A Genetic Algorithm for a capacitated lot-sizing problem with lost sales, overtimes, and safety stock constraints

Benoît Le Badezet, François Larroche, Odile Bellenguez, Guillaume Massonnet

A Genetic Algorithm for a capacitated lot-sizing problem with lost sales, overtimes, and safety stock constraints

Benoît Le Badezet[1], François Larroche[1,2], Odile Bellenguez[1], and Guillaume Massonnet[1]

IMT Atlantique, LS2N, La Chantrerie, 4 rue Alfred Kastler, 44307 Nantes France
VIF, 10 Rue de Bretagne, 44240 La Chapelle-sur-Erdre
benoit.le-badezet@etu.univ-nantes.fr, {francois.larroche, odile.bellenguez, guillaume.massonnet}@imt-atlantique.fr

Abstract. This paper deals with a complex production planning problem with lost sales, overtimes, safety stock and sequence dependent setup times on parallel and unrelated machines. The main challenge of this work is to propose a solution approach to obtain a good feasible plan in a short execution time (around 2 minutes) for large industrial instances. We develop a genetic algorithm that combines several operations already defined in the literature to solve the problem. Preliminary numerical results obtained with our algorithm are presented and compared to a straightforward MIP resolution. The method appears to be an appealing alternative on large instances when the computational time is limited.

1 Introduction

The problem presented in this paper is related to practical cases encountered in the food industry for production planning. In this context, manufacturers can generally use several production lines, each able to make several types of items. This complexity usually leads to problems that are too large to be solved optimally by off-the-shelf solvers. In addition, the models we consider in this paper also combines constraints from the lot-sizing and the scheduling literature, by assuming that the setup times between different types of items depends on the production sequence. This further limit the applicability of standard methods in practice, when the planners need to obtain "good" feasible solutions in reasonable time to test several machine configurations or shifts assignments and obtain quick insights to support their decisions.

This problem extends the field of lot-sizing, which has been extensively studied since the work of Wagner and Whitin [1]. Motivated by the physical constraints found in practical applications, the finite production capacity version of the problem (CLSP) has received a lot of attention, see [2] and [3] for a review of extensions and solution approaches. The problem we consider is an extension of the industrial problem with lost sales and shortage costs presented in [4], for which the authors introduce new classes of valid inequalities. The safety stock is seldom considered in the deterministic production and inventory literature. [5] define the safety stock as a lower bound on the number of units that must be held in the inventory at each period when [6] choose to penalize the missing units from the safety stock. The latest version is studied here. Versions of the problem with parallel machines and sequence dependent setups are less common in the literature. [7] develop new heuristics on a parallel machines problems. [8] present an industrial problem in which setup times depend on the sequence of production and propose a solution procedure based on subtour elimination and patching. [9] use a small bucket formulation to compute the sequence of production. [10] also present an extensive review of this extension and compare the efficiency of several methods to solve it. The possibility to exceed the production capacity is not common in the literature, see [11] for an overtime extension of a capacitated lot-sizing problem.

In terms of metaheuristics, various researches have been done on the previously detailed extensions of our problem. [12] propose a Genetic Algorithm (GA) to tackle a multi-items CLSP and on multiple production lines, using various crossovers and mutation. The authors also use a new operator called "siblings" that consists in a local search using a ranking system on the neighbours. [13] propose a Tabu-Search (TS) to solve the same problem. [14] and [15] propose hybridized GA to solve the CLSP with an overtime constraints. The hybridization introduces elements of Tabu-search and Simulated Annealing into the GA in order to improve the efficiency of the algorithm. On top of that, they also use multi-population on different version of the algorithm to tackle their instances. On the single-machine CLSP with sequence dependent setup times, [16] and [17] propose

a Threshold Accepting whereas [18] develop a Tabu-Search and [19] propose a GA. To the best of our knowledge however, none of the previous problems incorporate a target-stock constraint similar to our case.

In the following, we denote CLSSD-PM the *multi-item capacitated lot-sizing problem with lost sales, safety stock, overtimes, and sequence dependent setups on parallel machines*. A previous work in [20] focus on a part of this problem without safety stock. To the best of our knowledge, this whole problem has never been studied in the literature before.

2 Problem Definition

The CLSSD-PM is a extensive version of the capacitated lot-sizing problem which is proven to be NP-hard ([21]). The goal is to plan the production of N different items, over T time periods and on M parallel unrelated production lines. There is a demand d_t^i for each item $i \in \{1, ..., N\}$ in each period $t \in \{1, ..., T\}$ that must be satisfied if units of i are available in stock. When that is not the case, the demand can be (partially or totally) lost, incurring a per-unit lost sales cost l_t^i. Any production of item i in period t on line $m \in \{1, ..., M\}$ is an integral number of batches, i.e. a multiple of a fixed quantity Q^i of units. The production of one such batch incurs a cost p_{mt}^i and requires a production time τ_m^i. In addition, the production of items of type k immediately after items of type $i \neq k$ during a given period on machine m induces a setup time γ_m^{ik}.

Each line m at each period t has a (planned) time capacity of C_{mt}, but production overtimes are allowed up to a maximum total production time $\bar{C}_{mt}$. When production occurs during the planned capacity, the corresponding cost of line usage is c_{mt} per unit of time, but this cost increases to $c_{mt} + \bar{c}_{mt}$ when the production needs overtime, i.e. for any usage that exceeds C_{mt}.

We model item storage by the mean of a target stock S_{it} for each item i in each period t. Any unit of stock of i in period t that exceeds S_{it} induces an excess storage cost of h_{it}^+, while missing inventory to reach the target stock incurs a per-unit penalty equal h_{it}^-.

We also make the following hypothesis on our problem :

- Demand and inventory are satisfied and consumed following a FIFO rule. This implies that it is impossible to choose to loose some demand of an item that is held in stock.
- Setup times between items follow the triangle inequality rule.
- At the beginning of each period, each line is in a neutral state, and the setup time to start the production of the first item in any period is null.

The objective of our problem is to minimize the total cost of the production planning (line usage, production, storage and lost sales combined). For conciseness reasons, we do not present the MILP formulation here and instead refer the interested readers to the [22].

3 Genetic Algorithm

We now develop a genetic algorithm to address the CLSSD-PM. We start by introducing the general structure of the procedure, before presenting in mode details the chromosome representation, crossover and mutation operators.

3.1 Genetic Algorithm Pseudo-Code

We use a generational genetic algorithm (GA), which creates successive generations of a population of individuals, by using specific operators inspired by nature and called crossovers and mutations. We keep some overlapping between consecutive generations, i.e. some of the best elements obtained in the current population are retained for the next generation, to keep the most interesting information of what has been done in previous iterations.

To avoid being in a local optimum for too long, the algorithm sometimes performs a reset that re-generates randomly a large portion of the current population. This operation is done only after a long period without improvement of the best known solution. A pseudo-code of the procedure is presented in Algorithm 1.

Algorithm 1: *Genetic Algorithm*

```
curGen ← GeneratePopulation() ;
s* ← bestIndividual ;
while stopping criterion not met do
    nextGen ← overlap(curGen);
    while /nextGen/ < maxPopSize do
        if condCrossover then
            parent1, parent2 ← selectionCross(curGen);
            nextGen.add(crossover(parent1, parent2));
        end
        if condMutation then
            mutated ← selectionMutation(curGen);
            nextGen.add(mutation(mutated));
        end
    end
    if condReset then
        reset();
    end
    if cost(nextGen.bestIndividual) < cost(s*) then
        s* ← nextGen.bestIndividual ;
    end
    curGen ← nextGen ;
end
return s*
```

3.2 Chromosome representation

The problem requires two type of decisions: The first one assigns the production of items to periods and machines, while the second one aims at designing the production sequences. As a consequence, we propose the following independent variables that serve as chromosomes:

- x_{mt}: Set of tuples $\langle$ item ; quantity $\rangle$ produced on m during t.
- w_{mt}: Contains the ordered sequence of production on m during t.

Other necessary information to represent a solution are deduced from these two variables, using the dependent variables below:

- $cost$: Total cost of the solution.
- u_{mt}: Time usage of line m in period t.
- $prod_t^i$: Number of batches of i produced in period t.
- $stock_t^i$: Stock of item i available at the end of period t.
- L_t^i: Number of lost sales for item i in period t.

To ensure diversity within the population, we start a completely random chromosome generation. For each line in each period we draw randomly a subset of items and affect to each of them a random production quantity. The sequence is determined as the items are drawn. Since the goal is to minimize the objective function of our problem, we keep a fitness parameter $fitness = \frac{1}{cost(solution)}$ updated to ensure that the gaps between the costs of different solutions are proportional. Finally, the selection is made based on a roulette wheel mechanism, applied to the fitness of the population.

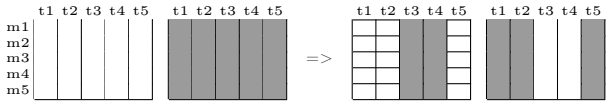
3.3 Crossover

In order to explore a large variety of solutions, we apply several crossovers from one generation to the next, in a similar fashion as the GA presented in [12]. In our case, we have three different crossover :

- On periods.
- On items.
- On sequences.

Crossover on periods. This crossover is heavily inspired by [12]. It basically consists of a two-point crossover applied on the periods of the solutions. The concept is to choose randomly a subset of periods and exchange all the production quantities of the two parents in the selected periods. Table 1 illustrates a crossover on periods 3 and 4.

Table 1. Illustration of the period crossover



Crossover on items. This crossover is inpired by [12] For a given machine m and a given time period t, this crossover iterates following the item information stored in the chromosomes x_{mt} of both parents. For each m and t, we consider the union the items produced by the two parents and draw for each of them a random boolean. If we draw 0 then the first child takes the first parent's production, and the second child takes the second parent's. Otherwise the first child takes the second parent's production, and the second child takes the first parent's. Table 2 shows a practical example of this crossover for a given period and line.

Table 2. Example of item crossover for a given period and line

Parent 1 :

item	1	4	3
quantity	15	13	9

Parent 2 :

item	2	9	4
quantity	17	10	5

Union and random draws :

union items	1	4	3	2	9
random draws	1	1	0	0	0

Child 1 :

item	4	3
quantity	5	9

Child 2 :

item	1	4	2	9
quantity	15	13	17	10

Crossover on the sequences. This crossover is inspired by [19] This crossover enables us to change the sequence of production. For each line and in each period, we form the set containing the common items from the two parent solutions. We then create a new sequence in the following manner:

1. Draw a random integer X between 1 and the number of common items
2. Order the X first item as they are in the sequence of the first parent. The remaining items follow the same order they have in the sequence of the second parent.
3. Create the sequences of the children using the parents sequences in which the common items are reordered.

Table 3 shows a practical example of this crossover for a given period and line.

3.4 Mutation

We consider a mutation that swaps the positions of two randomly selected items in the sequence of production, as represented in Table 4. As we also do not want to alter the totality of the individual we will add a parameter to describe the amount of of information that will be altered in a mutated individual.

Table 3. Example of sequence crossover for a given period and line

Parent 1 :	sequence	1	2	3	4	5	6	7	11
Parent 2 :	sequence	10	9	8	7	6	5	4	3
Intersection :	common items	3	4	5	6	7			
Random number draw X : 3									
Order from the parents :	Order of the first X items on parent 1	3	4	5					
	Order of remaining items on parent 2	7	6						
New order :	New order	3	4	5	7	6			
Child 1 :	sequence	1	2	3	4	5	7	6	11
Child 2 :	sequence	10	9	8	3	4	5	7	6

Table 4. Example of a mutated sequence on items 2 and 6

sequence before mutation	1	2	3	4	5	6
sequence after mutation	1	6	3	4	5	2

3.5 Repair

Note that such movements may result in infeasible solution since some line usage may exceed its maximum capacity. When this situation arise, we repair them by removing the production of one or more items until we don't exceed the hard capacity anymore and then replace it if possible on previous periods. In order to have a minimum impact on the quality of the solution, we chose to remove the item having the highest ratio $\frac{prod_t^i}{demand_t^i}$ so that we can avoid most of the lost sales. The quantity the remove in order to make the period feasible, is stored and will be spread on the previous periods where the item was already in production.

4 Experimentation Results

The instances that we use for our numerical experiments are derived from practical applications defined by VIF, a software company specialised in solutions for the food industry.

4.1 Parameters

Our algorithm is tuned through 9 parameters that have been tested to choose the best possible values.

- Size of the population: 200 individuals.
- Number of generations: 15000 generations, limited to 2 minutes of execution.
- Percentage of overlapping population between generations: 10%.
- Percentage of rested population: 50%.
- Number of non-improving iterations needed to reset: 200.
- Crossover ratio: 90%.
- Mutation ratio: 10%.
- Percentage of information of an individual that will be mutated: 20%.
- Ratio between the different crossovers: 60% period crossover, 20% item crossover, and 20% sequence crossover.

4.2 Experimentation

Implementation and tests of the algorithms have been done in Java. Tests have been realised on a personal computer with the following characteristics :
OS : Ubuntu 18.04.4 LTS

Processor : Intel i5-7600K @ 4.200GHz × 4
GPU : NVIDIA GeForce GTX 1070
RAM : 16 Gb
Type : 64-bit

We tested our GA on 168 instances that combine the following parameters: Number of items $\in \{20, 30, 40, 50, 75, 100, 125\}$, number of lines $\in \{1, 2, 4, 6\}$ and number of periods $\in \{15, 30\}$. The lower bound and upper bound considered are based on the results computed by CPLEX in 4 hours using a MIP formulation of the problem. We compare our results with the best lower bound (LB) obtained by CPLEX using settings presented in [22] and compute the gap achieved by our procedure with the following formula:

$$Gap = \frac{GA.cost - LB}{LB} \times 100$$

4.3 Results

We tested the GA presented in this paper with a maximum computational time of 2 minutes and compared the solutions obtained with the ones found by CPLEX in 4 hours. Note that the latter are used as a baseline and do not represent a viable option for practitioners to do its large computational time. In fact except for the smallest instances, CPLEX rarely even finds a feasible solution within 2 minutes, which already gives the GA an edge in the specific application that is targeted. In addition we observe that in 45 out of the 168 tested instances, our GA obtains a better solution in 2 minutes than the one obtained by CPLEX in 4 hours. The distribution of these 45 instances is as follows:

- 0 case for 20 items.
- 1 case for 30 items (0 for 15 periods, 1 for 30 periods).
- 7 cases for 40 items (0 for 15 periods, 7 for 30 periods).
- 10 cases for 50 items (2 for 15 periods, 8 for 30 periods).
- 9 cases for 75 items (3 for 15 periods, 6 for 30 periods).
- 11 cases for 100 items (7 for 15 periods, 4 for 30 periods).
- 7 cases for 125 items (5 for 15 periods, 2 for 30 periods).

The table 5 compares the gaps obtained by CPLEX and our GA on groups of 6 instances of same size. For each group we retain the minimal gap obtained, the maximal gap and the mean gap for all 6 instances.

This table also shows clearly the great differences that can appear between solutions found by CPLEX on 2 instances of same size (example for instance of size 50-1-30 where we have a minimal gap of 292% and a maximal gap of 14 145%) whereas our GA shows closer values (min : 999%, max : 3 453%). In general, the consistency of the results obtained by the GA is better across instances of the same size: In particular it appears that the solutions from CPLEX seem more sensitive to the number of periods that our procedure. Even if the results obtained by our GA are far behind the ones obtained by the MIP for the smallest instances, they become competitive on larger ones. For the largest instances, our heuristic consistently outperforms in 2 minutes the feasible solution computed by CPLEX in 4 hours.

These results clearly demonstrate the tendency of metaheuristics, in this case a genetic algorithm, to deal quickly with complex problems, and their usefulness in practice to tackle large industrial instances compared to MIP formulations and commercial solvers. Finally, note that the two approaches can also be used in combination, where the solution find by the GA can serve as a first feasible solution for the MIP solver, in an attempt to speed up the its convergence towards an optimal solution.

5 Conclusion

In this work, we apply the well-known genetic algorithm paradigm to develop a dedicated algorithm that is able to run quickly on large industrial instances of a complex practical production planning

Table 5. Comparison of gaps obtained by CPLEX (4 hours) and our GA (2 minutes) by group of same size instances

Instances	CPLEX Gap(%)			GA Gap(%)		
items-lines-periods	Min	Max	Mean	Min	Max	Mean
20-1-15	0.1	4.1	**1.7**	381.9	625.8	**534.0**
20-1-30	1.8	19.9	**8.9**	492.1	836.4	**731.4**
20-2-15	0.1	10.0	**2.5**	258.0	520.3	**374.2**
20-2-30	2.4	13.2	**7.0**	527.8	1108.7	**707.8**
30-1-15	1.5	12.0	**5.0**	602.4	1 203.1	**907.3**
30-1-30	0.8	1 163.3	**277.1**	1 000.8	1 911.9	**1 243.8**
30-2-15	1.8	18.8	**9.5**	513.2	1 122.8	**792.2**
30-2-30	7.4	544.4	**190.9**	379.9	1 298.2	**975.5**
40-1-15	7.6	75.1	**40.2**	685.7	1 471.5	**1 094.8**
40-1-30	533.8	2 853.7	**1 198.4**	685.0	2 245.4	**1 352.4**
40-2-15	6.0	789.9	**145.7**	391.5	1 553.2	**1 025.3**
40-2-30	322.2	7 127.8	**3 177.8**	640.1	1 768.2	**1 383.7**
50-1-15	83.0	3 573.4	**842.4**	968.8	1 778.6	**1 411.2**
50-1-30	292.0	14 145.4	**4 063.2**	998.6	3 452.6	**2 033.9**
50-2-15	4.9	1 305.0	**735.0**	1 014.6	1 598.5	**1 351.5**
50-2-30	530.9	4 277.8	**2 543.6**	1 681.4	2 312.7	**1 914.2**
75-2-15	774.5	2 811.1	**1 713.8**	2 070.9	3 459.9	**2 891.2**
75-2-30	432.6	6 829.5	**2 79.3**	2 139.9	4518.3	**3 328.3**
75-4-15	464.6	2 134.6	**1 744.8**	1 883.3	3 079.3	**2 582.1**
75-4-30	3 141.6	7 866.4	**4 537.1**	1 912.0	4 281.5	**3 387.3**
100-2-15	1 163.1	4 824.4	**2 494.2**	2 055.4	5 426.2	**3 864.5**
100-2-30	4 406.2	8 608.9	**6 371.8**	3 921.6	5 789.1	**4 449.2**
100-4-15	742.3	5 212.6	**2 569.0**	2 787.8	4 832.3	**4 034.2**
100-4-30	3 243.8	6 843.9	**5 090.9**	4 273.1	6 017.9	**5 241.9**
125-4-15	3 126.3	11 547.6	**5 756.7**	3 945.9	6 101.4	**4 935.5**
125-4-30	4 960.9	19 640.6	**8 306.3**	5 348.4	7 110.8	**6 402.2**
125-6-15	2 437.1	6 862.7	**4 358.6**	3 169.0	6 314.2	**5 301.0**
125-6-30	5 021.4	17 563.0	**7 579.5**	5 423.7	7 130.9	**6 295.3**

problem. The main contributions of this study can be partitioned in two broad categories. First, the heuristic developed is the first one that takes into account several industrial extensions of classical lot-sizing problems, such as the combination of multiple unrelated machines and sequence-dependent setup times. Second, it provides a viable alternative to commercial solvers to deal with large industrial instances that displays a robust behavior with respect to the size of the problem considered. Note that the solution obtained using our procedure may serve as a warm start for an exact method.

While the first results obtained show that such metaheuristics are a viable alternative on large instances, additional work is necessary to improve the overall performances. In particular, the method would become a lot more reliable if the solutions on small instances were comparable to the ones computed by commercial solvers. Local search methods or more advanced concepts such as hybridization or multi-population could help reduce the gap in such cases. We could also seek to find dominance properties to reduce the search space and speed up the resolution.

Another research direction to achieve this goal is to apply the procedure to a simpler problem that approximates the original one. In a recent paper [22], we developed a procedure that computes clusters of items with small switching times, which enables the algorithm to primarily focus on positioning clusters in the production sequence rather than items. This approximation greatly reduces the size of the original problem and was proven successful when used in combination with classical heuristics from the lot-sizing literature. It is likely that the GA presented in this paper would also benefit from this reduction of the problem size to converge faster to good quality solutions.

References

1. Wagner, H.M., Whitin, T.M.: Dynamic Version of the Economic Lot Size Model. Management Science **5** (1958) 89–96
2. Quadt, D., Kuhn, H.: Capacitated lot-sizing with extensions: a review. 4OR **6** (2008) 61–83
3. Karimi, B., Fatemi Ghomi, S., Wilson, J.: The capacitated lot sizing problem: a review of models and algorithms. Omega **31** (2003) 365–378
4. Absi, N., Kedad-Sidhoum, S.: The multi-item capacitated lot-sizing problem with setup times and shortage costs. European Journal of Operational Research **185** (2008) 1351–1374
5. Loparic, M., Pochet, Y., Wolsey, L.A.: The uncapacitated lot-sizing problem with sales and safety stocks. Mathematical Programming **89** (2001) 487–504
6. Absi, N., Kedad-Sidhoum, S.: The multi-item capacitated lot-sizing problem with safety stocks and demand shortage costs. Computers & Operations Research **36** (2009) 2926–2936
7. Beraldi, P., Ghiani, G., Grieco, A., Guerriero, E.: Rolling-horizon and fix-and-relax heuristics for the parallel machine lot-sizing and scheduling problem with sequence-dependent set-up costs. Computers & Operations Research **35** (2008) 3644–3656
8. Clark, A.R., Morabito, R., Toso, E.A.V.: Production setup-sequencing and lot-sizing at an animal nutrition plant through atsp subtour elimination and patching. Journal of Scheduling **13** (2010) 111–121
9. Gicquel, C., Minoux, M., Dallery, Y., Blondeau, J.M.: A tight mip formulation for the discrete lot-sizing and scheduling problem with parallel resources. In: 2009 International Conference on Computers & Industrial Engineering, IEEE (2009) 1–6
10. Guimarães, L., Klabjan, D., Almada-Lobo, B.: Modeling lotsizing and scheduling problems with sequence dependent setups. European Journal of Operational Research **239** (2014) 644–662
11. Özdamar, L., Bozyel, M.A.: The capacitated lot sizing problem with overtime decisions and setup times. IIE Transactions **32** (2000) 1043–1057
12. Hung, Y.F., Shih, C.C., Chen, C.P.: Evolutionary algorithms for production planning problems with setup decisions. The Journal of the Operational Research Society **50** (1999) 857
13. Hung, Y.F., Chen, C.P., Shih, C.C., Hung, M.H.: Using tabu search with ranking candidate list to solve production planning problems with setups. Computers & Industrial Engineering **45** (2003) 615–634
14. Özdamar, L., Birbil, Ş.İ.: Hybrid heuristics for the capacitated lot sizing and loading problem with setup times and overtime decisions. European Journal of Operational Research **110** (1998) 525–547
15. Özdamar, L., Bilbil, Ş.İ., Portmann, M.C.: Technical note: New results for the capacitated lot sizing problem with overtime decisions and setup times. Production Planning & Control **13** (2002) 2–10
16. Fleischmann, B., Meyr, H.: The general lotsizing and scheduling problem. Operations-Research-Spektrum **19** (1997) 11–21
17. Meyr, H.: Simultaneous lotsizing and scheduling by combining local search with dual reoptimization. European Journal of Operational Research **120** (2000) 311–326

18. Laguna, M.: A heuristic for production scheduling and inventory control in the presence of sequence-dependent setup times. IIE Transactions **31** (1999) 125–134
19. Sikora, R.: A genetic algorithm for integrating lot-sizing and sequencing in scheduling a capacitated flow line. Computers & Industrial Engineering **30** (1996) 969–981
20. Larroche, F., Bellenguez-Morineau, O., Massonnet, G.: Approche de résolution d'un problème industriel de lot-sizing avec réglages dépendant de la séquence. In: ROADEF 2020 : 21ème Congrès Annuel de la Société Française de Recherche Opérationnelle et d'Aide à la Décision, Montpellier, France (2020)
21. Florian, M., Lenstra, J.K., Rinnooy Kan, A.H.G.: Deterministic Production Planning: Algorithms and Complexity. Management Science **26** (1980) 669–679
22. Larroche, F., Bellenguez-Morineau, O., Massonnet, G.: Clustering-based solution approach for a capacitated lot-sizing problem on parallel machines with sequence-dependent setups. To appear in International Journal of Production Research (2020)