Evaluation of solution approaches for a stochastic lot-sizing and sequencing problem

Kseniya Schemeleva, Xavier Delorme, Alexandre Dolgui

Evaluation of solution approaches for a stochastic lot-sizing and sequencing problem

Kseniya Schemeleva[a], Xavier Delorme[b], Alexandre Dolgui[c]

[a] *Univ Lyon, Université Lyon 2, LAET, F-69007, Lyon, France*
[b] *Mines Saint-Etienne, LIMOS UMR CNRS 6158, F-42023 Saint-Etienne, France*
[c] *IMT Atlantique, LS2N, CNRS, F-44307 Nantes, France*

Abstract

A stochastic multi-product lot-sizing and sequencing problem is studied. Two kinds of uncertainties are integrated into the model: defective items due to the process imperfections and random lead times because of randomly arising machine breakdowns and uncertain repair times. There are also sequence-dependent set-up times between two items of different types. The objective is to find a sequence of lots and lot sizes maximizing the probability of demand satisfaction for all products. A decomposition approach has been proposed in the literature to reduce this problem to a sequence of known optimization problems with different algorithms available for each of them. However, a proper evaluation of the practical performance of the whole method has never been presented. The goal of this paper is to analyze and compare the behavior of different solution frameworks (with and without decomposition) and techniques for the considered problem.

Keywords: Stochastic production lines, Lot-Sizing, Sequencing, Decomposition, Dynamic programming, Genetic Algorithms

1. Introduction

We study a lot-sizing and sequencing problem under uncertainty. The goal is to find optimal sequence of lots and lot sizes maximizing the probability to satisfy the whole demand, i.e. the demand for all product lots. In the literature the problem of demand satisfaction is often considered from cost point of view with the objective to minimize total backlog and holding cost. But in practice it is very difficult to accurately calculate the backlog cost because of indirect consequences of stock-outs such as potential losses of clients. When the average cost evaluation is not possible, or average cost criterion cannot be used for decisions, the service level criterion is often applied.

The impetus for this research initially came from the design of a fully automated production facility in the electronics industry that processes different conductor patterns and

Email addresses: kseniya.schemeleva@laet.ish-lyon.cnrs.fr (Kseniya Schemeleva), delorme@emse.fr (Xavier Delorme), alexandre.dolgui@mines-nantes.fr (Alexandre Dolgui)

assembles them into printed circuits. However, the conclusions drawn by our study could be of interest for other similar situations where the same random phenomena arise and the same objective function of service level maximization is used.

The facility considered is composed of (see figure 1): 1) a manufacturing line that processes items of several types; 2) an automatic storage device that stocks processed items; 3) an assembly line assembling final products with previously stored items. As shown in figure 1, the manufacturing line consists of m sequentially placed machines and is a paced flow line. Every item passes through all machines in the same fixed order. Lines producing electronic components are particularly subject to rejects, which cannot be fixed. Defective items are detected only after the last machine and are not placed into the storage system (they are excluded from the future process). When changing the product type, some amount of time (*set-up time*) is needed for setting up the machines. To perform this changeover, the processing of all items of the previous product type should be finished and the production line should be empty. The set-up time is sequence-dependent, i.e. it depends on both outgoing and incoming products. The machines are subject to breakdowns that involve line stoppages and engagement of repairs.

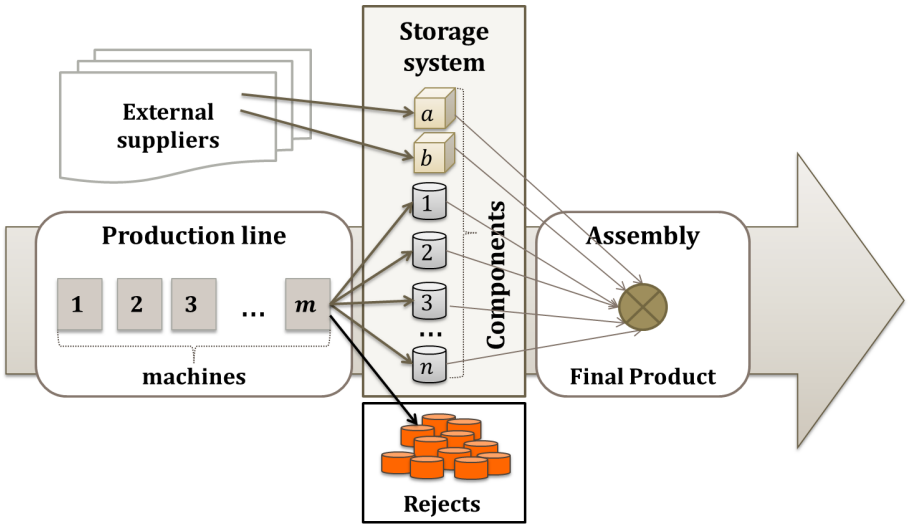

Figure 1: Production system

Capacity of the automatic storage system between production and assembly lines is limited by one day of stock approximately (one day is equal to 24 hours of autonomous work). Moreover, it is forbidden to keep the parts in this area more than a day otherwise they will degrade. So the whole facility should work in a *just-in-time* mode. It is necessary to ensure that all items needed for assembly day D are manufactured and loaded into the storage system at the end of day $D-1$. Otherwise, final products will not be delivered on time, and huge backlog costs will be incurred. Thus, the objective of lot-sizing and scheduling is to increase the probability to have all necessary items for the assembly process by the due date and thus to avoid stockout and assembly line stoppage.

We only consider the first part of a facility - production line which manufactures several types of items by lots with sequence dependent set-up times between lots. The decision to take daily is to determine how many items of each type (lot size) should be launched in the morning of the day $D-1$ on the production line to obtain all items needed for the assembly line by the end of the day. Some additional items of each product should be foreseen to compensate rejects, and some safety time should be added to perform machine repairs. Indeed, increasing the size of a given lot, increases the probability to obtain all required items of this type, but it reduces the time remaining for the subsequent lots and the time remaining for machine repairs (planned safety time) after failures. On the other hand, increasing the safety time reduces the time available for manufacturing. Furthermore, the chosen sequence of products to process also has an impact on the total processing time, and can increase or decrease the theoretical safety time intended for machines repairs.

We assume that :

- the demand level and unitary processing time are known for each type of product;
- defective items cannot be reworked, so they are rejected;
- the probability to obtain a quality item is given and can be different for different types of product;
- each machine is subject to failures and the Mean Time to Failure (MTTF) and Mean Time to Repair (MTTR) are also known.

This problem was originally considered in Dolgui et al. (2005). The authors presented probabilistic models of these uncertainties (rejects and breakdowns) and proposed a decomposition approach to solve the problem optimally but they didn't perform any experiment. The proposed decomposition approach is composed of three levels: 1) fix the type i of the last product $i = 1, \ldots, n$; 2) having last product i, find the sequence of other lots minimizing the total set-up time; 3) for the given sequence, find the sizes of lots maximizing the total probability to satisfy the overall demand. These three levels of decomposition lead to an exact optimal solution if at levels 2 and 3 the corresponding problems are solved to optimality. The first level is a *complete enumeration* of n possible solutions for the last lot. The *sequencing* decision (the second level of the approach) is equivalent to the Asymmetric Traveling Salesman Problem (ATSP). The *lot-sizing* one (the last level) is an extension of the Knapsack problem. Both the second and third level problems are NP-hard.

Dolgui et al. (2005) proposed to use a Dynamic Programming (DP) procedure to solve the lot-sizing part of the problem. A DP based method proposed in Dolgui et al. (2005) for lot-sizing part of the problem gives an optimal solution, but it is only able to solve problems with a very low number of lots. Thus for the instances of industrial scale the lot-sizing subproblem should be solved using approximate methods. Schemeleva et al. (2012) focused on the lot sizing problem and proposed a genetic algorithm. Hereby, till now only the overall decomposition scheme and corresponding lot-sizing problem was studied, the sequencing problem was left out since it was considered as well known.

The objective of this paper is to complete a cycle of work on this new problem and respond to the open questions concerning the global problem solution. The exact decomposition approach for the global problem has not been tested yet and no heuristics has been proposed. In our previous publications, we suggested interesting metaheuristics for lot-sizing part, and so to finish the study the question is: what are the most efficient techniques to solve the global problem?

This paper seeks to analyze the behavior and effectiveness of possible solution approaches. Our objective is to determine the limits of the optimal approach by decomposition proposed by Dolgui et al. (2005) and to propose three different resolution frameworks to obtain an approximate solution and evaluate their efficiency:

1. Skipping the enumeration step for the last lot in the decomposition approach, i.e. applying only the second and third level models.
2. Using the decomposition framework with heuristics for the lot-sizing problem, as suggested by Schemeleva et al. (2012), and for the scheduling problem.
3. Using a heuristic for the whole problem without decomposition.

While the decomposition implies tackling n resolution processes of similar problems, and could induce many redundant computations, a global heuristic means considering a larger decision space. Obviously, an intelligent search within the decision space should allow to avoid an increase of the computational time of the same magnitude than the size of the search space. As a consequence, it is not possible to know a priori which framework would be more efficient for this problem.

In order to achieve this goal, all these frameworks have been tested using algorithms from the literature, save for the latter for which a new memetic algorithm has been developed. Indeed, the purpose of this paper is not to evaluate each of these algorithms individually but rather to gather information on the possible resolution frameworks.

The rest of the paper is organized as follows. The problem statement and its mathematical model are presented in Section 2. Section 3 is designed to give a review of related literature. In section 4 we expose the resolution approach based on the decomposition. A short description of methods used to solve each sub-problem is given. A new MA is presented in Section 5. Further, Section 6 contains the numerical results and their analysis. Finally, Section 7 includes conclusion remarks.

2. Problem statement

We consider that the production line manufactures n types of products and consists of m machines. Each machine can treat one item at the same time and an item's transfer time between two adjacent machines is included in the processing time. At the beginning of a day the line is empty and is in the state 0 - initial set-up time is needed to adjust the machines to start the processing. Let for each type of product i, $i = 1, \ldots, n$ the following parameters are given:

- d_i - demand for items of type i;

- $t_{i,q}$ - processing time for an item of type i on machine q. We deal with the paced flow line, so without loss of generality we can consider that the processing time of an item of type i is the same for all machines and is equal to $t_i = \max\limits_{1 \leq q \leq m} \{t_{i,q}\}$, $q = 1, \ldots, m$ (see figure 2);

- $s_{i,j}$ - set-up time necessary to switch production from the product of type i for the product of type j, $s_{i,j} \geq 0$, $s_{i,i} = 0$, where $i, j = 1, \ldots, n$. As for the processing time, we consider that each set-up time is the longest duration needed for tools changing $s_{i,j} = \max\limits_{1 \leq q \leq m} \{s_{i,j,q}\}$, $q = 1, \ldots, m$ (see also figure 2);

- $s_{0,i}$ - initial set-up time to start the manufacturing process, if i is the first product to process, $s_{0,i} \geq 0$;

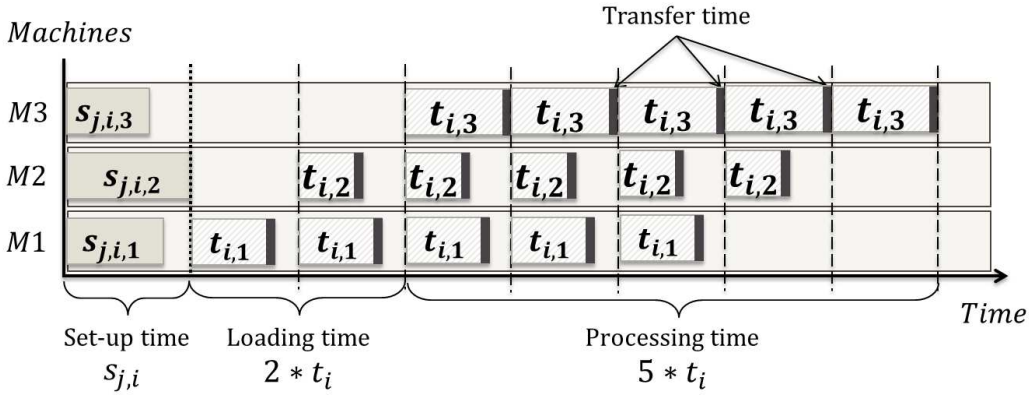

Figure 2: Set-up and processing time for product i, with $m = 3$ and 5 items of product i

We assume that triangle inequality is satisfied for the set-up times: $s_{i,k} \leq s_{i,j} + s_{j,k}$, $i = 0, \ldots, n$ and j, $k = 1, \ldots, n$. It is not difficult to show that each optimal solution of the sequencing sub-problem can be reorganized to include only one lot of each product, if it is not already the case.

The overall production process for each lot of product i contains the following steps:

- Set-up time $s_{j,i}$ (or initial set-up time $s_{0,i}$) to prepare the machines. During the set-up time, items processing is impossible;

- Loading time $(m-1) * t_i$ needed for the first item of each lot to achieve the last machine;

- Processing time t_i multiplied by the quantity x_i of items in the lot of product i .

Sometimes the normal production process is interrupted by breakdowns, in which case some repairing time is needed. We assume that breakdowns cannot appear during the set-up time. A schema of manufacturing line output is presented in figure 3.

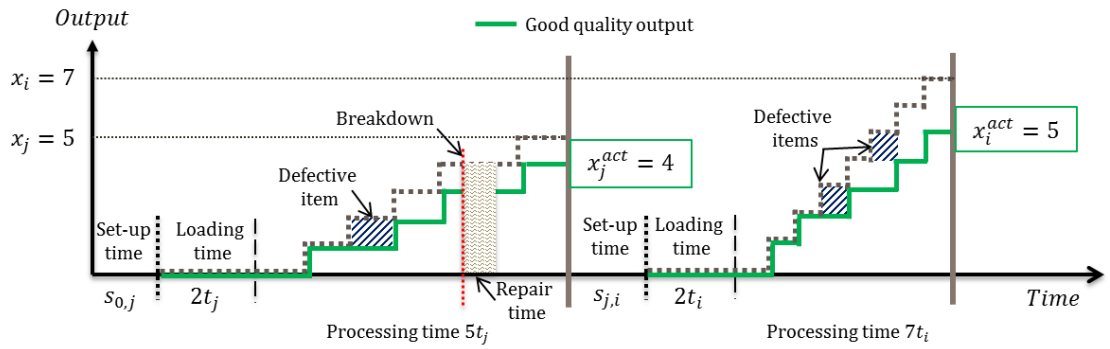

Figure 3: Example of output for lots j and i

The objective of the problem is to find the optimal sequence of lots and their sizes to maximize the probability of satisfying overall demand. This decision should be made at the beginning of each period, so we deal with a single-period model with T as a period duration.The decision variables are the sizes of lots $x = (x_1, x_2, \ldots, x_n)$, where x_i is the quantity of items of type i to produce, and their sequence $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$, where π_i is the type of product on the i^{th} position, $\pi_i \in \{1, \ldots, n\}$ and $\pi_i \neq \pi_j$ for $i \neq j,\ i, j = 1, \ldots, n$.

Remember, the objective is to maximize the probability to obtain at least d_i good items of each of products $i = 1, \ldots, n$:

$$\text{Maximize} \quad P\left(x_i^{act} \geq d_i,\ i = 1, \ldots, n |\ (x, \pi)\right) \tag{1}$$

where x_i^{act} is the quantity of items of a good quality obtained with x_i items launched. Note, if the lot size (x_i) was overestimated, the remaining items will be added into the planning of the work for the next period. We assume that the probabilities to obtain quality items of each type are known. We use the Bernoulli distribution to calculate the probabilities to have a given number of quality items of each type at the end of period taking into account the lot sizes.

The line is considered as one equivalent machine. Breakdowns and repairs of the line are expressed using two Poisson processes and renewal theory. Thus, the number of failures depends on the length of total processing time necessary to produce all lots and the breakdown and repair rates. The length of periods between two successive breakdown or repair events follows exponential probability densities. As it was mentioned above, the MTTR and MTTF of each machine are considered as known, so the MTTR and MTTF for the whole line can be calculated. The renewal process model proposed in Dolgui (2002) is used to calculate the probability that available cumulative working time (the sum of all periods of effective processing) is sufficient to process all lots.

If we know which lot will be the last one in the sequence, we can reformulate the objective function (1) as follows:

$$\text{Maximize} \quad \prod_{i=1}^{n-1} P\left(x_i^{act} \geq d_i \,|\, (x, \pi)\right) * P\left(x_n^{act} \geq d_n \,|\, (x, \pi), R\right) \tag{2}$$

where R is a cumulative repair time. In this formulation, we consider, without loss of generality, that all repair times are cumulated at the end of a planning period. Indeed, the lots are processed sequentially. An excessive cumulated repair time can consume the time of the last lot, which means there is no time left to launch the last lot. In this case the probability for the last lot will be equal to zero and the objective function will also be equal to zero. Thus, only the cases where the last lot can be processed, perhaps only partially, are considered in (2)). This reformulation shows that the probabilities to satisfy the demand for the first $n-1$ products (the first member of the product in equation (2)) depend only on rejects. Only the probability for the last lot with product n depends on rejects and cumulative repair time, depending in its turn on the total item's processing time.

Thus, if we fix the last lot the decision variables on sequence of lots and lot sizes become independent. If the last lot is not fixed these variables are dependent: the probability to satisfy the overall demand can be different for two solutions with equivalent set-up time if the last lot is not the same, and the optimal sequence of lots can change if we increase or decrease the quantity of items to process.

3. Literature review

The problem studied in this paper is recent and was stated in Dolgui (2002) and Dolgui et al. (2005). It concerns lot-sizing and sequencing decisions for a manufacturing line with imperfect production and sequence-dependent set-up times. Thus, it can be positioned on the intersection of two research domains: lot-sizing for imperfect production systems, and deterministic scheduling with batching.

3.1. Lot-sizing under uncertainty

Issues of lot-sizing under uncertainty are very common in industry and in literature. In most cases, uncertainty is related to the uncertain demand, random yield or random lead time. A state of the art of non-deterministic lot-sizing models was recently published by Aloulou et al. (2014). This review covers only recent papers, published since 2000. The authors proposed to use a five-field notation to classify lot-sizing models : number of periods, number of products, number of machines, uncertain parameters and modeling approaches. According to this classification, our problem can be represented as (1,n, m, $\{yie$, $lead\}$, $prob$).

Random yield occurs when the difference between the quantities of items launched and the quantity of items obtained is not known. A survey of the literature on the lot-sizing with random procurement and production yield can be found in Yano and Lee (1995). Another review concerning the MRP environments was given by Dolgui and Prodhon (2007). There exists a multitude of choices to model this type of uncertainty. One of the most common is probability distributions. The latest was used by Gerchak and Grosfeld-Nir (1998) in their problem of multiple lot-sizing. The Bernoulli process to model the random yield was used by Teunter and Flapper (2003). Another approach consists in using the proportional yield, where a certain percentage of items are defectives (quantity of defective items always depends on the lot size). If this percentage is fixed, the problem is deterministic and the

number of defective items in each lot can be calculated directly. Some examples of problems with proportional yield can be found in Papachristos and Konstantaras (2006), Wang and Gerchak (2000) and Dolgui et al. (2010). Sometimes, the interrupted geometric yield is used. The principal idea of this method is that the manufacturing process can become out of control with a given probability. In this case, all subsequent items are defective (see, for example, Salameh and Jaber (2000) and Maddah and Jaber (2008), etc.).

Random lead time is often a consequence of random processing times (see, for example, Çakanyildirim et al. (2000)), order and delivery delays (Arda and Hennet (2006)) or breakdowns. We focus here on breakdowns. The most used notions to model breakdowns are the Mean time to (between) failure(s) (MTTF or MTBF) and Mean time to repair (MTTR): Lin and Gong (2006) considered the Economic production quantity model with random breakdowns ($MTBF = 1/\lambda$, where λ is the breakdown rate which is a parameter of the exponential distribution) and constant repair times. Giri and Yun (2005) and Giri and Dohi (2004) examined the optimal lot-sizing problem with at most two failures during the planning period (both failures and repair times follow a general (arbitrary) distribution.

Both random yield and random lead times are considered in Wang and Gerchak (2000). Another example can be found in Dolgui et al. (2010), where the proportional yield was used to model both defective items, as a function of the size of the lot, and repair times, as a function of the total processing time.

3.2. Combined lot-sizing and scheduling problems

Deterministic problems combining both lot-sizing and scheduling decisions are widely represented in the literature. Sikora (1996) studied a problem of lot-sizing and sequencing with sequence dependent set-up times on a flow line issue from printed circuit board manufacturing. Allahverdi et al. (2008) proposed a review of scheduling with batching and set-up times and (or) costs for different line configurations. Palaniappan and Jawahar (2011) examined the minimization of total costs of procurement lot sizing and assembly scheduling for a flow line with sequence dependent setup times.

In the literature some authors treat this kind of problem without decomposition, namely by using meta-heuristics. As an example, we can cite Sikora (1996), where a genetic algorithm (GA) was developed for the lot-sizing and sequencing problem with set-ups. Allahverdi et al. (2008) pointed out in their survey that meta-heuristics are very effective for these joint lot-sizing and sequencing problems. Recently, Palaniappan and Jawahar (2011) proposed a GA for simultaneous lot-sizing and scheduling.

4. Resolution via Decomposition

To make the present paper self-sufficient, we give a brief description of the decomposition framework initially presented in Dolgui et al. (2005) and the algorithms chosen to solve each of the obtained sub-problems. This section is organized as follows. In subsection 4.1 three steps - enumerating, sequencing and lot-sizing - of decomposition is exposed. subsection 4.2 is intended to present several solution methods for the sequencing decision. We decided to use a linear model from Sherali and Driscoll (2002) to obtain an optimal solution (in 4.2.1)

and a genetic algorithm of Nagata and Soler (2012) to solve the problem approximately (subsection 4.2.2). Further, subsection 4.3 contains the methods proposed earlier to solve the lot-sizing part of the problem: exact algorithm for the lot-sizing part (DP procedure by Dolgui et al. (2005)) described in 4.3.1 and a meta-heuristic approach (MA) from Schemeleva et al. (2012) recapitulated in 4.3.2.

4.1. Decomposition approach

Let initial problem A be to maximize the probability of overall demand satisfying (2) and (x^*, π^*) its optimal solution, where $\pi_i \in \{1, \ldots, n\}$ and $x_i \in [x_i^{min}; x_i^{max}]$. We can define the inferior x_i^{min} and superior x_i^{max} limits for x_i, $i = 1, \ldots, n$ in the following manner:

$$x_i^{max} = \min_{z \geq d_i} \left(z | P\left(x_i^{act} \geq d_i\right) \geq 1 - \epsilon \right), \quad x_i^{min} = \min_{z \geq d_i} \left(z | P\left(x_i^{act} \geq d_i\right) \geq \beta \right) \tag{3}$$

where β is the *service level*, i.e. the minimum acceptable probability that the demand for a given product will be satisfied, and ϵ is a relatively small positive value. So x_i^{max} is the minimum lot size of product i such that the probability to satisfy the demand for the product i is close to 1; x_i^{min} is the minimum quantity of items of product i necessary to produce in order to obtain the probability to satisfy the corresponding demand not less than the required service level. Obviously, x_i^{max} is the maximum possible value of x_i taking into account a reasonable threshold.

Problem A can be decomposed into n equivalent sub-problems $A(i)$. Each $A(i)$, $i = 1, \ldots, n$ is to solve the problem (2) under condition of fixed last lot i, i.e. $\pi_n = i$. Once all of n problems $A(i)$ are solved, the optimal solution (x^*, π^*) of A is the best solution among $A(i)$, and can be obtained by a complete *enumeration*.

Each of $A(i)$ on its turn can be decomposed in two sub-problems:

- *Sequencing* $A(i)_{seq}$: by selection of a sequence for first $n-1$ products, planned safety time is modified because the total set-up time depends on the selected sequence. The planned safety time is equal to the duration of the planning period minus the set-up time and working time. Decreasing the total set-up time permits to increase the planned safety time and, subsequently, to increase the probability to repair all breakdowns within the safety time, so as to process all the lots by the end of the period. Thus, to optimize the sequencing decision, we should minimize the total set-up time:

$$A(i)_{seq}: \quad \text{Minimize} \quad s_{0,\pi_1} + \sum_{j=2}^{n} s_{\pi_{j-1},\pi_j}, \quad \pi_n \equiv i \tag{4}$$

  which is equivalent to the Asymmetric Traveling Salesman problem.

- *Lot-Sizing* $A(i)_{lot}$: consists of determining the optimal sizes of lots for a fixed sequence:

$$A(i)_{lot}: \quad \text{Maximize} \quad P\left(x_i^{act} \geq d_i,\ i = 1, \ldots, n \mid (x, \pi)\right) \tag{5}$$

  This problem is an extension of the Knapsack problem with a specific objective function.

Both sub-problems $A(i)_{seq}$ and $A(i)_{lot}$ are NP-hard.

4.2. Resolution of the sequencing part $A(i)_{seq}$ of problem

In the previous section it was pointed out that the sequencing sub-problem $A(i)_{seq}$ is equivalent to the Asymmetric Traveling Salesman problem, studied by numerous authors (see Choi et al. (2003), Xing et al. (2008), etc.). Thus, we decided to adapt some existing methods to solve it. First of all, we were looking for an effective in terms of CPU time linear model formulation to obtain an optimal solution. The chosen model is presented in section 4.2.1. Then we decided to implement a meta-heuristic algorithm (see subsection 4.2.2) which can provide a quality approximate solution but more rapidly.

4.2.1. A linear model formulation

A review of the literature has shown that the ATSP formulation proposed by Desrochers and Laporte in 1991 is still very effective. Recently, Sherali et al. (2006) introduced several new ATSP formulations and published the results of their comparison. Öncan et al. (2009) have also compared 24 different formulations of the ATSP (using some already proved domination relationships and proposing a several number of new ones) in terms of strengths of their LP relaxations. We were interested in a non-dominated models with low CPU time, so Sherali and Driscoll's model (proposed in Sherali and Driscoll (2002)) called ATSP-SD, was selected. Moreover, it was the fastest to find the lower bound with the barrier solver of CPLEX. To be sure about the effectiveness of this formulation for our problem instances, we have implemented another model - ATSP-O7 - taken from Sherali et al. (2006) based on different sub-tour eliminating constraints. Numerical results of preliminary tests proved that the ATSP-SD model is the fastest for our instances.

As was mentioned before, $A(i)_{seq}$ is an extension of the ATSP, so the chosen ATSP-SD model should be adapted to this extension. More precisely, compared to the standard ATSP problem, we have to find the *Hamiltonian path* always beginning from the vertex 0, because of the initial set-up time. The last lot (vertex) is fixed, so we should add several precedence constraints. For $i, j = 0, \ldots, n$ let:

$$x_{ij} = \begin{cases} 1 & \text{if arc } (i,j) \text{ belongs to the optimal solution} \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

$$u_i = \text{position of lot } i \text{ in the optimal path, } u_0 \equiv 0 \text{ - the } \textit{first lot} \text{ is fixed} \tag{7}$$

$$y_{ij} = u_i * x_{ij} = \begin{cases} \text{position (starting from 0)} & \\ \text{of the arc } (i,j) \text{ on the optimal tour,} & \textbf{if belong} \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

Note, y_{ij} are defined for $i, j = 1, \ldots, n$.

Our adapted ATSP-SD model can be formulated as follows:

$$\text{Minimize} \sum_{i=0}^{n} \sum_{j=0}^{n} s_{i,j} x_{ij} \tag{9}$$

Subject to:

$$\sum_{j=0,j\neq i}^{n} x_{ij} = 1 \quad \forall i = 0,\ldots,n \tag{10a}$$

$$\sum_{i=0,i\neq j}^{n} x_{ij} = 1 \quad \forall j = 0,\ldots,n \tag{10b}$$

$$\sum_{j=1,j\neq i}^{n} y_{ij} + nx_{i0} = u_i \quad i = 1,\ldots,n \tag{10c}$$

$$\sum_{i=1,i\neq j}^{n} y_{ij} = u_j - 1 \quad j = 1,\ldots,n \tag{10d}$$

$$y_{ij} \geq x_{ij} \quad i = 1,\ldots,n,\ j = 1,\ldots,n,\ i \neq j \tag{10e}$$

$$y_{ij} \leq (n-1)x_{ij} \quad i = 1,\ldots,n,\ j = 1,\ldots,n,\ i \neq j \tag{10f}$$

$$u_j + (n-1)x_{ij} + nx_{ji} - y_{ij} - y_{ji} \leq n \quad i = 1,\ldots,n,\ j = 1,\ldots,n,\ i \neq j \tag{10g}$$

$$x_{ji} + u_j - y_{ij} - y_{ji} \geq 1 \quad i = 1,\ldots,n,\ j = 1,\ldots,n,\ i \neq j \tag{10h}$$

$$u_j + x_{0j} - (n-2)x_{j0} \geq 2 \quad j = 1,\ldots,n \tag{10i}$$

$$u_j + (n-2)x_{0j} - x_{j0} \leq n-1 \quad j = 1,\ldots,n \tag{10j}$$

Let *last* be the number of the last lot (fixed). We have:

$$\begin{aligned}
&x_{ij} \in \{0,1\} \quad i,j = 0,\ldots,n \\
&x_{last,0} = 1 \\
&u_i, y_{ij} \in [0,n], \quad u_i, y_{ij} \in \mathbb{N}, \quad i,j = 0,\ldots,n \\
&u_{last} = n
\end{aligned}$$

In this model, (10a) and (10b) represent the standard assignment constraints assuring that each city appears only once in the optimal tour. The (10c) verifies that the position of the arc (i_1, i_2) in the tour is equal to the position of the lot of product i_1. If i_1 is the last lot, all y_{i_1,i_2} are equal to 0. Constraint (10d) complements the previous one by assuring that the position of any arc (i_1, i_2) is equal to the position of lot of product $i_2 - 1$ in the tour. Constraints (10e) and (10f) give the lower and upper bounding restrictions for y_{ij}: $x_{ij} \leq y_{ij} \leq (n-1)x_{ij}$. Indeed, if the arc (i,j) does not belong to the optimal tour, x_{ij} and y_{ij} equal to 0; otherwise, $x_{ij} = 1$ and inequalities $1 \leq y_{ij} \leq (n-1)$ are true because $y_{ij} \in [1; n-1]$ for each arc (i,j) of the optimal tour, $i > 0$. Constraints (10g) and (10h) are the bounding restrictions for y_{ij} and y_{ji}, and (10i) and (10j) are the lower and upper bounding restrictions for u_j.

The problem ATSP (and its extension also) is NP-hard, so exact methods stay effective up to a certain size. Beyond, explosion of calculation time and/or memory requires using an approximative method.

4.2.2. Genetic Algorithm for the ATSP problem

Applying meta-heuristics is very common in solving the ATSP problem. A survey of meta-heuristic algorithms for the ATSP and a new hybrid approach were proposed by Xing et al. (2008). Recently, Nagata and Soler (2012) published a MA for the ATSP. The experimentation (for 27 instances from TSPLB) shows that MA of Nagata and Soler (2012) is the most efficient comparing it with the most-known meta-heuristics (Choi et al. (2003), Buriol et al. (2004) and Xing et al. (2008)), so we have decided to use it. All steps were implemented in accordance with the considered problem:

- *Initial population*: N_{pop} solutions $h_1, \ldots, h_{N_{pop}}$ constituting the initial population are created by applying a variant of the *3-opt neighborhood* local search N_{pop} times. To make the local search procedure more reactive, the "don't look bits" strategy was employed (Bentley (1990)).

- *Crossover*: The *Edge Assembly Crossover (EAX)* operator was chosen for the crossover (see Nagata (2004)). Each chromosome h_i, $i = 1, \ldots, N_{pop}$ is randomly selected once for both parent p_a and p_b. A couple of parents can generate multiple offspring. Their number is limited by either the chromosomes structure or a parameter fixing the maximum number of offspring.

- *Selection*: If the best obtained offspring off_{best} is better than p_a and closer in terms of different arcs to p_a than to p_b, then p_a is replaced by off_{best}. If the off_{best} is more similar to p_b and is better than p_a and p_b, it replaces p_b in the population.

The algorithm stops when a termination condition is met: in our case - when the allocated time is elapsed.

4.3. Resolution of the lot-sizing sub-problem $A(i)_{lot}$

The lot-sizing sub-problem $A(i)_{lot}$ is to find solution $x = (x_1, \ldots, x_n)$ maximizing the probability of demand satisfying when the sequence of lots π is fixed. Dolgui et al. (2005) pointed out that this problem can be considered as a specific extension of the Knapsack problem. Consequently, it is NP-hard. The authors proposed a DP approach to find the optimal solution. Its short version is described in subsection 4.3.1. The results of numerical experiments on the DP were exposed in Schemeleva et al. (2012). Because of the huge computational expenditures, this algorithm can only be used for instances with up to 10 lots, which is often insufficient for industrial problems. So the authors proposed the MA approach to solve the problem. The main steps of this algorithm are described in subsection 4.3.2.

4.3.1. Dynamic programming

The main idea is the assignment of a time interval (corresponding processing time) for each lot i, $i = 1, \ldots, n-1$, such that $x_i^{min} t_i \leq$ time interval $\leq x_i^{max} t_i$. Further, let V^i be the time interval available to manufacture i first lots, $1 \leq i \leq n-1$, then

$$\sum_{j=1}^{i} x_j^{min} t_j \leq V_i \leq \min \left\{ \sum_{j=1}^{i} x_j^{max} t_j, T - S(\pi^*) - \sum_{j=i+1}^{n} x_j^{min} t_j \right\}, \tag{11}$$

where $S(\pi^*)$ is the total set-up time corresponding to the sequence π^*. Let $H^0(\cdot) \equiv 1$, then for $i = 1, \ldots, n-1$ the recurrent relation is as follows:

$$H^i(V^i) = \max_{\sum_{j=1}^{i} x_i t_i = V^i} \left\{ H^{i-1}(V^{i-1}) * P(x_i^{act} \geq d_i | x_i) | V^i - V^{i-1} \in [x_i^{min} t_i; x^{max} t_i] \right\} \quad (12)$$

The DP can be represented as fulfilling the matrix structure with a variable quantity of elements in each line. The first line contains the probabilities $P(x_1^{act} \geq d_1 | x_1)$ where $x_1 \in [x_1^{min}, x_1^{max}]$, the second - $P(x_1^{act} \geq d_1 | x_1) * P(x_2^{act} \geq d_2 | x_2)$ where $x_i \in [x_i^{min}, x_i^{max}]$, $i = 1, 2$, etc. The last line contains the probabilities of overall demand satisfying obtained in the following manner:

$$\max \left\{ H^{n-1}(V^{n-1}) * P\left(x_n^{act} \geq d_n \mid (x, \pi^*), \text{ Cumulative repair time}\right) \right\} \quad (13)$$

The lot-sizes x_i, $i = 1, \ldots, n$ can be found using the backtracking.

4.3.2. Memetic algorithm

The decision variables here are the sizes of lots. Each solution is a vector of integer number each of which gives the size of the corresponding lot. As the probability for the last lot is calculated separately (see equation (2)) , so for a problem with n types of products, the length of each solution is equal to $n-1$. The fitness value for each solution is calculated using equation (2) under assumption that $x_n = x_n^{max}$.

The MA from Schemeleva et al. (2012) consists of the following steps:

- *Initial population*: The initial population is composed of Pop_size solutions obtained with three greedy algorithms $G1$, $G2$, $G3$, and (Pop_size-4) random solutions. Here, we are going to present their short overview only. We will use the upper (x_j^{max}) and the lower (x_j^{min}) limits for each lot size x_j defined before.

  First of all, algorithm $G1$ initializes a solution x^1 with the minimal feasible values x_j^{min} for each $j = 1, \ldots, n-1$. Then, in each iteration, we increase by 1 the size of lot corresponding to the minimal value of probability $P(x_j^{1-act} \geq d_j | x_j^1)$, that can increase the total probability of demand satisfaction. The process continues while the current solution stays feasible.

  Algorithm $G2$ is opposite to algorithm $G1$ and consists of the following: initialize a solution x^2 with $x_j^2 = x_j^{max}, \quad j = 1, \ldots, n-1$. In most of the cases, this solution is not feasible, otherwise the lot-sizing sub-problem is solved: the optimal solution is $x^* = (x_1^{max}, x_2^{max}, \ldots, x_n^{max})$. If x^2 is not feasible, then decrease by 1 the size of the lot j which has the maximal probability $P(x_j^{2-act} \geq d_j | x_j^2)$ until the solution becomes feasible. If further decreasing of the lot with maximum probability permits to increase the total probability of the demand satisfying (because in this case, we increase the time remaining for the last lot), we continue to decrease the corresponding value, otherwise the algorithm stops.

  The greedy algorithm $G3$ is developed to find an *average* solution. We initialize x^3 as following: $x_j^3 = \lfloor (x_j^{min} + x_j^{max})/2 \rfloor$, $j = 1, \ldots, n-1$. Numerical experiments have

shown that this approach gives solutions of a good quality when they are feasible. To make a solution feasible, we proceed in the same manner as in algorithm $G2$: decrease at each iteration the lot size by 1 for the lot for which the probability is maximal until the solution becomes feasible.

Approach $G4$ generates the sizes of lots randomly, respecting lower x_j^{min} and upper x_j^{max} bounds, $j = 1, \ldots, n-1$. Note, a solution is considered as feasible if we have enough time to process at least d_n items of the last product. If an obtained solution is not feasible, we proceed in the same manner as in $G2$ and $G3$.

To have a heterogeneous population the distance between any pair of solutions should be not less than 3. The *distance metric* d_L for two lot-sizing parts x^a and x^b of solutions is defined as follows:

$$d_L(x^a, x^b) = \sum_{k=1, k \neq last\ lot}^{k=n} |x_k^a - x_k^b|, \tag{14}$$

- *Crossover*: The chromosomes chosen using the tournament selection participate in the crossover. Their number is equal to $Cross\_ratio$. The crossover probability for each pair of parents x^a and x^b to generate 2 offspring is $Cross\_prob$. With the same probability $Cross\_prob$, gene i ($i = 1, \ldots, n-1$) of both offspring is generated randomly between x_i^a and x_i^b, otherwise gene i takes the corresponding value of one of its parents.

- *Mutation*: The mutation probability $Prob\_mut$ is employed twice: 1) to decide if an offspring off will mutate; 2) to decide if a given gene i of off will mutate. If it is the case, its value is randomly generated in the following interval:

$$\{-\lceil \left(x_i^{max} - x_i^{min}\right)/4 \rceil, \ldots, \lceil \left(x_i^{max} - x_i^{min}\right)/4 \rceil\} \cap \{\lfloor x_i^{min} - off_i, -2 \rfloor \cup \lfloor 2, x_i^{max} - off_i \rfloor\}$$

- *Selection*: The redundant and close solutions (distance ≤ 3) are eliminated. If the population size is still greater than Pop_size, the elitist selection is employed, otherwise the population is completed with the necessary number of random solutions.

- *Local search*: The neighborhood for a given chromosome includes all solutions for which the distance is equal to 1. If the best solution cannot be ameliorated, we apply the local search to the chromosome with the second best fitness value if it can be ameliorated, etc.

5. Solving the global problem without decomposition

The global problem is then composed of two NP-hard problems which must be solved simultaneously. We propose to address it using a new meta-heuristic approach which considers jointly lot-sizing and sequencing issues simultaneously. This will be a memetic algorithm following the standard scheme. We will present the way of solution coding in subsection 5.1

and fitness calculating in subsection 5.2. The procedure of the initial population generating is reported in subsection 5.3. The selection mechanism is suggested in subsection 5.4 and the crossover and mutation operators in subsections 5.5 and 5.6 respectively. The algorithm is finished with a local search described in subsection 5.7.

5.1. Coding of solutions

Each solution (x, π) is coded as shown in Table 1. In $x = (x_1, \ldots, x_n)$, each x_j is a quantity of items for a corresponding product j to manufacture. Considering the example of the solution presented in Table 1: a lot of 5 items of product π_1 will be launched, then a lot of 12 items of product π_2, then 54 items of product π_3, etc. Vector $\pi = (\pi_1, \ldots, \pi_n)$ is a sequence of lots, where each π_j gives the product's type at position j, i.e. $\pi_1 = 1$ means that the lot of product of type 1 will be the first to manufacture, $\pi_2 = 5$ - that the second lot will be the lot of product 5, etc.

Table 1: An example of solution with *Total Set-up Time* (TS)= 1.63 hours and *Fitness*= 0.862

Position	1	2	3	4	5	6	7	8	9	10
Product π	1	5	10	8	2	7	4	6	3	9
Lot Size x	5	12	54	9	31	41	23	4	45	32

Note that π_j are ordered following their passing through the production line (product π_1 is the first, π_2 is the second, etc.). Whereas in the lot-sizing part x of solution, elements x_j are ranged in the product's type order (x_1 is the size of a lot for product 1, x_2 for product 2, etc.).

For a given sequence, the total set-up time can be calculated.

5.2. Solution's Fitness

Fitness f of each solution (x, π) is the probability that demands for all products will be satisfied. Let x_j^{act} be the quantity of quality items among x_j processed items of product j. The formula is identical to equation (2) from section 2 :

$$f(x,\pi) = \prod_{i=1}^{n-1} P\left(x_i^{act} \geq d_i \mid (x,\pi)\right) * P\left(x_n^{act} \geq d_n \mid (x,\pi), R\right) \tag{15}$$

5.3. Creation of initial population

Let *Pop_size* be the quantity of chromosomes in the population. To have a heterogeneous population consisting of quality solutions, we have created several heuristic algorithms to generate a sequence of lots ($H1$, $H2$ and $H3$) and sizes of lots ($G1$, $G2$, $G3$ and $G4$) taking into account the problem morphology. Note, to generate a complete solution: 1) a sequence should be found to know how much time remains for items processing (this time is equal to the period duration minus the sum of the sequence dependent set-up times); 2) sizes of

lots should be found taking into account the time available for items processing and the last product to process.

The first algorithm $H1$ is a greedy heuristic building a sequence iteratively. Each time the next product with the minimum set-up time is chosen.

The second heuristic $H2$ is similar to the first one, but we apply it in the reverse order, starting from the last product and choosing iteratively the product with smallest set-up time possible. This heuristic is repeated n times - once for each product at the last position.

The third heuristic algorithm $H3$ (see Algorithm 1) is an extension of Kruskal's algorithm that finds a minimum spanning tree for a connected weighted graph. Let each product $i, \quad i = 1, \ldots, n$ represents a vertex. We add an additional vertex labeled 0 to deal with initial set-up time $s_{0,i}$. Then $s_{i,j}$ is a weight (length) of the edge joining vertexes i and j starting from i. Matrix S is asymmetric, so the obtained graph is oriented and without loops ($s_{i,i} = 0, \ i = 1, \ldots, n$). Let $F = \{0, 1, 2, \ldots, n\}$ be the forest containing all vertexes, and set ES includes all edges $s_{i,j}$, $i = 0, 1, \ldots, n$, $j = 1, 2, \ldots, n$, $i \neq j$. The first specificity of the problem is that the graph is oriented, so the forest will contain only oriented trees. Secondly, we are looking for the Hamiltonian Path of minimum weight, therefore, the degree of each vertex i in the forest F should not exceed 2 (each tree is actually a simple directed path). This constraint guarantees that each vertex is visited only once.

Algorithm 1: Extension of Kruskal's algorithm (H3)

Data: Forest F includes separate vertexes $0, 1, \ldots, n$;
Set ES contains all the edges $s_{i,j}, i = 0, \ldots, n, \ j = 1, \ldots, n, \ i \neq j$
Result: Sequence of lots π^{n+2} (minimum spanning tree)
while $ES \neq \emptyset$ && *F is not yet spanning* **do**
  Find in ES the edge of minimum weight;
  if *that edge connects two different trees* **then**
    Add it to the forest, combining two trees into a single tree;
  end
  Remove it from ES;
end

As was mentioned earlier, we have developed 4 functions to generate lot-sizing parts of solutions - three greedy algorithms $G1$, $G2$ and $G3$, and function $G4$ that generates a random solution. Detailed description and pseudo-codes of these heuristics were given in Schemeleva et al. (2012) (see also section 4.3.2).

Thus, to obtain the initial population, we generate the $n+2$ sequences of lots : one using the algorithm $H1$, n with $H2$ and one with $H3$, and then for each sequence, we apply in the following order the algorithms $G1$, $G2$, $G3$ and $G4$ to obtain new solutions. Applying four algorithms to the same sequence makes sense because if the sequence of lots changes, the total set-up time changes. Moreover, the type of product for the last lot can change, which means that the calculation of the probability to have all necessary products will change.

As a result, we will obtain $4 * n + 8$ solutions. We have decided to fix the population size dependent on the number of lots as follows: $Pop\_size = 4 * \lceil \sqrt{n} \rceil < 4 * n + 8$. So there are many more solutions than needed. To keep the population size stable, we will select

$4 * \lceil \sqrt{n} \rceil$ solutions by using the selection procedure presented below.

5.4. Selection

Selection procedure consists of eliminating redundant solutions and solutions close to each other. We consider that two solutions are *close* if both lot sizes and sequence are close to each other. The distance metric d_L for lot sizes was presented in section 4.3.2. Now we will define the distance between sequences.

Let *Insertion* be the movement of any lot of a sequence (except the last one) to any other position in the sequence (except the last one). Then *distance* d_S between two sequences π^a and π^b can be defined as follows: $d_S(\pi^a, \pi^b) =$ the quantity of insertions necessary to apply to sequence π^a to obtain sequence π^b (and vice versa). The distance d_S between two sequences π^a and π^b is equal to 1 if $\exists$ pos_1 and pos_2, $pos\_1 < pos\_2$ such that:

$$\pi_i^a = \pi_i^b, \quad i \in (0, pos\_1) \bigcup (pos\_2; n] \tag{16}$$

$$if \pi_{pos\_1}^a = \pi_{pos\_2}^b \text{ then } \pi_{i+1}^a = \pi_i^b, \quad i \in [pos\_1; pos\_2) \tag{17}$$

$$if \pi_{pos\_2}^a = \pi_{pos\_2}^b \text{ then } \pi_i^a = \pi_{i+1}^b, \quad i \in [pos\_1; pos\_2) \tag{18}$$

In this paper we consider that two solutions (x^a, π^a) and (x^b, π^b) are close in the following cases:

- They have identical sequences, i.e. $\pi^a = \pi^b$ and the *distance* d_L between x^a and x^b is less than 3;
- The distance d_S between two sequences π^a and π^b is equal to 1 (see the definition above), the last lot is the same for both sequences $\pi_n^a = \pi_n^b$, and lot sizes are close $d_L(x^a, x^b) < 3$.

If two close solutions are found, the one with the smallest fitness is eliminated.

After selection, the population is sorted in decreasing order of fitness value. If there are too many chromosomes, we apply the elitist selection to save the best part of the population. If the number of chromosomes is smaller than the necessary size, the population is completed with solutions generated randomly.

5.5. Crossover

Usually, a crossover operation serves to generate some new solutions which preserve the good characteristics of the best existing solutions. To select $Cross\_rat$ percent of parents for the crossover, a *tournament selection* will be applied. There is a probability $Cross\_prob$ that a given pair of parents will produce the offspring.

Let chromosomes $p^a = (x^a, \pi^a)$ and $p^b = (x^b, \pi^b)$ are chosen. Starting from these two parents, we generate six offspring in the following way:

- Create two child sequences π_{off}^a and π_{off}^b (see Algorithm 2) with π^a and π^b;
- Having two parent vectors x^a and x^b, generate two child lot size vectors x_{off}^a and x_{off}^b;

- Mix new sequences and lot size vectors with parents' data to obtain six offspring:

$$(x^a_{off}, \pi^a), (x^a, \pi^a_{off}), (x^a_{off}, \pi^a_{off}), (x^b_{off}, \pi^b), (x^b, \pi^b_{off}), (x^b_{off}, \pi^b_{off})$$

To create new lot size vectors x^a_{off} and x^b_{off}, a hybrid crossover operation is employed (see the short description in section 4.3, and the detailed explanation in Schemeleva et al. (2012)).

Algorithm 2: Crossover for lot sequences

Data: Two parent sequences of lots π^a and π^b
Result: Offspring sequence π^a_{off}
Start from the last node: $\pi^a_{off\_n} = \pi^a_n$;
Create a set of "legitimate" nodes for $\pi^a_{off\_n}$: $LN = \{1, 2, \ldots, n\} \setminus \pi^a_n$;
for ($i \leftarrow n$ **to** 3) **do**
  Let α be the foregoing node for $\pi^a_{off\_i}$ in π^a, and β in π^b;
  if (α ***AND*** β *are both legitimate*) **then**
    if ($s_{\alpha,\pi^a_{off\_i}} \leq s_{\beta,\pi^b_{off\_i}}$) **then**
      Set $\pi^a_{off\_(i-1)} := \alpha$;
    else
      Set $\pi^a_{off\_(i-1)} := \beta$;
    end
  else if (α ***OR*** β *is legitimate*) **then**
    Let α is legitimate;
    Find in LN a node γ such that value $s_{\gamma,\pi^a_{off\_i}}$ is minimal;
    if ($s_{\alpha,\pi^a_{off\_i}} \leq s_{\gamma,\pi^b_{off\_i}}$) **then**
      Set $\pi^a_{off\_(i-1)} := \alpha$;
    else
      Set $\pi^a_{off\_(i-1)} := \gamma$;
    end
  else
    Find in LN a node γ such that value $s_{\gamma,\pi^a_{off\_i}}$ is minimal;
    Set $\pi^a_{off\_(i-1)} := \gamma$;
  end
  Delete the assigned node from LN;
end
Include the last remaining in LN node into π^a_{off};

To create two new sequences π^a_{off} and π^b_{off}, we developed a new approach (Algorithm 2) based on the idea of *Sequential Constructive Crossover* (SCX). The SCX operator constructs an offspring using better edges, present (or not) in the parents' structure (see Ahmed (2010)). The main idea is as follows: assign the first element of one of the two parents (or a randomly selected element); on each step take the last assigned to the offspring element π_i and search

for the foregoing element in both parents. If such elements exist, choose and assign the one with minimal set-up value to the offspring, otherwise look for an element which will generate the minimum set-up time among remaining elements not assigned to the offspring. In contrast to the classic SCX approach we fulfill the offspring starting from the end: $\pi^a_{off\_n} = \pi^a_n$ and $\pi^b_{off\_n} = \pi^b_n$.

After the crossover operation, the population size is $4*\lceil\sqrt{n}\rceil$ to $4*\lceil\sqrt{n}\rceil*(1+Cross\_rat*3)$ chromosomes.

5.6. Mutation

A mutation operation is applied to all offspring to increase the diversity of the current population. We introduce the parameter Mut_prob - the probability that a given chromosome will mutate. The mutation is performed in two steps: lot sizes mutation and sequence mutation. The general mutation approach is outlined in algorithm 3.

Algorithm 3: Mutation operation

Data: Offspring (x^i_{off}, π^i_{off})
for ($i \leftarrow 1$ **to** *Offspring_ quantity*) **do**
  Generate randomly a number $m \in [0, 1)$;
  if $(m < Mut\_prob)$ **then**
    Apply $Lot\_Sizes\_Mutation()$ to x^i_{off};
  Generate once again a number $m \in [0, 1)$;
  if $(m < Mut\_prob)$ **then**
    Apply $Sequence\_Mutation()$ to π^i_{off};
  Replace (x^i_{off}, π^i_{off}) with the obtained solution;
end

$Lot\_Sizes\_Mutation()$ is the same as in section 4.3. To perform $Sequence\_Mutation()$ we use *displacement* mutation without *inversion* approach. The displacement is a movement of a randomly selected subsequence to another place, where positions *begin*, *end*, and *insertion points* are also generated randomly. *Insertion* and *end* points cannot be at the end of a chromosome.

5.7. Local Search

The local search procedure is applied to the best solution of each generation. If the best solution cannot be ameliorated, we take the one with the highest fitness value from those that can be ameliorated yet. In this paper, the local search (LS) consists of three procedures: $LS\_lots()$ for sizes of lots; $LS\_sequence\_1()$ and $LS\_sequence\_2()$ applied to the sequences of lots.

The procedure $LS\_lots()$ is the same as in Schemeleva et al. (2012) see section 4.3.

$LS\_sequence\_1()$ changes only the last element in the sequence by moving elements from position k, $k = 1, \ldots, n-1$ to the last position. In such a way, we create $n-1$ new sequences. Note, as we change the last element, the comparison of set-up times is not sufficient to find the best solution from $n-1$ obtained - we have to calculate the fitness for each of them. The sequence with the best fitness is retained.

LS_sequence_2() deals with all sequences π^{ls} that can be obtained from a given one π by one *insertion*, i.e. $d_S(\pi, \pi^{ls}) = 1$. The neighborhood for this LS contains $(n-2)^2$ solutions. In this case, we evaluate all these chromosomes by calculating total set-up time for each of them.

The overall local search procedure is as follows: we apply successively *LS_sequence_1()*, *LS_sequence_2()* and *LS_lots()* to the considered solution.

6. Experimental results

In this section we present the results of computer experimentation. All tests were performed on a SUN UltraSPARC IIIi with 1593 Mhz CPU and 16 Gb of RAM. The algorithms were coded in C++. The linear models ATSP-SD were solved with ILOG CPLEX 11.0.

Let n be the number of different lots for each problem instance. Algorithms were tested using the problem instances from Schemeleva et al. (2012). For each problem size n, 10 instances were chosen with the following parameters: demand in the range $[10; 50]$ items, mean time to failure in the range $[50; 500]$ hours, mean time to repair in the range $[0.3; 1]$ hours.

6.1. What is the maximal size of problems which can be solved with the exact decomposition method proposed in Dolgui et al. (2005)

The objective of this section is to demonstrate the limits of the exact method based on the decomposition with *Enumeration + MIP + DP* procedures. Remember 10 instances were solved for each problem size n, $n \in [6; 15]$. The CPU times of each run of the DP and MIP were limited by 1000 seconds.

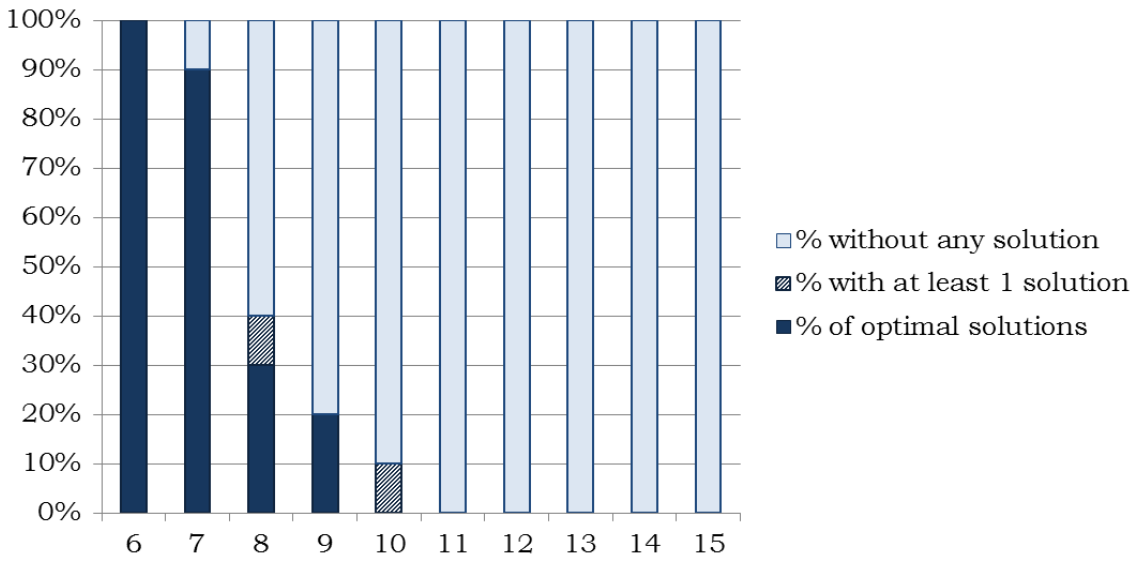

Figure 4: Percentage of instances solved to optimality with *Enumeration + MIP + DP*

First of all, we will show the number of instances solved by the procedure for each n. Figure 4 demonstrate the following information:

- The percentage of instances solved (i.e. the optimal solution was obtained). Note, to obtain the optimal solution, all of the n solutions (one for each lot i, $\quad i = 1, \ldots, n$ on the last position) should be found;
- The percentage of instances for which at least one solution could be obtained, i.e. the number of solutions obtained is in range $[1; n-1]$, $n \in [6; 15]$;
- The rest of instances for which no solution was obtained.

As we can see, only for $n = 6$ all of 10 instances were solved optimally; for $n = 7$ - 90% (or 9 out of 10) instances. With n increasing this percentage decreases drastically: 30% for $n = 8$ and 20% for $n = 9$. We have no solution for $n >= 11$.

The percentage of instances partially solved (from 1 to $n-1$ solutions) is quite small (one instance for $n = 8$ and one instance for $n = 10$). Generally, the impossibility to find any solution for an instance is explained by the lack of computer memory for *DP* and not by the time limitation. For the *MIP*, the limit of time was never reached. To give a rough idea, the average CPU time of MIP for $n = 10$ is less than 0.1 second.

6.2. Assessment of the importance of enumeration on the last lot

Now we will compare the results obtained for different lots on the last position, i.e. $\pi_n = i,\ i = 1, \ldots, n$. More precisely, we want to see what is the difference in terms of solution quality between the optimal solution and the other solutions obtained for the same instance for the different last lots. In other words, is the enumeration step very important? Obviously, this comparison can be made only for problems for which the optimal solution was obtained.

Table 2: Is the enumeration important?

Number of lots	Number of instances	Ratio with the optimal fitness		
		Minimum	Average (of n)	Best sequence
6	10	96.67%	98.28%	98.68%
7	9	91.26%	95.57%	98.33%
8	3	87.80%	93.85%	98.07%
9	2	88.54%	94.34%	99.31%

In Table 2 we report the ratios between the best fitness value and the others obtained for the same instance. The first column of the Table contains the number of lots, the second reports the number of instances solved optimally. In the third column the average of ratios $Minimum\ fitness/Optimal\ fitness$ for each problem size is presented. The fourth column gives the $Average\ fitness/Optimal\ fitness$ for each $n = 6, 7, 8$ and 9. This

column shows the expected solution quality if the last lost is chosen randomly. Finally, the last column shows the average of ratios $Fitness\ of\ solution\ with\ the\ optimal\ sequence/\ Optimal\ fitness$. Remember, the optimal or best sequence of n lots is that which minimizes the total set-up time.

We can see than the average ratio is between 93% and 98%, so if the last lot is chosen randomly, the average gap between obtained solutions and the optimal ones is from 2% to 7%. The "best sequence" ratios reported in the last column are about $98\% - 99\%$, so if an error of 1% to 2% is acceptable, the enumeration step can be omitted in the decomposition scheme. For tested instances, in the worst case, the gap between solutions obtained without enumeration and the optimal solutions was 12.20%

From all the above, we should conclude that the decomposition approach with DP procedure proposed in Dolgui et al. (2005) gives an optimal solution, but has a number of limitations related to the execution of Dynamic Programming procedure. Beginning at $n = 8$ the DP encounters execution problems because of the lack of CPU. And from $n = 11$ lots, it is not an option at all.

To solve the instances with more than $10 - 12$ lots, we should either replace the DP procedure by a heuristic (or meta-heuristic), like a memetic algorithm proposed in Schemeleva et al. (2012), or develop an algorithm able to solve the entire problem (sequencing and lot-sizing) simultaneously.

6.3. Is the approximate method proposed in Schemeleva et al. (2012) a suitable alternative?

The goal of this section is to see if we can replace the DP procedure with a heuristic and solve efficiently the instances of larger sizes. We will start with a comparison of solutions obtained with this approximate approach with known optimal solutions.

6.3.1. Evaluation for the optimally solved (or small) instances

In this subsection we are going to compare two approaches:

a) Decomposition approach with DP procedure *Enum + MIP + DP*;

b) *Enum + MIP +* GA_{LS} approach obtained by replacing of the DP in decomposition scheme by the memetic algorithm GA_{LS} from Schemeleva et al. (2012) to optimize the sizes of lots;

In Table 3 the average and minimum ratios with the optimal fitness (obtained with *Enum + MIP + DP*) for *Enum + MIP +* GA_{LS} for each lot size $n = 6, 7, 8$ and 9 are presented.

As we can see, solutions obtained with *Enum + MIP +*GA_{LS} are pretty much the same as the optimal ones. This means that the GA_{LS} is very efficient for the instances tested and generally gives the optimal lot sizes. So, for the small instances, the Dynamic Programming procedure can be replaced by the Genetic algorithm to solve the lot-sizing sub-problem. We cannot guarantee the optimality, but for the tested instances the gap was less than 0.2% on average. The maximal gap was 0.37%.

Table 3: Quality of solutions obtained with *Enum + MIP* $+GA_{LS}$

Number of lots	Number of Instances	Ratio with the optimal fitness	
		Average (of n)	Minimum
6	10	100.00%	100.00%
7	9	100.00%	100.00%
8	3	99.95%	99.86%
9	2	99.81%	99.63%

6.3.2. Is the approach proposed in Schemeleva et al. (2012) efficient to deal with larger problems

In this subsection, we discuss the CPU time of the decomposition approach with the MIP (to solve the ATSP) and memetic algorithm GA_LS, and more precisely, the CPU time of the sequencing part (MIP) of this approach.

Firstly, we executed the program for 5 different instances for each problem size $n = \{20, 30, 40, 50\}$ to see how the CPU time of MIP evolves with the increasing of n. In this experimentation the CPU time of the MIP was unbounded. In figure 5 we present the minimum, average and maximum CPU times of MIP and two exponential trend curves: one for the average *Expon.* (Average) and another for the maximum *Expon.* (Maximum) CPU time.

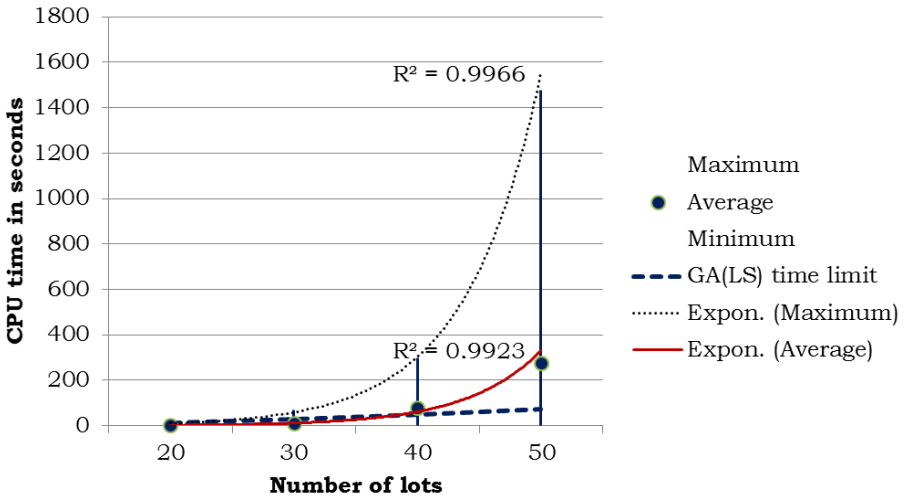

Figure 5: Minimum, Average and maximum CPU time of MIP

We can see that the CPU time grows promptly : if for $n = 20$ lots the average CPU time is less than 2 seconds, for $n = 30$ it is about 12.4 seconds, and for $n = 50$ the average running time achieves 277 seconds. There are wide variations between minimums (less than

1 second even for $n = 50$ lots) and maximums (about 70 seconds for $n = 30$, 300 seconds for $n = 40$ and 1480 seconds for $n = 50$), thus the CPU time is highly dependent on instances tested. Taking into account the tendency curves (r-squared values are also presented in the figure) when increasing the number of lots, the CPU time grows exponentially.

Without any limit for MIP's CPU time, a run duration will explode with increasing of n. To overcome this limitation we have four possibilities: 1) put a reasonable upper bound on the MIP's CPU time and accept to have an approximate solution with CPLEX; 2) replace the MIP model with a meta-heuristic; 3) take the $MIP + GA_{LS}$ approach without enumerating with the same global calculation time; 4) develop a meta heuristic able to solve the entire problem.

The following subsection intends to evaluate these four possibilities, but before starting the other comparisons, we will discuss the cases 2) and 3). The time needed to CPLEX to find the optimal solution grows very quickly with n. So CPLEX will be able to obtain less and less optimal sequences. To see the proportion of optimal solutions as a function of the problem size, we limited the CPU time of the MIP in $Enum + MIP + GA_{LS}$ by $0.02 * n^2$ seconds per run (accordingly to the time limit fixed in the next section). The results are presented in Table 4.

First three columns present the number of lots, the total number of runs of CPLEX across all instances for each n, $n = \{20, ...80\}$ and the upper bound on the calculation time in seconds per run, respectively. The fourth column of Table 4 shows the percentage of runs where CPLEX was able to get an optimal solution. The next column shows the percentage of runs where an optimal solution with a tolerance has been found. Sixth column "Time limit & integer solution" reports the percentage of cases where the time limit was exceeded, nevertheless an integer solution exists. Finally, the last column presents the percentage of cases where the time limit was exceeded and no integer solution was found. In the latter case a suboptimal sequence was obtained with a greedy heuristic.

Table 4: Percentages of optimal and suboptimal sequences for $Enum + MIP + GA_{LS}$

Number of lots	Number of runs	Time limit (s)	Optimal Solution	Opt. sol. with tolerance	Time limit & integer sol.	No solution
20	200	8	97.00%		3.00%	
30	300	18	74.00%	0.33%	12.67%	13.00%
40	400	32	27.25%	0.25%	4.00%	68.50%
50	500	50	15.80%		2.20%	82.00%
60	600	72	8.00%		0.17%	91.83%
70	700	98	1.57%			98.43%
80	800	128				100.00%

As we can see, starting from $n = 40$ lots, the percentage of cases where a greedy solution was created for the sequencing part of the approach is predominant. Thus the CPLEX needs

much more time to calculate an optimal solution.

For $MIP + GA_{LS}$ without enumerating, the limit of calculation time was fixed at $0.02{*}n^3$, that represents 160 seconds for $n = 20$, 540 seconds for $n = 30$, 1280 seconds for $n = 40$, etc. For this method, all runs (10 for each n from 20 to 80) have resulted in the optimal (or optimal with tolerance) solution.

6.4. What is the best way to design a heuristic algorithm to tackle large instances?

In this section, the total CPU time for all approaches is bounded by $0.04 * n^3$ seconds for each $n = \{20, 30, 40, 50, 60, 70, 80\}$. Remember we compare 4 following algorithms:

a) $Enum + MIP + GA_{LS}$ approach where the CPU time of both parts - MIP and GA_{LS} - are limited by $0.02 * n^2$ seconds each (thus, each iteration takes $0.04 * n^2$ seconds, and after n iterations the time limit fixed above will be achieved);

b) $Enum + GA_{TSP} + GA_{LS}$ under the same CPU time limits as above, where GA_{TSP} is the genetic algorithm solving the sequencing sub-problem presented in section 4.2.2;

c) $MIP + GA_{LS}$ without enumerating where the CPU time of both parts - MIP and GA_{LS} - are bounded by $0.02 * n^3$ each;

d) Memetic algorithm solving the whole problem (section 5) under the time limit is equal to $0.04 * n^3$.

10 different instances for each problem size $n \in \{20, 30, 40, 50, 60, 70, 80\}$ were executed. We employ the box-and-whisker diagram (figure 6) introduced by John W. Tukey in 1969 to have a general idea of the solutions' quality. Each box shows the first Q_1 and third Q_3 quartiles (box bounds) and the median. Whiskers are the maximum and the minimum (if not smaller than $Q_1 - 1.5 * (Q_3 - Q_1)$, otherwise $MIN = Q_1 - 1.5 * (Q_3 - Q_1)$) values.

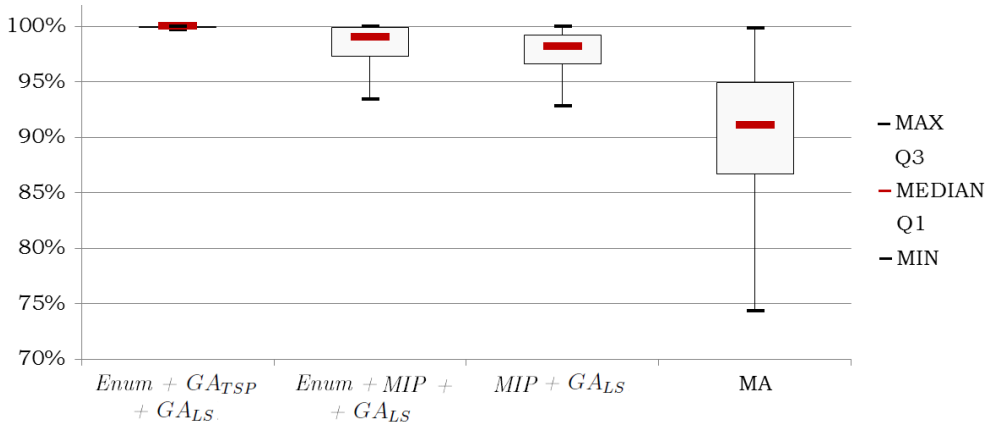

Figure 6: Solution quality

The best results were obtained with $Enum + GA_{TSP} + GA_{LS}$. We cannot see the minimum and maximum values because they are very close to each other. Among 70 instances tested, the $Enum + GA_{TSP} + GA_{LS}$ gives the best solution in 50% of cases, and the average ratio with the best obtained solution is 99.86%.

Despite the fact that $MIP + GA_{LS}$ was able to obtain the optimal sequence for each instance, it seems that the overall quality of its solutions ($obtained\ fitness/best\ fitness$) is not as good as for those of $Enum + MIP + GA_{LS}$ where the heuristic $H2$ (see section 5.3) was often employed to obtain an admissible sequencing solution. The medians of $Enum + MIP + GA_{LS}$ and $MIP + GA_{LS}$ are very close to each other, but the probability that the first method gives a better solution than the second one is 0.997944.

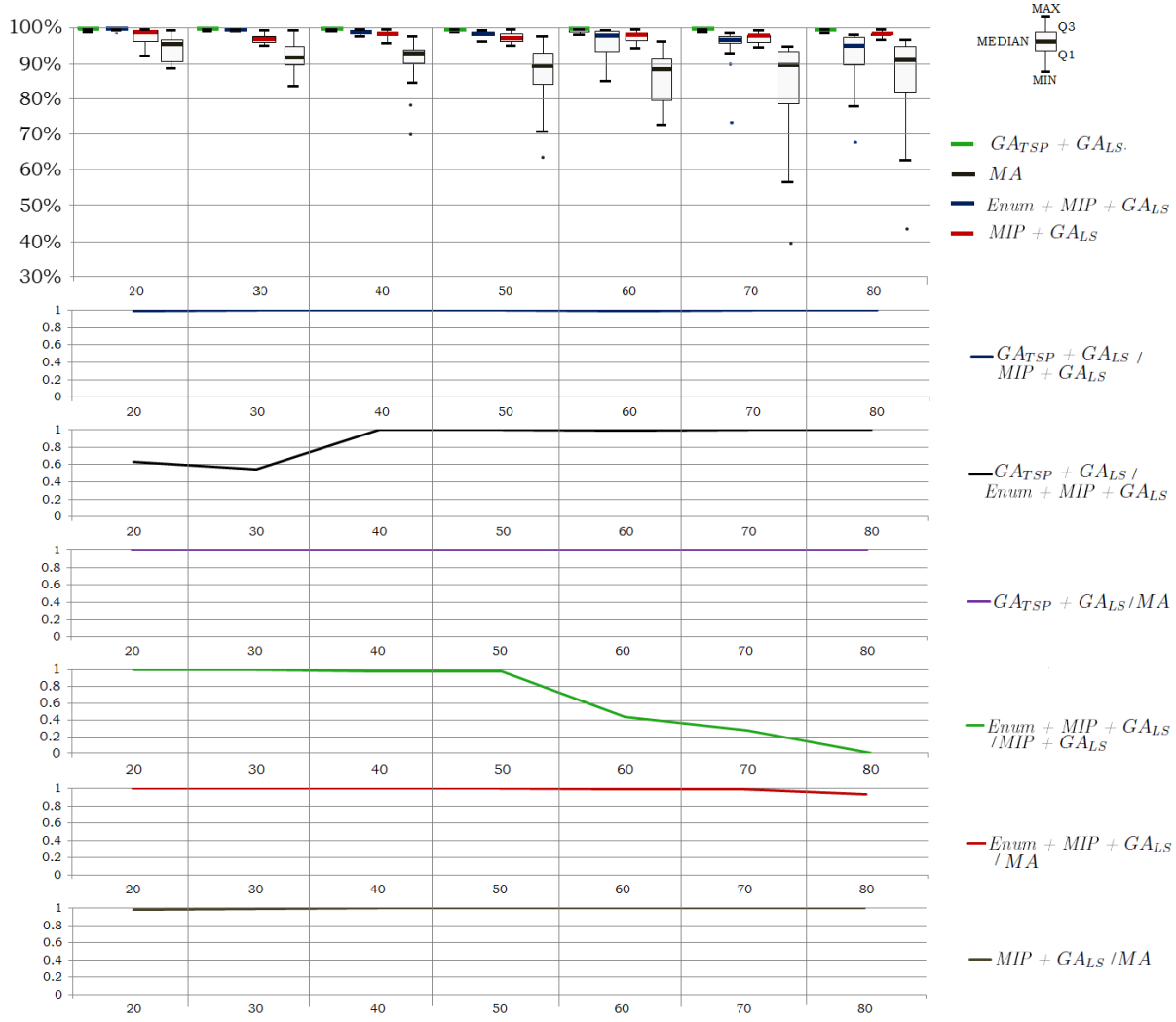

Figure 7: Comparison of 4 solution methods

To show the strengths and difficulties of each method, we present the box-and-whisker diagrams (top part of figure 7) for the four approximate methods for each $n = 20$, 30, 40, 50, 60, 70 and 80. The outliers - points that were not included in the boxes - are also represented. The bottom part of the same figure gives the probabilities that a method will provide a better solution than another. These results were obtained with the Mann-Whitney U tests (see Mann and Whitney (1947)) for each couple of methods.

We can state that among the four approaches compared here, it is almost guaranteed that $Enum + GA_{TSP} + GA_{LS}$ will give the best solution. $Enum + GA_{TSP} + GA_{LS}$ and $Enum + MIP + GA_{LS}$ give comparable performance for problems size 20 and 30 for which MIP has found the optimal sequence in most of the cases. But beginning from $n = 60$ the quality of $Enum + MIP + GA_{LS}$ degrades compared to $MIP + GA_{LS}$. Generally the quality of $Enum + MIP + GA_{LS}$ deteriorates with the increasing of n, whereas the quality of $MIP + GA_{LS}$ solutions is stable.

6.5. What is the best way to share the CPU time?

All algorithms were tested with three different distributions of allocated time: a) fifty-fifty between lot-sizing and sequencing parts (the case in the previous section); b) 25% for sequencing(ATSP) resolution and 75% for lot-sizing sub-problem; c) 75% for ATSP and 25% for lot-sizing. In this subsection we will study these time allocations for the most efficient algorithm, i.e. $Enum + GA_{TSP} + GA_{LS}$.

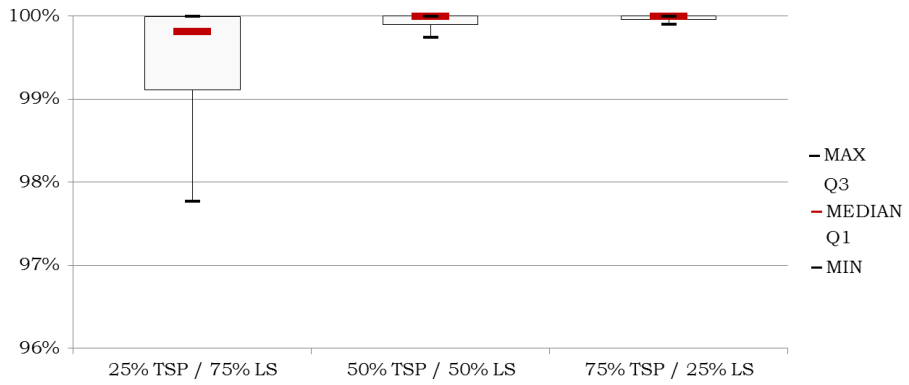

Figure 8: Solution quality of $Enum + GA_{TSP} + GA_{LS}$ for different time partitions

In figure 8 we present the box-and-whisker diagram where each box represents the quality of solutions (relative to the best solution found) for each of the three above-mentioned partitions of time.

As we can see the approach gives the best solutions when 75% of CPU time is allocated to solve the TSP sub-problem and 25% of time for the lot-sizing one.

In Table 5 the results of the Mann-Whitney U test are reported. Each cell of the Table shows the probability that the partition in the corresponding row will give better solutions than the partition in the corresponding column. As we can see, partition $50\%TSP$ / $50\%LS$

Table 5: Mann-Whitney U test for different time partitions of $Enum + GA_{TSP} + GA_{LS}$

Partitions	25% TSP / 75% LS	50% TSP / 50% LS	75% TSP / 25% LS
25% TSP / 75% LS		-	-
50% TSP / 50% LS	0.999984		-
75% TSP / 25% LS	0.999999	0.892099	

is better than $25\%TSP$ / $75\%LS$ and partition $75\%TSP$ / $25\%LS$ is better than both aforementioned.

7. Conclusions

A problem of optimal sequencing and lot-sizing for a production line with random breakdowns and rejects was studied. The objective was to maximize the overall service level. The benefit of this optimization is evident: without any additional resources, production cost can be reduced by limiting the stoppages for the assembly line that follows.

An optimal decomposition approach, including an enumeration step, the parts sequencing (MIP model) and lot-sizing decisions (DP procedure), is able to solve the small problems (up to 10 lots). To deal with larger instances, the GA_{LS} algorithm was used for the lot-sizing sub-problem. This combination of methods gives very good solutions; nevertheless the results have shown that up to 40 lots, the CPU times for the MIP model remain acceptable but increase very quickly. As a consequence, in order to solve larger problems, we needed to consider also an approximate solution of the sequencing sub-problem. Three possibilities were tested: a) bounding the computation time of CPLEX; b)using a heuristic (GA_{TSP}); c) removing the enumeration step for the same total CPU time. Finally, a last possibility d) was envisaged which consists of a metaheuristic MA dealing with the whole problem.

The results have shown that, contrary to what most people might assume, the most efficient method is the decomposition approach including two heuristic algorithms, which means that an integrated approach isn't well suited to this problem. Moreover, to design an effective heuristic algorithm based on a decomposition framework, two main results have been obtained:

1. skipping the phase of enumeration of the last lot isn't a good idea, and
2. it seems preferable to allocate more computational time for the lot-sizing sub-problem than the sequencing sub-problem.

Based on these results, further work can be guided to find more efficient heuristics to solve each sub-problem separately. As for any empirical study, it is theoretically possible that the global memetic approach may be more efficient for different data instances and algorithm parameters. Nevertheless, we performed many computational tests on many different instances and the gap between both approaches is always quite important.

Beside the choice of a framework, the improvement of the different heuristics used in this paper could increase the performance of the approaches. Studying the fitness landscape, and considering other neighborhoods, could be the most promising way to do that.

References

References

Z.H. Ahmed. Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator. *International Journal of Biometrics & Bioinformatics*, 3(6):96, 2010.

A. Allahverdi, C.T. Ng, T.C.E. Cheng, and M.Y. Kovalyov. A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187(3):985–1032, 2008.

M.A. Aloulou, A. Dolgui, and M.Y. Kovalyov. A bibliography of non-deterministic lot-sizing models. *International Journal of Production Research*, 52(8):2293–2310, 2014.

Y. Arda and J.-C. Hennet. Inventory control in a multi-supplier system. *International Journal of Production Economics*, 104(2):249–259, 2006.

J.L. Bentley. Experiments on traveling salesman heuristics. *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 91–99, 1990.

L. Buriol, P. M. França, and P. Moscato. A new memetic algorithm for the asymmetric traveling salesman problem. *Journal of Heuristics*, 10(5):483–506, 2004.

M. Çakanyildirim, J. H. Bookbinder, and Y. Gerchak. Continuous review inventory models where random lead time depends on lot size and reserved capacity. *International Journal of Production Economics*, 68 (3):217–228, 2000.

I.-C. Choi, S.-I. Kim, and H.-S. Kim. A genetic algorithm with a mixed region search for the asymmetric traveling salesman problem. *Computers & Operations Research*, 30(5):773–786, 2003.

A. Dolgui. Performance analysis model for systems described by renewal process. *Engineering Simulation*, 24(2):3–11, 2002.

A. Dolgui and C. Prodhon. Supply planning under uncertainties in MRP environments: A state of the art. *Annual Reviews in Control*, 31(2):269–279, 2007.

A. Dolgui, G. Levin, and M.A. Louly. Decomposition approach for a problem of lot-sizing and sequencing under uncertainties. *International Journal of Computer Integrated Manufacturing*, 18(5):376–385, 2005.

A. Dolgui, A.V. Eremeev, M.Y. Kovalyov, and P.M. Kuznetsov. Multi-product lot sizing and scheduling on unrelated parallel machines. *IIE Transactions*, 42(7):514–524, 2010.

Y. Gerchak and A. Grosfeld-Nir. Multiple lot-sizing, and value of probabilistic information, in production to order of an uncertain size. *International Journal of Production Economics*, 56:191–197, 1998.

B.C. Giri and T. Dohi. Optimal lot sizing for an unreliable production system based on net present value approach. *International Journal of Production Economics*, 92(2):157–167, 2004.

B.C. Giri and W.Y. Yun. Optimal lot sizing for an unreliable production system under partial backlogging and at most two failures in a production cycle. *International Journal of Production Economics*, 95(2): 229–243, 2005.

G.C. Lin and D.-C. Gong. On a production-inventory system of deteriorating items subject to random machine breakdowns with a fixed repair time. *Mathematical and Computer Modelling*, 43(7):920–932, 2006.

B. Maddah and M.Y. Jaber. Economic order quantity for items with imperfect quality: Revisited. *International Journal of Production Economics*, 112(2):808–815, 2008.

H.B. Mann and D.R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60, 1947.

Y. Nagata. The eax algorithm considering diversity loss. *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature, Lecture Notes in Computer Science*, 3242:332–341, 2004.

Y. Nagata and D. Soler. A new genetic algorithm for the asymmetric traveling salesman problem. *Expert Systems with Applications*, 39(10):8947–8953, 2012.

T. Öncan, İ K. Altınel, and G. Laporte. A comparative analysis of several asymmetric traveling salesman problem formulations. *Computers & Operations Research*, 36(3):637–654, 2009.

Pl.K. Palaniappan and N. Jawahar. A genetic algorithm for simultaneous optimisation of lot sizing and scheduling in a flow line assembly. *International Journal of Production Research*, 49(2):375–400, 2011.

S. Papachristos and I. Konstantaras. Economic ordering quantity models for items with imperfect quality. *International Journal of Production Economics*, 100(1):148–154, 2006.

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

K. Schemeleva, X. Delorme, A. Dolgui, and F. Grimaud. Multi-product sequencing and lot-sizing under uncertainties: A memetic algorithm. *Engineering Applications of Artificial Intelligence*, (25):1598–1610, 2012.

H.D. Sherali and P.J. Driscoll. On tightening the relaxations of miller-tucker-zemlin formulations for asymmetric traveling salesman problems. *Operations Research*, 50(4):656–669, 2002.

H.D. Sherali, S.C. Sarin, and P.-F. Tsai. A class of lifted path and flow-based formulations for the asymmetric traveling salesman problem with and without precedence constraints. *Discrete Optimization*, 3(1):20–32, 2006.

R. Sikora. A genetic algorithm for integrating lot-sizing and sequencing in scheduling a capacitated flow line. *Computers & Industrial Engineering*, 30(4):969–981, 1996.

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

Y. Wang and Y. Gerchak. Input control in a batch production system with lead times, due dates and random yields. *European Journal of Operational Research*, 126(2):371–385, 2000.

L.-N. Xing, Y.-W. Chen, K.-W. Yang, F. Hou, X.-S. Shen, and H.-P. Cai. A hybrid approach combining an improved genetic algorithm and optimization strategies for the asymmetric traveling salesman problem. *Engineering Applications of Artificial Intelligence*, 21(8):1370–1380, 2008.

C.A. Yano and H.L. Lee. Lot sizing with random yields: A review. *Operations Research*, pages 311–334, 1995.