A combinatorial optimisation approach for closed-loop supply chain inventory planning with deterministic demand

Pierre Desport, Anne Liret, Carla Di Cairano - Gilfedder, Gilbert Owusu, David Lesaint, Frédéric Lardeux

A Combinatorial Optimisation Approach for Closed-Loop Supply Chains Inventory Planning With Deterministic Demands

Abstract: Supply chains in equipment-intensive service industries often involve repair operations. In this context, tactical inventory planning is concerned with optimally planning supplies and repairs based on demand forecasts and in the face of conflicting business objectives. This paper considers closed-loop supply chains and proposes a mixed-integer programming model and a metaheuristic approach to this problem. The model is open to a variety of network topologies, site functions and transfer policies. It also accommodates multiple objectives by the means of a weighted cost function. We report experiments on pseudo-random instances designed to evaluate plan quality and impact of cost weightings. In particular, we show how appropriate weightings allow to emulate common planning strategies (e.g., just-in-time replenishment, minimal repair) and validate approaches.

Keywords: Supply Chains Planning; Planning Strategies; Mixed Integer Programming; Metaheuristics; Closed-Loop

1 Introduction

The concept of reverse logistics and closed-loop supply chains (CLSC) have gained attention in recent years following growing concerns over the environmental impact of products reaching end-of-life (Krikke et al., 2003; Savaskan et al., 2004). Generally speaking, the aim is to create or recapture value from used products by means of recycling, repair or disposal operations (Govindan et al., 2015). Reverse logistics represents both a challenge and an opportunity to industry, notably in equipment-intensive sectors that consume, produce and supply parts in high volumes (automotive, electronics, etc.) (Fleischmann et al., 1997; Stock, 1992).

Technically, reverse and forward supply chains can be combined to create a closed-loop supply chain that manages both "forward" and "reverse" flows (Akçalı et al., 2009). A CLSC describes the "design, control, and operation of a system to maximize value creation over the entire life cycle of a prod- uct with the dynamic recovery of value from different types and volumes of returns over time" (Guide and Van Wassenhove, 2006). Telecommunications operators, for instance, rely on multi-level CLSC to support service maintenance and repair tasks carried out by field engineers (Desport et al., 2015). These tasks require many different parts with varying degrees of reusability (cables, network cards, IT equipment, etc.) In this context, the function of the CLSC is to replenish field depots with spare parts and transfer used parts back to warehouse centres for recycling or repair.

CLSC are operated through inventory systems that control, allocate and plan inventory through supply and repair decisions (see (Govindan et al., 2015; Ilgin and Gupta, 2010; Stindt and Sahamie, 2014) for a comprehensive review). On the one hand, systems are tied to the underlying network topology, on the other hand they depend on inventory management

strategies (e.g., centralisation, cross-filling) and on the interconnection between repair facilities. Examples of topologies based on multi-level networks include networks with single inventory and repair warehouse centres, field depots with remanufacturing capabilities (Lieckens et al., 2013), or warehouse centres meeting local demand (Cattani et al., 2011). Inventory systems also differ based on whether demand is stochastic (e.g., faulty parts are generally unknown prior to field repair tasks) or deterministic (e.g., maintenance and provision tasks are generally well-specified and scheduled in advance). Inventory control models based on reactive policies (**?**) prevail in stochastic environments and can be coupled with analytical or simulation methods to maintain high-quality service and minimise inventory costs (Guide et al., 2003; Gupta et al., 2013). In deterministic environments however, centralised inventory planning systems based on combinatorial optimisation methods provide a suitable alternative.

In this paper, we restrict our attention to single-item spare parts inventory planning systems under deterministic demand. We consider the general situation where demand for spare parts matches with supply of used parts, that is, when a used part is placed in a field depot, a spare part is immediately taken from on-hand inventory or backordered. For simplicity, we assume that replenishment is confined to the supply chain (i.e., no external procurement) and that all used parts may be repaired (i.e., no scrap). The inventory planning problem specifically consists in generating a plan of supply (transfer of healthy item), return (transfer of faulty items) and repair actions for all sites (e.g., warehouse centres, repair centres, hubs, field depots). The planning horizon is discretised into consecutive time periods (e.g., a weekly plan spanning over 5 days) and the plan is built based on an exact forecast specifying the number of parts to deliver or pick-up, per site and time period. Planned actions are then performed at the start of each period until a new plan is generated.

Our problem is close to many logistics problems. A major aspect of our problem is the delivery of spare parts to customers. This goal points out that the Tactical Distribution Planning Problem (TDPP) is close to the classic Inventory Routing Problem (IRP) (Bertazzi and Speranza, 2012; Coelho et al., 2013; Campbell et al., 1998). However, the goal of the IRP is to define a policy that can be applied repeatedly to minimise transport cost while meeting the demand (considered as a periodic constant volume per product). In our case, objectives are multiple and include storage and repair costs. Moreover, while demand forecast are deterministic, they can vary from one time period to another and delivery schedules are predefined. Another well-known problem is the Multi-Level Lot Sizing Problem. Although this problem takes place in a multi-level supply chain, it is more concerned with scheduling the production activities and does not focus on distribution and refurbishing aspects(Drexl and Kimms, 1997; Cornelli et al., 2006).

We present a mixed-integer program to model this tactical distribution planning problem (TDPP) that is applicable in a wide range of environments. The model leaves full flexibility as to the stocking capability of each site (spare parts, used parts or both), its repair capability, and whether it fulfils demand or not. The supply networks for spare parts and used parts join on repair sites but their topology is otherwise not limited by constraints (from arborescences to complete graphs). A schematic example is illustrated in Figure 1 where a warehouse $S1$ provides spares to two sites ($S2$ and $S3$) where demands occurs. These sites can send back faulty parts to $S1$ and consequently to a repairer that retrieve them as healthy spares later on. The model does not enforce any restriction either on supply and repair lead times or on schedules. Inventory is managed centrally in this approach and flow constraints are enforced to compute and maintain inventory levels across sites and over time. Constraints on spares match on-site demand, backorders, and outgoing inventory (as per transfer decisions) with

on-hand and incoming inventory including repaired parts (as per repair decisions). Flow constraints for used parts follow similar patterns. As expected, integrity constraints forbid backordering if inventory is available and prevent from transferring or carrying inventory if orders are outstanding. Lastly, the objective function is a weighted sum of linear functions costing the total number of units that are backordered, held in inventory, shipped or repaired. No start-up penalty applies and weights are user-defined to support different prioritisation strategies between service levels and operational costs.

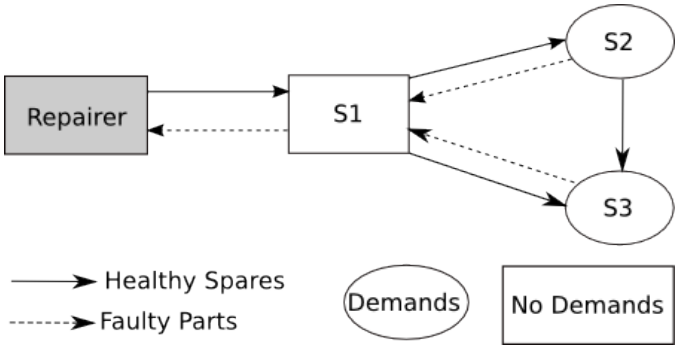


Figure 1 A 2-echelon CLSC system allowing lateral transshipments of spare parts.

Integrity constraints put aside, TDPP reduces to a minimum cost network flow problem. The core problem may indeed be represented by a space-time graph with nodes subject to conservation-of-flow equations, arcs labelled with convex cost functions, and a global cost function itself convex. This sub problem reduces to the minimum convex network flow problem Guisewite and Pardalos (1990) and is therefore polynomial. We show however that TDPP is NP-hard due to the above mentioned integrity constraints which enforce "mutual exclusion" rules between backordering decisions and decisions to carry or release inventory. The result is established by reducing the subset sum problem (Kellerer et al., 2004). Alternatively to the MIP model, we introduce a metaheuristic to solve TDPP. We compare both approaches on a range of pseudo-random instances generated from a concrete case in the telecommunications service sector. Instances feature different profiles of demand and initial stock levels but share the same topology (i.e., a 3-echelon CLSC with single distribution and repair centres) and temporal structure (i.e., lead-times and schedules). Results attest the superiority of the metaheuristic in terms of solution fitness (as measured by the objective function), run time and scalability.

From a decision support perspective, the TDPP solver can be packaged in a periodic planning system as sketched in 2. The key parameter of this system is the plan recomputation frequency.

This paper is organised in 5 sections. Section 2 introduces the MIP model for TDPP, proves it is NP-hard by reduction of Subset Sum, and casts the core sub problem into a minimum convex network flow problem. Section 3 introduces the metaheuristic and its key components. Section 4 presents the case study, the experiments comparing MIP and metaheuristic approaches and the sensitivity analysis. Section 5 concludes and discusses possible extensions to this work.

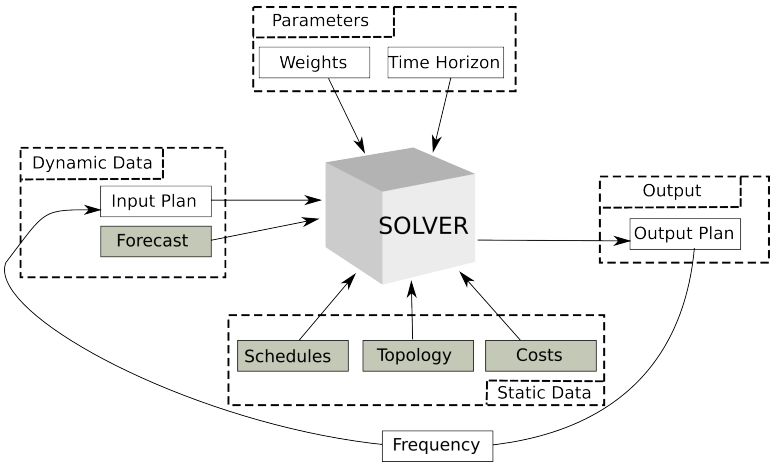


Figure 2 A periodic planning system for CLSC.

2 Problem Description

We introduce in this section the Tactical Distribution Planning Problem (TDPP). The proposed version is limited to single item planning as, in our case study, items are independent. We also consider that faulty parts are all either repaired or replaced by new ones by the repairer. We propose in section 2.2 a list of possible extensions to our model (buying or selling items, capacity constraints, attrition rate, …). We show that the TDPP is close to flow problems such as the Minimum Cost Flow Problem (MCNFP).
We also demonstrate the complexity of the TDPP by reducing the Subset Sum Problem to it. Note that in the rest of the paper we use the terms "healthy items" to designate spare parts and "faulty items" to nominate used parts.

2.1 Formalisation

We cast below the TDPP as a class of finite domain constraint optimisation problems (COP). TDPP is parameterised with a planning horizon T, a set of sites L and a space-time graph G^* whose vertices are all the possible pairings of sites and time points ($\#L \times \#T$) and arcs the routes interconnecting sites through the time horizon. Specifically, G^* can be divided into 3 subgraphs G^h, G^f and G^r where G^h describes the supply routes for healthy items, G^f the return routes for faulty items and G^r the repair routes. Parameters are more formally defined in Table 1. Figure 3 is an example of space-time graph. These subgraphs describe the network and schedules of the CLSC. Note that this representation is generic and is suitable to any topology and schedule. The parameters define the structure of the problem which the sets of constants, variables, constraints and cost terms are based upon. Tables 2 and 3 present the constants and the variables of the problem which are indexed either using the sites of the CLSC (e.g., repair costs), the nodes of the space-time graph (e.g., inventory level per site and time bucket) or its arcs (e.g., transfers). We introduce the following notation x_z^y where x is the name of the constant (or variable), y the state (i.e., healthy, faulty, healthy stock,-...), z the space-time parameters (i.e., locations, time buckets).

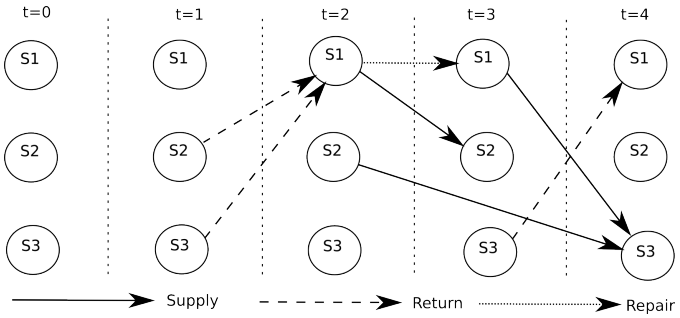


Figure 3 An example of space-time graph based on Figure 1

Table 1 Parameters of an instance

Paremeters	
Set of sites	$L : L \subseteq \mathbb{N}$
Time horizon	$T : T \subseteq \mathbb{N}$
Supply routes	$G^h : L \times L \times T \times T$
Return routes	$G^f : L \times L \times T \times T$
Repair routes	$G^r : L \times T \times T$

Table 2 Constants of an instance

Constants	
demand forecast	$d : L \times T \to \mathbb{N}$
inventory costs	$\mu^{s^h}, \mu^{s^f} : L \to \mathbb{R}$
backorder costs	$\mu^b : L \to \mathbb{R}$
transfer costs	$\mu^{m^h}, \mu^{m^f} : L \times L \to \mathbb{R}$
repair costs	$\mu^{m^r} : L \to \mathbb{R}$
weights	$\omega^s, \omega^b, \omega^{m^{hf}}, \omega^{m^r} \in \mathbb{N}$
initial state	I

The constants of a TDPP instance are the demand forecast d that defines the demand level for each site and time bucket of the horizon, the unit costs μ^* (note that it is possible to make dynamic the costs changing its definition by $L \times T \to \mathbb{N}$ or $L \times L \times T \times T \to \mathbb{N}$ or $L \times T \times T \to \mathbb{N}$), the user-defined costing weights ω^* and the initial state I. The initial state I is an assignment of all the variables at time bucket 0.

Variables are split into decision variables (in bold style) and auxiliary variables. The former model the set of actions defining a plan and the latter model the resulting state of the supply chain.

For two sites l and l', and time buckets t and t', decision variable $m^h_{l,l',t,t'}$ (respectively, $m^f_{l,l',t,t'}$) denotes the quantity of healthy (resp., faulty) items transferred from l at t and reaching l' at t'. Likewise, decision variable $m^r_{l,t,t'}$ denotes the quantity of faulty items whose repair starts at t and ends at t' on site l. Notice that Repair operations transform faulty items into healthy items and effectively link up the faulty and healthy subgraphs within a

Table 3 Variables of an instance

Variables	
inventory	$s^h, s^f : L \times T \to \mathbb{N}$
backorders	$b : L \times T \to \mathbb{N}$
healthy potential	$p^h : L \times T \to \mathbb{Z}$
faulty potential	$p^f : L \times T \to \mathbb{N}$
transfers	$m^h, m^f : L \times L \times T \times T \to \mathbb{N}$
repairs	$m^r : L \times T \times T \to \mathbb{N}$
costs	$c, c^s, c^b, c^{m^{hf}}, c^{m^r} \in \mathbb{R}$

repair time and cost. As such, they are a particular type of transfer operations topologically restricted to link a site with itself. Any decision variable that has no corresponding arc in the space-time graph is forced to 0. In effect, a plan is a labelling of the arcs of the space-time graph with positive or null values. The labelling of an arc represents the quantity of items (possibly null) being transferred or repaired between the site(s) connected by the arc.

Auxiliary variables are directly computed from the decision variables. They model the state and resulting cost associated to each site and time bucket. Given time bucket t and site l, variable $s^h_{l,t}$ (resp. $s^f_{l,t}$) denotes the inventory level for healthy (resp., faulty) items and $b_{l,t}$ the backorders. Variable $p^h_{l,t}$ (resp., $p^f_{l,t}$) represents the healthy (resp., faulty) potential of site l at time bucket t. $p^h_{l,t}$ is the difference between on-hand inventory and backorders. $p^f_{l,t}$ is the quantity of faulty items available to be sent out from site l at time bucket t. Lastly, four auxiliary variables are used to model the backordering (c^b), inventory (c^s), transfer ($c^{m^{hf}}$) and repair costs (c^{m^r}) of a plan.

Equations (1a) to (1c) characterize faulty items management. They define the stock requirement constraint (i.e. an item can be transferred only if the stock is positive). Specifically, Equation (1c) represents the quantity of faulty items that can be returned from a particular site l on a particular time bucket t. They also link demands and faulty items by adding a faulty item when a demand occurs.

$$\forall l \in L, t \in T, \qquad s^f_{l,t} = p^f_{l,t} - \sum_{t' \in T} m^r_{l,t,t'} - \sum_{\substack{l' \in L,\\ t' \in T}} m^f_{l,l',t,t'} \tag{1a}$$

$$\forall l \in L, t \in T, \qquad \sum_{t' \in T} m^r_{l,t,t'} + \sum_{\substack{l' \in L,\\ t' \in T}} m^f_{l,l',t,t'} \leq p^f_{l,t} \tag{1b}$$

$$\forall l \in L, t \in T \backslash \{0\}, \quad p^f_{l,t} = s^f_{l,t-1} + \sum_{\substack{l' \in L,\\ t' \in T}} m^f_{l',l,t',t} + d_{l,t} \tag{1c}$$

Equations (2a) to (2d) are conservation-of-flow constraints on healthy items that also give priority to the fulfilment of local demand. This prioritization policy forbids backordering when inventory is available and forbids transferring or carrying inventory when orders are outstanding on a site.

$$\forall l \in L, t \in T, \qquad s^h_{l,t} = max(0, p^h_{l,t} - \sum_{\substack{l' \in L,\\ t' \in T}} m^h_{l,l',t,t'}) \tag{2a}$$

$$\forall l \in L, t \in T, \qquad \sum_{\substack{l' \in L, \\ t' \in T}} m^h_{l,l',t,t'} \leq max(0, p^h_{l,t}) \tag{2b}$$

$$\forall l \in L, t \in T \backslash \{0\}, \quad p^h_{l,t} = s^h_{l,t-1} + \sum_{l' \in L, t' \in T} m^h_{l',l,t',t} + \sum_{t' \in T} m^r_{l,t',t} - b_{l,t-1} - d_{l,t} \tag{2c}$$

$$\forall l \in L, t \in T, \qquad b_{l,t} = \max(0, -p^h_{l,t}) \tag{2d}$$

Equation (3e) models the fitness function which is a weighted sum of the four different costs computed by equations (3a) to (3d). All these costs are linear functions based on unit costs that are site- or route-specific. For instance, the backordering cost for a site is its total number of backorders times its unit cost for backordering. The backordering cost associated to a plan is then the sum of backordering costs over all sites. The weighted sum will allow us to emulate inventory management policies (see Section 4).

$$c^s = \sum_{l \in L, t \in T} (s^h_{l,t} \times \mu_l^{s^h} + s^f_{l,t} \times \mu_l^{s^f}) \tag{3a}$$

$$c^b = \sum_{l \in L, t \in T} b_{l,t} \times \mu^b_l \tag{3b}$$

$$c^{m^{hf}} = \sum_{\substack{l,l' \in L, \\ t,t' \in T}} (m^h_{l,l',t,t'} \times \mu^{m^h}_{l,l'} + m^f_{l,l',t,t'} \times \mu^{m^f}_{l,l'}) \tag{3c}$$

$$c^{m^r} = \sum_{\substack{l \in L, \\ t,t' \in T}} (m^r_{l,t,t'} \times \mu_l^{m^r}) \tag{3d}$$

Global min :

$$c = \omega^s \times c^s + \omega^b \times c^b + \omega^{m^{hf}} \times c^{m^{hf}} + \omega^{m^r} \times c^{m^r} \tag{3e}$$

2.2 Extensions

The TDPP is generic problem and can easily be updated to include a large variety of extensions. We sketch a non-exhaustive list of extensions that seem relevant in a supply chain management context.

The TDPP is limited to 3 basic operations (supply, return and repair) such that the number of items moving or installed in the supply chain is constant. Other operations can be added to the model to increase or decrease the quantity of items. For example, it is reasonable to include selling and purchasing operations, to be able to meet a surplus of demands or oppositely to evacuate unused items. Equally, our model is currently restrained to maintenance or replacement procedures. We can include other procedures such as installation (i.e., supply a spare to a new site) and dismantling (i.e., the faulty items are not replaced). These procedures can be easily incorporated to our model by adding 2 matrices of constants.

Another common constraint in supply chain management problems is the capacity

constraint. In the TDPP, capacity constraints could be applied on transfer, storage and repair quantities. Matrices and inequalities can easily model these constraints but we intuitively understand that they add some complexity to the problem. The model may also be extended to model the notion of safety stock. It can be represented either with a strict capacity constraint or by adding an objective to the fitness function. The second method provides the possibility to prioritize objectives (i.e. meet the demands and if possible keep the stock level over the safety stock level).
Other extensions include accounting for unrepairable and recyclable items (using an attrition rate and a repair rate), supporting interchangeable item types (to cope with product/service platform lifecycles) or inter-dependent item types (to reflect bills of materials associated to service operations).

2.3 TDPP *as a flow management problem*

The TDPP can be easily seen as a flow problem thanks to its space-time graph. We present in this section the minimum cost network flow problem (MCNFP) and demonstrate how to transform the core of the TDPP into this flow problem.

2.3.1 The MCNFP *problem*

The minimum cost network flow problem (MCNFP) is a classic flow problem occurring in a directed graph $I = (N_I, A_I)$ of k nodes (Goldberg et al., 1989; Guisewite and Pardalos, 1990; Pardalos, 1993). A k-vector π represents the potential value of each node. Each arc (i, j) has a flow value $x_{i,j}$ bounded by $a_{i,j} \leq x_{i,j} \leq b_{i,j}$ associated with a cost function $c_{i,j}$ such that its total cost is $c_{i,j}(x_{i,j})$.
The objective is then to solve:

$$\text{global min} \sum_{i,j \in A} c_{i,j}(x_{i,j}) \tag{4}$$

subject to

$$\sum_{(i,k) \in A} x_{i,k} - \sum_{(k,i) \in A} x_{k,i} = \pi_i, \forall i \in N_I \tag{5}$$

and

$$0 \leq a_{i,j} \leq x_{i,j} \leq b_{i,j}, (i,j) \in A_I \tag{6}$$

Note that a system is called consistent if $\sum_{i=1}^{n} \pi_i = 0$ and that a node with $\pi_i < 0$ is a sink, contrary to a node with $\pi_i > 0$ called a source. The concave version of this problem (i.e., with fixed charge costs) has been proven NP-Hard (Guisewite and Pardalos, 1990).

2.3.2 TDPP *and* MCNFP

We describe here the process to transform the core of a TDPP instance into an uncapacitated MCNFP instance (Goldberg et al., 1989). We reuse the notations introduced in Section 2.1.

1. For each pair (site $l \in L$, time bucket $t \in T$), create three nodes $n^h_{l,t}$, $n^b_{l,t}$ and $n^f_{l,t}$.

2. For each node, define the potential value:

$$\forall l \in L \quad \pi_{n^h_{l,0}} = s^h_{l,0} \tag{7a}$$

$$\forall l \in L \quad \pi_{n^b_{l,0}} = -b_{l,0} \tag{7b}$$

$$\forall l \in L \quad \pi_{n^f_{l,0}} = s^f_{l,0} \tag{7c}$$

$$\forall l \in L, t \in T\backslash\{0\} \quad \pi_{n^h_{l,t}} = \sum_{l' \in L, t' \in T, t' \leq 0} m^h_{l',l,t',t} + \sum_{t' \in T, t' \leq 0} m^r_{l,t',t} \tag{7d}$$

$$\forall l \in L, t \in T\backslash\{0\} \quad \pi_{n^b_{l,t}} = -d_{l,t} \tag{7e}$$

$$\forall l \in L, t \in T\backslash\{0\} \quad \pi_{n^f_{l,t}} = \sum_{l' \in L, t' \in T, t' \leq 0} m^f_{l',l,t',t} + d_{l,t} \tag{7f}$$

Note that Equation (7d) only involves variables $m^h_{l',l,t',t}$ and $m^r_{l,t',t}$ such that $t' \leq 0$. These variables are actually constants set with the initial state data.

3. For each couple of healthy node and backorder node, $n^h_{l,t} \times n^b_{l,t}$:

   - If $t > 0$ create an arc $(n^h_{l,t-1}, n^h_{l,t})$. Its value is the **healthy storage flow**. We label it $x_{(n^h_{l,t-1}, n^h_{l,t})}$ and its associated cost is 0 if $t - 1 = 0$, $\mu_l^{s^h}$ otherwise.
   - If $t > 0$ create an arc $(n^b_{l,t-1}, n^b_{l,t})$. Its value is the **backordering flow**. We label it $x_{(n^b_{l,t-1}, n^b_{l,t})}$ and its associated cost is 0 if $t - 1 = 0$, μ^b_l otherwise.
   - Create an arc $(n^h_{l,t}, n^b_{l,t})$. Its value is the **consumption flow**. We label it $x_{(n^h_{l,t}, n^b_{l,t})}$ and its associated cost is 0.

4. For each faulty node $n^f_{l,t}$ with $t > 0$, create an arc $(n^f_{l,t-1}, n^f_{l,t})$. Its value is the **faulty storage flow**. We label it value $x_{(n^f_{l,t-1}, n^f_{l,t})}$ and its associated cost is 0 if $t - 1 = 0$, $\mu_l^{s^f}$ otherwise.

5. Create a residual node n^g to ensure the global consistency of the instance so that

$$\pi_{n^g} = -\sum_{l \in L, t \in T} (\pi_{n^h_{l,t}} + \pi_{n^b_{l,t}} + \pi_{n^f_{l,t}}) \tag{8}$$

6. For each healthy node $n^h_{l,t}$ with $t = \#T$, create an arc $(n^h_{l,t}, n^g)$. It is the **healthy storage overflow**. We label its value $x_{(n^h_{l,t}, n^g)}$ and its associated cost is $\mu_l^{s^h}$.

7. For each backordering node $n^b_{l,t}$ with $t = \#T$, create an arc $(n^g, n^b_{l,t})$. It is the **backorder overflow**. We label its value $x_{(n^g, n^b_{l,t})}$ and its associated cost is μ^b_l.

8. For each faulty node $n^f_{l,t}$ with $t = \#T$, create an arc $(n^f_{l,t}, n^g)$. It is the **faulty storage overflow**. We label its value $x_{(n^f_{l,t}, n^g)}$ and its associated cost is $\mu_l^{s^f}$.

9. For each healthy transfer $m^h_{l,l',t,t'}$, create a transfer arc $(n^h_{l,t}, n^h_{l',t'})$. We label its value $x^{ht}_{(n^h_{l,t}, n^h_{l',t'})}$ and its associated cost is $\mu^{m^h}_{l,l'}$

10. For each faulty transfer $m^f_{l,l',t,t'}$, create a transfer arc $(n^f_{l,t}, n^f_{l',t'})$. We label its value $x^{ft}_{(n^f_{l,t},n^f_{l',t'})}$ and its associated cost is $\mu^{mf}_{l,l'}$

11. For each repair $m^r_{l,l,t'}$, create a repair arc $(n^f_{l,t}, n^h_{l,t'})$. We label its value $x^{rt}_{(n^f_{l,t},n^h_{l,t'})}$ and its associated cost is $\mu^{m^r}_l$.

Figure 4 shows the MCNFP instance (without labelling) obtained by transformation of the TDPP example proposed in Figure 1.

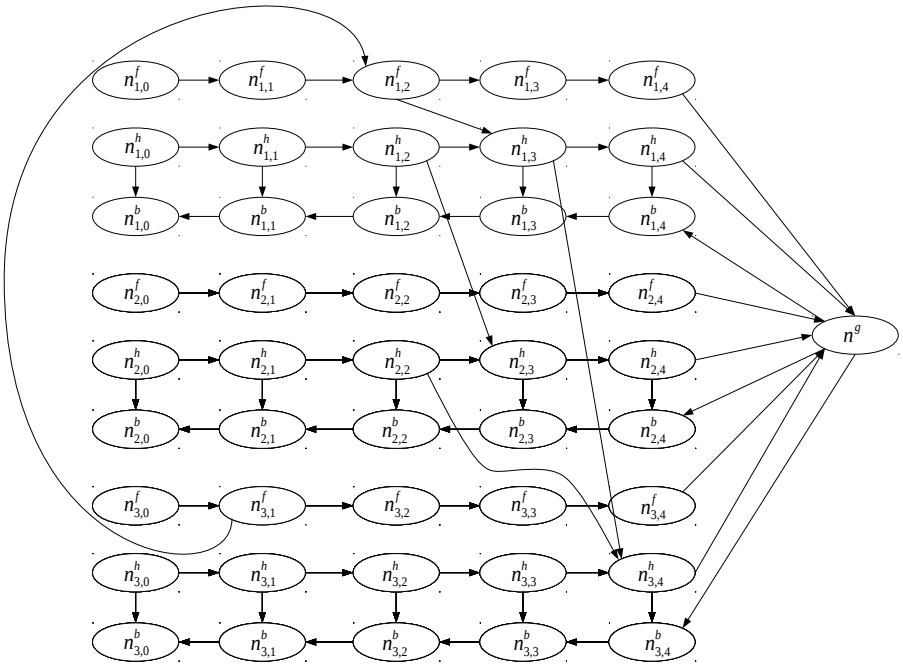


Figure 4 A flow modelisation

This transformation shows that the main part of the TDPP can be modelled as a convex MCNFP problem. This problem is polynomial as Fontes and Gonçalves (2007) have demonstrated that concave cost functions are mandatory to have a NP-Complete MCNFP problem. This transformation does not include the integrity constraint and we show in the next Section that this constraint transforms our problem in a NP-Hard one.

2.4 Reduction to the Subset Sum problem

Theorem 1 TDPP *is NP-Hard.*

Proof: We demonstrate that our problem is NP-Hard by reducing the Subset Sum Problem to the TDPP in polynomial time. The Subset Sum Problem is known to be NP-Complete (Garey and Johnson, 1990) and it is defined by only one question: Given a finite multiset of

z positive integers K $[y_1..y_z]$ and a positive integer B, is there a subset $K' \subseteq K$ such that $\sum_{k \in K'} y_k = B$? This reduction is based on the implicit constraint induced by equations (2d), (2c), (2a) and (2b) that ensure the consumption of available items on sites if demands occur on the same day. It allows us to ensure a non-separation of the outflow on the origin sites in order to have enough items in the supply chain to meet the demands in the stores. We propose the following reduction:

1. The time horizon T is set to 3 time buckets.
2. Create 3 distinct sets of sites $L1$, $L2$ and $L3$ such that $L1 = [1..z]$, $L2 = [z+1..3z]$ and $L3 = [3z+1..3z+2]$. We define $L = L1 \bigcup L2 \bigcup L3$.
3. We set up the initial stock levels such that $\forall l \in L1,\ s^h_{l,0} = y_l + 1$.
4. We create the demand forecast such that $\forall l \in L2,\ d_{l,1} = 1,\ d_{3z+1,3} = B$ and $d_{3z+2,3} = \sum_{k=1}^{z} y_k - B$.
5. We define the associated costs such that $\forall l \in L2,\ \mu^b_l = 0$, $\mu^b_{3z+1} = 1$ and $\mu^b_{3z+2} = 1$.
6. $\forall l \in L2$, create 2 healthy transfers $m^h_{l,l+z,1,2}$ and $m^h_{l,l+2z,1,2}$.
7. $\forall l \in L1$, create 2 transfers $m^h_{l+z,3z+1,2,3}$, $m^h_{l+2z,3z+2,2,3}$.
8. All the unmentioned costs are assumed to be 0.

If the $fitness$ is equal to 0 (i.e. we can meet all the demands occurring on $l = 3z+1$ and $l = 3z+2$) then it exists a subset $K' \subseteq K$ such that $\sum_{k \in K'} y_k = B$. □

Example. Figure 5 shows an example of transformation with $K = [8; 18; 9; 2]$ and $B = 17$.

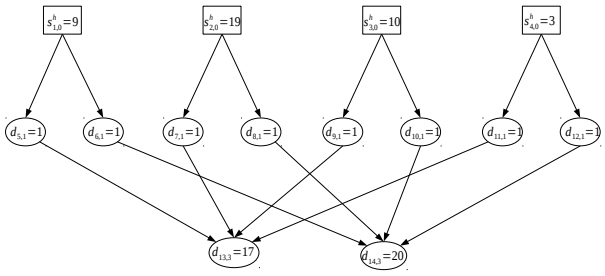


Figure 5 Subset Sum to TDPP for $K = [8; 18; 9; 2]$ and $B = 17$

3 Metaheuristics

Firstly, we have proposed a Mixed Integer Programming (MIP) model to solve the TDPP problem. The MIP is directly deducted from the equations presented in Section 2. This

exact method is working well but when the number of sites and the planning horizon grows up, it quickly has difficulty providing a solution (see Section 4).

In such case, metaheuristics (Gendreau and Potvin, 2010) are known to take up the slack. They potentially allow to find a very good solution in an acceptable time. We propose an algorithm denoted BIS(Best Improving Sequence) that can be classified as an ILS (Iterated Local Search) metaheuristic. Classically, a metaheuristic is defined by: a solution, a neighbourhood function, an evaluation function, and of course a choice of heuristics.

Search space and evaluation function

In our case, a solution is a plan which is a set of actions between nodes connected in the space-time graph G^* realised in a given time horizon. The search space is defined by: $\mathcal{S} = 2^{\mathcal{M}}$ where $\mathcal{M}$ is the set of possible actions given by the topology. In order to compare each solution an evaluation function ($fitness : \mathcal{S} \rightarrow \mathbb{R}$) is defined using the cost function given by Equation (3e).

Neighbourhood

The neighbourhood function explores the solutions reachable by a $move$ ($move$: $\mathcal{N} : \mathcal{S} \rightarrow 2^{\mathcal{S}}$). A $move$ between 2 plans corresponds to a sequence of unitary actions (only one item is transferred or repaired each time) improving the global objective function.
We introduce the notion of $sequence$. A sequence is a series of actions possibly empty. Actions included in a sequence are constrained temporarily and geographically such that given a sequence seq and 2 consecutive actions $m_{l1,l2,t1,t2}, m'_{l3,l4,t3,t4} \in seq$, we have $l2 = l3$ and $t2 \leq t3$. The type of action is also important. Supply and repair actions can only be followed by supply actions whereas a return can be followed either by another return or by a repair. For instance, a sequence could be compounded by a unique supply action or by a series of return, repair and supply actions. The maximum size of a sequence is $T - 1$ actions.

Local Search

For any given plan, it is computationally expensive to deduce the global impact of a sequence of actions. We then propose a method that evaluates locally the impact of a sequence. This method is denoted $improve(m, P, seq)$ with $m \in G^*$, a plan P and a sequence of actions seq.
It returns the maximum improvement value that a sequence initiated by seq and followed by action m ($seq \bigcup m$) could bring to the plan P. Note that the number of potential sequences evaluated with this method can increase very quickly. The computation of the improvement value only takes into account backorder, inventory, repair and transfer costs. Figure 6 describes an example of $improve(m, P, seq)$. In that case, 5 sequences of actions are executable after m and $improve(m, P, seq)$ will return the maximum positive improvement value that one of these sequences could bring and 0 otherwise. The computation of the improvement value could seem expensive but this cost is in fact limited by the usage of a data structure allowing to easily compute the improvement value of a sequence. This data structure is a delta matrix that details the improvement value of the different sequences based on actions. The matrix is created anti-chronologically and built around a propagation algorithm that consists in using subsequences to avoid the exploration of all the possible sequences. We also notice that even if the first computation of the matrix is costly, we only recompute it partially later on.

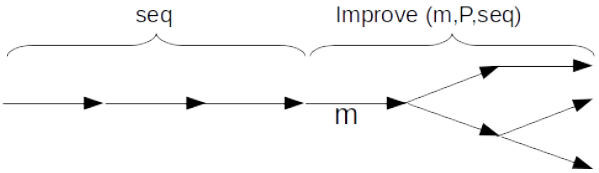


Figure 6 Example of $improve(m, P, seq)$

This $improve$ function allows to define the graph $G|_{seq}$. Given a plan P and a sequence seq, this graph represents the doable actions (satisfying the stock constraints) $m \in G^*$ following the sequence seq and with $improve(m, P, seq) > 0$. For instance, if the last action m of a sequence seq is a supply action from site 1 to site 2 completed on time bucket 10 then $G|_{seq}$ includes all the possible supply actions from site 2 after time bucket 10 whose improvement value is positive. Algorithms 1 and 2 describe the process of selection and application of a sequence and Algorithm 3 presents the BIS method. We can notice that we use the best improvement selection (i.e. we select action m such that $improve(m, P, seq)$ is maximum). It ensures us to only apply very efficient sequences of actions and provides good results in most of the cases. However, in some cases this selection will lead to a local optimum. It can be caused by different factors such as a complex topology, asynchronous schedules or stock positioning. Therefore, even if this selection process is good to drive the search, it has to be paired with a diversification process. This diversification process is represented by the α probability that allows us to explore the search space by selecting "less improving" sequences. This diversification probability is fixed to 1 in the first iteration of the BIS Algorithm and then decreases following an arithmetico-geometric series. It allows to bifurcate at each step of the creation of the sequence such that with $\alpha = 0$, we can explore all the sequences seq of the search space initiated by m with $improve(m, P, seq) > 0$.

Algorithm 1 Action Selection

Input: A Graph G, A Plan P, A Sequence seq, A Probability α
Output: An Action
1: With a probability α, return $argmax_{m \in G}(improve(m, P, seq))$; ▷ In case of equality, m is selected randomly amongst the best ones
2: With a probability $1 - \alpha$, Return $m \in G$;

Algorithm 2 Sequence Computation

Input: A Graph G, A Plan P, A Sequence seq, A Probability α
Output: A sequence
1: **while** $G|_{seq} \neq \emptyset$ **do**
2: Select an action m by using Algorithm 1 with $G|_{seq}$, P, seq and α;
3: Add m to seq;
4: **end while**
5: Return seq;

Algorithm 3 The BIS algorithm

Input: The constants of the problem, A number of iterations,
Output: A Plan
1: Set a probability α to 1
2: **while** Number of iterations is not reached **do**
3: Create an empty Plan P;
4: Create an empty sequence seq;
5: **while** $G|_{seq} \neq \emptyset$ **do**
6: Update seq by computing a sequence with Algorithm 2 with $G|_{seq}$, P, seq and α;
7: Execute the sequence seq and add it to P;
8: Empty seq;
9: **end while**
10: Evaluate P;
11: α decreases following an arithmetico-geometric series
12: **end while**
13: return the best Plan

4 Real Case

This section focuses on tactical inventory planning problems arising in equipment-intensive industries. It is based on a concrete case study in the telecommunications sector where large quantities and varieties of items are required for service maintenance and repair tasks at customer premises or company exchanges. Specifically, we consider a multi-echelon supply chain and tackle the problem of determining an optimal stock distribution plan given a demand forecast. The supply chain discussed here relies on a classical design with a tree (hub and spoke) topology organised around a warehouse inventory centre as presented in Figure 7. Experiments presented in this paper are limited to this single topology as it matches our case study and is one of the most spread network configuration for supply chains (O'Kelly, 1998).

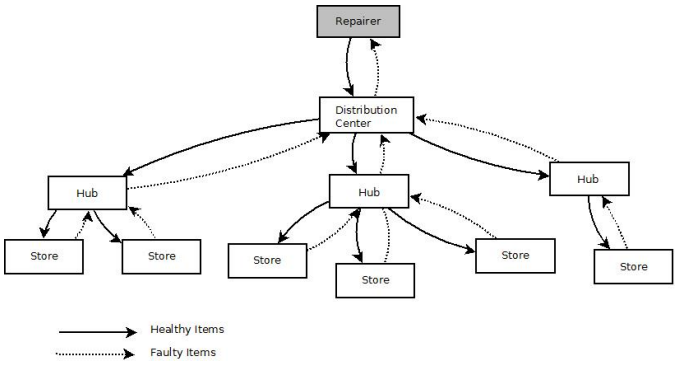


Figure 7 A simple example of real case supply chain.

The distribution centre *DC* (e.g.- warehouse centre) replenishes stores (e.g.- field depots) with items through intermediate hubs (e.g.- exchange points) and channels faulty items back

to a warehouse repair centre. Pick-up and delivery schedules are periodic with constant transportation lead times that are specific to each store. The same applies to the “repair loop" linking the inventory and repair warehouses. Operationally, a request for an item is resulting from a service malfunction and is created when a faulty item is placed in a store. This request is immediately satisfied if the store has items on-hand and prior on-site demands have been met. Otherwise, it is delayed until fresh items are delivered which results in service down-time. The faulty item is itself collected at the next pick-up date and stored at the inventory warehouse.

Decisions to supply stores with items or to repair faulty items are made ahead of time by an analyst team and revised periodically until execution. This repair and supply plan prescribes the volume of items to ship at each time bucket (i.e.- period of time) of the planning horizon. It is based on a demand forecast that estimates the number of items requested for each item type in each store at each time bucket. Such forecasts are considered accurate enough for particular items (e.g.- high-volume materials) or specific activities (e.g.- maintenance tasks).

Experiments are carried out on pseudo-random instances. The objective is to assess the impact of the key features of problem instances onto the plan’s fitness (demand distribution, stock levels, etc.) and to investigate the ability to “emulate" global planning strategies through different cost weightings. To this end, but also to reduce bias in the analysis, we generate all instances using the same topology, time horizon, transfer schedules, lead-times, transition period, and demand pattern. Settings are chosen consistently with data emanating from our case study.

We have generated many sizes of instances with similar results, but we present in this paper only two representative sizes of instances (for other sizes the behaviour remains the same). A small one compounded of 25 sites and a larger one compounded with 100 sites. The supply network includes 1 distribution centre, 9 (resp. 33) hubs and 15 (resp. 66) stores. Each hub deserves an exclusive set of stores. The DC serves all hubs and is the unique repair centre. The time horizon is initialised to 60 day-buckets. Transfers between connected sites all take 1 day. To fit more precisely the real case, the repair time is set to 3 days instead of only one day as proposed in section 2. Sites also have identical pickup and delivery frequencies (every 5 days) with synchronous schedules (identical delivery and pickup days). All transfers are performed according to schedule even if there is not item to deliver or pick up. Therefore we will ignore transfer cost in our experiments. We introduce a “transition period” set to the first 7 days to coincide with the earliest delivery time for a store. Demands raised within this period are supposedly addressed by previous planning decisions. Hence, the results only depend on our model without any interferences from previous choices and allow us to devise confidently about the quality of our method. For this reason, instances have no demands within the transition period, and ongoing transfers all complete within that time window. Similarly and consistently, with the need to avoid side-effects induced by past decisions, instances have no residual faulty items.

Instances also share the same demand distribution pattern. Specifically, all stores have a single demand request that recurs every 5 days. However, demands may occur on different days for different sites. The actual variability amongst instances comes from the volume and allocation of healthy stocks which we generated using different schemes. As for stock allocation, healthy items are either all placed in the DC (scheme `DC`), all placed elsewhere (scheme `Stores`) or evenly distributed between the DC and the other stores (scheme `Mix`). For the last two schemes, allocation to stores is performed randomly using a uniform distribution law. In terms of volumes, the total number of items across all sites is either

set to 100% of the total demand (scheme `High`), 50% (scheme `Med`) or 0% (scheme `Low`). Combining the two schemes yield seven classes of instances (stock allocation is irrelevant for scheme `Low`). The unit costs are configured to respect the hierarchy used in the real case. Solving the backorders is the ultimate goal of the supply chain; thus its cost is set to 1000, sending an item to repair costs 100, store an item in a store or in a hub costs 10 and store an item in a DC costs 1. In these experiments, weights are represented as a triplet $< \omega^b, \omega^{m^r}, \omega^s >$. As explained before, in this study we ignore transfer costs and thus transfer weight. The MIP model is implemented using CPLEX 12.6 and tests run on a *i5-3380M @ 2.90GHz* architecture. We allocate a maximum running time of 1 hour as the model is ultimately supposed to be running on a sliding window independently on many thousands of items. In case the maximum time is reached, we retrieve the lowest bound found by the model.

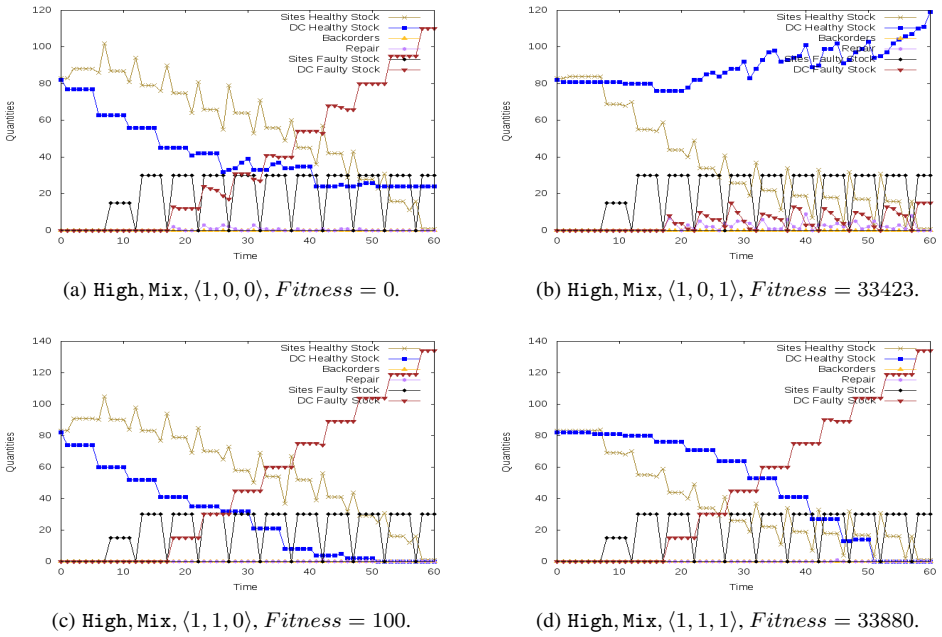
4.1 Policies

Supply chain management in major telecommunications companies is often based on global policies. It is motivated by different objectives such as customer satisfaction, costs or even advertising campaigns.

Classical Policies
In the case of a closed-loop supply chain, a policy can be seen as a triple of policies. The first one is the faulty items management policy. In our experimental study, this policy is fixed and can be ignored in this part. The second one is the repair policy. 3 main choices are possible: *repair everything*, *repair only when needed*, *never repair*. *Never repair* can be neglected as it would lead to an impossibility to meet demands. *Repair everything* is adopted by companies that want to ensure a maximum customer satisfaction, especially with a perfectible forecast. *Repair only when needed* is used by companies with a high quality forecast. It can also be applied when companies want to reduce the amount of items moving in the supply chain. The third one is the distribution policy. We distinguish two main policies, although they can be refined to fit more efficiently concrete cases. The well-known $just - in - time$ policy consists of dispatching the items just before the demands happen. This policy is efficient with a high quality forecast. At the contrary, another policy consists of sending items as soon as possible to the sites that will have demands. This can be adopted when uncertainties occur on when the demands are going to happen. Note that all the distribution policies described here are meant to be proactive. They should only be used with a reliable forecast otherwise it will lead to low customer service and mispositioning the items.

Weighting and Policies
We now analyse some of the generated plans to get insights into the impact of weightings and compare them with the previous policies. We consider the class of instances defined by schemes `High` and `Mix` and analyse the plans obtained with 3 weightings. For each plan, we study the evolution of different metrics over time, namely, the number of delayed demands (`Backorders`), the number of healthy and faulty items stored in the locations (`DC healthy stocks`, `DC faulty stocks`, `Site healthy stocks`, `Site faulty stocks`) and the number of repairs (`Repair`). Figure 8 plots the results.

For weighting $\langle 1, 0, 0 \rangle$, the only objective is to meet demands on time regardless of costs. The goal is consequently to minimise the quantity of backorders but



(a) `High`, `Mix`, $\langle 1, 0, 0 \rangle$, $Fitness = 0$.

(b) `High`, `Mix`, $\langle 1, 0, 1 \rangle$, $Fitness = 33423$.

(c) `High`, `Mix`, $\langle 1, 1, 0 \rangle$, $Fitness = 100$.

(d) `High`, `Mix`, $\langle 1, 1, 1 \rangle$, $Fitness = 33880$.

Figure 8 Policies analysis.

they may still occur in particular cases where they can not be physically solved. This objective is perfectly met as shown on Figure 8(a) (curve `Backorders`). Note that the stock of faulty items across stores and hubs is near-periodic (`Site faulty stocks`). This follows from the fact that faulty items are systematically sent back to the DC, the topology is regular and demand profiles similar for all sites. Since stock volumes are not penalised, we can also notice that unnecessary repairs and transfers from the DC are initiated. This weighting matches a basic customer satisfaction oriented policy.

Weighting $\langle 1, 0, 1 \rangle$ pursues an additional objective which is to minimise both backorders and storage in the supply chain and is presented in Figure 8(b). This weighting allows to simulate the $just - in - time$ distribution policy. Items are sent to sites just before demands occur. Note that this policy is due to the hierarchy of costs of our instances (hub and stores storage cost $> DC$ storage cost). With the opposite hierarchy, we would have simulated the other distribution policy. Regarding the repair policy, we note that repair transfers are performed randomly all along the time horizon. It can be assimilated to the *repair everything* policy as we avoid any storage cost while repairing.

Weighting $\langle 1, 1, 0 \rangle$ pursues a third objective which is to minimise $Repair$ in the supply chain and is presented on Figure 8(c). It prevents replenishment of the DC as shown by the opposite evolution of healthy and faulty stocks in the DC. Healthy stocks across sites decrease accordingly since the DC is the single feed (`Site healthy stocks`). Note that service level remains unaffected due to the high volume of initial items at the DC (`Backorders`). The repair policy can be assimilated to the *repair only when needed* policy while the distribution policy is more stochastic as long as we meet the demands in time.

The last weighting $\langle 1, 1, 1 \rangle$ (plotted on Figure 8(d)) minimises both lateness, repair and storage cost in the supply chain. Note that all demands are met in time. Demands are met by

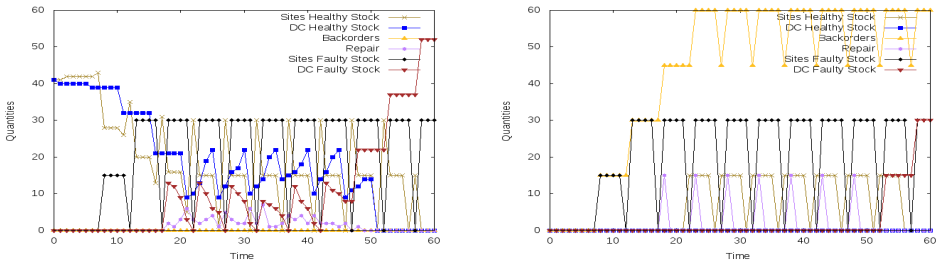
firstly consuming site inventories and secondly transferring healthy items to empty stores from the DC. Repairs are only performed in case there are not enough items in the supply chain to meet all the demands. In our case, only one repair transfer is necessary to avoid backorders. According to the hierarchy of the costs presented, items are kept in the DC as long as possible. The repair policy is *repair only when needed* and the distribution policy is $just - in - time$.

Note that we can easily perform both the distribution policies regardless the hierarchy of costs by dividing the $wStorage$ into two weights ($wSitestorage$ and $wDcstorage$) that represent respectively the weights associated with the costs to store an item in a hub or in a store, and the weight associated with the cost to store an item in a DC.

4.2 Experimental Results

Table 4 presents the results. The first two columns denote the schemes characterising each class of instances and the third column denotes the cost weightings used for the test. Experiments have been run on all combination of weightings. The remaining columns provide for the 2 sizes of instances (small and large), few information depending on the method used: *MIP - Fitness* is the best bound retrieved by the MIP in 1 hour, *MIP - Time* the run time of the MIP in milliseconds, *BIS - Fitness* the mean value of the metaheuristics on 100 iterations, *BIS - s.d.* the standard deviation of the metaheuristics, and *BIS - Time* the average run time of the metaheuristics in milliseconds. Note that if *MIP - Time* indicates $'-'$, it means that the MIP has reached the time limit. A bold value in the column *BIS - Fitness* means that either the MIP did not return a bound or that the bound returned is worse than the average value retrieved by the metaheuristics.

As expected, initial stock levels strongly influence service levels. Scheme `High` dominates scheme `Med` when the weight on backorders is activated for any given distribution scheme and weighting. Likewise, `Med` dominates `Low`. Figures 8(d), 9(a) and 9(b) describe the evolution of the metrics specified in 4.1 for 3 different stock levels with similar weightings. Stock distribution also plays a key role. Comparing when the weight on



(a) `Med`, `Mix`, $\langle 1, 1, 0 \rangle$, $Fitness = 30523$. (b) `Low`, $\langle 1, 1, 1 \rangle$, $Fitness = 2652165$.

Figure 9 Plan analysis.

backorders is activated again, distribution `DC` dominates `Mix` which dominates `Stores` for any given stock volume scheme and weighting. It highlights the importance of well locating items. This is particularly true in our case as healthy items cannot be sent back from the stores to the DC. Thus, bad positioning the items is a major risk and is highly depending on the quality of the forecast. When looking at the instances with weighting $\langle 1, 1, 1 \rangle$, we

Table 4 Comparison Mip vs Metaheuristics

			Small					Large				
		Weights	MIP		BIS (Meta)			MIP		BIS (Meta)		
			Fitness	Time	Fitness	s.d.	Time	Fitness	Time	Fitness	s.d.	Time
High	DC	<1,0,0>	0	4554	0	0	17	0	57000	0	0	220
		<0,1,0>	0	3205	0	0	3	-	-	**0**	0	42
		<0,0,1>	22725	4175	22726	6	8	99990	20500	99998	14	121
		<0,1,1>	23085	4555	23086	2	6	101574	23100	101581	13	80
		<1,1,0>	0	3562	2	14	15	0	32480	29	57	221
		<1,0,1>	25500	4554	25658	199	20	112200	-	113073	702	280
		<1,1,1>	25860	5870	25964	163	18	113784	-	114258	382	256
	Mix	<1,0,0>	0	2125	0	0	11	0	26220	0	0	147
		<0,1,0>	0	2085	0	0	3	0	10780	0	0	40
		<0,0,1>	31253	1182	31253	0	5	133862	2980	133869	50	70
		<0,1,1>	31613	1008	31613	1	3	135446	2840	135446	0	41
		<1,1,0>	100	3134	113	33	10	0	29500	12	35	143
		<1,0,1>	33423	2696	33573	193	14	143562	-	144348	587	191
		<1,1,1>	33880	4134	33959	169	12	145146	-	145508	389	164
	Sites	<1,0,0>	0	1072	0	0	5	27000	2730	27000	0	69
		<0,1,0>	0	820	0	0	3	0	2070	0	0	40
		<0,0,1>	58005	943	58005	0	6	256128	2030	256128	0	62
		<0,1,1>	58365	791	58365	0	3	257712	2030	257712	0	40
		<1,1,0>	1700	1329	1708	27	4	35600	3430	35625	48	71
		<1,0,1>	58565	1332	58585	59	6	285983	3510	286053	108	95
		<1,1,1>	60574	1423	60590	59	5	295909	3930	295995	137	78
Med	DC	<1,0,0>	0	5105	0	0	15	0	38000	0	0	221
		<0,1,0>	0	15161	0	0	3	-	-	**0**	0	40
		<0,0,1>	18326	7597	18327	2	8	80751	35410	80757	11	110
		<0,1,1>	18686	6863	18687	2	6	82335	31500	82343	18	82
		<1,1,0>	8300	-	8306	24	19	36300	-	36336	67	221
		<1,0,1>	21101	6321	21275	212	20	92961	-	93839	691	285
		<1,1,1>	29512	-	29650	192	19	129756	-	130237	379	252
	Mix	<1,0,0>	0	3120	0	0	13	0	25680	0	0	185
		<0,1,0>	0	9871	0	0	3	0	120500	0	0	40
		<0,0,1>	19407	1620	19408	6	6	88669	18250	88675	29	85
		<0,1,1>	19767	1569	19767	1	4	90253	14240	90272	55	54
		<1,1,0>	8300	-	8303	17	13	36300	-	36315	38	188
		<1,0,1>	22112	4543	22323	278	17	100339	-	101192	608	237
		<1,1,1>	30523	-	30623	130	15	-	-	**137643**	451	215
	Sites	<1,0,0>	31000	2542	31000	0	10	93000	11730	93000	0	148
		<0,1,0>	0	1536	0	0	3	0	4480	0	0	39
		<0,0,1>	25895	1261	25895	0	4	114648	6051	114648	0	63
		<0,1,1>	26255	1514	26255	0	3	116232	3900	116232	0	43
		<1,1,0>	39300	-	39303	17	12	129300	-	129310	36	148
		<1,0,1>	59100	3458	59141	78	13	217263	19240	217482	228	171
		<1,1,1>	67511	-	67564	81	12	254058	-	254342	281	163
Low	-	<1,0,0>	2625000	-	2625090	319	6	11550000	-	11550700	1923	94
		<0,1,0>	0	744	0	0	3	0	2000	0	0	40
		<0,0,1>	15045	1291	15045	0	4	66198	8410	66198	0	65
		<0,1,1>	15405	1408	15405	0	3	67782	8140	67782	0	40
		<1,1,0>	2635500	-	2635540	196	6	11603200	-	**11596400**	409	96
		<1,0,1>	2641620	-	2641710	289	7	11623128	-	11623500	1063	104
		<1,1,1>	2652165	-	2652160	2	7	11669526	-	11669900	1285	102

notice that the results returned by the scheme `High` and the scheme `Med` are close. It gives us the hint that the optimum quantity needed to solve all the backorders and limit the storage cost is between these two schemes. We can also notice on the graphs that the quantity of repair needed also vary a lot depending on the initial healthy stock level. Indeed, the more healthy items on the initial time bucket the less repairs you will have to proceed to meet the demands. However, running the algorithm on a really long time horizon will lead somehow

to a periodic repair loop. The distribution also affects the running time. For a particular scheme, distribution `DC` is slower than `Mix`, itself slower than `Stores`. Indeed, the more items are positioned in stores in the initial situation the less transfers you can perform. Hence, it indirectly cuts the search space and impacts the solving time.

We now compare the MIP with the metaheuristics. We first denote that in all cases the metaheuristics reaches at least once the best bound returned by the MIP. The results validate the well-functioning of the metaheuristics on simple instances.The standard deviation remains low and validate the stability of our method. This result was expected as the instances proposed here are symmetric and don't include any particular cases that can disallow the metaheuristics. In terms of scalability and running time the metaheuristics appears to be the best method. In most of the cases, the metaheuristics is at least 100 times faster than the MIP model. We note that the bigger the instances are, the harder it is for the MIP to complete. Indeed, it reaches the time limit in 10 cases on instance 25_165 and in 18 cases on instance 100_726. We also remark that on instance 100_726 the MIP fails to return a bound in 3 cases and in 1 case returns a higher bound than the mean of the metaheuristics. It confirms that the MIP faces a scalability issue due to the increasing number of possible transfers. On the contrary, the metaheuristics reaches very good quality solutions in a reasonable time.

5 Summary and Outlooks

This paper presented TDPP - an optimisation problem to address tactical distribution planning problem in CLSC. This problem is applicable to a wide variety of CLSC featuring different topologies, transfer policies or business objectives. We modelled TDPP as a MIP and showed that it is NP-Hard by reducing the Subset Sum problem. We also introduced a metaheuristic to solve TDPP. We discussed a real case from the Telecommunications domain and experimentally compared the MIP and the metaheuristic approaches on pseudo-random instances built on this real case. Experiments demonstrate the ability to implement classic planning strategies through appropriate weightings of the objective function. They also show that the MIP approach faces scalability issues whereas the metaheuristic appears to be both scalable and effective

Future directions will address how to match weightings with well-known inventory management policies in order to provide better decision support. We will also investigate the ability to build robust plans on a running window. Indeed, our model is working reasonably well in the absence of any uncertainty on problem data but in some cases, forecasts are inaccurate and the challenge is to produce robust and consistent plans over successive time periods. In this context, the proposed method is the central part of a periodic planning decision support system presented in Figure 2. One of the next steps of our work is to evaluate the impact of the recomputation frequency to face uncertainties while keeping consistency in successive plans.

References

Akçalı, E., Çetinkaya, S., and Üster, H. (2009). Network design for reverse and closed-loop supply chains: An annotated bibliography of models and solution approaches. *Networks*, 53(3):231–248.

Bertazzi, L. and Speranza, M. G. (2012). Inventory routing problems: an introduction. *EURO Journal on Transportation and Logistics*, 1(4):307–326.

Campbell, A., Clarke, L., Kleywegt, A., and Savelsbergh, M. (1998). The inventory routing problem. In *Fleet management and logistics*, pages 95–113. Springer.

Cattani, K., Jacobs, F., and Schoenfelder, J. (2011). Common inventory modelling assumptions that fall short: arborescent networks, poisson demand, and single echelon approximations. *Journal of Operations Management*, 29(5):488–499.

Coelho, L. C., Cordeau, J.-F., and Laporte, G. (2013). Thirty years of inventory routing. *Transportation Science*, 48(1):1–19.

Cornelli, M., Gourgand, M., and Lemoine, D. (2006). A review of tactical planning models. In *Service Systems and Service Management, 2006 International Conference on*, volume 1, pages 594–600. IEEE.

Desport, P., Lardeux, F., and Lesaint, D. (2015). Tactical inventory planning in the telecommunications service industry : a case study. *Roadef Marseille France*.

Drexl, A. and Kimms, A. (1997). Lot sizing and scheduling—survey and extensions. *European Journal of Operational Research*, 99(2):221–235.

Fleischmann, M., Bloemhof-Ruwaard, J. M., Dekker, R., Van der Laan, E., Van Nunen, J. A., and Van Wassenhove, L. N. (1997). Quantitative models for reverse logistics: A review. *European journal of operational research*, 103(1):1–17.

Fontes, D. B. and Gonçalves, J. F. (2007). Heuristic solutions for general concave minimum cost network flow problems. *Networks*, 50(1):67–76.

Garey, M. R. and Johnson, D. S. (1990). *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA.

Gendreau, M. and Potvin, J.-Y. (2010). *Handbook of Metaheuristics*. Springer Publishing Company, Incorporated, 2nd edition.

Goldberg, A. V., Tardos, É., and Tarjan, R. E. (1989). Network flow algorithms. Technical report, DTIC Document.

Govindan, K., Soleimani, H., and 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.

Guide, V. and Van Wassenhove, L. (2006). Closed-loop supply chains: An introduction to the feature issue (part 1). *Production and Operations Management*, 15(3):345–350.

Guide, V. D. R., Harrison, T. P., and Van Wassenhove, L. N. (2003). The challenge of closed-loop supply chains. *Interfaces*, 33(6):3–6.

Guisewite, G. and Pardalos, P. (1990). Minimum concave-cost network flow problems: Applications, complexity, and algorithms. *Annals of Operations Research*, 25(1):75–99.

Gupta, A., Tewari, P., and Garg, R. (2013). Inventory models and their selection parameters: a critical review. *International Journal of Intelligent Enterprise*, 2(1):1–20.

Ilgin, M. A. and Gupta, S. M. (2010). Environmentally conscious manufacturing and product recovery (ecmpro): a review of the state of the art. *Journal of environmental management*, 91(3):563–591.

Kellerer, H., Pferschy, U., and Pisinger, D. (2004). *Introduction to NP-Completeness of knapsack problems*. Springer.

Krikke, H., Bloemhof-Ruwaard, J., and Van Wassenhove, L. (2003). Concurrent product and closed-loop supply chain design with an application to refrigerators. *International journal of production research*, 41(16):3689–3719.

Lieckens, K., Colen, P. J., and Lambrecht, M. (2013). Optimization of a stochastic remanufacturing network with an exchange option. *Decision Support Systems*, 54(4):1548–1557.

O'Kelly, M. E. (1998). A geographer's analysis of hub-and-spoke networks. *Journal of transport Geography*, 6(3):171–186.

Pardalos, P. M. (1993). *Complexity in numerical optimization*. World Scientific.

Savaskan, R., Bhattacharya, S., and Van Wassenhove, L. (2004). Closed-loop supply chain models with product remanufacturing. *Management science*, 50(2):239–252.

Stindt, D. and Sahamie, R. (2014). Review of research on closed loop supply chain management in the process industry. *Flexible Services and Manufacturing Journal*, 26(1-2):268–293.

Stock, J. (1992). *Development and Implementation of Reverse Logistics Programs*. Council of Logistics Management.