty and static–dynamic uncertainty. In like manner, Tempelmeier and Herpers (2011) solved a dynamic multi-item capacitated LSP under random period demands. Rossi et al. (2015) studied the SLSP in which the demand is assumed uncertain and non-stationary. They proposed four models based on the static–dynamic uncertainty strategy, i.e. stochastic lot-sizing, α-service level constraints, penalty cost scheme, and β-service level constraints. The scenarios approach is also used to deal with the demand uncertainty in SLSP. Gutiérrez et al. (2004) addressed the single item SLSP where the costs and demand distribution depend on the scenario are considered. They solved the problem based on a branch-and-bound approach. Helber et al. (2013) used demand scenarios to approximate a nonlinear version of the SLSP. Machine breakdowns are assumed stochastic by Nourelfath (2011) for the multi-period and multi-item SLSP. The proposed model includes a set of constraints to ensure some minimum probability of meeting the customer service level, and it is solved by a two-phase solution approach. Hnaien and Afsar (2017) provided the min–max robust LSP with uncertain lead times. They explored the incapacitated and capacitated cases with and without lost sales, with discrete scenarios. Taş et al. (2019) formulated the LSP with stochastic setup times and overtime as a two-stage stochastic programming problem. They applied a sample average approximation procedure to obtain upper bounds and a statistical lower bound to evaluate the solutions of two proposed heuristics.

Recently, various research areas are studied the reverse LSP, which groups different activities such as disassembly, repair or recycling of end-life products. Among them, we focus on the disassembly process and especially the disassembly scheduling problem (DSP). Most of the related research treating the deterministic disassembly scheduling problem (Gupta & Taleb, 1994). For example, Kim et al. (2006) developed a two-phase heuristic where Langella (2007) presented an integer program and several heuristics to solve the problem. Compared to its deterministic counterpart, the literature on stochastic disassembly scheduling problem is still very scarce. As the SLSP, existing studies show interest in demand and/or yield uncertainty. Typically, the random yield does not necessarily the result of defective items, but it can also arise due to shortages from the suppliers (Moon et al., 2012).

There exist two approaches to represent random yield in a production context. The first group of research (e.g., Teunter and Flapper (2003)) represents random yield with a Bernoulli process, where a certain fraction of goods is defective, and the number of defective items depends on the lot size.

The second group of research (e.g., Salameh and Jaber (2000)) considered that all produced items are defective and random yields are modeled with a geometric distribution. Inderfurth and Langella (2006) proposed a one-to-one and a one-to-many heuristics to solve single-period DSP with yield uncertainty. Kongar and Gupta (2006) used a fuzzy goal programming technique to solve the disassembly to order system under uncertainty. They considered the uncertainty regarding the disassembly process by introducing various rates of broken, defective and replaced items. Barba-Gutiérrez and Adenso-Díaz (2009) extended the model proposed by Barba-Gutiérrez et al. (2008) by integrating demand uncertainty. To solve this problem, they proposed a fuzzy logic algorithm. Kim and Xirouchakis (2010) studied the DSP with stochastic demand and multiple product types with a two-level product structure. They proposed a stochastic inventory model and developed a Lagrangian relaxation method to solve it. After that, Wang and Huang (2013) considered demand uncertainty and they proposed a two-stage linear programming model using a scenario-based approach. Inderfurth et al. (2015) presented a two-root, three-leaf mathematical model that illustrates the effect of yield uncertainty in stochastically proportional and binomial models. Liu and Zhang (2018) dealt with a single-reference and multi-period disassembly scheduling problem with capacity constraint and with random yields and demands. This problem is formulated as a nonlinear mixed integer programming model also an algorithm based on an external approximation is proposed to solve it. A chance constraint approach is introduced to ensure that the probability of meeting the demand is greater than a predetermined service level and then approximated by a second order cone constraint.

2.2. Vehicle routing problem with stochastic demands

In this work, the quantities of disassembled components are assumed uncertain. This uncertainty is compensated by the purchase of new components or the disassembly of the purchased returned products. Therefore, this uncertainty impacts the collected quantity of returned products and components. Thus, this problem corresponds to the vehicle routing problem (VRP) with stochastic demand (Tillman, 1969).

The VRP is one of the famous optimization problems in logistics. In the VRP, a set of vehicles is available for serving a set of customers with demands. Of the studies that focus on the VRP with stochastic demands (VRPSD), several different mathematical models and algorithms are proposed and explored (Gendreau et al., 2016). Sungur et al. (2008) introduced a robust optimization approach to solve the VRPSD. The goal is to minimize transportation costs while satisfying all demands in a given bounded uncertainty set. Lei et al. (2011) formulated a two-stage stochastic recourse model and proposed an adaptive large neighborhood search heuristic. Lee et al. (2012) investigated the VRP with two types of uncertainty sets with adjustable parameters for the possible realizations of travel time and demand. They proposed a Dantzig–Wolfe decomposition and a dynamic programming algorithm to solve the problem with data uncertainty. Marinaki and Marinakis (2016) developed a swarm optimization based algorithm hybridized with variable neighborhood search to solve the VRPSD. Different variants of VRPSD are studied such as VRPSD with time windows and VRPSD with pick-up and delivery. Li et al. (2010) considered the VRPTW with both stochastic travel times, service times, and introduced a chance-constrained and a stochastic recourse model. Zhang et al. (2016) studied the VRPTW involving stochastic demands and they proposed three probabilistic models to address on-time deliveries probability while minimizing the expected total cost. Hu et al. (2018) studied the VRPTW under demand and travel time uncertainty. They proposed a robust optimization model to tackle small-sized instances and a two-stage algorithm based on a modified adaptive variable neighborhood search heuristic to tackle large instances. Corso and Wallace (2015) handled demand uncertainty in pick-up and delivery problem using the genetic algorithm. Goodson et al. (2017) solved the multiple-vehicles case under the re-optimization approach with approximate dynamic programming methods. Florio et al. (2020) studied the single VRPSD with optimal restocking as a Markov decision process. A wait-and-see model is proposed and it is used within a parallel heuristic to solve larger literature instances with up to 150 nodes and with a Poisson distributed demands.

2.3. Integrated production management and vehicle routing

The inventory routing problem is a generalization of the VRP. It combines inventory management at the customer and vehicle routing without considering the production planning at the central plant (Bell et al., 1983). The production routing problem combines lot-sizing decisions, inventory management, and routing (Chandra, 1993). Both the IRP and PRP are NP-hard combinatorial problems where different variants are extensively studied by focusing on developing efficient algorithms. We will focus on the matheuristic approach that is one of the efficient methods used to solve integrated production and routing problem. For the deterministic version, Absi et al. (2015) provided a matheuristic approach to solve a production routing problem. In the latter, the production routing problem is split into classic production planning problem and vehicle routing problem. The lot sizing model includes a visiting cost to account for distribution step. Consistent with methods from literature, the iterative approach has the smallest CPU time (less than 2 s) for instances with one vehicle. However, for instances with large transportation costs, it has difficulties reaching the optimal solution. The lack of intensification techniques and the weak method of diversification used may be the weak point of this approach. Russell (2017) proposed two mathematical programming-based heuristics to solve the PRP. The differences between the Absi et al. (2015) approach and the approaches proposed by Russell (2017) the manner in which vehicle routing costs are artificially incorporated, the use of predetermined routes, initial seed routes, and the vehicle routing methodology employed. These proposed multi-phase approaches achieve many new best-known solutions to test problems from the literature. However, scalability issues limit its application to large-scale problems having more than 100 retailers and a twenty-period planning horizon. Solyalı and Süral (2017) proposed a multiphase heuristics to solve the PRP. In their studied, they considered the case where a plant produces and distributes a single item to multiple retailers over a multi-period time horizon. Results show that the matheuristic managed to find new best solutions for the 65% of benchmark instances at the expense of higher computing times on large instances.

A few numbers of researchers extended the integrated problem to other more complicated problem that considers the stochastic case. For instance, Babagolzadeh et al. (2020) addressed the stochastic IRP with energy and emission costs. They considered the quantity delivered to each retailer, the route for distribution of products and the lost sale cost depend on scenarios. They proposed a two-stage stochastic programming and a matheuristic algorithm based on iterated local search and a MILP to solve the problem under 65 scenarios from a real case studied. Gruler et al. (2020) proposed an hybrid approach that combines the Variable Neighborhood Search algorithm (VNS) with a simulation to solve the stochastic IRP with a single period and stock-out requirements. Adulyasak et al. (2015) studied the stochastic PRP under demand uncertainty. They considered that the production quantity, the quantity delivered to the customer, the amount of unmet demand at customer depend on scenarios. They proposed a two-stage formulation, multi-stage problems, a bender decomposition-based branch-and-cut approach and a sample average approximation method to solve the problem with 100, 500, and 1000 scenarios. Results on instances with a large number of scenarios, show that their approaches provide significant solution speed improvements compared to a classical branch and cut algorithm. Other work studies the same problem such as the work of Shuang et al. (2019). They considered that the number of worn-out items available for pickup at customer and the number of worn-out items shipped from the customer to remanufacturing facility depends on scenarios. They proposed a two-stage stochastic MILP and generate three scenarios.

Recently, Chitsaz et al. (2019) studied the assembly routing problem (ARP) as an extension of the PRP. It presents the case of the collection of all the needed components from various suppliers to accomplish the assembly process, which is interrupted if a component is missing. ARP aims to minimize the sum of costs such as production, inventory, and procurement; subject to several types of capacity constraints over a finite and discrete-time horizon. They formulated the ARP as a mixed-integer linear programming model solved by a three-phase decomposition matheuristic. Afsar and Hnaien (2020) proposed a new formulation for the same problem but with dynamic aspect of demand.

In Table 1 we present a summary of the works closest to the problems studied in this article. Based on this table, we observe that the assembly routing problem, one of the most important problems in the supply chain, is received less attention. To the best of our knowledge, Frifita et al. (2021) are the first authors that considered the disassembly assembly routing problem, where the yields are uncertain due to the unknown situation of returned products. They introduced the formulation of the robust dynamic disassembly assembly routing problem with returns defined by discrete scenarios. However, they addressed only instances with 14 nodes under 20 scenarios. Motivated by the above observations, this paper is devoted to the studied of this new planning problem that combines assembly and disassembly operations with vehicle routing in the stochastic case.

Table 1
Articles on assembly and disassembly problems.

References	Assembly		Disassembly			Transport	Resolution methods
	Multi-item	Capacity	Multi-item	Capacity	Stochastic	LTL	
Kim and Xirouchakis (2010)			*	*	*		Lagrangian heuristic
Prakash et al. (2012)				*			Constraint-based simulated annealing algorithm
Ullerich and Buscher (2013)				*			Heuristics
Godichaud et al. (2015)				*			MILP, Genetic algorithm
Inderfurth et al. (2015)					*		Empirical analysis
Hein and Almeder (2016)		*				*	MILP
Hrouga et al. (2016)				*			Genetic algorithm, Fix-and-optimize
Ji et al. (2016)			*	*			Lagrangian heuristic
Díaz-Madroñero et al. (2017)	*	*					MILP
Liu and Zhang (2018)				*	*		Nonlinear programming, Outer approximation
Tian and Zhang (2019)			*	*			Particle swarm optimization algorithm
Habibi et al. (2019)			*	*	*	*	MILP, matheuristic
Chitsaz et al. (2019)		*				*	MILP, matheuristic
Chitsaz et al. (2020)						*	branch-and-cut
Slama et al. (2020)			*	*			MILP
Afsar and Hnaien (2020)						*	MILP, Benders decomposition
Pour-Massahian-Tafti et al. (2020)				*			3 MILP, Heuristics
Frifita et al. (2021)		*		*	*	*	MILP

3. Robust disassembly assembly routing problem with returns

3.1. Problem description

We consider a single disassembly and assembly site having a common storage capacity C_0 for components, returned and new products. The production operation is done to satisfy the external demand, d_t, at each period t by assembling a set of new and disassembled components with respecting the plant's production capacity C. Lost sales are allowed in the model with a penalty cost (pr). Without loss of generality, we assume that one unit of each component is needed to make one unit of the new product. The location of suppliers and warehouses as well as the production plant is modeled as nodes on a directed symmetric graph $G = (N, A)$, with the node-set $N^+ = N \cup \{0\}$, where 0 represents the plant, and the arc set $A = \{(u, v) : u, v \in N^+, u \neq v\}$. Each arc (u, v) has a transportation cost $c_{uv} > 0$. The set of nodes regroups the suppliers who provide a unique component and warehouses in which there are a stock of unique type product to be disassembled. These products are either at the end of their useful lives or the end of their consumer use. A bijective function ψ defines the supplier of a component ($v = \psi(j)$).

Fig. 1 presents an example of the deterministic case of the studied problem. Depending on the demand d_t for a single type of finished products, the manufacturer will produce the necessary quantities at the periods $t = 2$ and $t = 3$ using one unit of each type of raw materials and used parts. These used parts are screened and recovered by disassembling the returned products. Returned products are located in limited quantity at warehouses. We, therefore, assume here that the returned products are sufficient for the acquisition process. For the demand of the first period, it is satisfied with the shortage of final product at the factory depot. Even if there is no production in the first period, the vehicles collect components to ensure the production in the later periods. Each component whether it is new or disassembled has a size b_i, the same for the new and returned products (b). Size is taken into account for vehicle and storage capacities during the planning. The total cost includes disassembly, assembly, inventories, lost sales, purchasing and routing costs. For each product produced or disassembled in a period, the incurred cost includes a setup cost. In this example, the r_i are assumed to be known, but in the stochastic case, it can take a binary value of 0 or 1.

In some cases, the quality of each type of disassembled parts cannot be known in advance. Therefore, we generated a set of scenarios with random yields. The scenarios are equiprobable and the yields follow the Bernoulli distribution law. All returned products have a two-level structure, which means that returned products and disassembled components are considered as root and leaf items, respectively. Each returned product has a unit size b the same as the new product. This size influences the capacity of vehicles on routed and storage capacity of the plant. We consider a unit production cost f_t and setup cost f'_t at the plant level. We assume that an unlimited fleet of homogeneous vehicles with a capacity of Q is stationed at the plant, to pickup components and returned products from suppliers and warehouses represented by the set $N = \{1, \ldots, n\}$. The planning horizon comprises T periods ($t \in \{1, \ldots, |T|\}$) where the demand is known but dynamic in time, as are all the costs except transport.

3.2. Mathematical formulation

The full list of notations used throughout this paper is given in what follow:

- Sets:
  T : Planning horizon $t \in \{1, \ldots, |T|\}$.
  N^+: Set of nodes, $N^+ = \{0, \ldots, n\}$, where 0 represents the plant.
  N: Set of suppliers and specific depots that provide component and returned product, $N = N^+ \setminus \{0\}$.
  A: Set of arcs, $A = \{(u, v) : u, v \in N^+, u \neq v\}$.
  K: Set of components.
  S: Set of scenarios.

- Decision variables:
  p_t: Production quantity for period t at the plant.
  y'_t: Equal to 1 if there is disassembly operation at the plant for period t, 0 otherwise.
  y_t: Equal to 1 if there is production at the plant for period t, 0 otherwise.
  Ir_t: Inventory of returned product at the plant at the end of period t.
  I_t: Inventory of items at the plant at the end of period t.
  Ic_{it}^s: Inventory of component i at the plant at the end of period t in scenario s.
  x_{uvt}: Number of times a vehicle traverses the arc $(u, v) \in A$ for period t.
  F_{uvt}: Forward vehicle load on the arc $(u, v) \in A$ for period t.
  F_{vut}: Backward remaining vehicle capacity on the arc $(v, u) \in A$ for period t.
  z_{ut}: Equal to 1 if node $u \in N$ is visited in period t, 0 otherwise.
  q_{ut}: Shipment quantity of returned product from node $u \in N$ to the plant for period t .
  qc_{ut}: Shipment quantity of components from node $u \in N$ to the plant for period t.
  dis_t: Disassembly quantity of returned product at the plant for period t.
  l_t: Quantity of lost sales at period t.

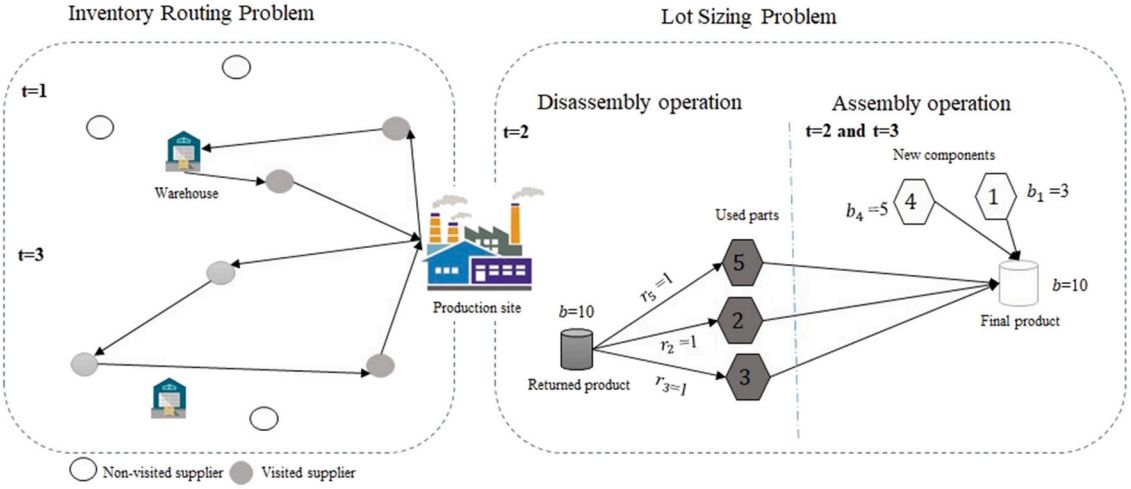

Fig. 1. Example of a SDARP-R instance.

- Parameters:

f_t:Unit production cost for new items for period t.
f'_t: Setup cost for new items for period t.
a_{it}, ar_t: Unit purchase cost for component i and returned product for period t, respectively.
h_t: Unit holding cost of new items and at the plant for period t.
hr_t: Unit holding cost of returned product at the plant for period t.
hc_{it}: Unit holding cost of component i at the plant for period t.
c_{uv}: Transportation cost between nodes u and v, $(u, v) \in A$.
C: Production capacity.
C_0: Global inventory capacity at the plant.
Q: Vehicle capacity.
S_{ut}: Supply of returned product at node $u \in N$ in period t.
Sc_{ut}: Supply of component at node $u \in N$ for period t.
r_i^s: Yield ratio of leaf item i to root item in scenario s.
b_i: Unit size of component i.
b : Unit size of product.
d_t: Demand for the final products at the plant for period t.
pr: Unit cost of lost sales.
σ_{ut}^s : Approximated visiting cost at node $u \in N$ for period t in scenario s.

The minimax robust problem modifies the objective function of deterministic problem as minimax objective function after introducing a set of uncertain yields scenarios. We propose a mathematical model based on the two-commodity flow formulation (Afsar & Hnaien, 2020). The binary variable x_{uvt} takes the value 1, if the arc *(uv)* is traversed (in any direction) in period t . The nodes 0 and *n+1* correspond to the depot node. This model presents a Minimax objective function as follows:

$$\min \quad \left(\sum_{t\in T}(f'_t(y_t+y'_t)+f_t p_t+prl_t+hr_t Ir_t+h_t I_t+\sum_{u\in N} ar_t q_{ut}+\sum_{i\in K} a_{it} qc_{\psi(i)t}+\sum_{(u,v)\in A} c_{uv}x_{uvt}+max_{s\in S}(\sum_{i\in K} hc_{it}Ic_{it}^s)\right) \quad (1)$$

s.t

$$I_{t-1}+p_t+l_t=d_t+I_t \qquad \forall t\in T \quad (2)$$

$$l_t \le d_t \qquad \forall t\in T \quad (3)$$

$$Ic_{i,t-1}^s+qc_{\psi(i)t}+r_i^s dis_t=p_t+Ic_{it}^s \qquad \forall t\in T,\forall i\in K,\forall s\in S \quad (4)$$

$$Ir_{t-1}+\sum_{u\in N} q_{ut}=dis_t+Ir_t \qquad \forall t\in T \quad (5)$$

$$p_t \le Cy_t \qquad \forall t\in T \quad (6)$$

$$dis_t \le Cy'_t \qquad \forall t\in T \quad (7)$$

$$q_{ut} \le S_{ut} \qquad \forall t\in T,\forall u\in N \quad (8)$$

$$qc_{ut} \le Sc_{ut} \qquad \forall t\in T,\forall u\in N \quad (9)$$

$$\sum_{i\in K} b_i Ic_{it}^s+b(I_t+Ir_t)\le C_0 \qquad \forall t\in T,\forall s\in S \quad (10)$$

$$b_{\psi^{-1}(u)}qc_{ut}+bq_{ut}\le Qz_{ut} \qquad \forall t\in T,\forall u\in N \quad (11)$$

$$\sum_{u\in N^+|u\neq v} x_{uvt}=\sum_{u\in N^+|u\neq v} x_{vut} \qquad \forall t\in T,\forall v\in N^+ \quad (12)$$

$$\sum_{u\in N^+} x_{uvt}=z_{vt} \qquad \forall t\in T,\forall v\in N \quad (13)$$

$$F_{uvt}+F_{vut}=Qx_{uvt} \qquad \forall t\in T,\forall u\in N^+,\forall v\in N^+ \mid u\neq v \quad (14)$$

$$\sum_{v\in N^+} F_{uvt}-\sum_{v\in N^+} F_{vut}=2(bq_{ut}+b_{\psi^{-1}(u)}qk_{ut}) \qquad \forall t\in T,\forall u\in N \quad (15)$$

$$\sum_{u\in N} F_{0ut}=\sum_{u\in N} b_{\psi^{-1}(u)}qc_{ut}+\sum_{u\in N} bq_{ut} \quad \forall t\in T \quad (16)$$

$$\sum_{u\in N} Qx_{0ut}-(\sum_{u\in N} b_{\psi^{-1}(u)}qc_{ut}+\sum_{u\in N} bq_{ut})=\sum_{u\in N} F_{u0t} \quad \forall t\in T \quad (17)$$

$$\sum_{v\in N^+} F_{n+1vt}=Q\sum_{v\in N^+} x_{0vt} \qquad \forall t\in T \quad (18)$$

$$p_t, dis_t, l_t\ge 0, y'_t, y_t\in\{0,1\} \qquad \forall t\in T,\forall u\in N \quad (19)$$

$$Ic_{it}^s, Ir_t, I_t\ge 0 \qquad \forall t\in T,\forall i\in K,\forall s\in S \quad (20)$$

$$qc_{ut}, q_{ut}\ge 0 \qquad \forall t\in T,\forall u\in N \quad (21)$$

$$F_{uvt}\ge 0 \qquad \forall t\in T,\forall u\in N^+,\forall v\in N^+ \quad (22)$$

$$z_{ut}, x_{uvt}\in\{0,1\} \quad \forall t\in T,\forall u\in N^+,\forall v\in N^+ \quad (23)$$

The objective function (1) minimizes the total setup (both disassembly and assembly operations), production, lost sales, inventory holding costs for new and returned products, purchase, and transportation costs as well as the maximum inventory holding costs of components over all scenarios. Only inventory costs of the plant are taken into account. The constraints from (2) to (11) are relevant to LSP with returns while the

constraints from (12) to (18) define the transportation problem. The constraints (2) ensure the manufactured final product inventory flow. The amount of the lost sales is less than the demand (constraint (3)). Constraints (4) and (5) ensure the components (new or disassembled) and returned product flow balance, respectively. Constraints (6) and (7) fix the setup decisions variables for the assembly and disassembly operations, respectively. Constraints (8) and (9) ensure that each visited node can provide the requested quantity of components or returned products. Constraints (10) limit the storage capacity of the final product, the components, and returned products at the plant. Constraints (11) impose the limit of total components and returned product shipment quantity from each node in each period by the vehicle capacity.

Constraints (12) state that the inflow is equal to the outflow at each node. Constraints (13) imply that if a node u is visited, there is only one arc *(uv)* is traversed by a vehicle. Constraints (14) state that the sum of forward vehicle load and the backward remaining capacity is equal to the total vehicle capacity at the traversed arc. The total change in forward vehicle load and the backward remaining capacity is twice the volume of the purchased amount from a visited node (constraints (15)). The forward vehicle load leaving the depot is equal to the total purchasing quantities (Constraints (16)). Constraints (17) state that the total backward remaining vehicle capacity re-entering depot node is the total capacity of all used vehicles minus the total volume of purchased components. Constraints (18) state that the total backward remaining capacity and the total capacity of used vehicles are equal. Constraints (19)–(23) are domain constraints.

The proposed model is nonlinear due to the nested min–max operator. We use ϕ to denote the maximum value of the components inventory cost at the plant. The mathematical model equivalent to the minimax model becomes the following linear programming model:

$$\min \sum_{t\in T}\left(f'_t(y_t+y'_t)+f_t p_t+prl_t+hr_t Ir_t+h_t I_t+\sum_{u\in N} ar_t q_{ut} + \sum_{i\in K} a_{it} qc_{\psi(i)t}+\sum_{(u,v)\in A} c_{uv}x_{uvt}\right)+\phi \tag{24}$$

s.t (2)–(23)

$$\sum_{t\in T}\sum_{i\in K} hc_{it} Ic^s_{it} \le \phi \qquad \forall s\in S \tag{25}$$

3.3. Flexibility levels

We analyze the following five assumptions H_j to identify which new model's decisions variables are the most beneficial to have flexibility and be dependent on scenarios:

- H_0 (zero flexibility): no decision variable is dependent on scenarios (p_t, l_t, dis_t, q_{ut}, qc_{ut}, x_{uvt}). The preceding mixed integer linear model presents this flexibility level.
- H_1: the production and lost sales related decisions are flexible and depend on scenarios (p^s_t, l^s_t, dis^s_t). This hypothesis allows adding flexibility to production, which means that the quantities to be assembled and disassembled depend on scenarios.
- H_2: the decisions of purchased quantities of returned products and components depend on scenarios (q^s_{ut}, qc^s_{ut}).
- H_3: procurement decisions that include the purchased quantities and the routing decisions depend on scenarios (q^s_{ut}, qc^s_{ut}, x^s_{uvt}). This hypothesis enlarges the flexibility of H_2 by including routing decisions.
- H_4: all decisions variables depend on scenarios (p^s_t, l^s_t, dis^s_t, q^s_{ut}, qc^s_{ut}, x^s_{uvt}). In this hypothesis, production and procurement depend on scenarios.

4. Two-phase matheuristic approach

In this section, we present a matheuristic algorithm based on a MIP model and a VNS method for the proposed problem. Matheuristics are heuristic methods that make use of an exact methods inside heuristics framework. This method is successfully implemented in different optimization problems such as the PRP (Absi et al., 2015), the collection disassembly problem in the work of Habibi et al. (2017), and ARP in the work of Chitsaz et al. (2019).

The VNS method is an extension of the classical local search that includes shaking procedure as a diversification mechanism (Mladenović & Hansen, 1997). VNS is a single solution based method that searches in the neighborhood of the local optimum found by local search to generate a new best solution. According to Funke et al. (2005), local search and local search-based metaheuristics are currently among the best ways to solve large vehicle routing and scheduling problems. Although simple, VNS approach combines several local search neighborhoods and it is relatively easy-to-implement, do not contain a large number of parameters requiring time-consuming setting processes, and offer an excellent trade-off in terms of solutions quality as well as in terms of computing times (Frifita & Masmoudi, 2020). We can see it is employed in the most recent works on vehicle routing problems (e.g. Kuo et al. (2022) to solve VRP with drones and time windows, Guo et al. (2022) for VRP with ride-sharing, Xu and Cai (2018) to provide excellent solutions for consistent VRP). That is why we are motivated to use the VNS algorithm and we intended to combine it with a mixed integer programming to present matheuristic.

The proposed matheuristic (Fig. 2) performs two main steps. The first step consists in solving the assembly and disassembly problem (Step A). It is addressing the decisions of dis^s_t, p_t, Ic^s_t, q^s_{ut}, qc^s_{ut}, l_t, and z^s_{ut}. Using the value of z^s_{ut}, the set of nodes visited in each period and each scenario is obtained. Also, q^s_{ut} and qc^s_{ut} values reveal the shipped quantity that forms each node. The second step (Step B), is to solve the routing problem by VNS.

We first generate a lower bound of the total costs (the sum of production cost and transportation cost). Then we initialize the value of σ^s_{ut} according to the lower bound. The parameter σ^s_{ut} is used as an approximate visiting cost (updated throughout the algorithm) of collection from node u at period t under scenario s. Consecutively, the routing problem is solved and the routing cost is obtained. Once the route of all period is built, we implement a memorization process inspired from tabu search (Step C). At each iteration, we keep in memory the best transport solution found in step B. After that, the value of σ^s_{ut} is updated based on this solution and the objective function is recalculated (see Section 4.3). This step is followed by a procedure of update for the worst case (see Section 4.4). Before resolving the LSP we check if this solution already exists in the memory. If it does, we proceed to step E of the diversification mechanism. If not, we go to step A. This technique allows us to escape local optimums to speed up the resolution. The iterative procedure stops when a stopping criteria is reached (after 10 iterations without improvement). The whole scheme is repeated until a maximal number of iterations ($iter2_{max}$) is achieved.

In some cases, the time limit for MILP in step A gives solutions with a high gap value. This leads us to define a variable time limit that increases as soon as there are no cost improvements. Then, returns to its initial value.

In the following subsections, we describe the different steps of the proposed matheuristic.

4.1. MILP with approximated transportation cost

The (step A) is solved by a MILP which minimizes a set of costs as presented in Eq. (26). Those costs are the assembly, disassembly, lost

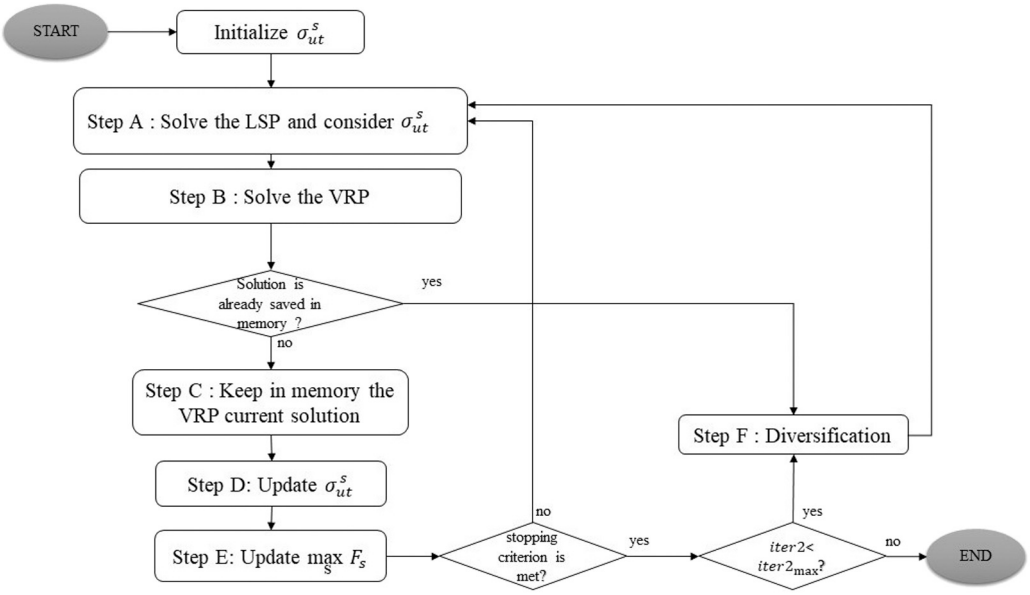

Fig. 2. The general steps of our algorithm.

sales, and inventories costs as well as an approximated transportation costs instead of the transportation costs over all scenarios.

$$\min \quad \Big(\sum_{t\in T}(f_t'(y_t+y_t') + f_t p_t + prl_t + hr_t Ir_t + h_t I_t + max_{s\in S}(\sum_{i\in K} hc_{it} Ic_{it}^s + \sum_{u\in N} ar_t q_{ut}^s + \sum_{i\in K} a_{it} qc_{\psi(i)t}^s + \tau_t^s))\Big)$$
$$\text{s.t (2)–(11)} \tag{26}$$

We note that $\tau_t^s = \sum_{u\in A} \sigma_{ut}^s z_{ut}^s$ is an approximated of the total transportation cost using the node visit variables z_{ut}^s and the approximated visiting cost σ_{ut}^s. This is the crucial link between production and transport problems.

4.2. Variable neighborhood search algorithm

Solving the problem with an approximate transport cost by the MILP allows us to have the set of visited nodes z_{ut}^s in each scenario as well as the purchased quantities of returned product q_{ut} and components qc_{ut}. Therefore, the routing and purchasing decision variables related to each period and each scenario are fixed as parameters for the VNS (Algorithm 1) to solve one VRP for each period and each scenario. The VNS was first proposed by Mladenović and Hansen (1997) and it takes into account a set of k_{max} neighborhoods and executes alternately a local search procedure and a shaking procedure to escape from the local optima. Our VNS starts with an initial solution obtained by applying a random construction heuristic. At each iteration, a random solution is generated from a current neighborhood and then a local search procedure is applied to improve the solution. If the new solution is better than the incumbent solution X_{best}, then the procedure is repeated by reinitializing the neighborhood, otherwise by passing to the next neighborhood . Four neighborhood structures are used both in local search and shaking procedures (see Frifita et al. (2020)):

- ShiftIntra; a node is removed from its current position to another position in the same route.
- Swapintra; two nodes from the same route are swapped.
- ShiftInter; a node is removed from one route to another one.
- Swapinter; two nodes from two different routes are swapped.

The local search strategy following a best improvement strategy. The above-mentioned movements are performed in constant time.

Algorithm 1 Variable Neighborhood Search algorithm

```
X = initial solution from the construction heuristic
X_best = X
repeat
    k = 1
    while k < k_max do
        X_1 = Shake(X, k)
        X_2 = Local search (X_1)
        if f(X_2) < f(X) then
            X_best = X_2
            k = 1
        else
            k++
        end if
    end while
until iter > iter_max
```

4.3. Approximated visiting cost

To update σ_{ut}^s we used two techniques: In the first technique, we suppose that u is a node visited in a route r of period t at scenario s obtained by solving Routing Problem. Let *pre* denote the predecessor of u with the cost c_{upre} and *suc* is its successor with the cost c_{usuc}. M_t^s is the set of nodes $u \in N$ for which $z_{ut}^s = 1$. If $u \in M_t^s$ then $\sigma_{ut}^s = c_{upre} + c_{usuc} - c_{psuc}$, which means that if a node u is eliminated from its current route, an acceptable route can be obtained by connecting the predecessor and successor nodes. Otherwise inserting u into the available vehicle routes in period t at scenario s, so $\sigma_{ut}^s = \Delta_{ut}^s$ where Δ_{ut}^s would be the cheapest insertion cost among all the routes in that period at the correspondent scenario s. Preliminary planning has shown that purchase and pickup of goods was not done in certain periods. So by inserting the node u in an unserved period t, an acceptable route can be obtained. With this technique, we impose visits to the nodes in different periods which can influence productions periods. The second technique divides the total transportation cost of the served period among the visited nodes. Let $cost_t^s$ be the total transportation cost in period t at scenario s, we define $\sigma_{ut}^s = cost_t^s(c_{0u}/\sum_{v\in M_t^s} c_{0v})$ if $u \in M_t^s$

Table 2
Results for a small instance with 4 scenarios.

	s = 1	s = 2	s = 3	s = 4
i = 1	**17 188**	15 902	17 053	16 327
i = 2	**16 570**	15 851	16 465	15 885
i = 3	16 639	15 933	15 775	15 921
i = 4	**16 639**	15 851	15 738	15 977
i = 5	**16 534**	15 851	16 501	15 921
i = 6	15 839	15 968	**16 093**	15 885
i = 7	17 188	15 887	17 053	16 363
i = 8	15 839	15 881	15 738	**15 885**
bestF				15 885

else $\sigma^s_{ut} = (cost^s_t + \Delta_{ut})(c_{0u}/\sum_{v\in M^s_t\cup\{u\}} c_{0v})$. One technique is used for a defined number of iterations ($iter1 = iter1_{max}$), and then we switch to the second one and vice versa. As results, most of the time we get better results compared to using any one of these two techniques alone.

4.4. Update the best worst cost procedure

The transportation solution provided by VNS presents the best set of routes for each scenario with the best costs. In step *E*, a procedure was defined to recalculate the true worst cost for each iteration *i* of the algorithm. In this procedure, we recalculate for each scenario *s* the total cost obtained by the MILP (f^s_i (MILP)) plus the transport cost obtained by the VNS (f^s_i(VNS)). Thus, $f^s_i = f^s_i(MILP) + f^s_i(VNS)$. This procedure allows to change the value of the best worst cost from one iteration to another. For each scenario $s \in \{1, \dots, |S|\}$, we compare two consecutive iterations *i* and *i+1* keeping the minimum value $F_s = \min(f^s_i, f^s_{i+1})$. We recover the maximum value $\max_{s=1..S} F_s$ on all scenarios *S*. For each iteration *i*, we update the worst cost corresponding to the minimum value given by $bestF = \min(\max_{s=1..S} F_s)$. If there is a change, we update the corresponding best solution. In the following example (Table 2), we present the value of the objective function recalculated by our algorithm for a number of scenarios equal to *4* and a number of iterations equal to *8*. As a result, *best F= 15 885* is the worst cost among all the scenarios. The bold numbers in the table show the changes in the $\max_{s=1..S} F_s$ value between iterations.

4.5. Diversification mechanisms

To prevent a quick convergence to the best solution, two diversification mechanisms are used (Absi et al., 2015): a multi-start procedure and an update diversification mechanism. These two diversification mechanisms reinitialize visiting costs σ_{ut} and restarts the iterative procedure. At each iteration restart we randomly chose one of two mechanisms.

The multi-start procedure: reinitializes approximate visiting costs σ^s_{ut} through multiplication with a random value ϵ in [0.5, 1.5] ($\sigma^s_{ut} = (c_{0u} + c_{0u})^*\epsilon$).

The update diversification mechanism modifies σ^s_{ut} according to the best known solution. It consists of multiplying σ^s_{ut} by the number of nodes served M^s_t plus one (to avoid zero multiplication) $\sigma^s_{ut} = |M^s_t|\ \sigma^s_{ut} + 1$.

5. Computational results

The matheuristic is coded in Java with Eclipse Version: 4.9.0. The Cplex library 12.9 is used within a java code to solve the mathematical models. All tests are run on HP desktop computer Intel (R) Core (TM) i7 2.9 GHz with 16 GO of RAM on Windows 10 (64-bit). We note that the statistical tests are performed with *IBM SPSS Statistics* software. The main parameters of the algorithm $iter1_{max}$ and $iter2_{max}$ are carefully tuned as they impact on the quality of the solution and computational time. They are fixed with the Friedman test by using *Minitab.18* software. The results value are equal to 5 for $iter1_{max}$ and 20 for $iter2_{max}$. The time limit for solving each instance by CPLEX is set to one hour (3600 s). For the limit of the execution time for CPLEX in the matheuristic, we impose between 60 and 120 s. The value of the maximum VNS iteration $iter_{max}$ is 100.

5.1. Data sets

To test the efficiency of the proposed model and matheuristic algorithm, we generate a set of instances based on the Chitsaz et al. (2019) benchmark which is adapted from PRP instances of Archetti et al. (2011). The new instances include 6 periods and a number of nodes equal to 14 and 50. These nodes present the suppliers or warehouses. These instances keep the original horizon of production capacity, and the distances between the nodes. For the inventory limits at the plant, we chose to share it between return products, new products, and components. A dynamic demand per period is randomly generated by uniform distribution between $0.5 * D$ and $1.5 * D$ where D is the demand per period in instances of Chitsaz et al. (2019). A dynamic setup, purchase, inventory and production costs are randomly generated by uniform distribution ±30% of the values obtained from instances of Chitsaz et al. (2019). For each instance, we generate a number of scenarios equal to 5, 50 and 100 that represent the random yields. In each instance, we check the non-redundancy of the scenarios.

5.2. Managerial insights

In order to identify the impact of the different hypotheses explained in Section 3.3 a set of tests is performed. Tables 3 and 4 present the results and the deviations *Dev0(%)* between the optimal solutions obtained under the hypothesis H_0 and the other hypotheses H_x ($Dev0(\%) = \frac{sol.H_0 - sol.H_x}{sol.H_x}$) and the average (Avg) also the median (*Med*) of those Deviations.

We start our experimentation by comparing the results under five scenarios for 10 instances with 14 nodes. According to the results shown in Table 3, it can be observed that solving the problem with H_0 is faster than with the remaining hypotheses because there are fewer decisions that are impacted by the scenarios. Consequently, with H_0 also for H_1 and H_2, the problem is solved to the optimal.

Comparing the results of H_0 with that of H_1 show that adding flexibility to production improves slightly the total cost ($Avg\_Dev0 = 0.10\%$), where the produced quantity with H_1 remains unchanged even if it depends on the scenarios. However, the quantity to be disassembled changed. Solving the problem with H_1 allows us to find the majority of the same optimal solutions founded by H_0 (9 instances among the 10) but with a higher CPU time.

Moreover, adding flexibility to the purchased quantities with H_2 improves the results of H_0 with $Avg\_Dev0 = 5.70\%$. Comparing the results of H_0 and H_3 shows a significant improvement in the total cost. This improvement can reach 22.80%. With the two hypotheses H_4 and H_3, we find the majority of the same solutions but with different CPU times. Indeed, the resolution with H_4 is faster in most cases in comparison with H_3.

Based on the deviation results presented in Table 4, H_1 slightly improves H_0. Conversely, H_2, H_3 and H_4 significantly improve H_0. These improvements are practically equivalent ($Avg\_Dev0 = 6.30\%$ and $Avg\_Dev0 = 6.40\%$). In contrast, having more flexibilities in the decisions improves the total cost in each hypothesis, but the difficulty of solving the problem increases and need more CPU time. Therefore, when solving the problem with H_3 and H_4, the majority of the instances are not solved to the optimal.

Following this observation, we can neglect H_0 and H_1 and solve more instances of different sizes with the last 3 hypotheses.

Table 3
Results for instances with 14 nodes under 5 scenarios.

Instance	H_0		H_1		H_2		H_3		H_4	
	CPU (s)	Gap (%)	CPU (s)	Gap (%)	CPU (s)	Gap (%)	CPU (s)	Gap (%)	CPU (s)	Gap (%)
1	4	0.0	5	0.0	77	0.0	3600	0.2	3600	0.2
2	18	0.0	28	0.0	36	0.0	3600	0.2	2830	0.0
3	69	0.0	69	0.0	7	0.0	75	0.0	52	0.0
4	12	0.0	24	0.0	76	0.0	1088	0.0	546	0.0
5	90	0.0	85	0.0	136	0.0	1217	0.0	426	0.0
6	12	0.0	23	0.0	13	0.0	2017	0.0	1676	0.0
7	22	0.0	48	0.0	115	0.0	3600	0.5	3600	0.3
8	32	0.0	55	0.0	35	0.0	3600	0.1	864	0.0
9	27	0.0	32	0.0	204	0.0	3600	0.8	3600	0.8
10	55	0.0	127	0.0	191	0.0	1455	0.0	1594	0.0
Med	25	0.0	40	0.0	77	0.0	2809	0.1	1635	0.0
Avg	34	0.0	49	0.0	89	0.0	2385	0.2	1879	0.1

Table 4
Deviations of solutions for instances with 14 nodes under 5 scenarios.

Instance	Dev0(H_0/H_1)	Dev0(H_0/H_2)	Dev0(H_0/H_3)	Dev0(H_0/H_4)
1	0.0	2.4	2.5	2.5
2	0.0	2.0	2.9	3.1
3	0.0	21.7	22.8	22.8
4	0.0	5.4	6.8	6.9
5	0.0	7.5	7.6	7.6
6	0.0	6.6	7.0	7.0
7	0.0	4.1	4.1	4.3
8	0.0	5.7	7.7	7.8
9	0.8	0.0	0.4	0.8
10	0.0	1.4	1.5	1.5
Med	0.00	4.75	5.45	5.60
Avg	0.10	5.70	6.30	6.40

5.3. Experimental results

In this section, the numerical results obtained from the MILP, the matheuristic algorithm as well as from the *MILP&WS* are explained. Note that a total of 200 instances with 14 and 50 components under 5, 50 and 100 scenarios are considered to compare the results between H_2, H_3 and H_4.

We summarize in Tables 5, 6, 7 and 8, the results for the H_2, H_3 and H_4 for each group of instances (instances with 14 and 50 nodes). Tables from A.13 to A.22 in Appendix give the detailed numerical results.

In our preliminary experiments, we find that the solver cannot find optimal solutions for instances with 50 nodes under 50 scenarios after 3600 s. Thus, we warm start the MILP with the best solution given by the matheuristic approach. This technique is chosen to evaluate the quality of the matheuristic solution and improve it when possible. Using warm start helps also to reduce the solution CPU time.

In those tables, column *MILP* gives the best solution obtained using Cplex. Column *Mat* presents the best solutions of the matheuristic algorithm. Column *MILP&WS* presents the solutions of the MILP with an initial solution from the matheuristic algorithm. Columns *CPU(s)* show the computing time in seconds.

Column *Gap(%)* shows the gap given by CLPEX in 3600 s of computation time. Column *Dev(%)* presents the deviation between the best know solution from H_2 (Bks_{H_2}) and the best know solution from H_3 (Bks_{H_3}) ($Dev(\%) = \frac{Bks_{H_2} - Bks_{H_3}}{Bks_{H_3}}$). Column *Dev1(%)* presents the deviation between the best know solution from H_2 (Bks_{H_2}) and the best know solution from H_4 (Bks_{H_4}) ($Dev1(\%) = \frac{Bks_{H_2} - Bks_{H_4}}{Bks_{H_4}}$).

The CPU times for *MILP&WS* are calculated as the sum of the CPU time for finding the initial solution plus the CPU time of the resolution with *MILP&WS*. For all tables the minimum (*Min*), the maximum (*Max*), the average (*Avg*), and the median (*Med*) values are presented. In all tables, negative deviations values indicate an improvement provided by H_2.

Table 5 shows the summary of the CPU time and the Gap of 20 instances with 14 nodes and 10 instances with 50 nodes under 5 scenarios.

For the instances with 14 nodes, MILP can prove optimality on all instances with H_2. However, it can prove optimality only for 11 of the 20 instances within the time limit with a $Max\_gap = 2.5\%$ and $Avg\_Gap = 0.4\%$ for H_3. For H_4, it can prove optimality only for 13 of the 20 instances within the time limit with a $Max\_gap = 1.7\%$ and $Avg\_Gap = 0.3\%$.

As can be seen from Tables A.13, A.14 and A.15, the *MILP&WS* is capable to find 20 optimal solutions for all hypotheses. The matheuristic converge to the same results for almost of instances for the three hypotheses.

For all hypotheses, the matheuristic algorithm and the *MILP&WS* are more efficient than MILP. The matheuristic algorithm can find a good solution very fast with an $Avg\_CPU = 20$ s for H_2, $Avg\_CPU = 64$ s for H_3 and $Avg\_CPU = 59$ s for H_4.

The results for the instances with 50 nodes state that the difficulty of resolution increases with the nodes size, for H_2 the $Avg\_Gap = 40.7\%$, for H_3 the $Avg\_Gap = 70.1\%$ and for H_4 the $Avg\_Gap = 56.4\%$. From these instances, the Avg_Gap is tighter for H_2 than for H_3 and H_4.

The solution obtained for H_2 by *MILP&WS* leads to an improvement in the gap (compared to the MILP) of 0.7% on average, and 1.5% as the maximal value of gap. For all instances, *MILP&WS* finds the best solutions in comparison with the other methods but with larger CPU times, which is equal to the sum of the CPU time of the matheuristic algorithm and the *MILP*.

To further assess the performance of the proposed methods, we executed the second round of tests on larger sized instances under 50 scenarios. For H_2, the average gaps for the MILP and the *MILP&WS* obtained for 20 instances with 14 nodes under 50 scenarios are 7.4% and 0.0%, respectively. As can be seen from Table A.16, the *MILP* is not able to find any feasible solution for the 10 instances with 50 nodes under 50 scenarios. However, the *MILP&WS* can solve the 20 instances with $Avg\_Gap = 3.6\%$. The results show that the matheuristic algorithm can find good solutions, which are improved by the *MILP&WS*. For H_3, the *MILP&WS* improves the solutions found by the MILP with an $Avg\_Gap = 0.1\%$ for instances with 14 nodes. The results show that the MILP has the worst performance; it is far from the optimal solution with an $Avg\_Gap = 77.1\%$ for the instances with 14 nodes. In addition, it cannot find any feasible solution for instances with 50 nodes (see Table A.17), where the *MILP&WS* proves its efficiency with an $Avg\_Gap = 3.9\%$.

For H_4, the *MILP&WS* improves the solutions found by the MILP with an $Avg\_Gap = 0.5\%$ for instances with 14 nodes and with an $Avg\_Gap = 4\%$ for instances with 50 nodes. The results show that the MILP has the worst performance; it is far from the optimal solution with an $Avg\_Gap = 73.4\%$ for the instances with 14 nodes. In addition, it cannot find any feasible solution for instances with 50 nodes (see Table A.18).

Table 5
Computational performance for instances with 14 and 50 nodes under 5 scenarios.

		H_2					H_3					H_4				
		MILP		Mat	MILP&WS		MILP		Mat	MILP&WS		MILP		Mat	MILP&WS	
		CPU (s)	Gap (%)	CPU (s)	CPU (s)	Gap (%)	CPU (s)	Gap (%)	CPU (s)	CPU (s)	Gap (%)	CPU (s)	Gap (%)	CPU (s)	CPU (s)	Gap (%)
14	Max	2084	0.00	52	212	0.00	3600	2.50	136	921	0.00	3600	1.70	141	962	0.00
	Min	7	0.00	5	14	0.00	75	0.00	14	82	0.00	52	0.00	11	136	0.00
	Med	70	0.00	13	81	0.00	2822	0.00	59	406	0.00	2035	0.00	44	374	0.00
	Avg	185	0.00	20	96	0.00	2502	0.40	64	436	0.00	2143	0.30	59	463	0.00
50	Max	3600	85.30	951	4551	1.50	3600	99.80	421	4021	2.50	3600	99.20	1036	4636	5.70
	Min	3600	6.00	40	3640	0.20	3600	17.50	65	3665	0.30	3600	29.00	62	3662	0.80
	Med	3600	20.90	90	3691	0.80	3600	98.80	226	3826	1.30	3600	41.20	277	3878	2.30
	Avg	3600	40.70	199	3799	0.70	3600	70.10	230	3830	1.30	3600	56.40	487	4087	2.90

Table 6
Computational performance for instances with 14 and 50 nodes under 50 scenarios.

		H_2					H_3					H_4				
		MILP		Mat	MILP&WS		MILP		Mat	MILP&WS		MILP		Mat	MILP&WS	
		CPU (s)	Gap (%)	CPU (s)	CPU (s)	Gap (%)	CPU (s)	Gap (%)	CPU (s)	CPU (s)	Gap (%)	CPU (s)	Gap (%)	CPU (s)	CPU (s)	Gap (%)
14	Max	3600	18.1	948	4548	0.4	3600	99.5	985	4585	0.8	3600	99.1	549	4149	2.5
	Min	3600	0.8	29	3629	0.0	3600	12.6	83	3683	0.0	3600	1.6	124	3724	0.0
	Med	3600	6.7	161	3761	0.0	3600	93.6	315	3915	0.0	3600	95.1	217	3817	0.3
	Avg	3600	7.4	298	3899	0.0	3600	77.1	459	4059	0.1	3600	73.4	253	3853	0.5
50	Max	–	–	975	4575	9.8	–	–	2095	5695	6.2	–	–	3509	7109	7.8
	Min	–	–	342	3942	0.2	–	–	821	4421	0.7	–	–	421	4021	0.7
	Med	–	–	502	4102	2.3	–	–	1003	4603	3.6	–	–	1455	5055	3.6
	Avg	–	–	629	4229	3.6	–	–	1181	4781	3.9	–	–	1616	5216	4.0

(–): No feasible solution is available.

Table 7
Computational performance for instances with 14 nodes under 100 scenarios.

		H_2					H_3					H_4				
		MILP		Mat	MILP&WS		MILP		Mat	MILP&WS		MILP		Mat	MILP&WS	
		CPU (s)	Gap (%)	CPU (s)	CPU (s)	Gap (%)	CPU (s)	Gap (%)	CPU (s)	CPU (s)	Gap (%)	CPU (s)	Gap (%)	CPU (s)	CPU (s)	Gap (%)
14	Max	3600	45.9	975	4575	3.3	3600	99.9	1369	4969	6.3	3600	99.7	2543	6143	6.5
	Min	3600	0.2	125	3725	0.0	3600	37.0	145	3745	0.0	3600	66.7	169	3769	1.1
	Med	3600	9.8	346	3946	0.9	3600	94.8	698	4298	3.1	3600	99.0	567	4167	3.7
	Avg	3600	13.5	478	4078	1.0	3600	55.9	705	4305	3.1	3600	77.0	695	4300	3.1

Table 8
Deviations for instances with 14 and 50 nodes under all scenarios.

		5		50		100	
		Dev (%)	Dev1 (%)	Dev (%)	Dev1 (%)	Dev (%)	Dev1 (%)
14	Max	3.20	3.20	16.2	17.0	8.8	15.0
	Min	0.00	0.00	0.0	0.6	−3.4	−0.9
	Med	1.20	1.20	1.4	3.0	1.4	7.0
	Avg	1.10	1.20	3.1	3.7	1.4	4.8
50	Max	10.90	12.00	11.6	14.2	–	–
	Min	0.00	1.10	0.2	2.3	–	–
	Med	2.60	3.00	1.8	4.6	–	–
	Avg	3.10	3.70	3.2	5.5	–	–

(–): No feasible solution is available.

Table 9
Correlation between the CPU times, the Nodes, the Gap for each hypothesis.

Correlations		CPU	Hypotheses	Nodes	Gap
Pearson correlation	CPU	1,000	0,081	0,986	0,718
	Hypotheses	0,081	1,000	0,000	0,264
	Nodes	0,986	0,000	1,000	0,686
	Gap	0,718	0,264	0,686	1,000
P-value (1-tailed)	CPU	–	0,223	0,000	0,000
	Hypotheses	0,223	–	0,500	0,006
	Nodes	0,000	0,500	–	0,000
	Gap	0,000	0,006	0,000	–

Table 10
Regression model.

Model	R	R square	Adjusted R square	Std. Error of the estimate
1	0,990	0,979	0,979	251,658

Predictors: (Constant), Gap, Hypotheses, Nodes
Dependent variable: Cpu

Table 7 summarize the results for 20 instances with 14 nodes under 100 scenarios. For these 20 cases, the performance of the proposed *MILP&WS* is superior to that of the other methods (see Tables A.19, A.20 and A.21).

For H_2, the average gap found by MILP ($Avg\_Gap = 13.5\%$) is higher than that of *MILP&WS* ($Avg\_Gap = 1\%$). For H_3, the average gap found by MILP ($Avg\_Gap = 55.9\%$) is higher than that of *MILP&WS* ($Avg\_Gap = 3.1\%$). For H_4, the improvement of *MILP&WS* is significant, it reduces the Gap cplex from ($Avg\_Gap = 77\%$) to ($Avg\_Gap = 3.1\%$).

The difficulty of solving the problem increases with 100 scenarios for the MILP with H_3 and H_4. Therefore, in 7 and 4 out of 20 cases, the MILP cannot find a feasible solution for H_3 and H_4, respectively. In some instances, an initial solution is not sufficient to speed up the optimization to the respective limits, and the solver spends unnecessary time on estimation. An initial solution presents an upper limit with an objective function value close to the optimum but in many cases, it makes no difference. That is why the *MILP&WS* was not able to find a feasible solution for 3 instances out of the 20 instances (Table A.21).

Table 8 presents a summary on the deviations between H_2 and H_3 also between H_2 and H_4 for the different scenarios. According to these results, we notice that for 14 nodes with 5 scenarios, H_3 improves H_2

Table 11
ANOVA tests.

	Sum of squares	df	Mean square	F	P-value
Between Groups	28 942 957 190 581,300	2	14 471 478 595 290,600	6,561	0,002
Within Groups	586 682 532 907 323,000	266	2 205 573 431 982,420	–	–
Total	615 625 490 097 905,000	268	–	–	–

Table 12
Dunn's pairwise Post-Hoc tests.

Methods		Mean difference	Std. Error	P-value
MILP	MILPWS	697 188,63348*	222 009,22373	0,004
Mat	MILPWS	109,05200	221 388,21769	1,000

*The mean difference is significant at the 0.05 level.

at most by 3.20% and on average 1.1%. The hypothesis H_4 improves H_2 at most by 3.20% and on average 1.2%. Concerning the results with 50 scenarios, we observe that H_3 improves at most H_2 by 16.20% and on average 3.1%, and H_4 improves at most H_2 by 17% and on average 3.7%. For the results with 100 scenarios, H_3 and H_4 improve H_2 on average by 1.4% and 4.8%, respectively. Similarly, with 50 nodes and different scenarios, the results are in line with the previous findings (Table A.22).

5.4. Results analysis and statistical tests

In the previous section, we reported the results of different resolution methods for each hypothesis. The analysis of these results allows us to draw conclusions about the assumptions and the resolution methods used. Before analyzing the impacts of different levels of flexibility, we need to reassure the reliability of the resolution methods used.

For each hypothesis, we compared the results of PLNE with the matheuristic and *MILP&WS*. The results show that matheuristics can find good solutions. The performance of our method is reflected in the correct selection of different steps. Even if the studied problem has different costs and capacity constraints, our iterative method can easily prove its effectiveness. It is simple to adjust to solve all the hypotheses under different scenarios size, knowing that each hypothesis has its own characteristic. The memorization process inspired by the tabu search used in our method allows us to avoid finding solutions that have already been exploited. This technique allows us to escape from local optimums to speed up the resolution. That is why our matheuristic algorithm was able to find a good solution faster than Cplex. We also implement two transportation cost approximation mechanisms schedule where we switch between them in order to diversify the search.

The matheuristics presented in this work can be adapted and applied to many variants of integrated transportation and production management problems that consider distribution as part of the decision-making process. Moreover, we can add the energy constraints in the production process by integrating the problem of selection of energy sources.

One of the main contributions of this paper is the study of the impact of different flexibility levels given to decisions related to production, transportation and procurement activities. The analyses indicate that the total cost improves with the addition of flexibility to the different decisions but with a longer resolution time. On the other hand, with some flexibility given only to decisions on quantities purchased, we can guarantee that the production and routing schedules are not impacted by the changes in scenarios. Based on this managerial study we can conclude that giving flexibility only to purchasing decisions is the fastest and easiest to implement in the industry.

The analysis of results indicate that the difficulty of solving instances increases with node size and different flexibility levels. To statistically determine if there is a relationship between CPU times, deviation and node size for each hypothesis, a *correlation* analysis is applied.

Table 9 shows the results of the correlation tests. The correlation coefficients indicate that the CPU times are perfectly related in a positive linear sense with the nodes size (coefficient = 0,986) and the gap given by CPLEX (coefficient = 0,718). This point is already highlighted in the previous analyses which show that the size of the instances increases the difficulty of solving the problem. This implies a higher CPU times and a higher gap. To know how the increase of the size of the node and the gap affects the CPU times a multiple linear regression model is realized (Table 10). The coefficient of determination (R-squared = 0,979) indicates that the nodes size and the gap significantly contribute to 98% of the variability of the CPU times.

Another observation made from the analysis of results is that the matheuristic and *MILP&WS* have almost the same performance. To test whether the difference in performance between the proposed methods is significant, *ANOVA Tests* and *Dunn's pairwise Post-Hoc Tests* are applied. We consider instances with 5 scenarios where we get feasible solutions almost the time.

We examine the hypothesis H0 versus H1 as follows:

- H0: The methods have identical performances.
- H1: At least one method is different.

The results from the ANOVA Tests (Table 11) reject H0 with a *p-value* = 0.002 that is lower than the level of significance = 0.05. So at least one method is more performed from the other. Thus, we applied *Dunn's pairwise Post-Hoc* Tests (Table 12). The results prove that a significant difference exists between the *MILP&WS* and the MILP with a *p-value* = 0.004. However, the *MILP&WS* and the matheuristic have the same performances with a *p-value* higher than the level of significance.

6. Conclusion

In this paper, we study a dynamic disassembly assembly routing problem with returns under uncertain yields. This problem integrates simultaneous assembly and disassembly operations for the first time with vehicle routing to minimize various aggregate costs. This work allows the recovery of products at the end of their life by collecting and reusing their components. Especially, it considers the stochastic nature of yields to help managers in optimizing their supply chain network based on available yields scenarios.

Initially, we formulate the SDARP-R as a minimax mixed-integer linear programming model with a two-commodity flow formulation. Secondly, an efficient matheuristic is developed based on a variable neighborhood search algorithm and a *MILP*. In the first phase of the solution algorithm, a lot-sizing problem is solved with approximate routing costs. The second phase handles the dates approximate routing costs. The third phase presents a procedure to recalculate the true worst cost. The model and the matheuristic are tested on a set of instances derived from the literature and modified accordingly.

We started our experimentation by adding flexibility to the decisions considered in our problem. As a result, we found that adding flexibility in procurement, which includes the quantities purchased and routing decisions, improve the total cost. However, allowing flexibility only to purchasing decisions is most practical to the industry.

We compare the performance of the *MILP* with the matheuristic algorithm and the *MILP&WS*. According to the results, the matheuristic approach is faster than the other methods and provides good quality

Table A.13
Results for instances under 5 scenarios for H_2.

Nodes	Instance	MILP			Mat		MILP&WS		
		Obj	CPU (s)	Gap (%)	Obj	CPU (s)	Obj	CPU (s)	Gap (%)
14	1	15 890.09	77.20	0.0	15 890.09	8	15 890.09	82	0.0
	2	19 923.87	36.21	0.0	19 923.87	52	19 923.87	74	0.0
	3	17 218.15	6.70	0.0	17 218.15	9	17 218.15	14	0.0
	4	18 590.95	75.66	0.0	18 590.95	10	18 590.95	67	0.0
	5	14 348.89	135.83	0.0	14 348.89	9	14 348.89	161	0.0
	6	16 419.54	13.03	0.0	16 419.54	13	16 419.54	24	0.0
	7	25 596.61	114.74	0.0	25 596.61	45	25 596.61	169	0.0
	8	16 198.59	35.16	0.0	16 198.59	31	16 198.59	59	0.0
	9	24 556.68	204.29	0.0	24 556.68	17	24 556.68	173	0.0
	10	19 835.72	190.51	0.0	19 835.72	13	19 835.72	80	0.0
	11	9940.03	63.35	0.0	9940.03	6	9940.03	92	0.0
	12	14 530.81	178.39	0.0	14 530.81	28	14 530.81	212	0.0
	13	13 574.99	2084.08	0.0	13 574.99	31	13 574.99	116	0.0
	14	14 396.17	106.45	0.0	14 396.17	9	14 396.17	99	0.0
	15	16 391.04	204.48	0.0	16 391.04	40	16 391.04	115	0.0
	16	15 903.75	23.30	0.0	15 903.75	18	15 903.75	170	0.0
	17	10 690.69	32.26	0.0	10 690.69	8	10 690.69	35	0.0
	18	11 808.19	45.86	0.0	11 808.19	42	11 808.19	78	0.0
	19	15 222.69	20.90	0.0	15 222.69	5	15 222.69	25	0.0
	20	10 600.58	50.06	0.0	10 600.58	8	10 600.58	81	0.0
50	1	67 697.96	3600	53.7	31 984.11	304	31 681.17	3904	1.1
	2	38 262.00	3600	20.3	31 137.17	178	30 960.20	3778	1.5
	3	45 523.74	3600	16.9	38 098.20	71	38 070.75	3671	0.6
	4	42 752.64	3600	21.4	33 714.79	98	33 661.21	3698	0.2
	5	251 852.95	3600	82.6	43 990.34	951	43 852.06	4551	0.2
	6	42 021.93	3600	6.0	40 047.79	40	39 859.68	3640	0.9
	7	155 986.38	3600	82.9	27 029.76	57	26 904.26	3657	0.9
	8	51 307.91	3600	20.0	41 446.58	137	41 217.81	3737	0.4
	9	58 893.32	3600	18.2	49 094.89	83	48 682.31	3683	1.0
	10	329 672.55	3600	85.3	48 682.54	72	48 537.80	3672	0.4

Table A.14
Results for instances under 5 scenarios for H_3.

Nodes	Instance	MILP			Mat		MILP&WS		
		Obj	CPU (s)	Gap (%)	Obj	CPU (s)	Obj	CPU (s)	Gap (%)
14	1	15 886.72	3600	0.2	15 885.04	21	15 851.07	832	0.0
	2	19 740.79	3600	0.2	19 740.79	18	19 700.85	136	0.0
	3	16 987.57	75	0.0	16 987.57	21	16 987.57	87	0.0
	4	18 321.36	1088	0.0	18 321.36	17	18 321.36	82	0.0
	5	14 330.78	1217	0.0	14 330.78	14	14 330.42	402	0.0
	6	16 359.54	2017	0.0	16 359.54	136	16 357.93	204	0.0
	7	25 601.59	3600	0.5	25 555.55	115	25 475.37	520	0.0
	8	15 852.70	3600	0.1	15 837.55	58	15 837.55	246	0.0
	9	24 457.25	3600	0.8	24 318.38	26	24 268.61	755	0.0
	10	19 825.86	1455	0.0	19 827.72	38	19 825.86	118	0.0
	11	9914.63	827	0.0	9914.63	78	9914.63	632	0.0
	12	14 564.27	3600	1.6	14 356.38	135	14 334.7	547	0.0
	13	13 599.92	3600	2.5	13 330.36	96	13 259.72	200	0.0
	14	14 298.02	3600	0.4	14 295.95	61	14 240.95	410	0.0
	15	16 387.27	3600	1.7	16 187.27	54	16 108.93	109	0.0
	16	15 416.78	1256	0.0	15 416.78	55	15 416.78	800	0.0
	17	10 690.69	2038	0.0	10 690.69	88	10 690.69	892	0.0
	18	11 599.33	3491	0.0	11 600.28	113	11 599.33	921	0.0
	19	15 221.18	2030	0.0	15 222.69	77	15 221.18	363	0.0
	20	10 345.74	2154	0.0	10 345.74	67	10 345.74	467	0.0
50	1	9 473 842.14	3600	99.7	31 922.27	330	30 880.27	3930	0.6
	2	42 969.51	3600	29.3	31 120.25	236	30 946.70	3836	1.8
	3	45 535.31	3600	17.5	37 719.66	65	37 700.95	3665	0.3
	4	42 919.54	3600	30.0	31 213.96	236	30 360.96	3836	1.1
	5	19 658 169.33	3600	99.8	42 643.16	421	41 423.63	4021	1.9
	6	4 747 777.25	3600	99.2	39 918.52	86	38 812.32	3686	0.6
	7	3 960 572.04	3600	99.3	26 880.29	136	26 880.29	3736	1.4
	8	54 826.89	3600	28.4	40 044.14	216	40 044.14	3816	1.9
	9	7 595 305.48	3600	99.4	48 203.29	154	47 723.29	3754	2.5
	10	2 802 442.90	3600	98.3	47 429.04	416	47 126.83	4016	0.9

solutions on the small and medium test instances. However, for all hypothesis, the difficulty of solving the problem increases with 14 nodes under 100 scenarios. For these instances, the matheuristic algorithm presents acceptable results but with a CPU time how reach 2543 s because of the additional difficulty induced by capacity constraints in the assembly and disassembly model that is solved with MILP. One of the main perspectives is to develop a fast and effective heuristic for the assembly and disassembly phase. The *MILP&WS* is more efficient

Table A.15
Results for instances under 5 scenarios for H_4.

Nodes	Instance	MILP			Mat		MILP&WS		
		Obj	CPU (s)	Gap (%)	Obj	CPU (s)	Obj	CPU (s)	Gap (%)
14	1	15 886.72	3600	0.2	15 851.07	50	15 851.07	353	0.0
	2	19 700.85	2830	0.0	19 700.85	37	19 700.85	338	0.0
	3	16 987.57	52	0.0	16 987.57	11	16 987.57	136	0.0
	4	18 301.36	546	0.0	18 301.12	20	18 301.12	267	0.0
	5	14 330.78	426	0.0	14 330.78	27	14 330.78	621	0.0
	6	16 357.93	1676	0.0	16 357.93	16	16 357.93	441	0.0
	7	25 555.55	3600	0.3	25 475.37	31	25 475.37	951	0.0
	8	15 837.55	864	0.0	15 837.55	16	15 837.55	565	0.0
	9	24 357.25	3600	0.8	24 157.25	35	24 157.25	380	0.0
	10	19 825.86	1594	0.0	19 825.86	68	19 825.86	831	0.0
	11	9914.63	1442	0.0	9914.63	54	9914.63	599	0.0
	12	14 520.12	3600	1.3	14 334.7	141	14 334.7	357	0.0
	13	13 372.58	3600	0.9	13 259.72	104	13 259.72	400	0.0
	14	14 240.95	2393	0.0	14 240.95	16	14 240.95	305	0.0
	15	16 387.27	3600	1.7	16 108.93	136	16 108.93	336	0.0
	16	15 416.78	702	0.0	15 416.78	19	15 416.78	238	0.0
	17	10 690.69	3099	0.0	10 690.69	118	10 690.69	619	0.0
	18	11 599.33	1131	0.0	11 599.33	68	11 599.33	368	0.0
	19	15 221.18	912	0.0	15 221.18	105	15 221.18	186	0.0
	20	10 420.95	3600	0.7	10 345.74	104	10 345.74	962	0.0
50	1	1 160 964.09	3600	97.4	31 176.45	104	31 016.06	3704	1.2
	2	–	3600	–	31 050.07	1014	30 623.07	4614	5.0
	3	62 567.11	3600	41.2	37 126.56	102	37 072.01	3702	0.8
	4	43 878.50	3600	32.7	30 192.58	242	30 041.98	3842	1.7
	5	3 021 857.05	3600	98.7	42 184.22	943	41 936.50	4543	4.2
	6	4 673 638.09	3600	99.2	39 146.62	313	38 440.62	3913	5.7
	7	36 197.70	3600	29.0	26 419.41	62	26 067.38	3662	1.4
	8	4 504 898.81	3600	99.1	40 011.24	1036	39 965.59	4636	2.0
	9	68 861.18	3600	33.3	48 137.93	945	48 062.04	4545	4.4
	10	68 927.42	3600	33.2	47 376.46	109	47 211.54	3709	2.5

(–): No feasible solution is available.

Table A.16
Results for instances under 50 scenarios for H_2.

Nodes	Instance	MILP			Mat		MILP&WS		
		Obj	CPU (s)	Gap (%)	Obj	CPU (s)	Obj	CPU (s)	Gap (%)
14	1	19 981.20	3600	16.0	16 810.11	87	16 789.55	3687	0.0
	2	20 160.15	3600	1.1	19 958.04	117	19 947.95	3717	0.0
	3	19 815.76	3600	2.0	19 422.62	100	19 419.75	3700	0.0
	4	20 922.23	3600	5.7	19 732.99	278	19 732.39	3878	0.0
	5	16 651.64	3600	2.9	16 206.04	101	16 162.89	3701	0.0
	6	16 837.82	3600	1.1	17 016.33	176	16 648.23	3776	0.0
	7	28 135.71	3600	7.8	25 988.82	274	25 944.12	3874	0.0
	8	17 725.01	3600	7.9	16 486.59	41	16 331.32	3641	0.0
	9	28 173.56	3600	12.2	24 883.25	475	24 724.45	4075	0.0
	10	25 540.05	3600	18.1	20 925.04	29	20 919.64	3629	0.0
	11	13 768.58	3600	13.7	11 892.68	452	11 883.57	4052	0.0
	12	17 347.99	3600	13.5	15 043.23	667	15 038.14	4267	0.2
	13	15 580.06	3600	4.9	14 849.77	758	14 823.62	4358	0.0
	14	15 642.96	3600	7.8	14 503.18	205	14 427.61	3805	0.0
	15	20 218.13	3600	12.7	17 738.99	84	17 669.94	3684	0.1
	16	16 112.66	3600	0.8	15 976.35	948	15 976.63	4548	0.0
	17	12 666.45	3600	5.1	12 019.96	96	12 014.62	3696	0.0
	18	13 277.56	3600	3.7	12 786.89	842	12 786.98	4442	0.0
	19	17 437.40	3600	7.6	16 216.15	94	16 176.73	3694	0.4
	20	11 205.24	3600	3.6	10 907.54	146	10 800.43	3746	0.0
50	1	–	–	–	32 976.54	975	32 851.54	4575	4.4
	2	–	–	–	32 816.51	912	32 757.05	4512	1.7
	3	–	–	–	41 237.59	451	41 062.72	4051	1.3
	4	–	–	–	34 856.67	543	34 712.58	4143	2.4
	5	–	–	–	45 168.79	342	44 946.98	3942	1.5
	6	–	–	–	40 836.47	943	40 775.47	4543	2.1
	7	–	–	–	49 716.71	862	47 624.37	4462	6.0
	8	–	–	–	51 156.71	461	50 325.16	4061	9.8
	9	–	–	–	51 856.84	346	51 629.49	3946	6.3
	10	–	–	–	50 852.76	456	50 803.25	4056	0.2

(–): No feasible solution is available.

Table A.17
Results for instances under 50 scenarios for H_3.

Nodes	Instance	MILP			Mat		MILP&WS		
		Obj	CPU (s)	Gap (%)	Obj	CPU (s)	Obj	CPU (s)	Gap (%)
14	1	70 865.29	3600	77.4	16 048.64	245	16 048.64	3845	0.0
	2	28 452.41	3600	29.9	19 977.91	342	19 937.21	3942	0.0
	3	22 095.58	3600	12.6	19 320.40	83	19 320.40	3683	0.0
	4	28 304.94	3600	31.3	19 444.64	145	19 444.64	3745	0.0
	5	66 314.95	3600	77.0	15 254.37	256	15 254.37	3856	0.0
	6	1 783 766.85	3600	99.1	16 646.44	256	16 646.44	3856	0.0
	7	1 497 790.91	3600	98.3	26 002.91	259	25 942.87	3859	0.8
	8	32 921.71	3600	51.1	16 114.29	259	16 114.29	3859	0.0
	9	1 702 665.52	3600	98.6	24 572.13	123	24 495.43	3723	0.1
	10	1 387 092.73	3600	98.5	20 803.57	562	20 716.75	4162	0.5
	11	191 392.61	3600	94.7	10 224.36	287	10 224.36	3887	0.0
	12	953 120.43	3600	98.5	14 667.05	216	14 667.05	3816	0.0
	13	887 125.25	3600	98.4	14 367.02	356	14 367.02	3956	0.0
	14	28 139.80	3600	49.2	14 304.21	945	14 304.21	4545	0.0
	15	1 687 268.28	3600	99.0	16 285.45	961	16 285.45	4561	0.0
	16	211 019.89	3600	92.5	15 884.45	923	15 884.45	4523	0.0
	17	137 249.10	3600	91.3	11 995.51	945	11 995.51	4545	0.0
	18	22 741.81	3600	47.6	11 921.23	985	11 921.23	4585	0.0
	19	864 089.85	3600	98.2	15 622.96	425	15 491.76	4025	0.1
	20	2 318 123.09	3600	99.5	10 518.24	612	10 518.24	4212	0.0
50	1	–	–	–	32 546.02	1845	32 201.53	5445	3.4
	2	–	–	–	32 469.31	1256	32 469.31	4856	5.7
	3	–	–	–	40 436.28	845	40 436.28	4445	2.1
	4	–	–	–	31 107.30	975	31 107.30	4575	3.1
	5	–	–	–	42 523.38	1024	42 523.38	4624	0.7
	6	–	–	–	40 654.88	983	40 654.88	4583	3.7
	7	–	–	–	46 001.11	2095	45 601.11	5695	6.2
	8	–	–	–	50 019.67	945	48 319.67	4545	6.1
	9	–	–	–	51 521.22	1023	51 521.22	4623	6.2
	10	–	–	–	50 774.46	821	50 094.46	4421	1.5

(–): No feasible solution is available.

Table A.18
Results for instances under 50 scenarios for H_4.

Nodes	Instance	MILP			Mat		MILP&WS		
		Obj	CPU (s)	Gap (%)	Obj	CPU (s)	Obj	CPU (s)	Gap (%)
14	1	1 523 511.73	3600	99.0	16 036.79	213	15 902.38	3813	0.1
	2	25 825.30	3600	23.5	19 876.21	218	19 804.15	3818	0.2
	3	19 630.64	3600	1.6	19 319.23	546	19 306.79	4146	0.0
	4	28 217.27	3600	32.7	19 364.57	256	19 064.73	3856	0.4
	5	23 991.19	3600	38.0	15 250.46	216	15 249.42	3816	2.5
	6	1 484 584.59	3600	98.9	16 645.44	549	16 550.98	4149	0.1
	7	2 214 403.04	3600	98.8	25 808.50	146	25 670.38	3746	0.4
	8	151 479.71	3600	89.4	16 104.29	246	16 101.44	3846	0.1
	9	669 054.46	3600	96.4	24 568.04	145	24 411.28	3745	0.1
	10	1 453 002.07	3600	98.6	20 794.58	124	20 704.68	3724	1.2
	11	180 677.73	3600	94.4	10 154.78	543	10 154.78	4143	0.4
	12	604 089.37	3600	97.6	14 667.54	246	14 604.29	3846	0.4
	13	1 592 994.91	3600	99.1	14 353.71	126	14 345.52	3726	1.5
	14	19 963.72	3600	28.3	14 304.21	156	14 304.21	3756	0.0
	15	377 252.44	3600	95.7	16 285.45	245	16 285.45	3845	0.0
	16	194 161.11	3600	91.9	15 808.36	346	15 802.24	3946	0.4
	17	784 427.03	3600	98.5	11 624.74	246	11 528.35	3846	0.9
	18	17 710.48	3600	33.0	11 887.37	124	11 881.55	3724	0.2
	19	659 633.94	3600	97.7	15 606.79	156	15 469.38	3756	0.7
	20	23 244.64	3600	54.9	10 518.24	213	10 490.35	3813	0.1
50	1	–	–	–	32 109.70	2136	32 109.70	5736	3.7
	2	–	–	–	31 270.85	1346	31 270.85	4946	2.2
	3	–	–	–	39 203.20	421	38 203.84	4021	3.4
	4	–	–	–	31 093.32	1563	30 403.15	5163	1.4
	5	–	–	–	42 452.49	1096	42 449.02	4696	0.7
	6	–	–	–	39 453.15	945	39 453.15	4545	3.1
	7	–	–	–	45 960.13	2456	44 660.13	6056	4.4
	8	–	–	–	49 168.5974	3509	48 168.5974	7109	7.10
	9	–	–	–	50 179.28	2043	50 179.28	5643	7.80
	10	–	–	–	49 463.05	645	49 463.05	4245	6.6

(–): No feasible solution is available.

Table A.19
Results for instances under 100 scenarios for H_2.

Nodes	Instance	MILP			Mat		MILP&WS		
		Obj	CPU (s)	Gap (%)	Obj	CPU (s)	Obj	CPU (s)	Gap (%)
14	1	32 963.61	3600	45.9	17 923.44	346	17 913.44	3946	0.5
	2	21 195.66	3600	1.7	20 908.39	245	20 908.39	3845	0.3
	3	29 653.60	3600	34.4	19 475.97	265	19 475.97	3865	0.1
	4	22 168.39	3600	10.3	19 977.04	258	19 977.04	3858	0.5
	5	26 218.75	3600	23.9	19 961.08	125	19 961.08	3725	0.0
	6	28 726.26	3600	35.1	18 812.61	345	18 812.61	3945	0.9
	7	27 076.02	3600	0.2	27 935.16	148	27 935.16	3748	3.3
	8	20 133.75	3600	11.5	18 099.85	945	18 099.85	4545	1.5
	9	26 269.65	3600	1.4	25 962.97	965	25 962.97	4565	0.2
	10	22 305.28	3600	1.1	22 368.54	245	22 368.54	3845	1.4
	11	13 871.17	3600	13.8	12 051.59	695	12 051.59	4295	0.8
	12	17 373.00	3600	3.6	16 939.00	412	16 839.00	4012	0.3
	13	16 836.11	3600	8.6	15 599.21	695	15 599.21	4295	1.3
	14	21 172.63	3600	22.4	16 648.25	258	16 619.25	3858	1.1
	15	20 872.63	3600	12.7	18 396.97	145	18 396.97	3745	0.9
	16	17 377.05	3600	4.3	17 033.36	945	17 033.36	4545	2.4
	17	14 181.41	3600	9.2	13 096.15	236	13 096.15	3836	1.7
	18	14 131.76	3600	4.4	13 607.31	645	13 607.31	4245	0.7
	19	17 548.13	3600	6.4	16 569.94	664	16 569.94	4264	0.9
	20	17 548.13	3600	19.2	14 356.90	975	14 256.90	4575	0.5

Table A.20
Results for instances under 100 scenarios for H_3.

Nodes	Instance	MILP			Mat		MILP&WS		
		Obj	CPU (s)	Gap (%)	Obj	CPU (s)	Obj	CPU (s)	Gap (%)
14	1	61 688.27	3600	72.5	17 007.13	156	17 007.13	3756	0.3
	2	68 987.08	3600	69.9	21 065.52	365	21 065.52	3965	1.3
	3	30 815.67	3600	37.0	19 426.09	456	19 426.09	4056	0.0
	4	67 085.86	3600	71.0	19 509.90	145	19 509.90	3745	0.3
	5	139 632.76	3600	85.9	20 333.18	698	20 333.18	4298	3.3
	6	18 761 854.73	3600	99.9	18 556.09	385	18 556.09	3985	0.2
	7	–	3600	–	27 645.21	269	27 645.21	3869	4.2
	8	288 375.28	3600	94.1	17 405.98	1245	17 405.98	4845	2.8
	9	–	3600	–	26 241.73	1096	26 241.73	4696	2.8
	10	–	3600	–	22 157.82	698	22 057.82	4298	2.8
	11	199 062.40	3600	94.8	11 079.00	546	11 079.00	4146	5.8
	12	–	3600	–	16 541.64	875	16 541.64	4475	5.6
	13	1 195 781.71	3600	98.8	15 886.47	985	15 186.47	4585	3.4
	14	715 131.12	3600	97.7	16 120.98	746	16 120.98	4346	0.1
	15	–	3600	–	18 064.08	698	18 064.08	4298	4.8
	16	1 488 487.83	3600	98.9	17 414.25	783	17 414.25	4383	4.7
	17	1 553 425.20	3600	99.2	13 551.01	1069	13 551.01	4669	6.3
	18	1 169 954.50	3600	99.0	12 902.32	1369	12 902.32	4969	6.0
	19	–	3600	–	16 383.89	542	16 383.89	4142	2.4
	20	–	3600	–	14 733.95	975	14 733.95	4575	3.9

(–): No feasible solution is available.

in terms of solution quality but is time consuming because it calculated as the sum of the CPU time for finding the initial solution plus the CPU time of the resolution with MILP. Probably, improving the quality of the initial solution can reduce the CPU time taken by the *MILP&WS*.

Finally, we propose further research directions from both problem and method perspectives. In one direction, multi-type of products can be considered as a future research area. In another direction, considering other realistic parameters of the model to be stochastic such as the random customers' demands, would be another interesting area for future research. Besides, this study considers only a robust model, however, stochastic programming can be proposed. Future studies can address this issue by developing for example a two-stage stochastic programming to solve the studied problem. Except the methods used in our paper, some of the most representative computational intelligence algorithms can be used like elephant herding optimization (Li et al., 2020), Harris hawks optimization (Zhong & Li, 2022), Runge Kutta optimizer (Ahmadianfar et al., 2021).

CRediT authorship contribution statement

Sana Frifita: Methodology, Software, Validation, Formal analysis, Writing – original draft, Visualization. **Hasan Murat Afsar:** Conceptualization, Methodology, Validation, Formal analysis, Supervision. **Faicel Hnaien:** Conceptualization, Validation, Formal analysis, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors gratefully acknowledge the European Union (FEDER) which funded this research.

Table A.21
Results for instances under 100 scenarios for H_4.

Nodes	Instance	MILP			Mat		MILP&WS		
		Obj	CPU (s)	Gap (%)	Obj	CPU (s)	Obj	CPU (s)	Gap (%)
14	1	1 473 808.51	3600	98.9	16 456.11	546	16 456.11	4146	3.3
	2	1 107 279.01	3600	98.2	20 374.21	256	20 374.21	3856	2.8
	3	–	3600	–	20 334.74	257	–	3857	–
	4	3 721 999.09	3600	99.5	20 790.32	356	20 090.32	4056	5.1
	5	2 228 707.39	3600	99.2	18 578.96	254	18 578.96	3854	4.1
	6	6 392 087.54	3600	99.7	17 199.65	256	17 199.65	3856	1.4
	7	4 204 361.40	3600	99.4	28 021.55	259	28 021.55	3859	5.9
	8	1 268 604.69	3600	98.7	17 680.55	785	17 680.55	4385	4.5
	9	6 329 571.80	3600	99.6	25 243.16	236	25 243.16	3836	1.4
	10	5 155 764.18	3600	99.6	22 007.08	945	22 007.08	4545	4.5
	11	194 259.04	3600	94.7	10 479.87	967	10 479.87	4567	1.1
	12	4 149 682.17	3600	99.6	15 639.39	587	15 639.39	4187	1.4
	13	–	3600	–	15 386.09	326	–	3926	–
	14	657 702.72	3600	97.7	15 528.55	169	15 528.55	3769	3.7
	15	1 657 291.26	3600	99.0	17 023.82	754	17 023.82	4354	2.4
	16	–	3600	–	17 056.05	961	–	4561	–
	17	36 658.87	3600	66.7	13 015.87	2543	13 015.87	6143	6.2
	18	502 154.35	3600	97.6	12 360.13	1026	12 360.13	4626	3.9
	19	–	3600	–	16 723.98	1087	16 723.98	4687	6.5
	20	160 107.62	3600	92.5	12 495.52	1324	12 405.52	4924	3.1

(–): No feasible solution is available.

Table A.22
Different deviations for all scenarios.

Nodes	Instance	5		50		100	
		Dev (%)	Dev1 (%)	Dev (%)	Dev1 (%)	Dev (%)	Dev1 (%)
14	1	0.25	0.25	4.62	5.58	5.06	8.86
	2	1.13	1.13	0.05	0.73	−0.75	2.62
	3	1.36	1.36	0.51	0.59	0.26	–
	4	1.47	1.58	1.48	3.50	2.39	−0.56
	5	0.13	0.13	5.96	5.99	−1.83	7.44
	6	0.38	0.38	0.01	0.59	1.38	9.38
	7	0.48	0.48	0.00	1.07	1.05	−0.31
	8	2.28	2.28	1.35	1.43	3.99	2.37
	9	1.19	1.65	0.93	1.28	−1.06	2.85
	10	0.05	0.05	0.98	1.04	1.41	1.64
	11	0.26	0.26	16.23	17.02	8.78	15.00
	12	1.37	1.37	2.53	2.97	1.80	7.67
	13	2.38	2.38	3.18	3.33	2.72	–
	14	1.09	1.09	0.86	0.86	3.09	7.02
	15	1.75	1.75	8.50	8.50	1.84	8.07
	16	3.16	3.16	0.58	1.10	−2.19	–
	17	0.00	0.00	0.16	4.22	−3.36	0.62
	18	1.80	1.80	7.26	7.62	5.46	10.09
	19	0.01	0.01	4.42	4.57	1.14	−0.92
	20	2.46	2.46	2.68	2.96	−3.24	14.92
50	1	2.59	2.14	2.02	2.31	–	–
	2	0.04	1.10	0.89	4.75	–	–
	3	0.98	2.69	1.55	7.48	–	–
	4	10.87	12.05	11.59	14.17	–	–
	5	5.86	4.57	5.70	5.88	–	–
	6	2.70	3.69	0.30	3.35	–	–
	7	0.09	3.21	4.44	6.64	–	–
	8	2.93	3.13	4.15	4.48	–	–
	9	2.01	1.29	0.21	2.89	–	–
	10	2.99	2.81	1.41	2.71	–	–

(–): No feasible solution is available.

Appendix

See Tables A.13–A.22.

References

Absi, N., Archetti, C., Dauzère-Pérès, S., & Feillet, D. (2015). A two-phase iterative heuristic approach for the production routing problem. *Transportation Science*, *49*(4), 784–795.

Accorsi, R., Baruffaldi, G., & Manzini, R. (2020). A closed-loop packaging network design model to foster infinitely reusable and recyclable containers in food industry. *Sustainable Production and Consumption*, *24*, 48–61.

Adulyasak, Y., Cordeau, J.-F., & Jans, R. (2015). Benders decomposition for production routing under demand uncertainty. *Operations Research*, *63*(4), 851–867.

Afsar, H. M., & Hnaien, F. (2020). Formulations and solution algorithms for dynamic assembly routing problem. *International Journal of Productions Research*, *58*(3), 671–688.

Ahmadianfar, I., Heidari, A. A., Gandomi, A. H., Chu, X., & Chen, H. (2021). RUN beyond the metaphor: an efficient optimization algorithm based on Runge Kutta method. *Expert Systems with Applications*, *181*, Article 115079.

Archetti, C., Bertazzi, L., Paletta, G., & Speranza, M. G. (2011). Analysis of the maximum level policy in a production-distribution system. *Computers & Operations Research*, *38*(12), 1731–1746.

Babagolzadeh, M., Shrestha, A., Abbasi, B., Zhang, Y., Woodhead, A., & Zhang, A. (2020). Sustainable cold supply chain management under demand uncertainty and carbon tax regulation. *Transportation Research Part D: Transport and Environment*, *80*, Article 102245.

Barba-Gutiérrez, Y., & Adenso-Díaz, B. (2009). Reverse MRP under uncertain and imprecise demand. *International Journal of Advanced Manufacturing Technology*, *40*(3–4), 413–424.

Barba-Gutiérrez, Y., Adenso-Diaz, B., & Gupta, S. M. (2008). Lot sizing in reverse MRP for scheduling disassembly. *International Journal of Production Economics*, *111*(2), 741–751.

Bell, W. J., Dalberto, L. M., Fisher, M. L., Greenfield, A. J., Jaikumar, R., Kedia, P., Mack, R. G., & Prutzman, P. J. (1983). Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer. *Interfaces*, *13*(6), 4–23.

Ben-Tal, A., & Nemirovski, A. (1998). Robust convex optimization. *Mathematics of Operations Research*, *23*(4), 769–805.

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

Camargo, V. C., Toledo, F. M., & Almada-Lobo, B. (2014). HOPS–hamming-oriented partition search for production planning in the spinning industry. *European Journal of Operational Research*, *234*(1), 266–277.

Chandra, P. (1993). A dynamic distribution model with warehouse and customer replenishment requirements. *Journal of the Operational Research Society*, *44*(7), 681–692.

Chitsaz, M., Cordeau, J.-F., & Jans, R. (2019). A unified decomposition matheuristic for assembly, production, and inventory routing. *INFORMS Journal on Computing*, *31*(1), 134–152.

Chitsaz, M., Cordeau, J.-F., & Jans, R. (2020). A branch-and-cut algorithm for an assembly routing problem. *European Journal of Operational Research*, *282*(3), 896–910.

Corso, L. L., & Wallace, M. (2015). A hybrid method for transportation with stochastic demand. *International Journal of Logistics Research and Applications*, *18*(4), 342–354.

Díaz-Madroñero, M., Mula, J., & Peidro, D. (2017). A mathematical programming model for integrating production and procurement transport decisions. *Applied Mathematical Modelling*, *52*, 527–543.

Fachini, R. F., Esposto, K. F., & Camargo, V. C. B. (2017). Glass container production planning with warm-ups and furnace extraction variation losses. *International Journal of Advanced Manufacturing Technology*, *90*(1–4), 527–543.

Florio, A. M., Hartl, R. F., & Minner, S. (2020). Optimal a priori tour and restocking policy for the single-vehicle routing problem with stochastic demands. *European Journal of Operational Research*, *285*(1), 172–182.

Frifita, S., Afsar, H. M., & Hnaien, F. (2021). Robust disassembly assembly routing problem with returns under uncertain yields. *IFAC-PapersOnLine*, *54*(2), 354–359.

Frifita, S., & Masmoudi, M. (2020). VNS methods for home care routing and scheduling problem with temporal dependencies, and multiple structures and specialties. *International Transactions in Operational Research*, *27*(1), 291–313.

Frifita, S., Mathlouthi, I., Masmoudi, M., & Dammak, A. (2020). Variable neighborhood search based algorithms to solve a rich k-travelling repairmen problem. *Optimization Letters*, 1–15.

Funke, B., Grünert, T., & Irnich, S. (2005). Local search for vehicle routing and scheduling problems: Review and conceptual integration. *Journal of Heuristics*, *11*(4), 267–306.

Gendreau, M., Jabali, O., & Rei, W. (2016). 50Th anniversary invited article—future research directions in stochastic vehicle routing. *Transportation Science*, *50*(4), 1163–1173.

Godichaud, M., Amodeo, L., & Hrouga, M. (2015). Metaheuristic based optimization for capacitated disassembly lot sizing problem with lost sales. In *2015 international conference on industrial engineering and systems management (IESM)* (pp. 1329–1335). IEEE.

Goodson, J. C., Thomas, B. W., & Ohlmann, J. W. (2017). A rollout algorithm framework for heuristic solutions to finite-horizon stochastic dynamic programs. *European Journal of Operational Research*, *258*(1), 216–229.

Govindan, K., Soleimani, H., & Kannan, D. (2015). Reverse logistics and closed-loop supply chain: A comprehensive review to explore the future. *European Journal of Operational Research*, *240*(3), 603–626.

Gruler, A., Panadero, J., de Armas, J., Pérez, J. A. M., & Juan, A. A. (2020). A variable neighborhood search simheuristic for the multiperiod inventory routing problem with stochastic demands. *International Transactions in Operational Research*, *27*(1), 314–335.

Guide, V. D. R., Jr. (2000). Production planning and control for remanufacturing: industry practice and research needs. *Journal of Operations Management*, *18*(4), 467–483.

Guo, J., Long, J., Xu, X., Yu, M., & Yuan, K. (2022). The vehicle routing problem of intercity ride-sharing between two cities. *Transportation Research, Part B (Methodological)*, *158*, 113–139.

Gupta, S., & Taleb, K. (1994). Scheduling disassembly. *The International Journal of Production Research*, *32*(8), 1857–1866.

Gutiérrez, J., Puerto, J., & Sicilia, J. (2004). The multiscenario lot size problem with concave costs. *European Journal of Operational Research*, *156*(1), 162–182.

Habibi, M. K., Battaïa, O., Cung, V.-D., & Dolgui, A. (2017). Collection-disassembly problem in reverse supply chain. *International Journal of Production Economics*, *183*, 334–344.

Habibi, M. K. K., Battaïa, O., Cung, V.-D., Dolgui, A., & Tiwari, M. K. (2019). Sample average approximation for multi-vehicle collection–disassembly problem under uncertainty. *International Journal of Productions Research*, *57*(8), 2409–2428.

Hein, F., & Almeder, C. (2016). Quantitative insights into the integrated supply vehicle routing and production planning problem. *International Journal of Production Economics*, *177*, 66–76.

Helber, S., Sahling, F., & Schimmelpfeng, K. (2013). Dynamic capacitated lot sizing with random demand and dynamic safety stocks. *OR Spectrum*, *35*(1), 75–105.

Hnaien, F., & Afsar, H. M. (2017). Robust single-item lot-sizing problems with discrete-scenario lead time. *International Journal of Production Economics*, *185*, 223–229.

Hrouga, M., Godichaud, M., & Amodeo, L. (2016). Heuristics for multi-product capacitated disassembly lot sizing with lost sales. *IFAC-PapersOnLine*, *49*(12), 628–633.

Hu, C., Lu, J., Liu, X., & Zhang, G. (2018). Robust vehicle routing problem with hard time windows under demand and travel time uncertainty. *Computers & Operations Research*, *94*, 139–153.

Inderfurth, K., & Langella, I. M. (2006). Heuristics for solving disassemble-to-order problems with stochastic yields. *OR Spectrum*, *28*(1), 73–99.

Inderfurth, K., Vogelgesang, S., & Langella, I. M. (2015). How yield process misspecification affects the solution of disassemble-to-order problems. *International Journal of Production Economics*, *169*, 56–67.

Ji, X., Zhang, Z., Huang, S., & Li, L. (2016). Capacitated disassembly scheduling with parts commonality and start-up cost and its industrial application. *International Journal of Productions Research*, *54*(4), 1225–1243.

Kim, H.-J., Lee, D.-H., & Xirouchakis, P. (2006). Two-phase heuristic for disassembly scheduling with multiple product types and parts commonality. *International Journal of Productions Research*, *44*(1), 195–212.

Kim, H.-J., & Xirouchakis, P. (2010). Capacitated disassembly scheduling with random demand. *International Journal of Productions Research*, *48*(23), 7177–7194.

Kongar, E., & Gupta, S. M. (2006). Disassembly to order system under uncertainty. *Omega*, *34*(6), 550–561.

Kuo, R., Lu, S.-H., Lai, P.-Y., & Mara, S. T. W. (2022). Vehicle routing problem with drones considering time windows. *Expert Systems with Applications*, *191*, Article 116264.

Langella, I. M. (2007). Heuristics for demand-driven disassembly planning. *Computers & Operations Research*, *34*(2), 552–577.

Lee, C., Lee, K., & Park, S. (2012). Robust vehicle routing problem with deadlines and travel time/demand uncertainty. *Journal of the Operational Research Society*, *63*(9), 1294–1306.

Lei, H., Laporte, G., & Guo, B. (2011). The capacitated vehicle routing problem with stochastic demands and time windows. *Computers & Operations Research*, *38*(12), 1775–1783.

Li, J., Lei, H., Alavi, A. H., & Wang, G.-G. (2020). Elephant herding optimization: variants, hybrids, and applications. *Mathematics*, *8*(9), 1415.

Li, X., Tian, P., & Leung, S. C. (2010). Vehicle routing problems with time windows and stochastic travel and service times: Models and algorithm. *International Journal of Production Economics*, *125*(1), 137–145.

Liu, K., & Zhang, Z.-H. (2018). Capacitated disassembly scheduling under stochastic yield and demand. *European Journal of Operational Research*, *269*(1), 244–257.

Marinaki, M., & Marinakis, Y. (2016). A glowworm swarm optimization algorithm for the vehicle routing problem with stochastic demands. *Expert Systems with Applications*, *46*, 145–163.

Mathiyazhagan, K., Sengupta, S., & Poovazhagan, L. (2018). A decision making trial and evaluation laboratory approach to analyse the challenges to environmentally sustainable manufacturing in Indian automobile industry. *Sustainable Production and Consumption*, *16*, 58–67.

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, *24*(11), 1097–1100.

Moon, I., Ha, B.-H., & Kim, J. (2012). Inventory systems with variable capacity. *European Journal of Industrial Engineering*, *6*(1), 68–86.

Nourelfath, M. (2011). Service level robustness in stochastic production planning under random machine breakdowns. *European Journal of Operational Research*, *212*(1), 81–88.

Pour-Massahian-Tafti, M., Godichaud, M., & Amodeo, L. (2020). New models and efficient methods for single-product disassembly lot-sizing problem with surplus inventory decisions. *International Journal of Productions Research*, 1–21.

Prakash, P., Ceglarek, D., & Tiwari, M. (2012). Constraint-based simulated annealing (CBSA) approach to solve the disassembly scheduling problem. *International Journal of Advanced Manufacturing Technology*, *60*(9–12), 1125–1137.

Rossi, R., Kilic, O. A., & Tarim, S. A. (2015). Piecewise linear approximations for the static–dynamic uncertainty strategy in stochastic lot-sizing. *Omega*, *50*, 126–140.

Russell, R. A. (2017). Mathematical programming heuristics for the production routing problem. *International Journal of Production Economics*, *193*, 40–49.

Salameh, M., & Jaber, M. (2000). Economic production quantity model for items with imperfect quality. *International Journal of Production Economics*, *64*(1–3), 59–64.

Sawanishi, H., Torihara, K., & Mishima, N. (2015). A study on disassemblability and feasibility of component reuse of mobile phones. *Procedia CIRP*, *26*, 740–745.

Sheu, J.-B., Chou, Y.-H., & Hu, C.-C. (2005). An integrated logistics operational model for green-supply chain management. *Transportation Research Part E: Logistics and Transportation Review*, *41*(4), 287–313.

Shuang, Y., Diabat, A., & Liao, Y. (2019). A stochastic reverse logistics production routing model with emissions control policy selection. *International Journal of Production Economics*, *213*, 201–216.

Slama, I., Ben-Ammar, O., Dolgui, A., & Masmoudi, F. (2020). New mixed integer approach to solve a multi-level capacitated disassembly lot-sizing problem with defective items and backlogging. *Journal of Manufacturing Systems*, *56*, 50–57.

Solyalı, O., & Süral, H. (2017). A multi-phase heuristic for the production routing problem. *Computers & Operations Research*, *87*, 114–124.

Sungur, I., Ordónez, F., & Dessouky, M. (2008). A robust optimization approach for the capacitated vehicle routing problem with demand uncertainty. *IIE Transactions*, *40*(5), 509–523.

Taş, D., Gendreau, M., Jabali, O., & Jans, R. (2019). A capacitated lot sizing problem with stochastic setup times and overtime. *European Journal of Operational Research*, *273*(1), 146–159.

Tempelmeier, H., & Herpers, S. (2011). Dynamic uncapacitated lot sizing with random demand under a fillrate constraint. *European Journal of Operational Research*, *212*(3), 497–507.

Teunter, R. H., & Flapper, S. D. P. (2003). Lot-sizing for a single-stage single-product production system with rework of perishable production defectives. *Or Spectrum*, *25*(1), 85–96.

Tian, X., & Zhang, Z.-H. (2019). Capacitated disassembly scheduling and pricing of returned products with price-dependent yield. *Omega*, *84*, 160–174.

Tillman, F. A. (1969). The multiple terminal delivery problem with probabilistic demands. *Transportation Science*, *3*(3), 192–204.

Ullerich, C., & Buscher, U. (2013). Flexible disassembly planning considering product conditions. *International Journal of Productions Research*, *51*(20), 6209–6228.

Wagner, H. M., & Whitin, T. M. (1958). Dynamic version of the economic lot size model. *Management Science*, *5*(1), 89–96.

Wang, H.-F., & Huang, Y.-S. (2013). A two-stage robust programming approach to demand-driven disassembly planning for a closed-loop supply chain system. *International Journal of Productions Research*, *51*(8), 2414–2432.

Xu, Z., & Cai, Y. (2018). Variable neighborhood search for consistent vehicle routing problem. *Expert Systems with Applications*, *113*, 66–76.

Zhang, J., Lam, W. H., & Chen, B. Y. (2016). On-time delivery probabilistic models for the vehicle routing problem with stochastic demands and time windows. *European Journal of Operational Research*, *249*(1), 144–154.

Zhong, C., & Li, G. (2022). Comprehensive learning harris hawks-equilibrium optimization with terminal replacement mechanism for constrained optimization problems. *Expert Systems with Applications*, *192*, Article 116432.