

**2007/49**



Exact methods for the single-item multi-plant  
capacitated lot sizing problem coupled  
with transportation

Ayşe Akbalık and Bernard Penz

CORE DISCUSSION PAPER  
2007/49

**Exact methods for the single-item multi-plant capacitated  
lot sizing problem coupled with transportation**

Ayse AKBALIK<sup>1</sup> and Bernard PENZ<sup>2</sup>

July 2007

**Abstract**

In this paper we study the integration of production, transportation and storage decisions in a multi plant-distribution center supply chain structure. Multiple plants produce one type of item, each of them with different production capacity and costs, and send finished goods to the distribution center (DC) using capacitated vehicles. Customer demand is known at DC level and has to be satisfied without backlogging. This problem contains classical capacitated lot sizing problem as a subpart, which, in the general case, is NP-hard. We propose an exact pseudo-polynomial dynamic programming algorithm and show that the problem is NP-hard in the ordinary sense. We then compare the computational time of this dynamic program to that of a mixed integer linear program (MILP) which is selected among 4 different MILP formulations based on its lower computational time.

**Keywords:** multi-plant production, transportation, single-item capacitated lot sizing problem, dynamic program.

---

<sup>1</sup> CORE, Université catholique de Louvain, Belgium. E-mail: ayse.akbalik@uclouvain.be

<sup>2</sup> G-SCOP, INPG, Grenoble, France. E-mail: bernard.penz@inpg.fr

This paper presents research results of the Belgian Program on Interuniversity Poles of Attraction initiated by the Belgian State, Prime Minister's Office, Science Policy Programming. The scientific responsibility is assumed by the authors.

# 1 Introduction

Firms have continuously searched for ways to optimize their supply chain, *i.e.*, to minimize operational costs generated in each stage of the chain. In recent years, with an on-growing trend of delocalization of firms in different regions, transportation and inventory costs have become as important as production costs. This fact forces firms to optimize their entire chain activities globally rather than sequentially. Simultaneous optimization is based on the integration of all activity constraints into a single model to solve it in one time, contrary to the sequential optimization where each activity is planned independently. Research conducted until now on the integrated approach show the significant gain obtained in terms of the total cost and the reduction of inefficiencies and delays. For example Chandra and Fisher [2] show a reduction of 3% - 20% on the total cost after a simultaneous approach, rather than a sequential approach. Park [12] has a similar approach as Chandra and Fisher. The author compares total cost obtained by a simultaneous approach, where production is planned under transportation and production constraints, to the total cost obtained by a sequential approach, where production is optimized firstly and followed by the distribution optimization.

There are also many firms trying to integrate planning decisions of independently managed activities of their chain. In the literature one can find some real case studies. Matta and Miller [11] coordinate production, storage and transportation decisions between two plants. They study the influence of different parameters on the integration and on the total profit, using data obtained from a pharmaceutical firm. Gnoni *et al.* [6] introduce an industrial case concerning the supply chain of a multi-site, multi-product and multi-period manufacturing system, producing braking equipment components for the automotive industry. They propose an hybrid model (analytical + simulation model) to solve it.

In most of the cases, authors propose mixed integer programs to solve these integrated problems which become computationally intractable with the addition of many constraints. They propose then some relaxation or decomposition techniques or heuristics to solve them. Doğan and Goetschalckx [3] study a multi-period model of production and distribution, for a supply chain with potential suppliers, potential plants, distribution centers and customers, and solve the problem using Benders' decomposition. Wu and Gölbası [17] propose a multi-commodity flow network for a structure constituted of capacitated multi-plants and multi-products. They firstly analyze the single commodity model after the relaxation of the capacity constraints, then they propose a Lagrangean method for the integrated model.

For more detailed information on the studies done in integrated supply chain management, the reader can refer to the following reviews. Thomas and Griffin [15] give an overview of the papers in the area of the coordinated supply chain management and also some operational and strategic models. Sarmiento and Nagi [14] make a study on the integrated analysis of production-distribution systems. Erengüç *et al.* [4] propose a review on the integrated production and distribution planning in supply chains.

There are also some papers presenting polynomial time algorithms, sometimes based

on dynamic programming approach, in order to solve integrated models. Van Hoesel *et al.* [16] study a serial supply chain, with multiple levels of storage, and give polynomial time algorithms under some hypotheses on the costs and production capacities. Kaminsky *et al.* [8] analyze also a serial supply chain constituted of two capacitated production stages and the transportation and storage between these stages. They propose some polynomial time algorithms under different cost structures. Lee *et al.* [9] analyze an integrated inventory replenishment and outbound dispatch scheduling problem. They study a structure composed of one manufacturer supplying a warehouse, which then delivers to some distribution centers. In the first transportation echelon there is only a fixed cost generated if a positive amount is delivered, but between the warehouse and DCs each vehicle generates a fixed transportation cost. The authors propose a network approach to solve the problem and propose polynomial time algorithms using some optimal solution properties.

Li *et al.* [10] study two variants of the dynamic economic lot sizing problem. In the first one, the production is restricted to a multiple of a fixed batch size in each period, backlogging is allowed and all cost parameters are time varying. In the second one, a general form of product order cost structure is studied. Some polynomial time algorithms are given for these cases. Jin and Muriel [7] study a structure of one warehouse receiving a single product from a supplier and replenishing the inventory of  $n$  retailers with direct shipments. Ordering from the supplier and shipping to the retailers generate full truckload transportation costs with cargo capacity constraints. Giving some optimal solution properties, they study decentralized and centralized systems. In the decentralized system, each retailer and the warehouse decide how and when to replenish in an independent way which makes it easier than the centralized system where the decisions are interdependent. The authors propose an algorithm in  $\mathcal{O}(nT^2)$  with  $n$  being the number of retailers and  $T$  the number of periods. For the centralized system and single retailer case they propose an algorithm with complexity in  $\mathcal{O}(T^3)$ . They use Lagrangean decomposition to solve the multi-retailer model.

In this paper, we study the integrated planification of production, transportation and storage activities. The supply chain structure we analyze consists of multiple plants producing a single-item with a limited capacity, a distribution center (DC) to store the finished goods, and the transportation between plants and DC. This problem will be called MPP (multi plant planning) in the rest of the paper. A more detailed description of MPP structure, different hypotheses and cost assumptions is given in section 2.

One can easily see that the sub-structure of MPP with only one plant replenishing DC where the storage can take place is a special case of single-item capacitated lot sizing problem (CLSP). The lot sizing problem consists in finding optimal production quantities while minimizing production and holding costs. Batch sizes are computed as a function of capacity, demand and cost configurations. The reader can refer to Brahimi *et al.* [1] to learn more about lot sizing problem literature and its variants. See also Pochet and Wolsey [13] for the use of a mixed integer programming approach to solve efficiently production planning problems containing lot sizing substructure. For MPP, the production cost structure becomes more complicated with the possibility of production in

different plants. In addition to the capacitated multi-plant production planning, the fixed transportation cost associated to each vehicle shipped makes also the resolution harder than the classical CLSP, due to the step-wise cost structure. However, we propose an efficient way of computing optimal production and transportation quantities using a two-step pseudo-polynomial dynamic program. To compute one of these two steps, we use a similar dynamic programming formulation as the one proposed in the literature to solve the classical single-item CLSP.

The paper is organized as follows. In Section 2, a detailed description of the problem is given with all hypotheses made. A mathematical formulation by a mixed integer program and the idea of three other MILP formulations are also described, followed by the complexity result of MPP. In Section 3 a pseudo-polynomial dynamic program is given. Some computational experiments are reported in Section 4 and finally some concluding remarks are presented in Section 5.

## 2 Problem description and complexity

The structure of the supply chain studied in this paper contains multiple capacitated plants, a distribution center (DC) where the external demand is known and the transportation by capacitated vehicles between them (see figure 1). Shipments consist only of direct deliveries. The production activity generates a setup cost which is independent of the quantity produced and generates also a unit cost per unit of product. All finished goods produced at a given period are directly shipped to DC because of the absence of stock area in plants. Each vehicle shipped generates a fixed cost per vehicle and a unit cost per unit of product. With this assumption, one can easily deduce that one unit of finished product generates obligatory one unit of production cost and one unit of transportation cost, which will be integrated in the remaining of the paper under the notation  $p$ . If finished products are not used to satisfy demands at a given period, they are stored in DC and generate unit holding cost per unit of product and per period. However, demand delivered the same period to the customers do not generate any holding cost. All production, transportation costs and capacity limits are period and plant dependent. Holding cost is also time varying. The aim is to minimize the total cost of production, transportation and holding, while satisfying customer demands without backlogging.

The notation used in the paper and a mathematical formulation by a MILP are given as follows.

### Plant

- $N$ : number of plants,
- $C_{it}$ : production capacity of plant  $i$  in period  $t$ , assumed to be integral,
- $q_{it}$ : setup cost in plant  $i$  in period  $t$ ,
- $p_{it}$ : integrated unit production and transportation cost in plant  $i$  in period  $t$ ,

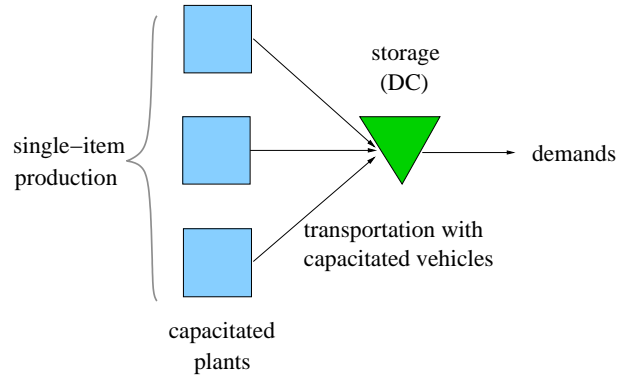


Figure 1: Multi-plant problem (MPP) structure.

### Transportation

- $f_{it}$ : fixed transportation cost generated by one vehicle shipment from plant  $i$  to DC in period  $t$ ,
- $V_{it}$ : capacity of the vehicle shipped from plant  $i$  to DC in period  $t$ , assumed to be integral,

### DC

- $h_t$ : unit inventory cost at DC at the beginning of period  $t$ ,

### Others

- $d_t$ : customers' aggregated demand in period  $t$ , known at DC level, assumed to be integral,
- $T$ : number of periods in the planning horizon,
- $\mathbf{1}_{\{x>0\}} = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{otherwise.} \end{cases}$

### Decision variables

- $x_{it}$ : production quantity of plant  $i$  in period  $t$ ,
- $s_t$ : inventory level at DC at the beginning of period  $t$ ,
- $y_{it} = \begin{cases} 1 & \text{if } x_{it} > 0, \\ 0 & \text{otherwise.} \end{cases}$
- $z_{it}$ : number of vehicles sent in period  $t$  from plant  $i$  to DC.

## 2.1 MILP formulation of the MPP

In this paper the aim of giving a MILP formulation is on one side to better describe the problem constraints and formalize it more clearly, and on the other side to provide an exact formulation in order to compare the computational time of the dynamic program proposed later in the paper. For single-item lot sizing substructure of MPP, four different formulations by MILP are given in the literature (see Pochet and Wolsey [13] for more details). In order to find the most efficient MILP formulation in terms of the execution time, we coded and compared these four formulations under different costs and capacity configurations for MPP (see section 4). The aim is to choose the fastest one before comparing its execution time to that of the dynamic program. In most cases AGG is found more efficient, thus we only give its formulation in this section. Some other instances where one of the other three formulations is faster are given in section 4. The decision variables of the other three formulations are also discussed very briefly in this section.

$$\min \sum_{t=1}^T (h_t s_t + \sum_{i=1}^N (q_{it} y_{it} + p_{it} x_{it} + f_{it} z_{it}))$$

*s.t.*

$$\sum_{i=1}^N x_{it} + s_t = s_{t+1} + d_t; \forall t \tag{1}$$

$$x_{it} \leq C_{it} y_{it}; \forall i, \forall t \tag{2}$$

$$x_{it} \leq V_{it} z_{it}; \forall i, \forall t \tag{3}$$

$$x_{it} \geq 0; \forall i, \forall t, s_t \geq 0; \forall t \tag{4}$$

$$y_{it} \in \{0, 1\}; \forall i, \forall t \tag{5}$$

$$z_{it} \in \mathbb{N}^+; \forall i, \forall t \tag{6}$$

The most natural formulation of the problem is given above (in the remaining of the paper we call it **AGG**-aggregated formulation). The objective function minimizes the total cost in the supply chain. Constraint (1) represents the material balance in each period at DC. The production capacity restriction and the value of the setup variables are fixed by constraint (2). In constraint (3), the number of vehicles sent from each plant to DC is computed as a function of the quantity produced in each period. Constraint (4) imposes the non-negativity restriction on the decision variables  $x_{it}$ ,  $s_t$  and constraints (5), (6) impose the integrality and non-negativity restrictions on  $y_{it}$ ,  $z_{it}$ .

We compared the performance of the above formulation with three other formulations. The first one is based on facility location problem formulation. Decision variables  $x_{it}$  become  $x_{iqt}$  which is the quantity to produce in plant  $i$  in period  $q$  to satisfy demand at DC in period  $t$ . One can write the objective function by eliminating either stock variable  $s$  or production variable  $x$ . Thus the facility location based formulation (**FAL**) can be entirely written using only decision variables  $x_{iqt}$ ,  $z_{iqt}$  and  $y_{iq}$ . For the single-

item uncapacitated lot sizing problem (ULSP), same authors have shown that the linear relaxation of FAL gives an integer optimal solution.

The second formulation is similar to FAL. It is based on a multi commodity formulation (**MC**) where each demand at a given period is considered as a separate commodity. The new decision variables are  $x_{iqt}$  and  $s_{qt}$ , respectively production quantity in period  $q$  to satisfy demand in  $t$ , in plant  $i$  and storage quantity at the end of period  $q$  destined to satisfy demand in  $t$ . The flow conservation constraint is written for each commodity. The linear relaxation of MC for the classical ULSP gives also an integer optimal solution. The third one is based on a shortest path formulation (**SP**) of ULSP. A new decision variable  $\phi_{qt}$  is integrated in the SP formulation: for the uncapacitated case  $\phi_{qt}$  is 1 if the cumulative demand  $\sum_{j=q}^t d_j > 0$  is produced in  $q$  to satisfy all demands between  $q$  and  $t$ , 0 otherwise. Additional capacity and multi-plant production and transportation constraints are added compared to SP for ULSP. This formulation is also very similar to the idea of the dynamic programming recursion given in section 3.2. Again for the linear relaxation of ULSP, SP formulation gives integer optimal solution. These 3 formulations loose their integrality property with the addition of production and transportation capacity restrictions.

As for the complexity of MPP, this problem includes a special case of the single-item capacitated lot sizing problem assuming only one production plant, transportation to DC and storage at DC. Florian *et al.* [5] give an NP-hardness proof for the single-item CLSP with equal demands and zero storage costs. The same authors show that single-item uncapacitated LSP remains NP-hard even for the case with equal demands, zero storage costs, zero setup cost and arbitrary cost functions. We assume that in MPP, all problem parameters can take arbitrary values. Thus, for the general case MPP, which contains single-item CLSP, is NP-hard. In the next section we propose a pseudo-polynomial dynamic program to solve MPP in its general form to optimality. The existence of a pseudo-polynomial dynamic programming algorithm makes the complexity of MPP NP-hard but only in the ordinary sense.

### 3 A pseudo-polynomial dynamic program

The dynamic programming algorithm is constituted of two parts: a general part and an assignment part. In the general part we compute the total minimum cost to pay over periods  $(1, \dots, T)$  for the entire supply chain. To be able to compute the optimal planning, one needs to know how to assign each production quantity to  $N$  plants at a given period. The assignment part ensures this computation. It is computed independently of the general part, for each feasible total production quantity, for each period. Once the total minimum cost to pay for producing any feasible quantity among  $N$  plants is known for any period  $t$ , this information is used in the optimal planning computing inside the general part.

Let us describe these steps in a more structured way. We begin by computing the assignment of production to each plant. For each period  $t$  over  $1, \dots, T$ , and for each feasible production quantity in  $[0, \sum_{i=1}^N C_{it}]$ , a minimum cost vector is computed, taking

into account only production and transportation costs. The upper bound of the feasible production quantity at  $t$  can be tightened to  $\min\{\sum_{j=t}^T d_j, \sum_{i=1}^N C_{it}\}$  using the assumption of zero storage level at the end of  $T$ . With this computation, for a given period  $t$ , we know exactly which quantity to produce in each plant for any feasible total production quantity. We have thus  $T$  minimum cost vectors. These vectors contain the information needed by the general part. As for the computation of the optimal planning, we consider a cumulated production capacity of all plants. Knowing the inventory quantity at the beginning of a period  $t$ , we decide which quantity to produce at this period. This time, the holding cost is also taken into account together with the minimum cost vector. For the recursive formula in the general part, we used a similar idea as Florian *et al.* [5] with the addition of transportation costs. In what follows we give more details on these two parts. In Appendix an example is given to illustrate the recursive formula of the dynamic program.

### 3.1 A pseudo-polynomial dynamic program for multi-plant production assignment

This computation is independent of the general part. For a given total production quantity in period  $t$ , the aim is to assign it to plants. Capacity and costs in each plant and for each vehicle fleet are period dependent, thus the computation is done for each period. The output is a set of  $T$  vectors, where each vector  $t$  has a dimension of the size of feasible production quantities ( $\min\{\sum_{j=t}^T d_j, \sum_{i=1}^N C_{it}\}$ ). Each of these  $T$  vectors contains the minimum total cost  $g_t(x)$  for a given quantity  $x$ .

To compute these vectors, we consider the state  $(t, i, x)$  corresponding to (period  $t$ , plant  $i$ , the quantity that can be produced in plants  $i, i + 1, \dots, N$ ). Before defining some lower and upper bounds, we give the following notation.  $LB_r(\bullet)$  : lower bound of  $r$  dependent on  $\bullet$ ,  $UB_r(\bullet)$  : upper bound of  $r$  dependent on  $\bullet$ .

For a given period  $t$  and plant  $i$ , the lower and upper bounds of  $x$  are respectively,  $LB_x(t, i)=0$  and  $UB_x(t, i)=\min\{\sum_{j=t}^T d_j, \sum_{j=i}^N C_{jt}\}$ . The lower bound  $LB_x(t, i)$  means that, it has been decided to produce all the orders in plants  $(1, \dots, i-1)$  and there remains nothing to produce in the plants  $(i, i + 1, \dots, N)$ . The upper bound  $UB_x(t, i)$  means that, it has been decided to produce nothing in plants  $(1, \dots, i-1)$  and there remains a quantity of  $\min\{\sum_{j=t}^T d_j, \sum_{j=i}^N C_{jt}\}$  to produce in the remaining plants  $(i, i + 1, \dots, N)$ .

For each state, one has to decide which quantity  $x_{i,t}$  to produce in plant  $i$  at  $t$  to minimize the total production and transportation costs. For a given state  $(t, i, x)$ , the value of  $x_{i,t}$  varies in the interval:  $[LB_{x_{i,t}}(t, i, x), UB_{x_{i,t}}(t, i, x)]$  where;  $LB_{x_{i,t}}(t, i, x) = \max\{0, x - \sum_{j=i+1}^N C_{jt}\}$  and  $UB_{x_{i,t}}(t, i, x) = \min\{x, C_{it}\}$ .  $LB_{x_{i,t}}(t, i, x)$  means that we have to produce at least the quantity which can not be satisfied in the remaining plants  $(i + 1, \dots, N)$ , and  $UB_{x_{i,t}}(t, i, x)$  is the minimum value between the production capacity of the plant  $i$  and the value  $x$  given in the state.

The objective function to minimize is  $g_t(x, i)$ :

$g_t(x, i)$ : The total minimum cost to pay to produce the quantity  $x$  in plants  $(i, i + 1, \dots, N)$  in period  $t$ . This cost is computed for each value  $x \in [LB_x(t, i), UB_x(t, i)]$ ,  $t \in [1, T]$  and  $i \in [1, N]$ .

Considering  $g_t(x, N + 1) = 0, \forall x, \forall t$ , backward recursion can be written as:

$$g_t(x, i) = \min_{x_{i,t}} \left\{ g_t(x - x_{i,t}, i + 1) + q_{it} \mathbf{1}_{\{x_{i,t} > 0\}} + p_{it} x_{i,t} + f_{it} \left\lceil \frac{x_{i,t}}{V_{it}} \right\rceil \right\} \quad (7)$$

where  $x_{i,t}$  has to be taken over the interval  $[LB_{x_{i,t}}(t, i, x), UB_{x_{i,t}}(t, i, x)]$ . As we see above, in the formulation of  $g_t(x, i)$ , we find the expression  $g_t(x - x_{i,t}, i + 1)$  which is the minimum total cost to pay for producing a quantity of  $x - x_{i,t}$  among the plants  $(i + 1, \dots, N)$ . Other expressions describe the total cost to pay for producing and transporting the quantity  $x_{i,t}$  in plant  $i$  in  $t$ .

The minimum cost vector to use in any state of the general formulation is  $g_t(x, 0), \forall x \in [0, \min\{\sum_{j=t}^T d_j, \sum_{i=1}^N C_{it}\}]$ . For short, we note  $g_t(x, 0)$  as  $g_t(x)$ .

### 3.2 General pseudo-polynomial dynamic program

To formulate the algorithm, consider  $T$  periods. In each period  $t$  there are  $BS_s(t)$  states, where  $BS_s(t)$  is the maximum quantity that can be stored at the beginning of a given period  $t$ . At a given period  $t$ , the beginning inventory can not exceed the maximum quantity that can be stored from the beginning of the horizon until  $t$  and also the total remaining demand to satisfy from  $t$  to  $T$ . Mathematically saying:

$$BS_s(t) = \min \left\{ \sum_{k=1}^{t-1} \left( \sum_{i=1}^N C_{ik} - d_k \right), \sum_{k=t}^T d_k \right\}.$$

Each state  $(t, s_t)$  means (period  $t$ , inventory level at the beginning of period  $t$ ), where  $s_t \in [0, BS_s(t)]$ . The decision is made on the quantity  $x_t$  that must be produced for a state  $(t, s_t)$  in order to minimize the total cost. The lower and the upper bounds of this quantity  $x_t$  are respectively,  $\max\{0, d_t - s_t\}$  and  $\min\{\sum_{i=1}^N C_{it}, \sum_{k=t}^T d_k\} - s_t$ , because one has to satisfy at least the demand at  $t$  which is not satisfied by the beginning inventory, and this production quantity can not exceed neither the total production capacity at  $t$  nor total demand in  $[t, T]$ , minus the quantity stored at the beginning of  $t$ . The aim is to find an optimal production planning which minimizes total production, transportation and storage costs. The objective function that minimizes the total cost is:

$\varphi_t(s_t)$ : The total minimum cost to pay for periods  $(t, t + 1, \dots, T)$ , knowing that there is a quantity  $s_t$  in the stock at the beginning of period  $t$ .

A backward recursion is carried out, assuming that  $\varphi_{T+1}(s_{T+1}) = 0$  with  $s_{T+1} = 0$ , we obtain the following formula:

$$\varphi_t(s_t) = \min_{x_t} \{h_{t+1}(s_t + x_t - d_t) + g_t(x_t) + \varphi_{t+1}(s_t + x_t - d_t)\}. \quad (8)$$

where the minimum in (8) has to be taken over all  $x_t \in [\max\{0, d_t - s_t\}, \min\{\sum_{i=1}^N C_{it}, \sum_{k=t}^T d_k\} - s_t]$ . In the formulation of  $\varphi_t(s_t)$ ,  $g_t(x_t)$  corresponds to the minimum cost value to produce a quantity  $x_t$  among plants  $\{1, \dots, N\}$  at  $t$  (see the previous section). The total minimum cost to pay over periods  $(1, \dots, T)$  corresponds to  $\varphi_1(s_1)$ , considering that at the beginning of the period 1, the stock level is  $s_1$ . We thus obtain an optimal planning taking into account production, transportation and storage constraints of the entire chain.

### 3.3 Algorithm complexity

The complexity of the recursive formulation in (8) is in  $\mathcal{O}(C^\Sigma \Sigma d^\Sigma)$  where  $C^\Sigma \Sigma = \sum_t \sum_i C_{it}$ , and  $d^\Sigma = \sum_t d_t$ . The computation of one state  $(t, s)$  is bounded by  $\min\{C^\Sigma, d^\Sigma\}$  where  $C^\Sigma = \sum_i C_{it}$ . The inventory level at the beginning of  $t$  is bounded by  $d^\Sigma$ . The recursive formulation complexity is thus  $\mathcal{O}(T d^\Sigma C^\Sigma) = \mathcal{O}(C^\Sigma \Sigma d^\Sigma)$ .

The complexity of the minimum cost vectors computation in (7) is in  $\mathcal{O}(N C^\Sigma \Sigma C_{max})$ . The state number is bounded by  $N C^\Sigma \Sigma$ . The computation of one state is bounded by  $C_{max}$  which is the maximum value of production capacity over horizon and over all plants. Thus, in total, the algorithm complexity is in  $\mathcal{O}(C^\Sigma \Sigma d^\Sigma + N C^\Sigma \Sigma C_{max})$  which is pseudo-polynomial.

## 4 Computational experiments

In this section, we present the results of the tests carried out with MILPs and the dynamic program. The aim is to compare the computational time of these methods under different parameters. Dynamic program is coded in Java, MILPs are coded using Java+Concert Technology and executed with Cplex 9.1. Tests are carried out on an Intel Xeon 3.2 GHz (bi-processor HT), 4 Gb RAM. For MILPs execution the gap to the optimal solution is fixed to 0.01%.

### 4.1 Experimental protocol

Firstly, we define the different instance sets used. Parameters influencing MILPs computational time are: number of periods, number of plants, demand configurations, production and vehicle capacities, different cost values. For the dynamic program, the parameters that influence the execution remain the same except cost configurations. Demand configurations and production capacities are given in Table 1, where their values are uniformly distributed in given intervals.

Demands are chosen as a function of the cumulative production capacity of plants. For example, D1 is used to generate a demand vector of size  $T$ , where demand values are randomly chosen in the interval  $U(0, 3 \cdot \sum_{i=1}^N C_{it}; 0, 8 \cdot \sum_{i=1}^N C_{it})$  for each period  $t$ , among integer values. Tests are carried out for 3 demand configurations: D1, D2, D3. In the

first one, demands are distributed far from the cumulative production capacity. In the second one they are closer to the capacity, and in the third one they are very tight. Plant production capacities are chosen in 3 intervals for each period: Small, medium and large capacities. Even if demands are scaled with capacities, for larger capacities production quantities have more variability.

Table 1: Instance sets.

<i>Demands</i>	<i>Production capacities</i>
<i>D1</i> : U(0.3, 0.6)	<i>C1</i> : U(8, 10)
<i>D2</i> : U(0.6, 1.1)	<i>C2</i> : U(80, 100)
<i>D3</i> : U(0.9, 1)	<i>C3</i> : U(800, 1000)

In Table 2, different cost configurations are given. Production setup cost is either fixed to a certain value or chosen from a uniformly distributed interval, and other costs are generated as a function of this setup cost.  $p_{it} = a \cdot q_{it}$ ,  $f_{it} = b \cdot q_{it}$ , and  $h_{it} = c \cdot q_{it}$ . For cost configurations (Cost1, Cost2, Cost3), setup cost is the same in each plant and period. In Cost2, transportation costs and in Cost3 holding costs are higher than setup cost. For Cost4, costs differ each period and for each plant significantly.

Table 2: Cost configurations.

SET	q	a	b	c
Cost1	30	0.01	0.1	0.001
Cost2	30	0.01	2	0.001
Cost3	30	0.01	0.1	2
Cost4	U(20,30)	0.01	0.1	0.001

## 4.2 Influence of the parameters on MILPs execution time

To obtain each value in Tables 3 to 10, 10 tests are carried out. In Tables 3 and 4, the instance chosen is  $T = 10$ ,  $N = 10$ , *Cost1*, *D1*. We give the relative gap between the best feasible solution and the best lower bound found after 1 min (see Table 3) and after 10 min (see Table 4), for each MILP formulation. Almost for all instances, the AGG gap is lower than the others. For instance, for demand configuration *D1* and production capacity *C1*, the gap of AGG varies between 0% and 0.6%. We observe that after a few seconds we obtain already this gap, which, even after 10 min can not be reduced significantly (see gap values in Table 4). A positive gap can remain even after 30 mins of execution. We also remark that with bigger variabilities on production quantities (for *P3*), this gap is smaller. On some instances the other formulations can be faster than AGG, for reasons that remain difficult to state.

Table 5 is given for demand configurations *D2* and *D3*, with a time limit of 1 min. This limitation is determined as a function of the maximum time that the dynamic

Table 3: Influence of production capacities and demands on MILP exec.time (*Cost1*).

		Gap LB-UB after 1min			
		AGG	FAL	MC	SP
D1	C1	[0-0.6]%	[0-0.6]%	[0-0.7]%	[0-0.6]%
	C2	[0-0.25]%	[0-0.3]%	[0-0.4]%	[0-0.3]%
	C3	[0-0.05]%	[0-0.18]%	[0-0.15]%	[0-0.1]%

Table 4: Influence of production capacities and demands on MILP exec.time (*Cost1*).

		Gap LB-UB after 10min			
		AGG	FAL	MC	SP
D1	C1	[0-0.55]%	[0-0.6]%	[0-0.5]%	[0-0.4]%
	C2	[0-0.2]%	[0-0.2]%	[0-0.2]%	[0-0.2]%
	C3	[0-0.04]%	[0-0.1]%	[0-0.09]%	[0-0.08]%

program takes under the same configurations. We remark that when average demands are closer to the production capacities, total execution times of all formulations are reduced, which is due to the reduction on the choice of production quantities. For D3, for which demands are very tight, the execution time is almost instantenous for all formulations.

Table 5: Influence of production capacities and demands on MILP exec.time (*Cost1*).

		Gap LB-UB after 1min			
		AGG	FAL	MC	SP
D2	C1	[0-0.2]%	[0-0.29]%	[0.1-0.44]%	[0-0.34]%
	C2	[0-0.05]%	[0-0.07]%	[0-0.1]%	[0-0.07]%
	C3	$\sim 0\%$	$\sim 0\%$	$\sim 0\%$	$\sim 0\%$
D3	C1	[0-0.25]%	[0-0.3]%	[0.1-0.32]%	[0.1-0.3]%
	C2	$\sim 0\%$	$\sim 0\%$	$\sim 0\%$	$\sim 0\%$
	C3	$\sim 0\%$	$\sim 0\%$	$\sim 0\%$	$\sim 0\%$

In Tables 3 to 5, the cost configuration was the same in each plant, each period, but in Table 6 we introduce some variability between plants in terms of cost configurations. For Table 6 the instance chosen is  $T = 10$ ,  $N = 10$ , *Cost4*. With this new cost configuration, all MILP formulations become very fast, they are all very close to the optimum in less than 1s. This can be due to the difference between feasible solution values, which allows to cut easily unnecessary solutions in the B&B tree.

In Table 7 the instance chosen is  $T = 10$ ,  $N = 10$ , *C1*. The aim of this table is to show the influence of the transportation and holding costs increase. The most interesting results in Table 7 are the values obtained with the cost configuration *Cost3*. When holding cost is very high compared to setup cost, the gap for 3 formulations AGG, FAL and MC can achieve 5% to 7% after 1 min of computing time. Even after 10 min, these gaps can not be reduced significantly. However, for the same instance, the gap for

Table 6: Influence of production capacities and demands on MILP exec.time (*Cost4*).

		Execution time			
		AGG	FAL	MC	SP
D1	C1	0.4s	0.2s	0.3s	0.1s
	C2	0.2s	0.7s	0.9s	0.2s
	C3	0.05s	0.3s	0.4s	0.07s
D2	C1	0.7s	1.1s	0.9s	0.5s
	C2	0.5s	1.5s	1s	0.7s
	C3	0.4s	2.5s	2.6s	0.2s
D3	C1	0.4s	0.3s	0.3s	0.2s
	C2	0.3s	0.8s	0.8s	0.3s
	C3	0.06s	0.09s	0.1s	0.08s

SP decreases very fast and we obtain solutions very close to the optimum.

Table 7: Influence of cost configurations and demands on MILPs execution time (C1).

		Gap LB-UB after 1min			
		AGG	FAL	MC	SP
D1	Cost1	[0-0.6]%	[0-0.6]%	[0-0.7]%	[0-0.6]%
	Cost2	[0-0.7]%	[0-0.7]%	[0-0.7]%	[0-0.63]%
	Cost3	[0.2-4.6]%	[3-6.2]%	[2.5-6.7]%	[0-0.05] %

For most of the cases, the AGG formulation runs faster than the others. Thus, we chose AGG formulation to compare its computational time to that of the dynamic program.

### 4.3 Comparison between AGG and dynamic program execution time

Demand and capacity configurations, number of periods and plants are changed to see their influence on the dynamic program execution time (see Table 8 for D1-D2, and Table 9 for D3). For small instances, the dynamic program execution time is instantaneous. In the same table, we give also the AGG gap after 1min. For a large number of periods and plants, tight demands and large production capacities, AGG takes less time. For the same instances (large T, N and P), the dynamic program execution time increases, which is due to its pseudopolynomial nature. Dynamic program is advantageous compared to MILPs, for small and medium instances, and particularly for cost configurations *Cost1* and *Cost3* coupled with small instances. We can make the same remarks on AGG time as before on the influence of tighter demand configurations.

In Table 10, we show some limits of the dynamic program with larger number of periods and plants. For instance, C=100, T=40 and N=60, dynamic program takes 1min. When C is increased to 1000 units, for the same instance, the dynamic program generates an out of memory error. For these instances, AGG generates a solution with a

Table 8: Comparison of AGG and dynamic program execution times.

		D1			D2		
		$AGG^{max}$	$AGG^{moy}$	DP	$AGG^{max}$	$AGG^{moy}$	DP
N=10, T=10	C1	0.6%	0.2%	0.06s	0.3%	0.12%	0.09s
	C2	0.25%	0.13%	0.47s	0.06%	0.04%	0.45s
	C3	0.05%	0.03%	46s	$\sim 0$	$\sim 0$	40s
N=10, T=20	C1	0.3%	0.12%	0.09s	0.1%	0.1%	0.1s
	C2	0.3%	0.15%	1s	0.07%	0.04%	0.9s
	C3	0.06%	0.03%	107s	0.03%	$\sim 0$	88s
N=20, T=20	C1	0.1%	0.05%	0.07s	0.09%	0.05%	0.06s
	C2	0.15%	0.1%	4.3s	0.1%	0.04%	4s
	C3	0.03%	0.01%	544s	$\sim 0$	$\sim 0$	374s

Table 9: Comparison of AGG and dynamic program execution times.

		D3		
		$AGG^{max}$	$AGG^{moy}$	DP
N=10, T=10	C1	0.3%	0.15%	0.06s
	C2	0.03%	$\sim 0$	0.44s
	C3	$\sim 0$	$\sim 0$	43s
N=10, T=20	C1	0.16%	0.1%	0.09s
	C2	0.05%	0.03%	0.9s
	C3	$\sim 0$	$\sim 0$	86s
N=20, T=20	C1	0.05%	0.03%	0.06s
	C2	0.05%	0.02%	3.8s
	C3	$\sim 0$	$\sim 0$	365s

gap of 0.1% in 1min.

Table 10: Influence of period-plant number and capacity on DP time.

	T=20, N=40	T=20, N=60	T=40, N=60
C1	0.3 s	0.3 s	0.9 s
C2	8.5 s	13 s	64 s
C3	1060 s	OM	OM

## 5 Concluding remarks

We studied the integrated planning of production and transportation of a single-item in a multi-plant environment. These two activities, with cost and capacity configurations that are dependent on each plant, are coupled with the storage activity in the unique distribution center. This problem can be seen as a generalization of the single-item

capacitated lot sizing problem with a more complicated capacity and cost structure due to plant dependent capacity limitation and piecewise cost structure in each plant. One can also interpret the production planning for the subpart of MPP with only 1-plant-DC as a generalization of CLSP. In the general case, MPP is thus NP-hard. The problem consists in finding an optimal production planning while minimizing production, transportation and storage costs in the entire chain.

Compared to the classical CLSP, the difficulty arises in computing total production and transportation costs, which consists in assigning some feasible production quantity to a finite number of plants, in each period. For this computation, we found a pseudo-polynomial dynamic programming algorithm, which can be computed independently from the rest of the general algorithm. The general algorithm itself consists in deciding the optimal production quantity and the storage level in each time period. Different ways of computing this general algorithm can be found in the literature of single-item CLSP, which is also formulated by a pseudo-polynomial dynamic program. For the total cost computation in the entire chain, there exists thus a pseudo-polynomial time dynamic program which consists of two independent algorithms. The complexity of MPP is thus NP-hard, but only in the ordinary sense.

We formulated the same problem by mixed integer linear programs. We found four MILP formulations in the literature of CLSP, which we adapted to MPP. We coded and tested them under different cost and capacity configurations, in order to choose the most efficient in terms of the execution time. We then compared the execution time of the most efficient MILP to that of the dynamic program. We gave instances where each method is more efficient. We showed also the limits of each one. Execution time of MILPs is quite unpredictable for different instances. All problem parameters influence their execution time. However, for some specific instances, we could observe similar values and interpreted these values regrouping them in tables. One can predict easily the execution time of the dynamic programming algorithm looking at its complexity. Particularly, when production capacities are small-medium size and cost configurations are hard for MILPs (ie *Cost1* or *Cost3*), dynamic program can be very advantageous. The main advantage of DP versus MILPs is that its execution time does not depend on cost configurations. However, for larger instances (large production capacities, large number of periods and plants), dynamic program can take more time and even return out of memory errors. Reader can find the influence of different parameters on methods execution time in Section 4.

As perspectives, one can reduce the MILP execution time, using different extended formulations proposed by Pochet and Wolsey [13] for extensions of lot sizing problem. Better formulations may be found exploring the problem structure. Dynamic program recursion may also be improved with new dominance rules related to the problem.

**Acknowledgement** We are grateful to Y.Pochet for his careful reading of this paper.

## References

- [1] N. Brahimi, S. Dauzère-Pérès, N.M. Najid, and A. Nordli. Single item lot sizing problems. *European Journal of Operational Research*, 168(1):1–16, 2006.
- [2] P. Chandra and M. Fisher. Coordination of production and distribution planning. *European Journal of Operational Research*, 72:503–517, 1994.
- [3] K. Doğan and M. Goetschalckx. A primal decomposition method for the integrated design of multi-period production-distribution systems. *IIE Transactions*, 31:1027–1036, 1999.
- [4] S.S. Erengüç, N.C. Simpson, and A.J. Vakharia. Integrated production/distribution planning in supply chains: An invited review. *European Journal of Operational Research*, 115:219–236, 1999.
- [5] M. Florian, J.K. Lenstra, and A.H.G. Rinnooy Kan. Deterministic production planning: Algorithms and complexity. *Management Science*, 26(7):669–679, 1980.
- [6] M.G. Gnani, R. Iavagnilio, G. Mossa, G. Mummolo, and A. Di Leva. Production planning of a multi-site manufacturing system by hybrid modelling: A case study from the automotive industry. *International Journal of Production Economics*, 85:251–262, 2003.
- [7] Y. Jin and A. Muriel. Single-warehouse multi-retailer inventory systems with full truckload shipments. Technical report, Department of Mechanical and Industrial Engineering, University of Massachusetts, 2005.
- [8] P. Kaminsky and D. Simchi-Levi. Production and distribution lot sizing in a two stage supply chain. *IIE Transactions*, 35(11):1065–1075, 2003.
- [9] C.-Y. Lee, S. Çetinkaya, and W. Jaruphongsa. A dynamic model for inventory lot sizing and outbound shipment scheduling at a third-party warehouse. *Operations Research*, 51(5):735–747, 2003.
- [10] C.-L. Li, V.N. Hsu, and W.-Q. Xiao. Dynamic lot sizing with batch ordering and truckload discounts. *Operations Research*, 52(4):639–654, 2004.
- [11] R.D. Matta and T. Miller. Production and inter-facility transportation scheduling for a process industry. *European Journal of Operational Research*, 158:72–88, 2004.
- [12] Y.B. Park. An integrated approach for production and distribution planning in supply chain management. *International Journal of Production Research*, 43(6):1205–1224, 2005.
- [13] Y. Pochet and L.A. Wolsey. *Production Planning by Mixed Integer Programming*. Springer, 2006.

- [14] A.M. Sarmiento and R. Nagi. A review of integrated analysis of production–distribution systems. *IIE Transactions*, 31:1061–1074, 1999.
- [15] D.J. Thomas and P.M. Griffin. Coordinated supply chain management. *European Journal of Operational Research*, 94:1–15, 1996.
- [16] S. van Hoesel, H.E. Romeijn, D.R. Morales, and A.P.M. Wagelmans. Polynomial time algorithms for some multi-level lot sizing problems with production capacities. Technical report, Maastricht University, University of Florida, Tinbergen Institute, 2002.
- [17] S.D. Wu and H. Gölbaşı. Multi-item, multi-facility supply chain planning: Models, complexities and algorithms. *Computational Optimization and Applications*, 28:325–356, 2004.

## Appendix

### Example to illustrate the dynamic program

With the example below, we illustrate the multi-plant production assignment and the general dynamic program recursion. We firstly give the problem instance. We consider 4 periods and 3 plants. Demands to satisfy over 4 periods are  $d=[4,3,5,4]$ . All costs and capacities are plant and period dependent and their values are given in Tables 11 and 12. Holding cost is given as  $h=[0.01,0.02,0.02,0.01]$ .

Table 11: Cost configurations for the example.

		Setup costs (q)				Unit prod. costs (p)				Transp. costs (f)			
		t=1	t=2	t=3	t=4	t=1	t=2	t=3	t=4	t=1	t=2	t=3	t=4
plants	1	5	5.2	4	3	0.2	0.3	0.4	0.3	1	2	1	1.5
	2	5	6	4	4	0.3	0.2	0.3	0.3	1	2	1	2
	3	4.7	5	5.1	5	0.19	0.2	0.3	0.2	2	1	2	1

Table 12: Production and vehicle capacities.

		Production capacities				Vehicle capacities			
		t=1	t=2	t=3	t=4	t=1	t=2	t=3	t=4
plants	1	2	3	1	4	2	2	2	2
	2	3	1	3	3	3	2	3	2
	3	2	4	5	2	1	2	3	1

First of all we compute the assignment of production to each plant (see section 3.1). For each period  $t$  and for each feasible production quantity  $x_t$ , the minimum cost vector  $g_t(x_t)$  is computed. In this stage, the only costs we take into account are production and transportation costs. Let us firstly compute the upper bound  $\min\{\sum_{j=t}^T d_j, \sum_{i=1}^N C_{it}\}$  of  $x_t$  for a given  $t$  ( $x_t$  means total production quantity at  $t$  without specifying in which plants). Lower bound of each  $x_t$  is 0 ( $LB_{x_t} = 0$ ).

$$t = 1; UB_{x_1} = \min\{16, 7\} = 7$$

$$t = 2; UB_{x_2} = \min\{12, 8\} = 8$$

$$t = 3; UB_{x_3} = \min\{9, 9\} = 9$$

$$t = 4; UB_{x_4} = \min\{4, 9\} = 4$$

With the recursive formulæ (7), one can obtain  $T$  minimum cost vectors  $g_t(x)$  for each  $x$  value in the interval  $[LB_{x_t}, UB_{x_t}]$  given above. In Table 13 we give  $g_t(x)$  values computed for each  $t$  and  $x$ .

To not complicate Table 13, we did not give detailed information on each value, but with this computation we know exactly which quantity  $x_{it}$  to produce in each plant  $i$  at

Table 13: Minimum cost values  $g_t(x)$ , (see section 3.1).

	Feasible production quantities (x)									
	0	1	2	3	4	5	6	7	8	9
$t = 1$	0	6.2	6.4	6.9	13	13.3	20.19	22.38	-	-
$t = 2$	0	6.2	6.4	7.6	7.8	15.3	15.6	17.9	26.1	-
$t = 3$	0	5.3	5.6	5.9	10.29	10.6	13.9	16.2	16.5	21.9
$t = 4$	0	4.8	5.1	6.9	7.2	-	-	-	-	-

a given period  $t$ . For example, for  $t = 1$ ,  $x = 5$  is composed of  $x_{11} = 2$  and  $x_{21} = 3$ . We will now show how to use these vectors in the computation of the general part. In the general computation, we are given the inventory quantity  $s_t$  at the beginning of the period  $t$  and we decide the quantity  $x_t$  to produce at this period. In Table 14 we give total minimum costs in each state. For example, for  $t = 3$  and  $s_3 = 2$ ,  $\varphi_3(2) = 13.1$  is the total cost to pay from period 3 to 4 with an initial stock of 2 units at the beginning of period 3.

Table 14: Total minimum cost computing for the entire chain, (see section 3.2).

	Inventory levels $s_t$ at the beginning of period $t$									
	0	1	2	3	4	5	6	7	8	9
$t = 1$	34.24*	-	-	-	-	-	-	-	-	-
$t = 2$	24.38	20.94	20.66	17.8	15.1	-	-	-	-	-
$t = 3$	17.8	16.58	13.1	12.8	10.68	7.2	5.98	5.14	4.86	0.08
$t = 4$	7.2	6.9	5.1	4.8	0	-	-	-	-	-

This time, to compute  $\varphi_t(s_t)$ , the holding cost is also taken into account together with the minimum cost vector  $g_t(x)$ . To detail how each value in the table is computed, we take:  $t = 3$ ,  $s_3 = 2$ ,  $\varphi_3(2) = 13.1$ . In the recursive formula we compute,

$$\varphi_t(s_t) = \min_{x_t} \{h_{t+1}(s_t + x_t - d_t) + g_t(x_t) + \varphi_{t+1}(s_t + x_t - d_t)\}$$

where  $x_t$  takes its values in  $[\max\{0, d_t - s_t\}, \min\{\sum_{i=1}^N C_{it}, \sum_{k=t}^T d_k\} - s_t]$ . For the instance chosen, this interval is  $[\max\{0, 5 - 2\}, \min\{9, 9\} - 2] = [3, 7]$ . Thus  $f_3(2)$  is computed for each value of  $x_3 \in \{3, \dots, 7\}$ .

$$x_3 = 3, f_3(2) = h_4(2 + 3 - 5) + g_3(3) + f_4(2 + 3 - 5) = 13.1^*$$

$$x_3 = 4, f_3(2) = h_4(2 + 4 - 5) + g_3(4) + f_4(2 + 4 - 5) = 17.2$$

$$x_3 = 5, f_3(2) = h_4(2 + 5 - 5) + g_3(5) + f_4(2 + 5 - 5) = 15.7$$

$$x_3 = 6, f_3(2) = h_4(2 + 6 - 5) + g_3(6) + f_4(2 + 6 - 5) = 18.7$$

$$x_3 = 7, f_3(2) = h_4(2 + 7 - 5) + g_3(7) + f_4(2 + 7 - 5) = 16.2$$

The minimum cost for  $t = 3$  and  $s_3 = 2$  is thus  $f_3^*(2) = 13.1$  with  $x_3^* = 3$ . The total minimum cost to pay over periods  $(1, \dots, T)$  corresponds to  $\varphi_1(s_1)$  with  $s_1 = 0$ .

In Table 14,  $\varphi_1(0)$  corresponds to 34.24 with  $x_1^* = 5$ . Optimal production quantities are then  $x^* = [5, 4, 3, 4]$  with  $x_{11}^* = 2$  and  $x_{21}^* = 3$  for the first period,  $x_{32}^* = 4$ ,  $x_{23}^* = 3$  and  $x_{14}^* = 4$  respectively for periods 2, 3 and 4.