A matheuristic approach for the design of multiproduct batch plants with parallel production lines

A matheuristic approach for the design of multiproduct batch plants with parallel production lines

Floor Verbiest[a,*], Trijntje Cornelissens[a], Johan Springael[a]

[a] *ANT/OR - Operations Research Group,*
Department of Engineering Management,
University of Antwerp, Prinsstraat 13, 2000 Antwerp, Belgium

Abstract

Batch processes are typically used to manufacture, among other, specialty and fine chemicals. As the construction of grassroot batch plants requires major investments, models for determining the optimal design of such plants have been developed over the past decades. These models are often formulated as Mixed Integer Linear Programming (MILP) models which are solved exactly. In a previous study, we introduced the concept of parallel production lines as a design option into existing mathematical plant design models. The design problem now also aims at optimising the number of production lines, their design and the allocation of products (and production quantities) to the installed lines. However, with this extension, the complexity increases significantly. To tackle this combinatorial divergence, we formulated a matheuristic solution approach which combines an iterated local search metaheuristic with exact MILP calculations. In this paper, the hybrid solution method is described and its performance, in comparison to an exact algorithm, is illustrated for several example problems. It was found that our matheuristic obtained very good solutions in significant lower computation time. As a consequence, this technique is suitable to solve more realistic instances and enables us to expand these design models with e.g. different objectives in the future.

Keywords: Metaheuristics, Chemical batch plant, Plant design, Matheuristics, Combinatorial optimisation.

*Corresponding author
Email addresses: floor.verbiest@uantwerpen.be (Floor Verbiest), trijntje.cornelissens@uantwerpen.be (Trijntje Cornelissens), johan.springael@uantwerpen.be (Johan Springael)

1. Introduction

1.1. Research context

The global market of specialty and fine chemicals, which are typically made in batch plants, is predicted to grow in the upcoming years (Technavio, 2016). This growth is accompanied by the need for upgrading existing and building new plants. Since this is associated with immense investments, appropriate capacity assessments are needed. Such strategic capacity decisions form the main subject of the batch plant design problem (BPDP), the purpose of which is to select the optimal number and size of equipment units for every production stage so as to optimise a strategic business objective, subject to design and horizon constraints. In literature, the BPDP has been studied extensively over the years, both for multiproduct and multipurpose plants, and the design models have been extended with different strategic choices on e.g. modes of operation, objectives and design options (see Verbiest et al. (2017) and references therein).

Recently, the option of parallel production lines has been introduced into the multiproduct BPDP (Verbiest et al., 2017). These parallel lines are installed on the same production site and operate independently but simultaneously. Since they have the same processing stages, production may be divided over all these lines (Hill et al., 2016). Moreover, the added value of parallel production lines is even more justified when the objective function includes, besides capital costs, also startup and contamination costs. The startup cost is incurred for every equipment unit, every time a series of batches of the same product starts, and represents a fixed product dependent cost of preparing the equipment. The contamination cost depends on the combination of products produced on the same line. Indeed, products with different characteristics, belonging to different product families, generate contamination if they are produced on the same equipment. The aforementioned paper revealed that the presence of parallel lines can lead to a reduction in capital and startup costs as not all products have to share all equipment any more, leading to smaller equipment and less to start up. Additionally, high contamination costs can be avoided through the dedication of these lines to product families.

With this new design option, the aim of the design problem is to determine the optimal number of lines to install, their individual design and the product assignment to these lines so as to minimise the total cost. Until now, this problem was formulated as a Mixed Integer Linear Programming (MILP) model that is solved using exact solution methods.

1.2. Literature review

In the research field on batch plant design, the problem is often described as an MILP model and solved with exact solution methods, such as (derivations of) branch and bound (Barbosa-Póvoa, 2007). However, due to the combinatorial nature of (mixed-)integer programming, there is a high computational burden that grows with the problem size (Chibeles-Martins et al., 2010). Since the multiproduct BPDP with parallel production lines involves additional integer decisions, large problems become intractable for exact solution methods. Therefore, other methods such as meta- and matheuristics are needed.

Heuristic solution techniques have been applied for the design of single line batch plants in a number of articles. Already in the 70's, Sparrow et al. (1975) applied a heuristic method, based on a single representative hypothetical product, to determine the design of a multiproduct batch plant with identical parallel equipment at a minimum investment cost. In the years following, heuristic rules have been formulated to solve both multiproduct and multipurpose BPDP's, with or without intermediate storage (Modi & Karimi, 1989; Tan & Mah, 1998). More recently, Corsano et al. (2007) specified heuristic rules for the combined design and scheduling problem of a (sequential) multipurpose batch plant so as to maximise profit. Besides specific heuristic procedures, different types of metaheuristics have been developed to solve the design problem. Local search metaheuristics for the design of multiproduct batch plants with intermediate storage and parallel equipment are reported by Patel et al. (1991) using simulated annealing and by Wang et al. (1999) implementing tabu search. The aim is in both cases to minimise investment costs. Similar types of metaheuristics are also developed for the design and scheduling of multipurpose plants, i.e. tabu search (Cavin et al., 2004) and simulated annealing (Chibeles-Martins et al., 2010). The first optimised a multiobjective retrofit case taking into account throughput and number of equipment units, whereas the latter paper considered a grassroot plant for the maximisation of profit. Jayaraman et al. (2000) applied ant colony optimisation for the design (and separate scheduling) of a multiproduct batch plant at minimal investment cost. This technique is an example of the use of constructive metaheuristics. Finally, there are a number of articles on population based algorithms, more specifically on genetic algorithms. Wang et al. (1996) and Ponsich et al. (2008), for example, solved the multiproduct design problem with intermediate storage and identical parallel equipment while minimising investment costs. Moreover, as genetic algorithms are especially suitable for multiobjective optimisation problems, examples of this technique for multiobjective design problems are given by

Dedieu et al. (2003); Dietz et al. (2006); Aguilar-Lasserre et al. (2009). In these last papers, the genetic algorithms are often combined with discrete event simulation for evaluating the chosen plant structure and deciding on a schedule.

Indeed, apart from pure metaheuristics, hybrid metaheuristics have been introduced in the last decades for solving combinatorial optimisation problems (Jourdan et al., 2009). Such hybrid metaheuristics combine elements of metaheuristics with other techniques for optimisation, such as constraint programming or mathematical programming. The combination with the latter is also referred to as matheuristics (Boschetti et al., 2009). As stated by Blum et al. (2011), the main reason for combining different algorithms is to exploit the complementary character of different optimisation techniques. More specifically, although metaheuristics do occur for optimisation problems containing continuous variables, these methods are not so well suited. Conversely, exact methods perform extremely well for (linear) optimisation problems without discrete variables. Hence, in case of a mixed problem, it could be advantageous to combine metaheuristics and mathematical programming. Depending on the combination and interaction of both techniques, Ball (2011) proposed four different classes of matheuristics: decomposition approaches, improvement heuristics, the use of mathematical programming algorithms to generate approximate solutions and relaxation based approaches. In the remaining of this paper, we will concentrate on the decomposition approaches. Concerning the BPDP, pioneering research on (decomposition) matheuristics for the multipurpose case was undertaken by Xi-Gang & Zhong-Zhou (1997), where they combined simulated annealing and linear programming. Simulated annealing was adopted for optimising the discrete decisions, such as number and size of equipment units, which are to be chosen out of a discrete set. The LP, on the other hand, was applied to determine the batch sizes of and time spent per product. This article assumed, however, a single line design problem. Other applications of matheuristics for the BPDP are hardly found.

1.3. Contributions

In this paper, we present a decomposition matheuristic that combines MILP calculations with a metaheuristic to solve the multiproduct BPDP with parallel lines. This problem is described into detail in Verbiest et al. (2017), where it was solved exactly. The main motivation for using matheuristics is to obtain very good solutions in a limited amount of time for realistic instances. This is achieved by exploiting the characteristics of both solution methods when optimising the different types of decisions.

Our first contribution concerns the metaheuristic employed within the matheuristic. We implemented an iterated local search (ILS) metaheuristic, a technique successfully applied to a variety of combinatorial optimisation problems, and illustrated its performance for the single line BPDP. This ILS is re-used in the developed matheuristic for specific cases of the multiple lines design problem, and contributes to a great extent to the efficiency of the matheuristic.

The second contribution of this paper is the development of the entire matheuristic. As indicated previously, the aim of the design problem with parallel production lines is threefold: determining the optimal number of lines to install, their individual design and the product assignment to these lines. This problem can be decomposed into 3 logical subproblems, each solved with their specific method, hence, legitimising the choice for a decomposition approach. Moreover, the design problem with multiple lines can be tackled in a hierarchical manner in which first higher-level decisions are fixed and then, in successive phases, the remaining lower-level variables are determined (Raidl, 2015). The first step of our approach will be to fix the number of lines, as this is a higher-level decision. Then, for every number of lines, the design and assignment subproblems are iteratively optimised, following the structure of an ILS algorithm. Within this comprehensive ILS structure, our developed single line ILS and different types of MILP calculations are implemented, depending on the subproblem. A general overview of the matheuristic with its different subproblems and how they relate is given in Figure 1.

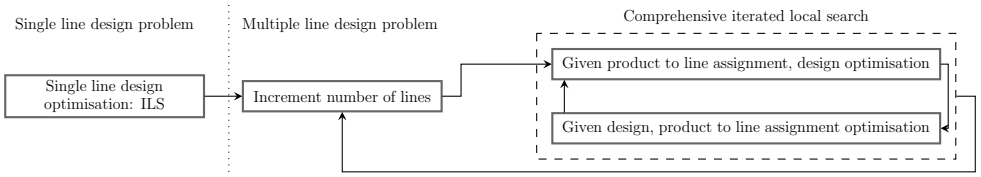

Figure 1: Overview of the matheuristic with 1) the single line design problem and 2) the multiple lines design problem, consisting of 3 subproblems: for a given number of lines, iterative procedure to determine the optimal assignment and design

The remainder of this paper is structured as follows: in the next section (Section 2), a description of the BPDP with parallel lines is provided, followed by the accompanying mathematical model (Section 3). The developed ILS metaheuristic for a single line design problem is described in Section 4. Then, in Section 5, a detailed explanation of our decomposition matheuristic, encompassing the previously described ILS, is presented. Both the ILS and matheuristic are validated by solving several example problems of

various size. The results, compared to an exact algorithm, are outlined in Section 6.

2. Problem description and assumptions

In this section, the parallel lines multiproduct chemical BPDP is explained, as presented in Verbiest et al. (2017). The nomenclature used is provided in Appendix A.

We assume that the plant to be designed can have at most L lines l. Every line consists of J stages j, needed to produce P products i. The demand for every product (Q_i) and the total production horizon (H) are known upfront, as well as the size factors (S_{ij}) and fixed batch processing times (τ_{ij}) for every product i at every stage j. The processing times are assumed to be independent of the product batch size. Further assumptions, as essentially specified by Voudouris & Grossmann (1992), are:

1. Only batch equipment is explicitly considered;
2. Zero-wait policy between the batch stages so that a batch is immediately transferred from one stage to the next;
3. Overlapping mode of operation to eliminate idle times (no need to wait until one batch has completely passed through all stages);
4. At most N identical parallel equipment per stage (n_j), operating out-of-phase, i.e. batches are supplied from the previous stage shifted in time;
5. Cycle time of product i is the longest stage cycle time over all stages: $\max_{j=1,\ldots,J} \tau_{ij}/n_j$;
6. Discrete set of S equipment sizes v_s for all stages j to choose from;
7. Single product campaign mode of operation and no intermediate due dates.

The latter assumption means that total demand of one product is produced before switching to another. This means that no explicit sequencing nor specified start and end times are needed (Applequist et al., 1997). Although this is a simplification of reality, it gives for now sufficient information to tackle the design problem at a strategic level. More explicit scheduling will be addressed in future work.

Within this setting, the aim of the problem is to determine the optimal:

1. Number of lines to install, with at most L lines
2. Design of every installed line, encompassing the number and size of equipment units at every stage (chosen out of a discrete set of options)
3. Assignment of products to the installed lines, where a product may be assigned to multiple lines

so as to minimise the total cost while satisfying design and timing constraints. The total cost consists of capital, startup and contamination costs.

3. Mathematical formulation and objective function

The parallel lines multiproduct chemical BPDP is mathematically formulated in this section. We refer to Verbiest et al. (2017) for the detailed formulation with explicit linearisation expressions.

3.1. Constraints

Production line design constraints:

In order to determine the existence of lines, the binary e_l is introduced, being 1 if line l exists and zero otherwise. For every line, the number of equipment per stage should be defined. This decision is captured in the binary variable z_{ljn}, which equals 1 if unit n at stage j of line l is installed. In any other case this variable is zero. Furthermore, the related assignment of products to the installed lines is indicated with variable t_{li}, being 1 if product i is produced on line l or zero if not. Hence, the following constraints apply:

$$(NJ)e_l \geqslant \sum_{j=1}^{J}\sum_{n=1}^{N} z_{ljn} \qquad \forall l \tag{3.1}$$

$$e_l \leqslant \sum_{n=1}^{N} z_{ljn} \qquad \forall l, j \tag{3.2}$$

$$t_{li} \leqslant e_l \qquad \forall l, i \tag{3.3}$$

Eq.(3.1) indicates that no equipment units can be installed on a line that does not exist. If the line is installed, there are at most N and at least 1 equipment units per stage. This is captured in Eqs.(3.1)-(3.2), imposing the upper and lower bounds respectively. Finally, Eq.(3.3) means that no product can be assigned to a line that does not exist.

Equipment design constraints:

A binary variable u_{ljs} was introduced to indicate whether or not equipment of stage j of line l has size s, which leads to the constraints:

$$\sum_{n=1}^{N} z_{ljn} \leqslant N \sum_{s=1}^{S} u_{ljs} \qquad \forall l, j \tag{3.4}$$

$$\sum_{n=1}^{N} z_{ljn} \geqslant \sum_{s=1}^{S} u_{ljs} \qquad \forall l, j \tag{3.5}$$

$$n_{li} \geqslant \sum_{s=1}^{S} \frac{q_{li} S_{ij}}{v_s} u_{ljs} \qquad \forall l, j, i \tag{3.6}$$

Eqs.(3.4)-(3.5) define the relation between the existence of a stage and the related choice of the equipment size. Eq.(3.4) induces the choice of a size if equipment is installed at stage j of line l, whereas Eq.(3.5) states that if no equipment is installed at stage j of line l, no size is chosen. Furthermore, for every stage j of line l the size of the equipment v_s should be large enough to hold a batch of every product i, multiplied by its size factor S_{ij}. Eq.(3.6) is obtained through the incorporation of the discrete set of sizes and the expression q_{li}/n_{li} to represent a batch of product i on line l, with q_{li} the amount of product i produced on line l and n_{li} the number of batches.

Horizon constraints:

The total time spent on every product i on stage j of line l (T_{lji}) corresponds to the time spent per batch (stage cycle time) multiplied by the number of batches (Eq.(3.7)). Since parallel equipment per stage operate out-of-phase, this affects the time necessary to produce a batch. Total time spent on every product i on line l (θ_{li}) is equal to the longest time spent on a stage of that line (Eq.(3.8)). Lastly, Eq.(3.9) states that total production time per line should not exceed the production horizon H.

$$\sum_{n=1}^{N} z_{ljn} T_{lji} = n_{li} \tau_{ij} \qquad \forall l, j, i \tag{3.7}$$

$$\theta_{li} \geqslant T_{lji} \qquad \forall l, j, i \tag{3.8}$$

$$\sum_{i=1}^{P} \theta_{li} \leqslant H \qquad \forall l \tag{3.9}$$

Demand constraint:

The production of product i split over multiple lines l should add up to the demand Q_i:

$$\sum_{l=1}^{L} q_{li} = Q_i \qquad \forall i \tag{3.10}$$

Boundaries:

Eqs.(3.11)-(3.14) pose an upper bound on the variables. M_i was included as an upper bound on the number of batches for every product i, which is formulated as $(Q_i S_i^{max})/v_0$. Eq.(3.15) indicates that if no amount is produced of product i on line l, this product is

not produced at all on that line ($t_{li} = 0$). Moreover, to allow t_{li} to be 1 when $0 < q_{li} < 1$ (e.g. $q_{li} = 0.7$), $\epsilon \in]0, 0.01]$ is included.

$$q_{li} \leqslant Q_i t_{li} \qquad \forall l, i \tag{3.11}$$

$$n_{li} \leqslant M_i t_{li} \qquad \forall l, i \tag{3.12}$$

$$T_{lji} \leqslant H t_{li} \qquad \forall l, j, i \tag{3.13}$$

$$\theta_{li} \leqslant H t_{li} \qquad \forall l, i \tag{3.14}$$

$$\epsilon t_{li} \leqslant q_{li} \qquad \forall l, i \tag{3.15}$$

3.2. Objective function

The objective function consists of three parts: capital, startup and contamination costs.

$$\min \left[\sum_{l=1}^{L} \sum_{j=1}^{J} \sum_{n=1}^{N} \left(\sum_{s=1}^{S} c_{js} z_{ljn} u_{ljs} + \sum_{i=1}^{P} Cstart_i \, z_{ljn} t_{li} + \sum_{f=1}^{F} Ccont \, b_{fl} z_{ljn} \, g_l \right) \right] \tag{3.16}$$

Capital costs. The first term is the capital cost expression with $c_{js} = \alpha_j v_s^{\beta_j}$ and α_j, β_j the stage dependent cost coefficients, with $\beta_j < 1$. This one time capital expenditure is assumed to be already converted to a uniform cost per horizon so that it can be correctly added up with the other cost components (Jelen et al., 1983).

Startup cost. The preparation of the equipment units at the start of every series of batches of product i is modelled in the second term, where $Cstart_i$ is a line and stage independent startup cost. If t_{li} equals one, all the equipment units of line l, given by $\sum_{j=1}^{J} \sum_{n=1}^{N} z_{ljn}$, have to be set up for producing product i.

Contamination cost. This cost represents a penalisation for producing products from different product families on the same equipment. As in this paper no exact sequencing of products is determined, a fixed cost $Ccont$ is assumed that occurs when changing from one product family to another for all equipment units on a production line. The binary variable b_{fl} is introduced to indicate if product family f is produced on line l. This is calculated via the constraint:

$$b_{fl} \geqslant d_{if} t_{li} \qquad \forall l, f, i \tag{3.17}$$

with d_{if} a given boolean matrix that specifies if product i belongs to family f. To avoid the generation of a contamination cost when there is only one product family on a line,

a correction via the binary g_l is included, together with the following constraints:

$$\sum_{f=1}^{F} b_{fl} \geqslant 2 - F(1 - g_l) \qquad \forall l \tag{3.18}$$

$$\sum_{f=1}^{F} b_{fl} - 1 \leqslant F g_l \qquad \forall l \tag{3.19}$$

These constraints force g_l to zero if there is no or 1 product family on the equipment, with F the total number of product families.

Eventually, the aim of this problem is to minimise the 3-fold objective function (3.16) subject to constraints (3.1)-(3.15) and (3.17)-(3.19). This problem contains two types of nonlinearities: products of a binary and a continuous variable (in Eq. (3.6) and Eq. (3.7)) and products of binary variables (in the three cost components of the objective function). We applied known linearisation techniques, as described in Glover (1975), to obtain an MILP model. The expressions are included in Appendix B.

4. Iterated local search metaheuristic for the single line plant design problem

Solving the model of the previous section with an exact method becomes very time consuming or even intractable for larger problems. Therefore, the BPDP with multiple lines is decomposed and specific methods are used to solve the different subproblems. It appears that a major strength of the proposed matheuristic lies in the ability to use a powerful metaheuristic for determining the optimal number and size of the equipment units in every stage of a single line ($L = 1$). As in this case there are only discrete decisions, metaheuristics are proven to be a useful technique (Sörensen & Glover, 2013). We have chosen to address this design problem with an ILS algorithm, as described in detail by Lourenço et al. (2010) and successfully applied to a variety of optimisation problems in the past (e.g. Stützle (2006); Öncan (2015)). The specific implementation of the algorithm and its parameter settings are discussed in Sections 4.1 and 4.2.

4.1. Implementation of ILS for the single line plant design problem

The implementation of the ILS algorithm for the single line BPDP, hereafter referred to as 1-ILS, is discussed. It follows the structure of a general ILS, as described in Figure 2.

Initial solution. As stated in Figure 2, an initial feasible solution s_0 is constructed. The solution considered is a plant with a maximal design: all stages have the maximum

Algorithm 1 Iterated Local Search

1: $s_0 \leftarrow$ *InitialSolution*
2: $s \leftarrow$ *LocalSearch*(s_0)
3: **while** Termination criterion is not met **do**
4: $s' \leftarrow$ *perturbation(s)*
5: s" $\leftarrow$ *LocalSearch(s').*
6: $s \leftarrow$ *AcceptanceCriterion(s", s).*
7: **end while**

Figure 2: Pseudocode ILS (Lourenço et al., 2010)

number of parallel equipment units of maximal size. This solution should satisfy the design and demand constraints, i.e. the designed plant must be large enough to produce all the products before the end of a given time horizon. If this solution violates a constraint, the problem is infeasible.

Local search. In order to improve the solution (s_0), neighbouring solutions are generated by considering two different move types:

- Move type (1): Decrease the number of batch units of a stage by one
- Move type (2): Decrease the size of the batch units of a stage by one (i.e. pick one size smaller out of the discrete set).

So, all neighbouring solutions are identical except for one stage. To determine which neighbouring solutions are constructed, the move type is selected through the generation of a random number (in [1,10]). If this number is lower than `threshold-ILS` move type (1) is selected, otherwise move type (2). `threshold-ILS` (in [1,10]) is one of the parameters of the metaheuristic that is to be set experimentally (see Section 4.2 hereafter). Hence, in case `threshold-ILS` is low, the chance of decreasing the number of units is lower than decreasing the size.

To decide on the acceptance of a neighbouring solution as the new solution, a steepest-descent strategy is used. This involves the generation of all neighbouring solutions, i.e. every stage is considered to perform the move on, and the selection of the one with the lowest cost (move acceptance criterion). This solution's neighbourhood is again generated and studied. The procedure ends when no better neighbouring solutions are obtained. In this case, a local optimum s has been found which is considered as the best solution found so far.

Perturbation. Next, a perturbation step is performed to escape from the local optimum s. In this step, a percentage of the stages (given by parameter `perturbationRate-ILS`) are

reset to the largest sizes from the discrete set or to the maximum number of equipment units. The choice for one of these perturbations is determined through the comparison of a random number and a parameter `thresholdPerturbation-ILS`, both in [1,10]. The perturbation is applied to the overall best solution s and the local search is restarted from this perturbed solution s' to obtain s".

Acceptance criterion. This local optimum s" will be compared to the best solution found so far s. If s" is better than s, i.e. if a design with a lower total cost is found, the solution s" is accepted as best solution found so far, if not the latter remains unchanged.

Termination criterion. The algorithm ends if there is no improvement of the best solution found so far for a fixed number of perturbations (as specified by parameter `noImprovement-ILS`).

4.2. Parameter tuning for 1-ILS

As 1-ILS is controlled by different parameters, the values for every parameter need to be set before the metaheuristic is used. To determine these values, a parameter tuning experiment is executed on a number of single line instances of different sizes. The parameters `noImprovement-ILS`, `perturbationRate-ILS`, `threshold-ILS` and `thresholdPerturbation-ILS`, described in Section 4.1, are summarised in Table 1 together with the tested and chosen values. The tested values were empirically selected from a few preliminary tests.

Table 1: Parameters 1-ILS

Parameter name	Possible values	Chosen value[a]
`noImprovement-ILS`	50, 100, 150	**100**
`perturbationRate-ILS`	20, 40, 60	**40**
`threshold-ILS`	4, 5, 6, 7	**7**
`thresholdPerturbation-ILS`	4, 5, 6	**4**

[a] for every instance, every combination is tested 10 times

A full factorial experiment is conducted for the values reported and tested 10 times for all possible combinations, resulting in $3 \times 3 \times 4 \times 3 \times 10 = 1080$ calculations, per test instance. As these instances have different sizes and the resulting total costs can have a different order of magnitude, normalised costs are used for comparability over the instances. Indeed, for every instance, the lowest cost found by 1-ILS over the 1080 runs is taken as a point of reference (i.e. gets value 1) and all other solutions are related to this reference. Next, for every parameter value, the average of the relative costs over all test instances, as well as the average computation times, are calculated and shown in

Figure 3. The closer the average cost to 1, the better the solution.

From Figure 3(a), it can be observed that the average relative cost decreases with increasing value of `noImprovement-ILS`. However, there is an almost linear relationship between number of perturbations and computation time. In order to preserve a good balance between computation time and solution quality, we chose 100 perturbations as the stopping criterion (`noImprovement-ILS`). The perturbation rate `perturbationRate-ILS` (Fig. 3(b)) that results in the best performance of the 1-ILS is 40 %. This can be explained by the fact that 20 % may prevent the algorithm to escape from poor local optima, whereas a 60 % perturbation rate may lead to a significant loss of information on the current local optimum. Finally, for `threshold-ILS`, a higher probability of decreasing the number of units (high value in Fig. 3(c)) combined with a lower probability of setting them again to the maximum (low value for `thresholdPerturbation-ILS` in Fig. 3(d)) is preferred. The chosen values are summarised in the last column of Table 1 and are used in all following 1-ILS calculations.

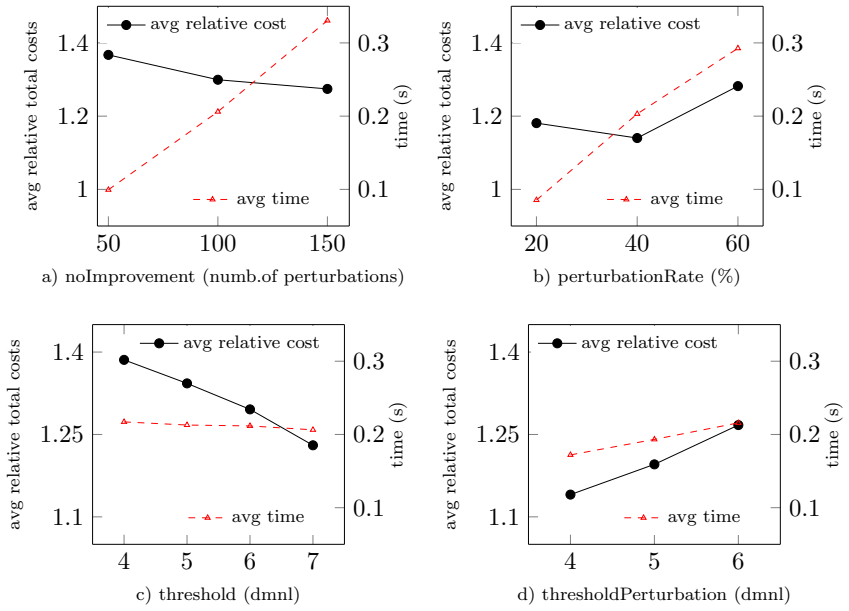

Figure 3: Full factorial experiment - 1-ILS

It is important to remark that the previously described 1-ILS can also be used for designing a plant with multiple lines, if the product to line assignment (t_{li}) is given and every product is produced on a single line. This implies that $q_{li} = Q_i$ and the lines can be designed separately. This is explained in greater detail in the next Section, describing

the matheuristic for the multiple lines BPDP.

5. Decomposition matheuristic for the multiple lines plant design problem

In this section, the matheuristic solution procedure for the multiple lines BPDP is described. We distinguish three main structures: the design of a single line plant, an outer loop to determine the number of lines and a comprehensive ILS procedure to solve the BPDP for a fixed number of lines.

5.1. Evaluation of the single line plant design

As shown in Figure 4, the procedure starts with initialising some parameters: the number of lines *NumbLin* is set to 1, a boolean *maxPlantFeasible* is flagged false and the cost of the overall best solution $s_{OverallBest}$ is set to a very large value $s_{maxCost}$. Next, the feasibility of the single line plant design is tested, by checking the demand constraint satisfaction considering the maximum and largest equipment units in every stage (maxPlant). If all products can be produced within the given horizon, the boolean *maxPlantFeasible* is set to true, the design problem is solved with the 1-ILS metaheuristic and $s_{OverallBest}$ is updated. Otherwise, this single line configuration is infeasible and the parameters remain unchanged. After this, the outer loop is entered.

5.2. Outer loop for determining the number of lines

In this outer loop, the number of lines are determined in an incremental manner and the optimal design is calculated for every incremented number of lines. The decision to fix the number of lines first is legitimate as the installation of an additional line has a large impact on a plant. Indeed, every additional line implies installing at least one unit in every stage and a reassignment of products to lines. Furthermore, the maximum number of lines that can be installed is mostly fairly limited, hence, the number of possibilities is restricted.

Before optimising the design for a specific number of lines, two tests have to be performed. Firstly, as long as *maxPlantFeasible* is flagged false, the feasibility of the plant with an incremented number of lines has to be checked. Hence, the demand constraint satisfaction is re-evaluated by considering the maximum and largest equipment units in every stage of every line (maxPlant) and by solving the product to line assignment problem for a fixed design (see further Section 5.3.1). If this is still infeasible, the number of lines is augmented once more. Once it is feasible to produce all products on time, the

parameter *maxPlantFeasible* is set to true and this test is no longer required. Indeed, once a line configuration is feasible, all subsequent (larger) line configurations are feasible as well. Secondly, a preliminary cost evaluation of the minimum plant (minPlant) is performed. This means the minimal total cost of the line configuration, i.e. smallest design installed and fewest contamination possible, is calculated ($s_{minPlant}$) and compared to $s_{OverallBest}$. Indeed, for solutions with multiple lines, it is possible that $s_{minPlant}$ is already higher than the lowest cost found so far. In that case, this particular line configuration can be discarded. The number of lines is again augmented, as the installation of more lines may decrease the total cost. After all, total operating costs may be reduced and may offset the (likely) increase in capital costs.

If both the feasibility and minimal cost conditions are satisfied, the procedure continues with the optimisation of the plant design for the determined multiple number of lines. This will be done by the comprehensive ILS procedure, described in the next subsection. After this optimisation, $s_{OverallBest}$ is updated if lower costs are found. The entire procedure is repeated until the maximum number of lines is reached.

5.3. Comprehensive iterated local search for a fixed number of lines

The procedure to optimise the design of a plant with a fixed number of lines is shown in Figure 5. As this procedure follows the structure of an iterated local search algorithm, we called it a comprehensive ILS: first, the products are assigned to the fixed number of lines ($PrToLin$) using a set of rules as explained in Section 5.3.1. Then, the corresponding optimal design ($design$) is determined with 1-ILS or with an exact solver (d-MILP) as described in Section 5.3.2. The total cost obtained is compared with the lowest cost found so far ($s_{OverallBest}$) and the better of the two is stored. Next, starting from this solution, the assignment and design problem are iteratively resolved until a local optimum is found. This iterative process is shown in the dashed rectangle in Figure 5: if for a given design, a better product to line assignment is found (i.e. one that lowers the total cost), this new assignment is kept ($PrToLin \leftarrow PrToLin^*$) and its optimal $design^*$ is calculated. If a less expensive design is found, this one is continued from ($design \leftarrow design^*$). For every new design, a cost evaluation is performed and $s_{OverallBest}$ is updated if required. When either the new assignment ($PrToLin^*$) or the new design ($design^*$) does not change any more, a local optimum is encountered.

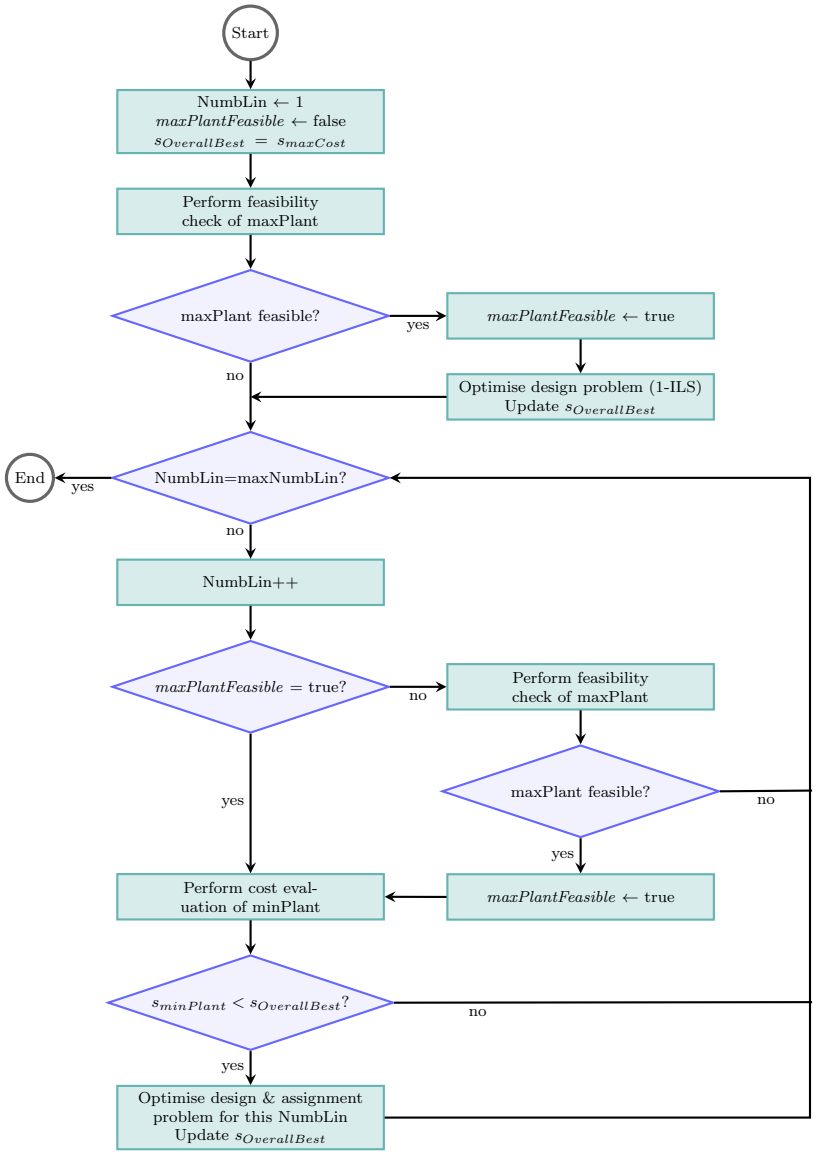

Figure 4: Determination of the number of lines - outer loop of solution procedure

To escape from this local optimum, a perturbation step is performed by changing a certain percentage of the current product to line assignment. This means that some products are randomly placed on a different line than they were on before. For this new assignment, a (first) optimal design is again determined and solving the assignment and design problems in an alternating manner is restarted. The entire procedure is repeated until a fixed number of perturbations without an improvement of the best solution is reached. In that case, there is a return to the outer loop (Fig. 4) and an increase of the number of lines.

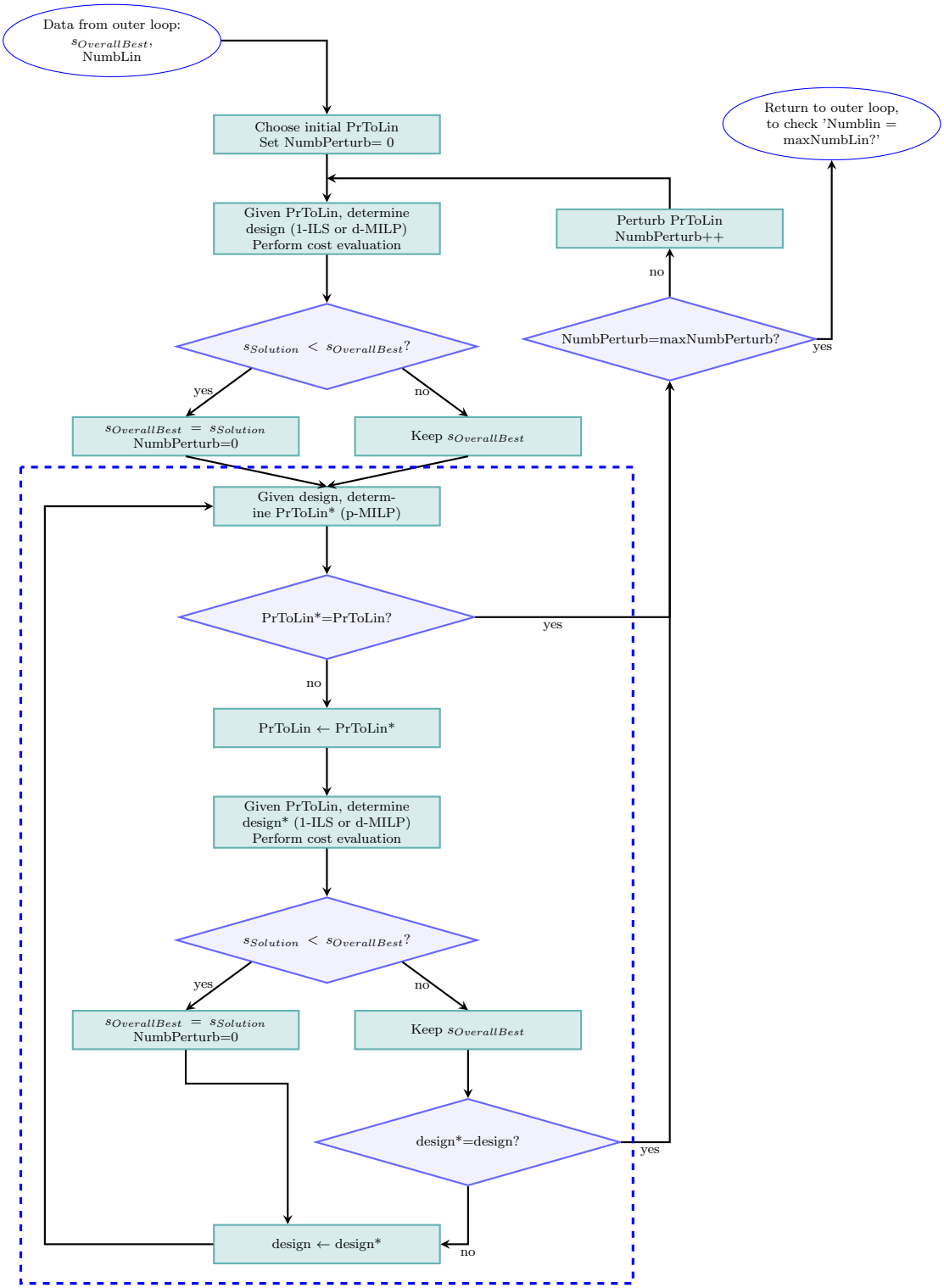

Figure 5: Comprehensive iterated local search procedure - iteratively solve design and assignment problem for a certain NumbLin

5.3.1. Product to line assignment

In case of multiple lines, the products have to be assigned to one or more lines. Two cases can be distinguished: the initial product to line assignment for an unknown design and subsequent product to line assignments for a given design.

Initial product to line assignment for an unknown design. The first product to line assignment is the starting point of the optimisation procedure for every line configuration. For this case, only the number of lines installed, the production data and the cost structure are known. Consequently, the initial assignment is determined based on a heuristic rule: products of the same product families are grouped together on lines as much as possible so as to minimise contamination costs.

There are three possible scenario's for the initial situation:

- number of lines $<$ number of product families: the different product families can not be entirely separated from each other. The product families are ranked based on their volumes and assigned to different lines in descending order of volume. The smallest families are produced together on the last remaining line.
- number of lines = number of product families: every family is assigned to its own line.
- number of lines $>$ number of product families: every product family is assigned to at least one line. The remaining lines are divided over the families according to descending order of volumes. The products belonging to product families that are dedicated to multiple lines are evenly spread over the lines.

For all cases, every product is assigned to a single line in order to minimise startup costs.

Product to line assignments for a given design (p-MILP). In this subproblem, the design is already determined and all products have to be optimally (re)allocated to the existing lines. This fixed design implies known variables e_l, u_{ljs} and z_{ljn} and is given by $\mathsf{Size}_{\mathsf{lj}}$ and $\mathsf{NumbPar}_{\mathsf{lj}}$ representing the size and number of equipment units at stage j of line l respectively. The aim is to determine the optimal product to line assignment t_{li} and, if applicable, the optimal distribution of production quantities q_{li}. This MILP, referred to as p-MILP, is solved using an exact solver:

$$\min \left[\sum_{l=1}^{L} \sum_{j=1}^{J} \left(\sum_{i=1}^{P} Cstart_i \, \mathsf{NumbPar}_{\mathsf{lj}} t_{li} + \sum_{f=1}^{F} Ccont \, b_{fl} \mathsf{NumbPar}_{\mathsf{lj}} \, g_l \right) \right]$$

s.t.

Design constraints:

$$n_{li} \geqslant \frac{q_{li} S_{ij}}{\mathsf{Size}_{\mathsf{lj}}} \qquad \forall l, j, i \tag{5.1}$$

Horizon constraints:

$$T_{lji} = \frac{n_{li} \tau_{ij}}{\mathsf{NumbPar}_{\mathsf{lj}}} \qquad \forall l, j, i \tag{5.2}$$

$$\theta_{li} \geqslant T_{lji} \qquad \forall l, j, i \tag{5.3}$$

$$\sum_{i=1}^{P} \theta_{li} \leqslant H \qquad \forall l \tag{5.4}$$

The demand constraint (Eq. (3.10)), the boundaries (Eqs. (3.11)-(3.15)) and contamination cost constraints (Eqs. (3.17)-(3.19)) remain unchanged in comparison to the original model. To avoid excessive running times for larger problems, the MILP solver is aborted when a certain time limit is exceeded and reports the best feasible solution so far.

5.3.2. Design calculation for a given product to line assignment

When there are multiple lines, two alternative techniques to calculate the design can be used: an exact calculation, referred to as d-MILP, and the previously described 1-ILS. The motivation to pick one or the other is the outcome of the product to line assignment subproblem (Section 5.3.1). If a product is assigned to multiple lines ($\sum_{l=1}^{L} t_{li} > 1 \quad \forall i$), the repartition of production over the lines becomes an additional continuous decision to make. In this case, an exact solver is called to optimise the MILP problem. This d-MILP problem is fairly similar to the original formulation described in Section 3, with the exception that t_{li} is given. In the other case, i.e. if every product is produced only on one line, the design problem, with only discrete decisions, can be solved for every line separately and thus, the 1-ILS metaheuristic is used.

5.3.3. Parameter tuning for comprehensive iterated local search

For the comprehensive ILS, 2 main parameters have to be tested as shown in Table 2: `noImprovement-math`, which has the same meaning as `noImprovement-ILS` described in Section 4.2, and `perturbationRate-math` that refers to the percentage of products that are assigned to another line to move away from a local optimum. A similar parameter tuning experiment was conducted and the results are shown in Table 2 and Figure 6. The final values of every parameter are summarised in the last column of Table 2. Although the parameter value for `noImprovement-math` that yields the lowest costs is 20 perturb-

ations, we have chosen to use this value only for small instances requiring a limited amount of computation time. In general, we selected 10 perturbations as, on average, the decrease in cost does not outweigh the increase in computation time.

Table 2: Parameters matheuristic

Parameter name	Possible values	Chosen value[a]
`noImprovement-math`	10, 15, 20	**10**
`perturbationRate-math`	20, 40, 60	**40**

[a] for every instance, every combination is tested 10 times

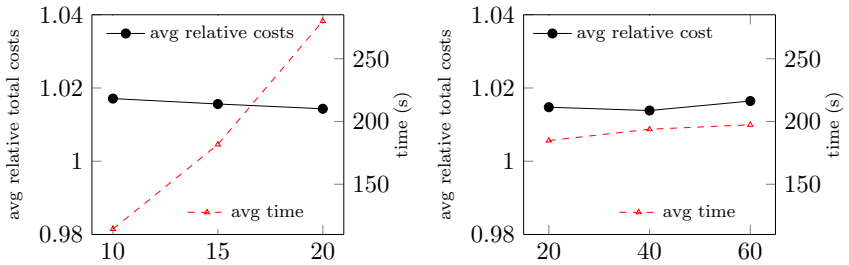

Figure 6: Full factorial experiment - matheuristic

6. Experimental results

In this section, the performance of both the 1-ILS and decomposition matheuristic is illustrated by solving several instances and comparing the obtained results to the solutions found with an exact method.

6.1. Input data

The different instances are presented in Table 3 and have the same structure: the demand for a number of products (Products), belonging to a certain number of product families (Product families), is to be produced within a fixed horizon. Therefore, a number of operations are required (Stages) for which the equipment sizes need to be chosen out of a set of different sizes (Sizes). Furthermore, due to plant surface area restrictions, a maximum number of units per stage (Max par. equip.) and a maximum number of parallel production lines (Max lines) can be installed. Finally, the remaining input data are size factors, processing times and cost structures.

All examples are analysed for 3 objective functions: the minimisation of (1) only capital costs; of (2) capital and startup costs and of (3) capital, startup and contamination

Table 3: Dimensions of the instances[a b]

	Products (P)	Product families (F)	Stages (J)	Sizes (S)	Max par.equip. (N)	Max lines (L)[c]
Instance 1	8	2	3	10	3	3
Instance 2	10	3	3	6	4	4
Instance 3	20	3	4	6	5	5
Instance 4	28	2	3	6	3	4
Instance 5	30	4	4	8	4	3
Instance 6	31	9	11	9	8	5
Instance 7	35	7	5	6	4	6
Instance 8	39	10	7	6	9	9
Instance 9	40	4	5	7	4	6
Instance 10	50	6	4	8	6	3
Instance 11	60	6	4	6	6	5
Instance 12	75	4	4	7	4	8
Instance 13	85	5	5	5	2	8
Instance 14	100	4	4	8	7	6
Instance 15	150	4	3	7	8	4
Instance 16	230	3	2	4	8	5
Instance 17	250	4	4	8	4	4

[a] Instances can be obtained upon request

[b] Total instance sizes correspond to:
variables: $L(4P + 1 + J(N(1 + S + P) + P(1 + S) + S)) + (LJPN)^* + (L(1 + F + 2JFN))^{**}$
constraints: $P + L(2 + 5P + J(3 + P(4 + 3S + 3N) + SN)) + (LJPN)^* + (L(2 + FP + 2JFN))^{**}$
(with $*$ included for objective function (2) and (3) and $**$ for objective function (3))

[c] for the validation of 1-ILS, Max lines is set to 1

costs. The problem is first solved for a single line with our 1-ILS (see Section 6.2) and, then, for multiple lines with our matheuristic (see Section 6.3). The obtained results are in both cases compared to an exact method, using the Gurobi Optimizer 7.0 with an optimality gap set to 0 %. All calculations are done on an Intel(R) Core(TM) i7-4790 CPU, 3.60GHz and 16 GB of RAM under a windows operating system.

6.2. Iterated local search for the single line design problem (1-ILS)

To demonstrate the efficiency of our 1-ILS, a single line design problem is solved for the instances presented in Table 3. For this case, the maximum number of lines to install is limited to 1. Hence, the last column of Table 3 is omitted. In Table 4, the results of 1-ILS are presented and compared with the exact solutions. For every instance, the best 1-ILS solution found over 10 runs is reported, as well as the corresponding gap with the exact solution. Moreover, also the gap between the exact solution and the average solution over 10 runs is given. Regarding computation times, both the total computation time with Gurobi as well as the average time of the 1-ILS runs are presented. Finally, the percentage of the 10 solutions that lie in a 2 % range of the best metaheuristic solution are reported to give an indication of the variation in results.

From the results in Table 4, it can be concluded that the 1-ILS algorithm finds the optimal solution for every example, except for the minimisation of capital and startup costs of instance 14 for which there is a small gap of 0.09 %. Moreover, the small gaps between the average 1-ILS and exact solution, and the high percentages in the 2 % dispersion-column, prove that the variations found in 1-ILS solutions are fairly low. Furthermore, from the moment the problems become larger, the 1-ILS algorithm is significantly faster.

6.3. Decomposition matheuristic for the multiple lines design problem

After the validation of the 1-ILS algorithm, the entire decomposition matheuristic, including the comprehensive ILS procedure, is tested.

6.3.1. Results of the multiple lines design problem

In Table 5, the results for multiple lines are presented and compared with the exact solutions. Every instance was again solved for the three different objective functions (Ca., Ca.St. and Ca.St.Co.). The same optimality gap of 0 % and a time limit of 12 hours (43 200 s) was imposed on the exact solution procedure. If the optimal solution was not found with Gurobi within this time limit, the best feasible solution calculated so far is reported.

The results indicate that our proposed matheuristic provides very good solutions in significantly lower computation times. For the instances that Gurobi could solve to optimality (instances 1-5 and 9-11), our matheuristic found the optimal solutions as well. Furthermore, it can be observed that for these 8 instances the gaps between the average matheuristic solution and optimal solution are always smaller than 5 % indicating a low variation in solutions. Note that, for the smallest instances (instance 1-5), a stopping criterion of 20 perturbations was allowed, as specified in the parameter tuning experiment. For instances 6, 7, 8 and part of 17 (for Ca.St. and Ca.St.Co) the exact solver could not reach optimality in 43 200 s, but the solutions are of reasonable quality (optimality gaps $\leqslant$ 45 %) to evaluate the performance of the matheuristic. In these cases as well, the matheuristic is significantly faster and provides in most cases, except for instance 7, better solutions. Finally, for instances 12 to 16 and 17 (Ca.) solutions with very large gaps or no feasible solutions are found with the exact solver within the given time limit. Even though the quality of the solutions cannot be evaluated, the matheuristic finds feasible solutions in still acceptable amounts of time whereas the limits of the exact solution technique are reached.

Table 4: Results 1-ILS for 1 line instances[a]

	Exact sol. (m.u.[b]) (Gurobi)	time (s) (Gurobi)	Opt. gap (%)	Best 1-ILS sol.[c] (m.u.)	gap (%) Best 1-ILS vs exact	gap (%) Avg. 1-ILS vs exact	avg time (s) (1-ILS)	2% disp. (%)
Instance 1: 8 products								
Ca.[d]	250 990	0.04	0	250 990	**0**	0	0.00	100
Ca.St.	379 875	0.10	0	379 875	**0**	0	0.00	100
Ca.St.Co.	449 875	0.07	0	449 875	**0**	0	0.00	100
Instance 2: 10 products								
Ca.	130 182	0.07	0	130 182	**0**	0	0.00	100
Ca.St.	210 182	0.07	0	210 182	**0**	0	0.00	100
Ca.St.Co.	306 182	0.07	0	306 182	**0**	0	0.00	100
Instance 3: 20 products								
Ca.	79 530	0.50	0	79 530	**0**	0	0.00	100
Ca.St.	159 530	0.32	0	159 530	**0**	0	0.00	100
Ca.St.Co.	177 530	0.32	0	177 530	**0**	0	0.01	100
Instance 4: 28 products								
Ca.	115 065	1.14	0	115 065	**0**	0	0.01	100
Ca.St.	173 035	1.27	0	173 035	**0**	0.7	0.00	90
Ca.St.Co.	205 035	1.26	0	205 035	**0**	0.9	0.00	90
Instance 5: 30 products								
Ca.	73 458	0.67	0	73 458	**0**	0	0.01	100
Ca.St.	133 458	0.77	0	133 458	**0**	0	0.01	100
Ca.St.Co.	149 458	0.77	0	149 458	**0**	0	0.01	100
Instance 6: 31 products								
Ca.	1 754 762	3267.37	0	1 754 762	**0**	3.7	0.27	50
Ca.St.	3 153 692	1028.57	0	3 153 692	**0**	0.6	0.28	80
Ca.St.Co.	3 801 692	729.39	0	3 801 692	**0**	0.2	0.25	90
Instance 7: 35 products								
Ca.	119 531	0.71	0	119 531	**0**	0.5	0.03	80
Ca.St.	382 031	0.59	0	382 031	**0**	0	0.02	100
Ca.St.Co.	473 031	0.60	0	473 031	**0**	0	0.01	100
Instance 8: 39 products								
Ca.								
Ca.St.	infeasible to produce all products on a single line within the horizon							
Ca.St.Co.								
Instance 9: 40 products								
Ca.	96 517	7.02	0	96 517	**0**	0	0.02	100
Ca.St.	996 517	1.42	0	996 517	**0**	0	0.03	100
Ca.St.Co.	1 016 517	1.78	0	1 016 517	**0**	0	0.01	100
Instance 10: 50 products								
Ca.	432 133	0.90	0	432 133	**0**	0	0.03	100
Ca.St.	882 133	0.87	0	882 133	**0**	0	0.03	100
Ca.St.Co.	918 133	0.90	0	918 133	**0**	0	0.03	100
Instance 11: 60 products								
Ca.	6958	3.38	0	6958	**0**	0	0.03	100
Ca.St.	24 958	2.97	0	24 958	**0**	0	0.04	100
Ca.St.Co.	26 158	3.03	0	26 158	**0**	0	0.03	100
Instance 12: 75 products								
Ca.								
Ca.St.	infeasible to produce all products on a single line within the horizon							
Ca.St.Co.								
Instance 13: 85 products								
Ca.								
Ca.St.	infeasible to produce all products on a single line within the horizon							
Ca.St.Co.								
Instance 14: 100 products								
Ca.	4 809 709	116.52	0	4 809 709	**0**	0.06	0.07	100
Ca.St.	7 077 709	110.94	0	7 084 350	0.09	0.2	0.07	100
Ca.St.Co.	7 256 269	133.34	0	7 256 269	**0**	0.05	0.06	100
Instance 15: 150 products								
Ca.	243 888	334.59	0	243 887	**0**	0	0.01	100
Ca.St.	2 646 474	220.36	0	2 646 474	**0**	0	0.02	100
Ca.St.Co.	3 094 474	209.06	0	3 094 474	**0**	0	0.02	100
Instance 16: 230 products								
Ca.								
Ca.St.	infeasible to produce all products on a single line within the horizon							
Ca.St.Co.								
Instance 17: 250 products								
Ca.	1 558 267	2441.30	0	1 558 267	**0**	0	0.06	100
Ca.St.	3 045 767	4383.17	0	3 045 767	**0**	0	0.06	100
Ca.St.Co.	3 129 767	4849.02	0	3 129 767	**0**	0	0.06	100

[a] All calculations are done on an Intel(R) Core(TM) i7-4790 CPU, 3.60GHz and 16 GB of RAM

[b] monetary units

[c] out of 10 runs

[d] minimisation of capital (Ca.), capital and startup (Ca.St.) and capital, startup and contamination (Ca.St.Co.) costs

Table 5: Results matheuristic procedure for multiple lines instances[a]

	Exact sol. (m.u.[b]) (Gurobi)	time (s) (Gurobi)	Opt. gap (%)	Best math. sol.[c] (m.u.)	gap (%) Best math. vs exact	gap (%) Avg. math. vs exact	avg time (s) (math)	2% disp. (%)
Instance 1: 8 products								
Ca.[d]	249 035	56.16	0	249 035[e]	**0**	0	60.75	100
Ca.St.	326 639	194.85	0	326 639[e]	**0**	2.3	2.36	60
Ca.St.Co.	360 326	162.47	0	360 326[e]	**0**	0	4.57	100
Instance 2: 10 products								
Ca.	128 808	1645.29	0	128 808[e]	**0**	0	80.80	100
Ca.St.	166 947	8271.63	0	166 947[e]	**0**	0.6	7.22	100
Ca.St.Co.	168 588	401.33	0	168 588[e]	**0**	1.3	13.06	90
Instance 3: 20 products								
Ca.	79 530	83.34	0	79 530[e]	**0**	0	0.02	100
Ca.St.	159 530	98.01	0	159 530[e]	**0**	0	0.01	100
Ca.St.Co.	177 530	23.00	0	177 530[e]	**0**	0	0.01	100
Instance 4: 28 products								
Ca.	115 065	213.74	0	115 065[e]	**0**	1.8	133.68	90
Ca.St.	173 035	661.47	0	173 035[e]	**0**	4.0	7.77	40
Ca.St.Co.	190 325	1210.90	0	190 325[e]	**0**	0	14.31	100
Instance 5: 30 products								
Ca.	73 458	27.78	0	73 458[e]	**0**	0	0.03	100
Ca.St.	133 458	27.84	0	133 458[e]	**0**	0	0.03	100
Ca.St.Co.	149 458	9.36	0	149 458[e]	**0**	0	0.03	100
Instance 6: 31 products								
Ca.	1 722 850	43 200[f]	42.7	1 716 443	-	-	9244.30	100
Ca.St.	2 777 785	43 200[f]	28.1	2 757 798	-	-	60.51	100
Ca.St.Co.	3 269 931	43 200[f]	35.2	3 251 535	-	-	143.04	100
Instance 7: 35 products								
Ca.	116 708	43 200[f]	14	117 269	0.5	0.9	373.02	100
Ca.St.	249 266	43 200[f]	10	249 304	0.02	0.2	4.40	100
Ca.St.Co.	296 184	43 200[f]	8.2	295 377	-	-	4.80	100
Instance 8: 39 products								
Ca.	447 797	43 200[f]	34.7	444 860	-	-	0.14	100
Ca.St.	1 805 202	43 200[f]	12	1 805 202	-	-	9.30	100
Ca.St.Co.	1 976 702	43 200[f]	19.7	1 976 702	-	-	9.68	100
Instance 9: 40 products								
Ca.	96 517	3551.62	0	96 517	**0**	0.4	374.89	90
Ca.St.	996 517	4892.11	0	996 517	**0**	1.3	3.8	70
Ca.St.Co.	1 016 517	5282.02	0	1 016 517	**0**	0	3.0	100
Instance 10: 50 products								
Ca.	432 133	516.08	0	432 133	**0**	0	0.10	100
Ca.St.	797 696	313.15	0	797 696	**0**	0	8.00	100
Ca.St.Co.	826 496	912.57	0	826 496	**0**	0	7.13	100
Instance 11: 60 products								
Ca.	6958	753.87	0	6958	**0**	0	0.11	100
Ca.St.	24 958	819.74	0	24 958	**0**	0	0.08	100
Ca.St.Co.	26 158	1543.72	0	26 158	**0**	0	0.08	100
Instance 12: 75 products								
Ca.	1 847 201	43 200[f]	82.4	1 753 517	-	-	17 192.67	100
Ca.St.	no feas sol[g] (LB: 375 549)	43 200[f]	-	1 953 779	-	-	3627.13	20
Ca.St.Co.	no feas sol[g] (LB: 438 813)	43 200[f]	-	2 003 488	-	-	19 554.48	70
Instance 13: 85 products								
Ca.	no feas sol[g] (LB:776 379)	43 200[f]	-	5 924 242	-	-	23 145.1	50
Ca.St.	no feas sol[g] (LB:957 995)	43 200[f]	-	6 718 251	-	-	7502.77	50
Ca.St.Co.	no feas sol[g] (LB:993 097)	43 200[f]	-	7 277 622	-	-	2373.11	80
Instance 14: 100 products								
Ca.	4 502 842	43 200[f]	67.3	4 455 575	-	-	6551.82	70
Ca.St.	6 482 296	43 200[f]	69.9	5 429 498	-	-	119.69	100
Ca.St.Co.	6 697 432	43 200[f]	66.8	5 573 375	-	-	509.28	100
Instance 15: 150 products								
Ca.	303 135	43 200[f]	87.9	229 485	-	-	11 957.23	60
Ca.St.	20 932 033	43 200[f]	97.7	821 012	-	-	560.27	50
Ca.St.Co.	2 916 612	43 200[f]	83.3	997 950	-	-	98.87	20
Instance 16: 230 products								
Ca.	no feas sol[g] (LB:558 817)	43 200[f]	-	6 920 729	-	-	14 321,74	100
Ca.St.	no feas sol[g] (LB:726 512)	43 200[f]	-	7 520 992	-	-	837.72	70
Ca.St.Co.	no feas sol[g] (LB:726 511)	43 200[f]	-	7 986 812	-	-	1581.69	30
Instance 17: 250 products								
Ca.	1 718 307	43 200[f]	62.8	1 558 267	-	-	11 983.6	100
Ca.St.	2 569 989	43 200[f]	45.0	2 557 661	-	-	63.5	100
Ca.St.Co.	2 698 969	43 200[f]	42.9	2 688 721	-	-	567.40	100

[a] All calculations are done on an Intel(R) Core(TM) i7-4790 CPU, 3.60GHz and 16 GB of RAM

[b] monetary units

[c] out of 10 runs

[d] min.of capital(Ca.), capital and startup(Ca.St.), capital, startup and contamination(Ca.St.Co.) costs

[e] 20 perturbations as stopping criterion

[f] time limit of 12 h

[g] no feasible exact solution was found within the allowed time of 12 h

6.3.2. Progress of the decomposition matheuristic

In this section the progress of the decomposition matheuristic is visualised for one example (Instance 2) for the minimisation of capital, startup and contamination costs. In Figure 7, the total cost of every intermediate solution found during the execution of the matheuristic is shown at the corresponding time point. Which component of the matheuristic that provides every intermediate solution is illustrated via the different (coloured) symbols. The blue filled squares and triangles represent the designs found after the initial product to line assignments for every new line configuration, or after leaving a local optimum by perturbing the products over lines. The squares refer to solving the design problem with 1-ILS, whereas the triangles visualise the d-MILP solutions. The red dots refer to the subsequent optimal product to line assignments for these given designs, found when solving p-MILP. The following (green) solutions are the optimal designs for these new product assignments. The green triangles represent the solutions found when solving d-MILP, whereas the green squares represent the 1-ILS solutions. Finally, every increment of the number of lines is indicated with an L-mark.

Figure 7 shows that, for Instance 2, the design subproblem is solved with both the exact d-MILP (green triangles) and the 1-ILS metaheuristic (green squares). It can be seen that these 1-ILS optimisations take (on average) a shorter computation time than employing d-MILP, as the horizontal distances between the previous assignment problem p-MILP (red dots) and the 1-ILS design problem (green squares) are smaller than for the d-MILP design problem (green triangles). The same holds for the distances between the (blue) perturbed design solutions (with 1-ILS or d-MILP) and the preceding ones. Moreover, the optimal solution, for this instance, is the design found for the initial product to line assignment when 3 lines are considered. Indeed, all products are grouped into product families and assigned as much as possible to a single line. For instance 2 with 3 product families and 3 lines, every family is dedicated to one separate line, which appears to be sufficient.

When comparing all three objective functions (Figures 7-9), it can be seen that the pattern in Figure 8 is less fluctuating in comparison to the other two cases, due to the absence of the startup and contamination costs. Indeed, after perturbing the solutions, often much worse solutions are found in Figures 7 and 9 since products are randomly grouped. Moreover, it is clear that the minimisation of only capital costs (Fig. 8) requires significantly more time than the other two optimisations. Indeed, for capital cost

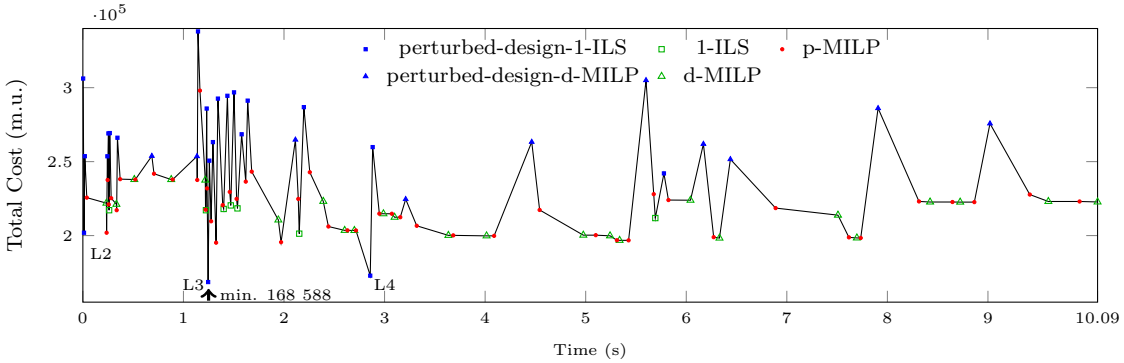

Figure 7: Progress of decomposition matheuristic for capital, startup and cont. costs (best sol.: 168 588)

minimisation only, it appears that it is always better to duplicate products over multiple lines so that the design problem is every time solved with the exact method (d-MILP), except for the initial designs for a new line configuration. The minimisation of capital and startup costs (Fig. 9), on the other hand, employs always 1-ILS for the design optimisation and is overall the fastest, confirming again the contribution of the 1-ILS algorithm to the efficiency of the matheuristic.

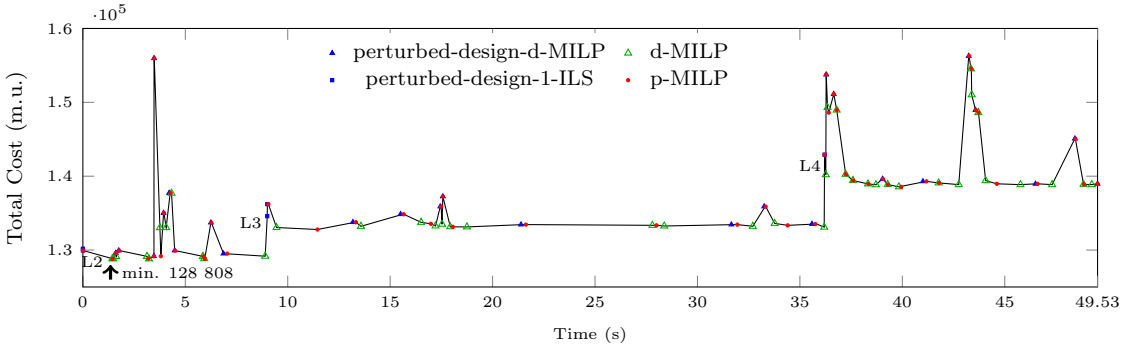

Figure 8: Progress of decomposition matheuristic for capital costs (best sol.: 128 808)

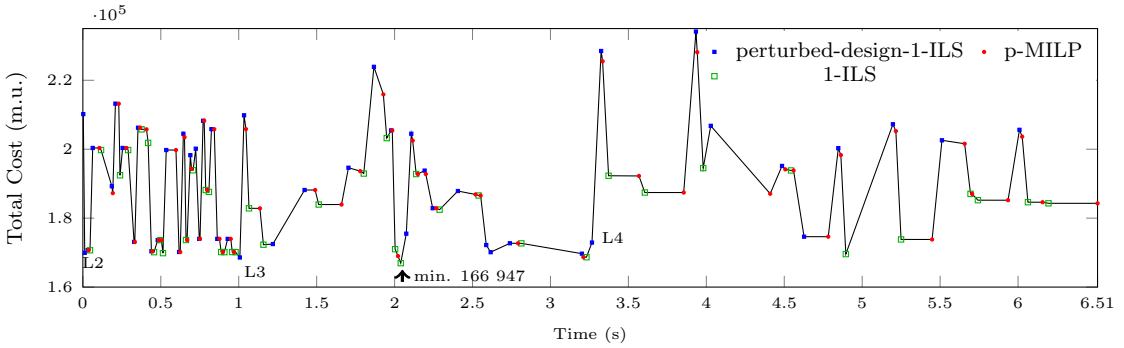

Figure 9: Progress of decomposition matheuristic for capital and startup costs (best sol.: 166 947)

7. Conclusion

A hybrid metaheuristic has been presented to address the problem of multiproduct batch plant design with parallel production lines. The decisions to optimise in this problem are 3-fold: the number of lines to install, their design and the product to line assignments. As this problem becomes intractable for exact solution methods, a decomposition matheuristic is implemented that consists of the following steps: At first, a single line design problem is solved with an iterated local search metaheuristic (1-ILS). Then, the number of lines are determined in an incremental manner in the outer loop followed by a comprehensive iterated local search procedure to find the optimal design and product to line assignments for every given number of lines. Within this comprehensive iterated local search, a combination of exact MILP calculations (p-MILP and d-MILP) and the 1-ILS is applied. This 1-ILS metaheuristic appeared to account to a great extent to the efficiency of the entire solution procedure and is adopted to solve the discrete design decisions for every line (when products are assigned to one line only). Several instances are solved both exactly and with our 1-ILS to demonstrate the applicability of the metaheuristic. Finally, the performance of the decomposition matheuristic is evaluated through, again, the comparison with an exact solution approach. From the multiple instances solved, it can be concluded that our proposed hybrid solution method provides very good results to this specific problem in a considerable lower computation time. In future research, we aim to employ this matheuristic to solve extended design problems. Such extensions include e.g. more explicit scheduling together with optimising the plant design.

Acknowledgements

This work was supported by the Research Foundation - Flanders (FWO).

References

Aguilar-Lasserre, A. A., Pibouleau, L., Azzaro-Pantel, C., & Domenech, S. (2009). Enhanced genetic algorithm-based fuzzy multiobjective strategy to multiproduct batch plant design. *Applied Soft Computing*, *9*, 1321–1330.

Applequist, G., Samikoglu, O., Pekny, J., & Reklaitis, G. (1997). Issues in the use, design and evolution of process scheduling and planning systems. *ISA Transactions*, *36*, 81–121.

Ball, M. O. (2011). Heuristics based on mathematical programming. *Surveys in Operations Research and Management Science*, *16*, 21–38.

Barbosa-Póvoa, A. P. (2007). A critical review on the design and retrofit of batch plants. *Computers & Chemical Engineering*, *31*, 833–855.

Blum, C., Puchinger, J., Raidl, G. R., & Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, *11*, 4135–4151.

Boschetti, M. A., Maniezzo, V., Roffilli, M., & Bolufé Röhler, A. (2009). Matheuristics: Optimization, simulation and control. In M. J. Blesa, C. Blum, L. Di Gaspero, A. Roli, M. Sampels, & A. Schaerf (Eds.), *Hybrid Metaheuristics: 6th International Workshop, Proceedings* (pp. 171–177). Berlin: Springer Berlin Heidelberg.

Cavin, L., Fischer, U., Glover, F., & Hungerbühler, K. (2004). Multi-objective process design in multi-purpose batch plants using a tabu search optimization algorithm. *Computers & Chemical Engineering*, *28*, 459–478.

Chibeles-Martins, N., Pinto-Varela, T., Barbosa-Póvoa, A. P., & Novais, A. (2010). A meta-heuristics approach for the design and scheduling of multipurpose batch plants. *Computer Aided Chemical Engineering*, *28*, 1315–1320.

Corsano, G., Montagna, J. M., Iribarren, O. A., & Aguirre, P. A. (2007). Heuristic method for the optimal synthesis and design of batch plants considering mixed product campaigns. *Industrial & Engineering Chemistry Research*, *46*, 2769–2780.

Dedieu, S., Pibouleau, L., Azzaro-Pantel, C., & Domenech, S. (2003). Design and retrofit of multiobjective batch plants via a multicriteria genetic algorithm. *Computers & Chemical Engineering*, *27*, 1723–1740.

Dietz, A., Azzaro-Pantel, C., Pibouleau, L., & Domenech, S. (2006). Multiobjective optimization for multiproduct batch plant design under economic and environmental considerations. *Computers & Chemical Engineering*, *30*, 599–613.

Glover, F. (1975). Improved linear integer programming formulations of nonlinear integer problems. *Management Science*, *22*, 455–460.

Hill, A., Cornelissens, T., & Sörensen, K. (2016). Efficient multi-product multi-bom batch scheduling for a petrochemical blending plant with a shared pipeline network. *Computers & Chemical Engineering*, *84*, 493–506.

Jayaraman, V. K., Kulkarni, B. D., Karale, S., & Shelokar, P. (2000). Ant colony framework for optimal design and scheduling of batch plants. *Computers & Chemical Engineering*, *24*, 1901–1912.

Jelen, F. C., Black, J. H., & of Cost Engineers, A. A. (1983). *Cost and optimization engineering*. McGraw-Hill.

Jourdan, L., Basseur, M., & Talbi, E. G. (2009). Hybridizing exact methods and metaheuristics: A taxonomy. *European Journal of Operational Research*, *199*, 620–629.

Lourenço, H. R., Martin, O. C., & Stützle, T. (2010). Iterated local search: Framework and applications. In M. Gendreau, & J.-Y. Potvin (Eds.), *Handbook of Metaheuristics* (pp. 363–397). Springer US.

Modi, A. K., & Karimi, I. A. (1989). Design of multiproduct batch processes with finite intermediate storage. *Computers & Chemical Engineering*, *13*, 127–139.

Öncan, T. (2015). Milp formulations and an iterated local search algorithm with tabu thresholding for the order batching problem. *European Journal of Operational Research*, *243*, 142–155.

Patel, A. N., Mah, R. S. H., & Karimi, I. A. (1991). Preliminary design of multiproduct noncontinuous plants using simulated annealing. *Computers & Chemical Engineering*, *15*, 451–469.

Ponsich, A., Azzaro-Pantel, C., Domenech, S., & Pibouleau, L. (2008). Constraint handling strategies in genetic algorithms application to optimal batch plant design. *Chemical Engineering and Processing: Process Intensification*, *47*, 420–434. 10th French Congress on Chemical Engineering.

Raidl, G. R. (2015). Decomposition based hybrid metaheuristics. *European Journal of Operational Research*, *244*, 66–76.

Sörensen, K., & Glover, F. W. (2013). Metaheuristics. In S. I. Gass, & M. C. Fu (Eds.), *Encyclopedia of Operations Research and Management Science* (pp. 960–970). Boston, MA: Springer US.

Sparrow, R. E., Forder, G. J., & Rippin, D. W. T. (1975). The choice of equipment sizes for multiproduct batch plants. heuristics vs. branch and bound. *Industrial & Engineering Chemistry Process Design and Development*, *14*, 197–203.

Stützle, T. (2006). Iterated local search for the quadratic assignment problem. *European Journal of Operational Research*, *174*, 1519–1539.

Tan, S., & Mah, R. S. (1998). Evolutionary design of noncontinuous plants. *Computers & Chemical Engineering*, *22*, 69–85.

Technavio (2016). *Global specialty chemicals market 2016-2020*. Technical Report. URL: https://www.technavio.com/report/global-specialty-chemicals-global-specialty-chemicals-market-2016-2020.

Verbiest, F., Cornelissens, T., & Springael, J. (2017). Design of a chemical batch plant with parallel production lines: Plant configuration and cost effectiveness. *Computers & Chemical Engineering*, *99*, 21–30.

Voudouris, V. T., & Grossmann, I. E. (1992). Mixed-integer linear programming reformulations for batch process design with discrete equipment sizes. *Industrial &*

Engineering Chemistry Research, *31*, 1315–1325.

Wang, C., Quan, H., & Xu, X. (1996). Optimal design of multiproduct batch chemical process using genetic algorithms. *Industrial & Engineering Chemistry Research*, *35*, 3560–3566.

Wang, C., Quan, H., & Xu, X. (1999). Optimal design of multiproduct batch chemical processes using tabu search. *Computers & Chemical Engineering*, *23*, 427–437.

Xi-Gang, Y., & Zhong-Zhou, C. (1997). A hybrid global optimization method for design of batch chemical processes. *Computers & Chemical Engineering*, *21*, S685–S690. Suppl. to Computers & Chemical Engineering.

Appendices

Appendix A. Nomenclature

Table A.1: Nomenclature

Indices			
i	products (P)	j	stages (J)
l	lines (L)	s	discrete sizes (S)
n	number of equipment in parallel (N)	f	product families (F)
Parameters			
S_{ij}	size factor of product i in stage j	S_i^{max}	largest size factor of product i ($\max_{j=1,\ldots,J} S_{ij}$)
τ_{ij}	processing time of product i in stage j	Q_i	total amount to produce of product i
H	horizon, total available production time	v_s	tank size s
N	maximum number of parallel equipment per stage	M_i	upper bound on number of batches of product i
α_j, β_j	cost coefficients of stage j	d_{if}	boolean matrix indicating if product i belongs
c_{js}	capital costs of stage j with size s ($\alpha_j v_s^{\beta_j}$)		to family f
$Cstart_i$	startup costs of product i	$Ccont$	fixed contamination cost between product families
Variables			
Continuous:			
q_{li}	amount produced of product i on line l	n_{li}	number of batches of product i on line l
T_{lji}	total time spent on product i on stage j of line l	θ_{li}	total time spent on product i on line l
Binary:			
e_l	if line l exists	z_{ljn}	if unit n of stage j of line l exists
t_{li}	if product i is produced on line l	u_{ljs}	if stage j of line l has units of size s
b_{fl}	if product family f is produced on line l	g_l	if there is more than 1 product family on line l

Appendix B. Mathematical formulation: linearisation expressions