Optimal synthesis of storageless batch plants using the Process Intermediate Storage Operational policy

Thomas Pattinson

Optimal synthesis of storageless batch plants using the Process Intermediate Storage Operational policy

by

Thomas Pattinson

A dissertation submitted in partial fulfillment

of the requirements for the degree

MEng: Chemical Engineering

in the

Faculty of Engineering, the Built Environment and Information Technology

University of Pretoria

Pretoria

Supervised by Prof. Thokozani Majozi

8th December 2007

Synopsis

A novel operational policy, the Process Intermediate Storage (PIS) operational policy, is introduced and used to synthesize, schedule and design multipurpose batch plants. The model is based on the State Sequence Network (SSN) and non-uniform discretization of the time horizon of interest model developed by Majozi & Zhu (2001). Two cases are studied to determine the effectiveness of the operational policy. A plant without dedicated intermediate storage is considered in the first case. In this case the throughput is maximized with and without the use of the PIS operational policy. The use of the PIS operational policy results in a 50% improvement in the throughout. The second case is used to determine the minimum amount of intermediate storage while maintaining the throughput achieved with infinite intermediate storage. This resulted in a 33% reduction in the amount of dedicated intermediate storage. The models developed for both cases are MILP models. A design model is then developed to exploit the attributes of the PIS operational policy. The design model is a MINLP due to the capital cost objective function. This model is applied to a literature example and an industrial case study and in both cases results in improved flowsheets and reduced capital cost.

Acknowledgments

I would like to acknowledge the South African Water Research Commission for their financial contribution under project K5/1625. I would also like to acknowledge contribution of Prof. Thokozani Majozi.

Declaration

UNIVERSITY OF PRETORIA

FACULTY OF ENGINEERING, BUILT ENVIRONMENT AND INFORMATION TECHNOLOGY

DEPARTMENT OF CHEMICAL ENGINEERING

The Department of CHEMICAL ENGINEERING places great emphasis upon integrity and ethical conduct in the preparation of all written work submitted for academic evaluation.

While academic staff teach you about systems of referring and how to avoid plagiarism, you too have a responsibility in this regard. If you are at any stage uncertain as to what is required, you should speak to your lecturer before any written work is submitted.

You are guilty of plagiarism if you copy something from a book, article or website without acknowledging the source and pass it off as your own. In effect you are stealing something that belongs to someone else. This is not only the case when you copy work word-by-word (verbatim), but also when you submit someone else's work in a slightly altered form (paraphrase) or use a line of argument without acknowledging it. You are not allowed to use another student's past written work. You are also not allowed to let anybody copy your work with the intention of passing if of as his/her work.

Students who commit plagiarism will lose all credits obtained in the plagarised work. The matter may also be referred to the Disciplinary Committee (Students) for a ruling. Plagiarism is regarded as a serious contravention of the University's rules and can lead to expulsion from the University.

The declaration which follows must be appended to all written work submitted while you are a student of the Department of CHEMICAL ENGINEERING. No written work will be accepted unless the declaration has been completed and attached.

I (full names): Thomas Pattinson

Student number: 22213440

Topic of work: Batch process scheduling

Declaration

1. I understand what plagiarism is and am aware of the University's policy in this regard.
2. I declare that the material handed in (e.g. essay, report, project, assignment, dissertation, thesis, computer programme, etc) is my own original work. Where other people's work has been used (either from a printed source, internet or any other source), this has been properly acknowledged and referenced in accordance with departmental requirements.
3. I have not used another student's past written work to hand in as my own.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as his or her own work.

Signature _____

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

1.1 Background

Batch plants are used for the production of low volume high value-added products such as fine chemicals, agrochemicals and pharmaceuticals. These plants are inherently flexible because, in most cases, the same unit can be utilized to manufacture many different products. This inherent flexibility is used to react to steep changes in market demand, thus gaining a competitive advantage in an ever-changing industry. As such these facilities are becoming more and more popular as opposed to continuous plants.

Batch processes differ from continuous processes in many ways, the main of which is that time is intrinsic in batch processes. In batch operations every task has a specific duration, with attendant starting and finishing times, whereas in continuous processes time is important during non-steady state operation. As a result of this, the scheduling of batch processes is vital to the operation of any batch facility. Although it is possible to do this by hand, in large complex facilities, the time and effort required is tremendous. There are mathematical methods to aid in the development of feasible schedules, however, to gain a truly competitive advantage the optimization of these schedules is imperative.

A large amount of research has gone into the development of optimization techniques for the scheduling of batch plants. This research has focused on reducing the time required to achieve the optimal solution through the development of mathematical models based on improved frameworks, such as the State Task Network (STN) (Kondili *et al.*, 1993), Resource Task Network (RTN) (Schilling & Pantelides, 1996) and the State Sequence Network (Majozi & Zhu, 2001). Furthermore, through improvements in the discretization of the time horizon and hybridization of methods such as the CP/MILP hybrids (Huang & Chen, 2005; Maravelias & Grossmann, 2004; Roe *et al.*, 2005; Harjunkoski & Grossmann, 2002; Jain & Grossmann, 2001). All of the abovementioned methods are based on mathematical programming, however, there is another class of methods, the graphical techniques.

Graphical techniques approach the problem in a different manner to their mathematical counterparts, by using graph theory to generate feasible schedules based on the recipe and then finding the optimal solution through searching algorithms, such as the S-graph approach (Sanmartí *et al.*, 1998). The approach in mathematical techniques is a so called 'black box' approach, where the user has no interaction in the solution of the model, while in graphical techniques the recipe is directly exploited as part of the solution algorithm.

Both mathematical and graphical methods have one thing in common, the final result is a schedule, whether it is simply feasible or optimal. In a typical batch operation units are idle and empty for large portions of the time horizon of interest, which provides a unique opportunity of using these units as storage, instead of, or in conjunction with dedicated intermediate storage. This would result in increased capital utilization of the equipment, and in the case of a design problem, this could result in a reduction in the size and capital cost of the facility. Mathematical and graphical techniques rely on operational policies to simulate operational conditions. In the context of batch plants there are six operational policies, i.e., Zero Wait (ZW), No Intermediate Storage (NIS), Finite Intermediate Storage (FIS), Unlimited Intermediate Storage (UIS), Mixed Intermediate Storage (MIS) and Central Intermediate Storage(CIS). Each operational policy corresponds to a particular operational philosophy. For example the ZW operational policy is used when intermediate

products are unstable. The NIS policy is used when there is no intermediate storage available. While the FIS and UIS operational policies are used when there is finite and infinite intermediate storage, respectively. In the case where intermediates are compatible the CIS operational policy is often used. The MIS operational policy is used when a combination of all the above mentioned operational are used. However, none of these policies takes into account the use of latent storage. To take the use of latent storage into account a novel operational policy, Process Intermediate Storage (PIS) operational policy, is introduced.

1.2 Aim

The aim of the project was to develop a mathematical model to take the inherent latent storage capacity into account when scheduling and designing a batch plant, in order to increase the capital utilization of a batch plant and to reduce the capital cost of constructing a new facility.

1.3 Approach

The techniques developed in this work are based on a mathematical programming and optimization approach. The work was approached in two steps. Firstly, the scheduling capabilities of the proposed model were explored, followed by the exploration of the design capabilities.

The project entailed three stages, i.e., the literature review, methodology development and the industrial application of the method. A more detailed description of each of these stages follows.

1.3.1 Literature review

Before the model was developed a thorough literature review was conducted. This review was done to ensure a comprehensive understanding of the current advances in the field of batch scheduling and also to identify possible opportunities to improve the current level of development in the field. The literature survey covered these advances from the early developments in 1975 to current research taking place in the field.

1.3.2 Methodology development

The approach used for the development of the methodology was to divide the problem into two subsections. Firstly, the scheduling effect on the use of latent storage was determined, wherein, the model was solved using an algorithmic approach to determine the minimum amount of intermediate storage required to achieve an optimal throughput. Secondly, the design opportunities were explored, where a model was developed to design storageless batch plants using the latent storage. Both models are presented in full detail in Chapter 3.

1.3.3 Application

To prove applicability of the developed models they were then applied to an industrial case study. The case study involved the design of a petrochemicals facility.

1.4 Thesis structure

Chapter 1 of the thesis introduces the nature of the problem at hand. Chapter 2 is the literature review, followed by the methodology developed in Chapter 3. Chapter 4 is the industrial application. The findings and further development of the work are discussed in Chapters 5 and 6.

CHAPTER 2

Literature Survey

2.1 The development of batch scheduling

The production of low-volume high-value-added products such as pharmaceuticals and agrochemicals is the premise of batch plants. This is mainly due to their inherent ability to adjust to steep changes in production and product type demands. Based on this inherent flexibility, batch plants can be divided into two classes, i.e., multiproduct and multipurpose batch plants. In multiproduct plants all products use essentially the same equipment and follow the same path through the plant. Whereas in multipurpose plants, there is no common path through the plant and in some cases successive batches of the same product can follow completely different paths and finish simultaneously. Therefore, multipurpose plants are not only more complex than multiproduct plants, but are also the superset of multiproduct plants. Accompanying the flexibility of batch plants, is inherent complexity that derives from the nature of the products.

There are six operational policies currently used in literature and in practice:

- Zero Wait (ZW),

- No Intermediate Storage (NIS),
- Finite Intermediate Storage (FIS),
- Unlimited Intermediate Storage (UIS),
- Mixed Intermediate Storage (MIS) (Weide Jr. & Reklaitis, 1987) and,
- Central Intermediate Storage(CIS) (Ku & Karimi, 1990).

In the ZW policy intermediate products cannot wait after they have been processed; so as soon as a batch has been processed in a unit it must be moved to the next step in its recipe. In general the resulting schedules have a stair-step appearance, as shown in Figure 2.1. The ZW policy is generally used for unstable products, where delays may have a detrimental effect on the product. The remaining operational policies are related to the nature of the intermediate storage. The NIS policy is used when there is no intermediate storage available, however, this does not imply that products cannot be stored in the process unit before processing or before moving to the next available processing unit. The FIS policy is more practical in nature, where there is an existing storage vessel of known capacity. However, this operational policy often assumes that there is a dedicated storage vessel for each intermediate product. The UIS policy is more of a theoretical operational policy, because the plant would have to be of infinite size to handle the unlimited capacity, however, this operational philosophy can be used at a design phase because it offers the highest degree of freedom of the previously mentioned policies. In practice it is more common to find these operational policies being used together in sections of the plant. To this end Weide Jr. & Reklaitis (1987) defined the MIS operational policy. In a situation where various intermediates are compatible, a CIS operational policy is recommended. When using the CIS operational policy there is a centralised intermediate storage unit which can be used by all products. The flowsheet in Figure 2.2 illustrates the MIS and CIS policies, where sections of a plant have different operational policies including the concept of shared storage.

Although these operational philosophies help with the operation of a batch plant, one

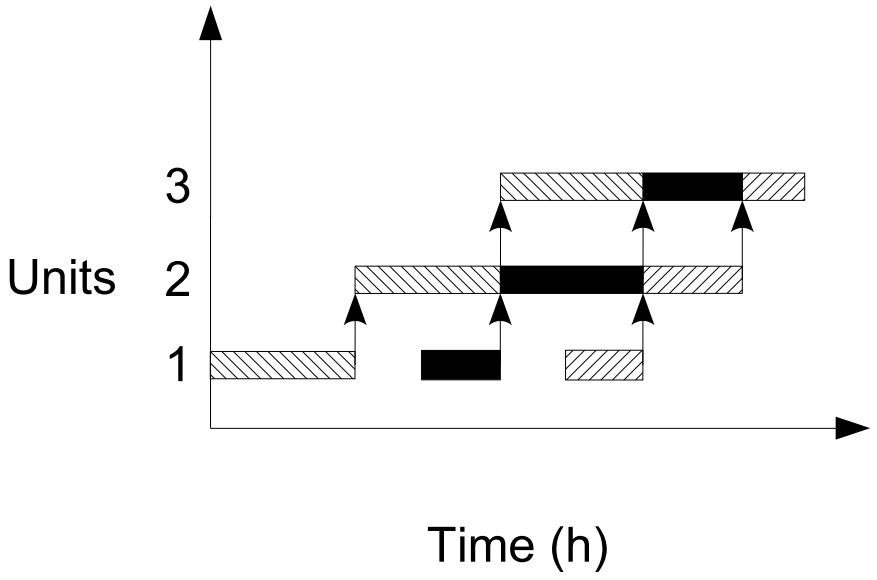

Figure 2.1: Stair-step nature of schedules with the ZW operational policy

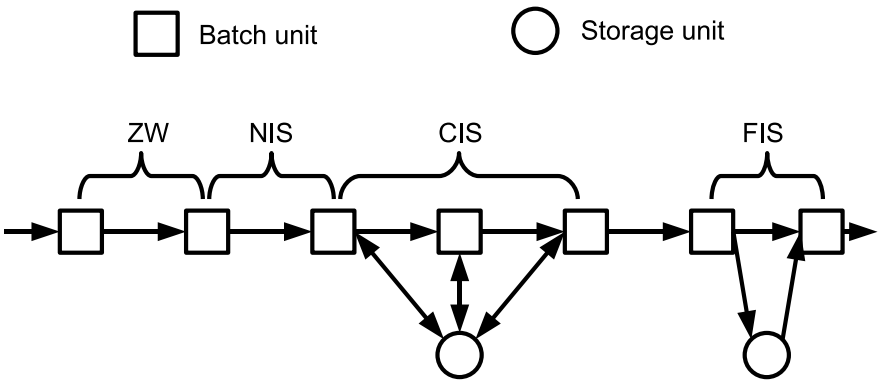

Figure 2.2: A serial process with the MIS and CIS operational policies

must understand that there is a major difference between continuous and batch plants which pertains to time. In continuous plants time is not a factor, however, in batch plants time is one of the most important variables. Due to the discrete nature of batch plants, scheduling of tasks is essential for their operation. The differences between batch and continuous processing is clearly shown in Figure 2.3. The discrete nature of batch processes can be clearly seen in Figure 2.3 a), where a task is processed in a unit and then once completed is transfered to the next unit for processing. Whereas, in a similar continuous process the discrete nature is only observed at startup and shutdown, as shown in Figure 2.3 b).

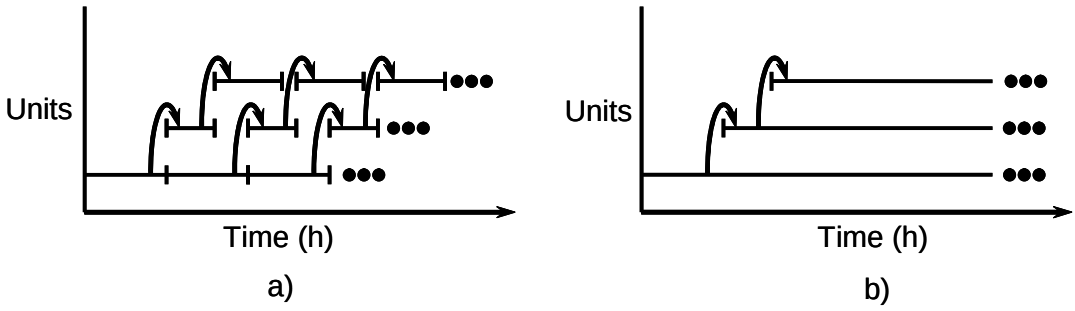

Figure 2.3: Batch vs. continuous processing

In its general form, the general scheduling problem entails determination of the optimal sequence of events using available resources. Formulation of this problem was initially proposed by Sparrow *et al.* (1975). They developed a mixed integer non–linear program (MINLP) formulation for multiproduct plants and introduced two solution strategies to solve the scheduling problem. The first strategy introduced the heuristic solution method, which is divided into three parts. Firstly, calculation of the exact equipment sizes, given the number of units in parallel is performed. Secondly, the exact sizes are converted to standard equipment sizes, and thirdly, the number of units which are in parallel at a given stage are chosen. The second solution strategy involved a deterministic branch and bound method.

It was concluded that the heuristic method was faster but might not lead to the optimal solution, whereas the branch and bound technique finds an optimum solution and is more flexible in that it allows constraints to be set, thus lowering computational time. As the

problem was an MINLP and there were no solution procedures to guarantee the global optimality, Grossmann & Sargent (1979), posed this problem as a geometric program and proved that the solution is global using the Karush-Kuhn-Tucker conditions. They also proved that the problem could be solved as a relaxed subprogram by disregarding the discreteness of equipment sizes. Ravemark & Rippin (1998) applied the same formulation as Sparrow *et al.* (1975) but used logarithmic transformations to ensure convexity of the MINLP.

Heuristic methods proposed by Sparrow *et al.* (1975) were used by Suhami & Mah (1982) to solve a multipurpose batch plant formulation. The drawback of heuristic methods is that they cannot guarantee global optimality. This is due to the fact that they are based on "rules of thumb", which are derived from experience.

The above formulations were all based on recipe networks. These networks are derived for continuous plants (i.e. flowsheets), but in batch plants this can lead to ambiguity. This led Kondili *et al.* (1993) to develop the State Task Network (STN) representation. The STN has two types of nodes; namely, state nodes, which represent feeds, intermediate and final products and task nodes, representing processing operations that transform material from input states to output states. Rectangular blocks and circles represent task and state nodes, respectively. Based on the STN a discrete time mixed integer linear program (MILP) formulation was developed. The resulting time intervals coincided with the beginning and end of a specific event, as shown in Figure 2.4. The problem with this formulation, however, is that the discretization of time with the attendant accuracy concerns, results in the creation of a large number of binary variables. In general the more binary variables a problem has, the more the computational effort required to find a solution. These computational issues were tackled in the second paper of the series (Shah *et al.*, 1993), where they developed methods to reduce the number of binary variables. However, due to the inherent large number of binary variables required, suboptimal results were still achieved. The large number of binary variables was in part due to the restriction placed on the time horizon. In order to alleviate this restriction the continuous time formulation was developed.

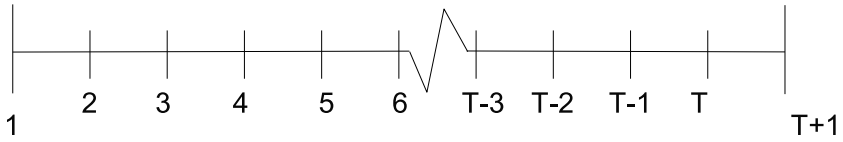

Figure 2.4: Formulations based on the discrete time framework require uniform discretization of the time horizon

Schilling & Pantelides (1996) developed a continuous time MINLP formulation where time slots of unknown length were used. The beginning and end of time slots coincided with the beginning and end of a task, as shown in Figure 2.5. The formulation was derived from the Resource Task Network (RTN) representation. In this representation, the plant is modelled as resources and tasks. Resources include raw materials, intermediates and products like the STN, but also include processing, storage, transportation and manpower. A task is an operation that converts a set of resources to another set. The tasks do not only include processing steps, but also include transportation and cleaning. In RTN, resources are produced and consumed.

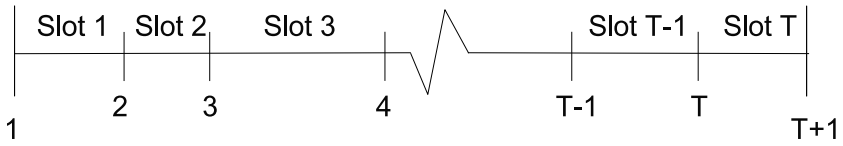

Figure 2.5: Time points and slots used by Schilling & Pantelides (1996)

Although continuous time formulations reduced the number of binary variables, problems still occurred in solving schedules for large-scale industrial plants. The assignment of a single binary variable to units (i) and tasks (j) at any time (n), is common to all previously discussed formulations, and leads to $i{\times}j{\times}n$ number of binary variables. This observation led Ierapetritou & Floudas (1998) to develop an MILP continuous-time formulation, based on the STN to address this problem. The main contribution of their formulation is the decoupling of task events from unit events. This is done by the introduction of binary variables for tasks *wv(j,n)* and units *yv(i,n)*, which represent the common binary variable *y(i,j,n)*. As a result, instead of $i{\times}j{\times}n$ binary variables, their formulation leads to $(i{+}j){\times}n$ binary variables. In this formulation a further reduction in binary variables is possible if there exists a one-to-one correspondence between tasks

and units. However, this reduction can become tedious in large-scale industrial plants. Following this observation Majozi & Zhu (2001) developed a scheduling representation and a continuous time formulation, which gives the least number of binary variables and does not require simplification.

2.2 The State Sequence Network

Majozi & Zhu (2001) introduced the State Sequence Network (SSN) representation consisting of states only. The SSN is a graphical representation of all the states that occur on the particular batch plant and is derived from the recipe. A state changes when it undergoes some process, such as mixing, separating or reacting. This is represented by an arc connecting two consecutive nodes. The building blocks of the SSN are shown in Figures 2.6, 2.7 and 2.8. From these building blocks its is easy to construct an SSN for any situation. The difference between the SSN and the STN is that in the SSN only states are considered while tasks are implicitly incorporated. The formulation makes use of time points proposed by Schilling & Pantelides (1996) and also used by Ierapetritou & Floudas (1998). The main difference between this model and the model proposed by Ierapetritou & Floudas (1998) is that a binary variable is not assigned to the task, so if a one-to-one correspondence between a state and a task does not exist, fewer binary variables will result when using the SSN.

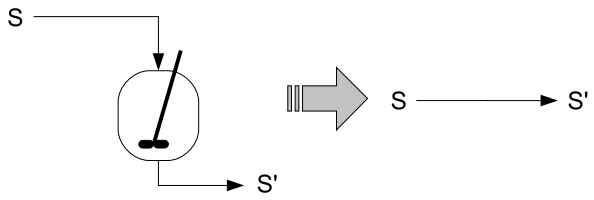

Figure 2.6: Simple unit operation

The defining of effective states is an integral part when using the SSN because this reduces the number of binary variables. Effective states are a subset of all the input states so only input states are considered. If a process requires multiple raw materials to make a particular product, then it is a fact that if one of the raw materials is fed then all of the

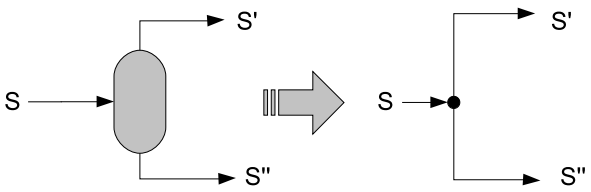

Figure 2.7: Unit operation where states mix

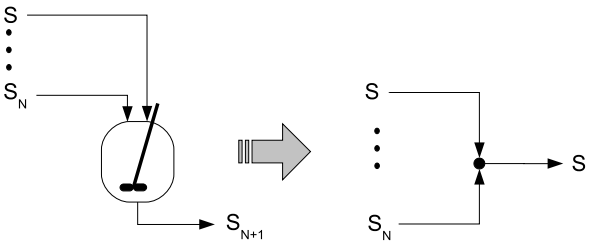

Figure 2.8: Unit operation where states split

other required feeds must also exist to make the product. By noting this, it is simple to see that only one of the states need to be defined as an effective state to ensure that all the states are fed to the particular process. For instance in the example in Figure 2.9, the second reactor requires two feeds, S2 and S3. This leads to two choices of effective state, either S2 or S3, the reason that only one of these states need contribute to the number of binary variables is that if S2 is fed to the reactor then S3 must also be fed to the reactor at the same time so there is no need for both to be effective states.

Once again using the example in Figure 2.9, where the SSN and STN are constructed from the given flowsheet, the attractiveness of the model proposed by Majozi & Zhu (2001) is illustrated. Applying the formulation of Ierapetritou & Floudas (1998) to the example in Figure 2.9, $n\times(3+3)$ binary variables (where n is the number of time points) would be required, but using the one-to-one correspondence between tasks and units, this would reduce to $3\times n$ binary variables. However, using the SSN and defining the effective states as S1, S3 and S4, $3\times n$ binary variables would be required. Both formulations result in the same number of binary variables but using the SSN no simplification was required.

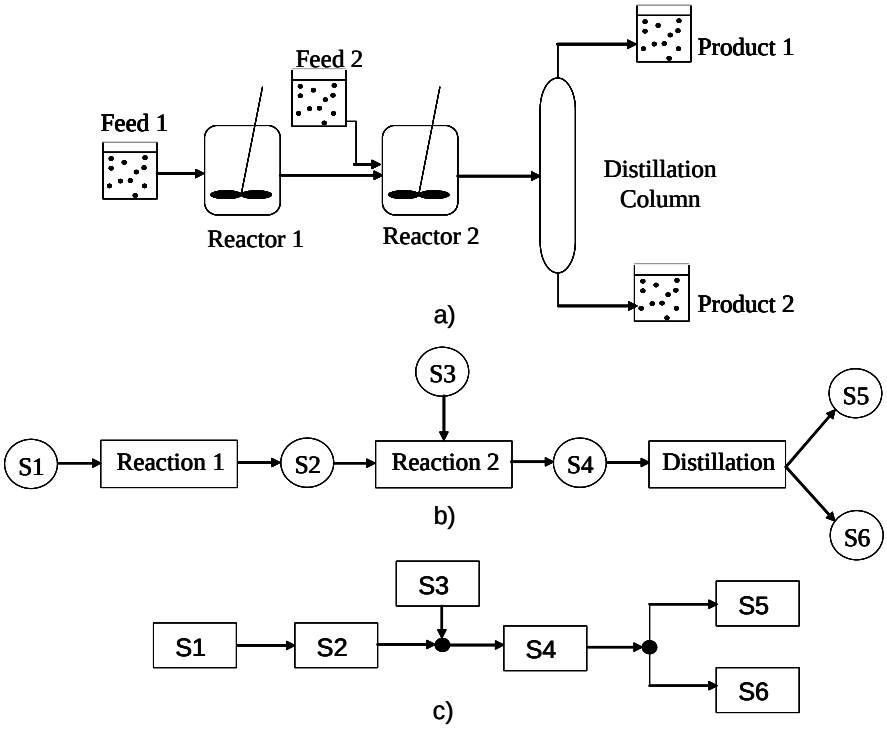

Figure 2.9: a) Flowsheet, b) The STN and c) SSN representation of (a)

2.3 S-Graph

Although mathematical methods are often encountered in this kind of research there are graphical techniques to schedule the operation of batch plants. The schedule-graph (S-graph) (Sanmartí *et al.*, 1998) representation is such a technique. All of the above scheduling formulations require the assignment of binary variables to either a state or resource, task or both, and all formulations rely on the discretization of the time horizon of interest, whether it be uniform or non-uniform discretization. The so-called continuous time formulations are not in fact continuous as their name suggests. Any method where there is discretization of the time horizon cannot be called continuous. The S-graph does not suffer from the problem of binary variables which increase the computational complexity of a problem. Furthermore, this method does not require the discretization of the time horizon so can truly be called continuous. Although this method seems to be the answer to all the problems associated with the mathematical models, the generation of the S-graph is complex and is ill suited to the case where tasks such as storage are not definite.

In this representation, nodes represent the production tasks and the arcs represent the precedence relationships between tasks. A node is also assigned to the final product. A number is placed over the arc which represents the duration of that task as shown in Figure 2.11. This figure is the representation of the master recipe as shown in Figure 2.10

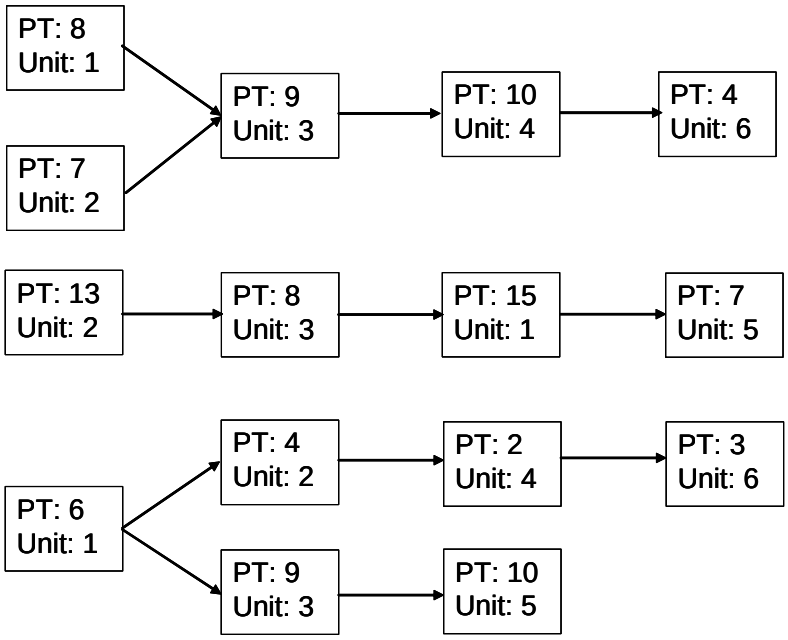

Figure 2.10: Master recipe for a batch production facility

After completion of the master recipe the precedence constraints are set for the units. Depending on the operational philosophy (UIS or NIS) the precedence arcs are set. The operational philosophy changes the start location of the arcs joining the same piece of equipment, for UIS the arcs start at the node of the same unit and terminate at the same unit completing the next task. However, for the NIS policy the arcs start at the following node and terminate as before. These arcs have a task duration of 0, or some value which gives one the time which is required for the transfer of mass from one unit to another. Figure 2.12 gives an example of an S-graph. One should note that this is one example of many possible S-graphs for this example.

The final goal of the formulation of Sanmartí *et al.* (1998) was the minimization of makespan with a feasible schedule. This is done by simply finding the longest pathway starting from the end node and terminating at the source. After the longest path has

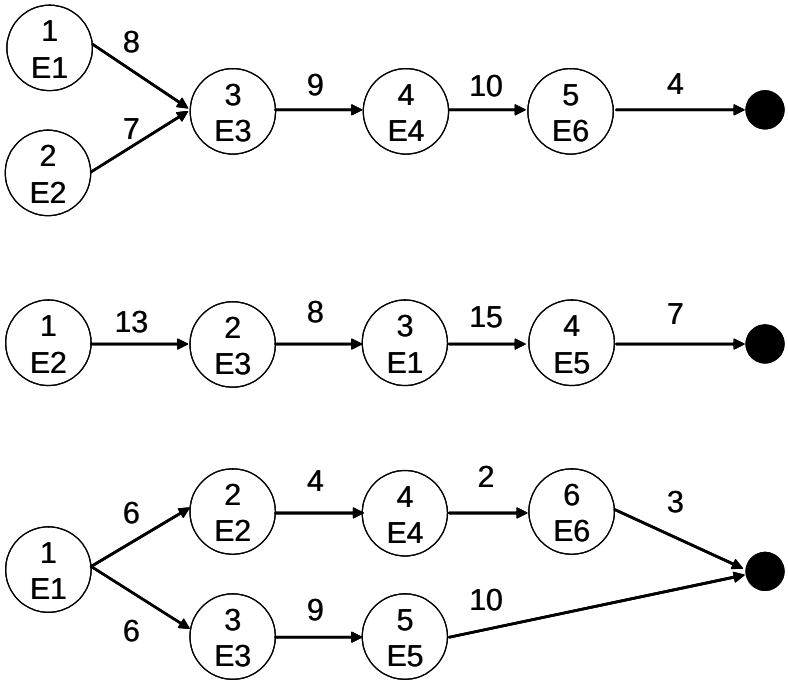

Figure 2.11: Graphical representation of recipes

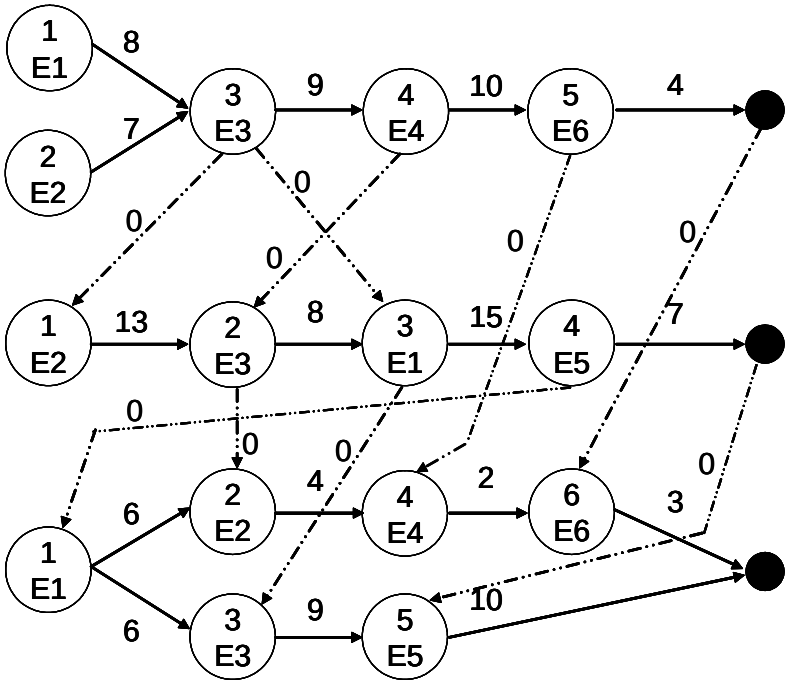

Figure 2.12: Schedule graph for the NIS sequence to minimise makespan

been found the schedule can be constructed from the appropriate S-graph. The basic assumption associated with this method is that of fixed batch size.

Further improvements on the method have been made, most significantly by Majozi & Friedler (2006). Their formulation extends the current formulation, where instead of minimising makespan they maximise throughput or profit over a given time horizon of interest, furthermore they propose a novel node cutting algorithm which improves the computational efficiency.

The way these cuts are performed is easily described using a two-dimensional example as shown in Figure 2.13 where the nodes represent the number of batches of product A or B. The search space is divided into two sections, namely, a region which can be excluded and the region where the optimal point lies, by a constant revenue line. The constant revenue line is derived firstly, by placing the product with the higher revenue contribution on the x–axis, secondly, noticing that the highest revenue that can be achieved using a single product is the point defined by $(\mathrm{N}_B)^U$, and thirdly, the actual line can be constructed by maintaining this revenue. Any point above this line will have a higher revenue than simply producing product B. After this cut has been made further cuts can be made by noticing, from the figure, that if you test node 4 and it is found to be infeasible then it is not necessary to test node 1, because if one cannot produce 3 batches of B and 2 batches of A within the time horizon then one certainly cannot produce an extra batch of A. Similarly for point 3, where there is an increase in the number of batches of B. Furthermore, node 2 does not need to be checked for the same reasons. Using both these cutting methodologies, the number of nodes that need to be checked reduces significantly, although this is a two-dimensional example the method can be easily extended to more dimensions.

Although the graphical methods described above seem to have many significant advantages over their mathematical counterparts, the S-graph requires definite knowledge of the tasks required to produce a batch. In the case where storage becomes a task this method becomes overly complex.

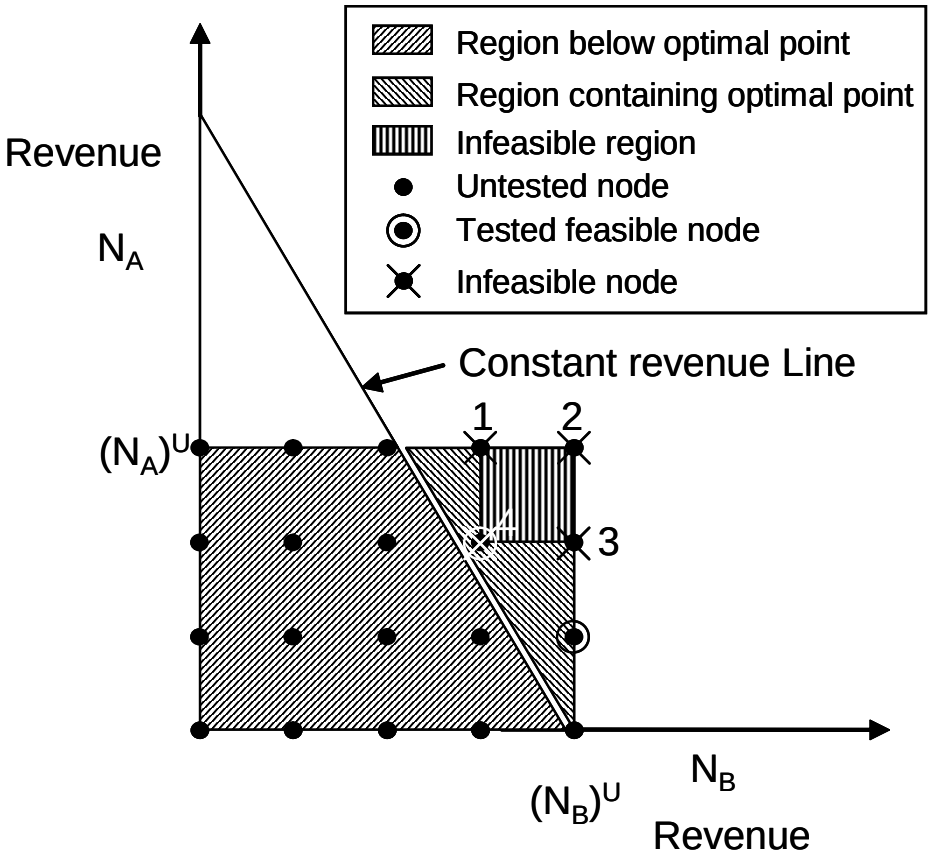

Figure 2.13: Search space for a two product system

2.4 Scheduling under uncertainty: Reactive scheduling

Most scheduling models assume that problem data are known and fixed. However, in practice this is not usually the case. Uncertainties in batch processing derive from two main sources. Firstly, process model parameters, i.e., processing times (catalyst deactivation) and equipment availability (equipment failure). Secondly, economic issues, i.e., due dates, product demand, cost of raw materials and price of product. Reactive scheduling is reactive in nature, i.e., once a problem has been identified steps are taken to minimise the effect of the problem, without resorting to rescheduling the entire plant operation. These steps include the local reordering of batches, reassignment of certain batches to other equipment items or simply shifting the starting times of the affected batches.

One approach to countering the variability of batch processes is to design intermediate storage between process units and to maintain a reserve of material for downstream processes (Karimi & Reklaitis, 1985a,b). This procedure allows decoupling of two units while allowing the original schedule to be used without modification. However, there are a number of concerns with this solution. Firstly, it requires the production of sufficient material which could be expensive, secondly, there need to be storage tanks for every intermediate which is also expensive and in multipurpose plants is in some cases impossible. Finally, if the products are unstable, and thus need to be used as soon as they are produced, intermediate storage is not an option.

Following this Cott & Macchietto (1989) came up with a method for solving variability problems. They stated that there are two ways in which a batch can be affected, firstly the processing time could be longer than expected and secondly, if a task is finished earlier than expected. Figure 2.14 shows the effects on a small two unit plant using the ZW operational policy. The first of the series of pictures illustrates the schedule under ideal operating conditions, i.e., when there are no disturbances. The second of the series illustrates what would occur if the first batch was processed faster than the schedule indicated, as one can see there is time wasted before processing the second

batch. In the final figure of the series the first batch takes longer than expected, causing the second batch to wait in unit 1. This results in a violation of the ZW operational policy. To counter these problems Cott & Macchietto (1989) developed an algorithm to dynamically modify the original schedule to improve the plant performance. This method shifts the starting times of batches. For example, if we revisit the example in Figure 2.14 and apply their algorithm, the results are shown in Figure 2.15. In the case where the processing time was shorter than expected the second batch is brought earlier to reduce the idle time of the units, while in the case where the processing time was longer than expected the starting time for the second batch is moved later so that the ZW operational policy is obeyed.

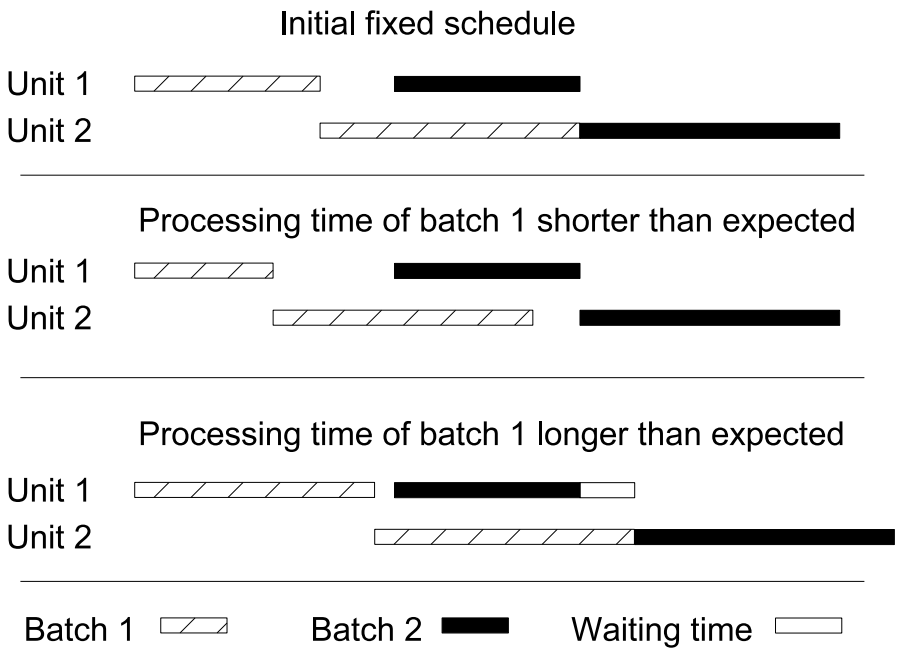

Figure 2.14: The effects of batch variability on a two unit operation

The scope of Cott & Macchietto (1989) research into this field was only based on the start time modification, which is an effective method. However, the approach does not take into account customer priorities and possible changes in unit functionality. This lead Kanakamedala *et al.* (1994) to develop a method which also takes into account unit replacement as well as time shifting. Following this Sanmartí *et al.* (1997) developed a scheduling model for multipurpose batch plants in the presence of equipment failure uncertainty. They introduced a *reliability index* which represents the discrete probability

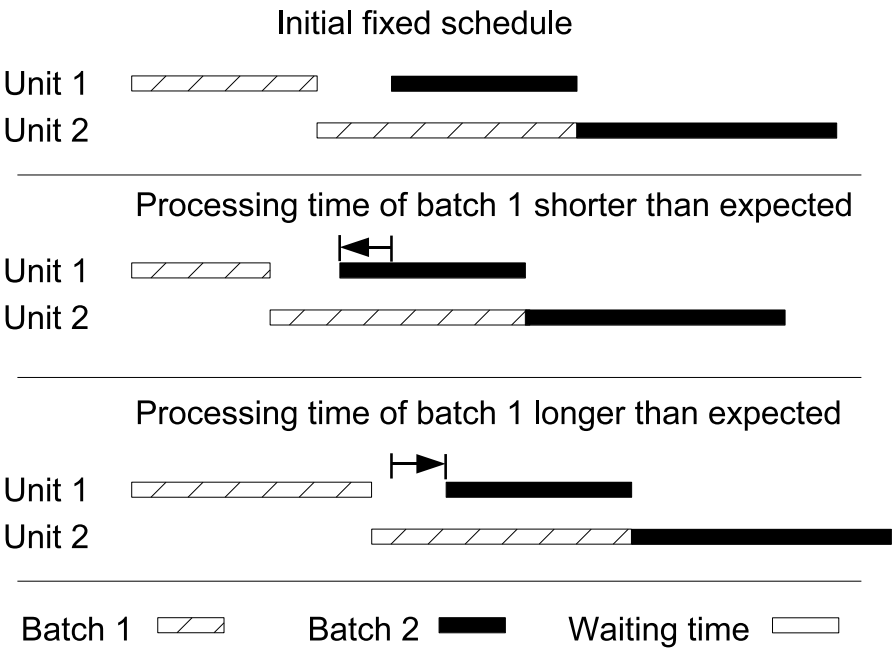

Figure 2.15: Schedule with online schedule modification

that the corresponding unit will be available to perform the required task. The value of this index depends on the failure history of the unit, its maintenance history and its intended use. They further assumed that the reliability decreases linearly with the number of times the unit is used. This method is based on 'rules of thumb', so global optimality cannot be assured.

Roslöf *et al.* (2001) developed an MILP based algorithm that can be used to improve sub-optimal schedules or to reschedule jobs in case of changed operational parameters. They adjusted the number of simultaneously released jobs, in order to reduce the computational complexity of the resulting MILP. Méndez and & Cerdá (2003) developed a similar method to the method described by Roslöf *et al.* (2001), however, their approach allows the the user to perform multiple rescheduling operations at the same time. Méndez and & Cerdá (2004) further improved on their earlier work by including the renewable discrete resources such as manpower, equipment and tools.

2.5 Pipeless batch plants

Batch plants are inherently flexible, however, this flexibility is directly proportional to the piping network that links the various pieces of equipment. Material can only be transported to process units that are linked to each other. In a large multipurpose plant, one can imagine that to completely exploit the flexibility, the piping network would be large, complex and in some cases impossible to construct due to size considerations. However if one was to completely remove this piping network and instead move the process units, one could come closer to exploiting the true flexibility and could, therefore, adapt to fast market changes (Niwa, 1993; Zanetti, 1992).

The main difference between these plants and conventional fixed pipe plants is that material is transported from stage to stage in *transferable vessels*. Processing takes place at *processing stations* and in general, material is processed in the same vessel as it is transfered in (Realff *et al.*, 1996). A simple blending operation, where component x is mixed with component y, is shown in Figure 2.16.

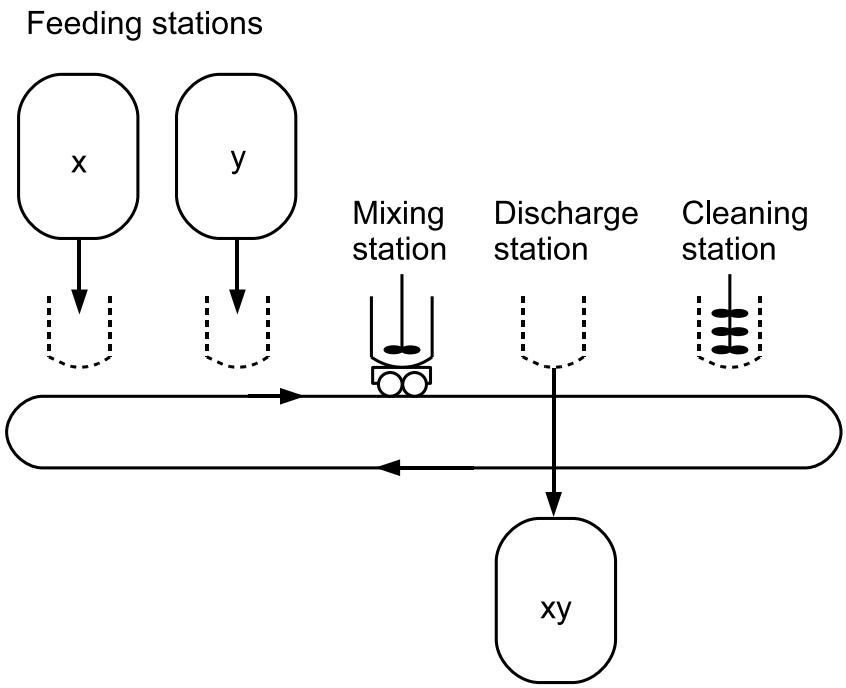

Figure 2.16: Pipeless batch plant

A rather novel analogy was drawn by Niwa (1993), where a pipeless batch plant is compaired to a chemical laboratory. In a laboratory, a beaker or test tube is the *transferable*

vessel and the laboratory equipment are the *processing stations*, such as scales, Bunsen burners and vacuum filters. The product is synthesised by moving the beaker or test tube to the required processing station to carry out the specific task.

The scheduling of these types of plants was done initially by Pantelides *et al.* (1995), who developed a discrete–time formulation based on the STN. This model was then extended to simultaneously design and schedule pipeless batch plants by Realff *et al.* (1996). Huang & Chung (2000) then extended the modeling to constraint satisfaction techniques. However, these formulations did not take into account the possibility of vessel collision, therefore Huang & Chung (2005) developed a model to integrate routing and scheduling.

2.6 Constraint and hybrid MILP/CP based approaches

Constraint satisfaction techniques (CST) are methods of expressing and solving constraint satisfaction problems (CSP's). These problems are similar to MILP or MINLP problems in that they are represented by constraints and constrained variables, where variables have their specific domains. A solution to a CSP is an assignment of values to the variables which satisfy all of the constraints simultaneously, giving only feasible solutions whereas a solution to an MILP problem is the optimal solution. An optimal solution can be found using these techniques by inserting a constraint that ensures that the next objective function should be larger than the last feasible solution (in the case of a maximization problem, and smaller in the case of a minimization problem) and repeating this until an optimal solution is found. As one can imagine, in some problems this would take a lengthly amount of time. This is the main disadvantage of the CP techniques.

The way the CST works, is by reducing the search space by using the constraints themselves. Furthermore, it uses existing constraints in a constructive way to deduce other constraints and find inconsistencies among possible solutions, which is known as constraint propagation and consistency checking.

To illustrate this method a simple example can be used. If we have two variables h and

w with domains {0, 1, 2, 3..., 14, 15} and {0, 1, 2, 3,..., 9, 10}, respectively, and constraints h<10, w<6 and h+w=12. Using the first two constraints unsuitable values can be removed. The domains of h and w now change to {0, 1, 2, 3..., 8, 9} and {0, 1, 2, 3, 4, 5}, respectively. These sets can be further reduced when the third constraint is set to {7, 8, 9} and {3, 4, 5}. To search for a solution, a choice point (called a *node*) is set by assigning either 7, 8 or 9 to h. If 7 is assigned to h, then constraint propagation will remove 3 and 4 from the domain of w, and set w to 5. After this potential solution has been found the program can backtrack and set $h = 8$ and then remove 3 from the domain of w, leaving $w = 4$, and similarly for $h = 9$ and $w = 3$. From this example it is clear to see that the search space has been greatly reduced, and that this method only provides the user with a set of feasible solutions and not the optimal solution. Furthermore, it should be noted that there isn't an objective function, but only the satisfaction of the constraints.

Using this approach Huang & Chen (2005) developed a batch scheduling model based on the CST and solved complex problems, however, due to the nature of these techniques optimality was not achieved in a reasonable amount of time. CST does have the advantage of cutting the search space using constraint propagation but has the disadvantage of not finding the optimal schedule. From the conclusions of Huang & Chen (2005), it is clear that a hybrid CSP/MILP technique which could take the search space reduction techniques of the CST and combine it with an MILP model, which can find optimal solutions, would lead to faster solution times.

There is a large amount of research going into this area (Maravelias & Grossmann, 2004; Roe *et al.*, 2005; Harjunkoski & Grossmann, 2002; Jain & Grossmann, 2001) to try to develop methods which use the complementary strengths of both methods. In general the way these methods are hybridised is by relaxing the MILP and solving for optimality, then using CP to check for feasibility. If the answer is feasible then it is optimal. However, if it is infeasible, the causes of infeasibility are inferred and further cuts are generated. Roe *et al.* (2005) followed a similar path, however, they decompose the problem differently. Firstly, an aggregate planning problem is solved using an MILP model, and then the

sequencing problem is solved using CP. These methods show large improvements when compared to solving pure MILP and CP approaches.

2.7 Conclusions

From this literature survey it is clear that there are two main types of methods, graphical and mathematical programming. Graphical methods exploit the structure of the recipe to determine the optimal schedule and are effective when applied to plants running on the NIS or UIS operational policies or where every task has a pre-specified duration. Furthermore, these methods do not require pre-specification of a number of time points prior to solution of the model, and are thus truly continuous. On the other hand, mathematical models require the presupposition of the number of time points and therefore are not continuous, regardless of whether the time horizon is discretized evenly or unevenly. However, mathematical methods are able to handle tasks of varying duration and handle all previously developed operational policies. One of the most advanced of the mathematical programming techniques is that of Majozi & Zhu (2001), which utilises the SSN and the non-uniform discretization of the time horizon approach. The model is also a MILP model and thus results in globally optimal solutions.

The techniques discussed in the literature survey have been developed to take many different operational policies into account. The reason for this is so that the practical environment can be more easily modeled. However, none of these operational policies exploit the latent storage found in most batch chemical facilities. As such the use of this latent storage requires the introduction of a new operational policy, the Process Intermediate Storage (PIS) operational policy.

CHAPTER 3

Methodology

The models developed to take the PIS operational policy into account are detailed in this chapter. The models are based on the SSN and continuous time model developed by Majozi & Zhu (2001), as such their model is presented in full. Following this the additional constraints required to take the PIS operational policy into account are presented, after which, the necessary changes to constraints developed by Majozi & Zhu (2001) are presented. In order to test the scheduling implications of the developed model, two solution algorithms are developed and applied to an illustrative example. The final subsection of the chapter details the use of the PIS operational policy as the basis of operation to design batch facilities. This model is then applied to an illustrative example. All models were solved on an Intel Core 2 CPU, T7200 2GHz processor with 1 GB of RAM, unless specifically stated.

The models developed in this section take the PIS operational policy into account. This operational policy is novel and thus requires further explanation. When a batch operation is scheduled a Gantt chart is usually generated, such as in Figure 3.1. From this figure it is

simple to identify the latent storage potential of the units. For example, units 1, 3, 4 an 6 are idle and empty for most of the time horizon of interest. This provides the opportunity of using these units as storage, instead of, or in conjunction with dedicated intermediate storage. This leads to a number of benefits, such as increased capital utilization of the equipment, possible reduction in the size required for the plant and a reduced capital cost associated with the construction of new batch facilities. In order to illustrate the idea even further, the following example was developed.

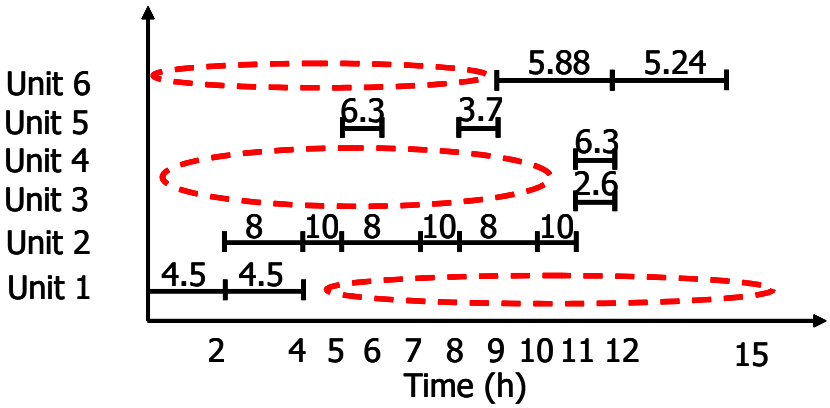

Figure 3.1: General schedule

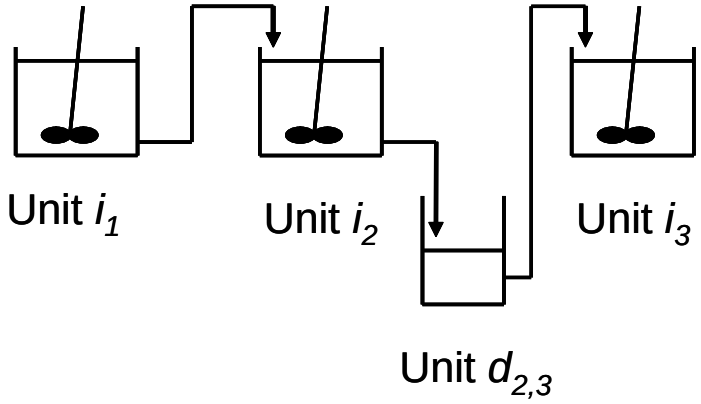

Figure 3.2: Flowsheet for the simple example

Table 3.1: Data for simple illustrative example

Unit	Capacity(ton)	Processing time (h)
i_1	100	2
i_2	100	3
i_3	50	2
$d_{2,3}$	50	-

Figure 3.2 shows the flowsheet for the illustrative example. The data for the example

is given in Table 3.1, where i_1, i_2, and i_3 are consecutive processing units, while $d_{2,3}$ is a dedicated intermediate storage vessel between processing units i_2 and i_3. The time horizon of interest in this example is 9 hours.

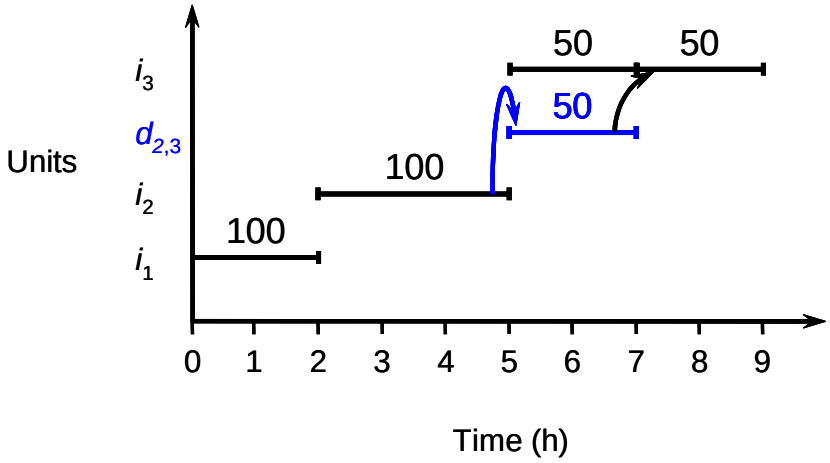

Figure 3.3: Schedule of the simple example without using PIS

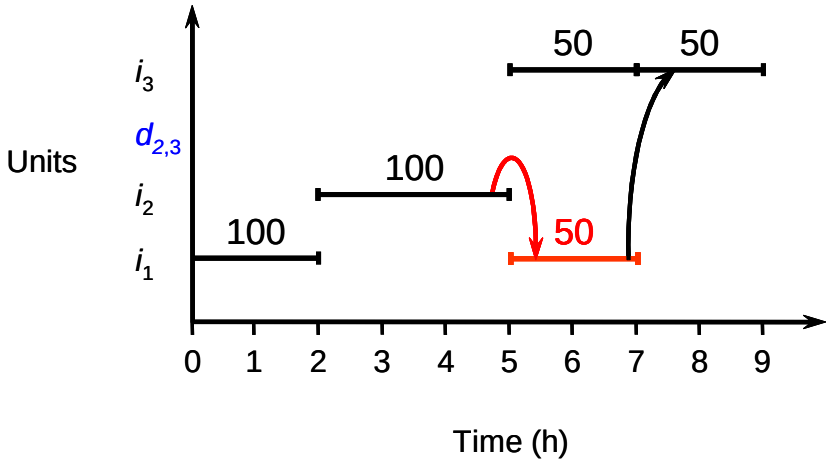

Figure 3.4: Schedule of the simple example without using PIS

When the PIS operational policy is not used, as in Figure 3.3, 50 tons of dedicated intermediate storage unit, $d_{2,3}$, was required. The reason for this is that the capacity of the final unit is only 50t, due to this, the 100t batch produced from unit i_2 must be split in half. Half of the batch is stored in dedicated intermediate storage while the remaining batch is processed, after which, the stored mass is then processed thus achieving the optimal throughput of 100t. However, when compared to the schedule in Figure 3.4, the 100 ton storage vessel is not needed. The reason for this is that 50t of the intermediate product produced from unit i_2 is moved to i_1 for storage, while the remaining 50t is

processed in unit i_3. This increases the capital utilization of unit i_1, while reducing the size required for the plant which achieves the same throughput. This also avails unit i_2 for further processing. Furthermore, if this possibility had been identified at the design phase, the cost of the 50t storage vessel could have been saved.

In order to illustrate the uses of this novel operational policy, i.e. PIS operational policy, this chapter has been divided into two parts. Firstly, the applicability of the operational policy will be proven and used to determine the minimum amount of intermediate storage required while maintaining the throughput achievable with infinite intermediate storage. Secondly, the PIS operational policy will be used to design storageless batch plants.

3.1 Scheduling implications

In order to test the applicability of the PIS operational policy the problem has to be clearly defined.

3.1.1 Problem Statement

The problem can be formally stated as follows,

Given:

(i) the production recipe for each product, including processing times in each unit operation,

(ii) the available units and their capacities,

(iii) the maximum storage capacity for each material, and

(iv) the time horizon of interest,

determine,

(i) the maximum throughput with zero intermediate storage with and without using the PIS operational policy

(ii) the minimum amount of intermediate storage, while maintaining the optimal throughput

3.1.2 Mathematical model

A mathematical model based on the mdel developed by Majozi & Zhu (2001) was developed to solve the stated problem.

Sets

$$
\begin{aligned}
P &= \{p|p = time\ point\} \\
J &= \{j|j = unit\} \\
S &= \{s|s = any\ state\} \\
S_{in} &= \{S_{in}|S_{in} = any\ input\ state\} \\
S_{out} &= \{S_{out}|S_{out} = any\ output\ state\}
\end{aligned}
$$

Binary Variables

$y_{in}(s,j,p)$	decision variable describing the processing of state s in unit j at time point p
$y^{lt}(s,j,p)$	decision variable describing the latent storage of state s in unit j at time point p
$e(j)$	decision variable based on whether unit j exists or not

Continuous Variables

$m_{in}(s,j,p)$	amount of state s consumed for processing in unit j at time point p
$m^{s}_{in}(s,j,p)$	amount of state s fed into storage from unit j at time point p

$m_{in}^{lt}(s,j,j',p)$	amount of state s fed into latent store j' from unit j at time point p
$m_{out}(s,j,p)$	amount of state s produced from unit j at time point point p
$m_{out}^{s}(s,j,p)$	amount of state s fed from storage to unit j at time point p
$m_{out}^{lt}(s,j',j,p)$	amount of state s from latent store in j' fed to unit j at time point p
$c(j)$	capacity of unit j
$t_{in}(s,j,p)$	time at which state s is used in unit j at time point p
$t_{in}^{lt}(s,j,p)$	beginning of latent storage for state s in unit j at time point p
$t_{out}(s,j,p)$	time at which state s is produced from unit j at time point p
$t_{out}^{lt}(s,j,p)$	end of latent storage for state s in unit j at time point p
$w(s,j,p)$	storage time for state s in unit j during latent store at time point p
$q(s,p)$	amount of state s stored at time point p
$d(s,p)$	amount of state s delivered to customers at time point p
Z	objective function to be evaluated

Parameters

V_j^{min}	Minimum capacity of unit j
V_j^{max}	Maximum capacity of unit j
$Q_s^0(s)$	Amount of state initially stored
$\tau(s)$	Processing time of state s
$W^U(s,j)$	Upper limit of the duration of the latent storage of state s in unit j

$W^L(s,j)$	Lower limit of the duration of the latent storage of state s in unit j
H	Time horizon of interest
A, B	Unit specific capital cost terms
α	Power function for capital cost objective function

Basic scheduling model

In the first section, the constraints developed by Majozi & Zhu (2001) are repeated for completeness of the model.

Capacity constraints

$$V_j^{min} y(s_{in}, j, p) \leq \sum_{s \in S_{in}} m_{in}(s, j, p) \leq V_j^{max} y(s_{in}, j, p) \qquad \forall\, s \in S_{in},\ j \in J,\ p \in P \quad (3.1)$$

Constraint (3.1) states that the mass entering a unit for processing must be between the minimum and maximum capacities of the unit. Furthermore, it ensures that if mass enters a unit, that unit becomes active.

Material Balances

$$\sum_{s \in S_{in}} m_{in}(s, j, p-1) = \sum_{s \in S_{out}} m_{out}(s, j, p) \qquad \forall\, j \in J,\ p \in P, p > 1 \quad (3.2)$$

$$q(s,p) = q(s,p-1) + \sum_{j \in J} m_{out}(s,j,p) - \sum_{j \in J} m_{in}(s,j,p)$$
$$\forall\, s \in S,\ S = intermediate,\ p \in P, p > 1 \quad (3.3)$$

$$q(s,p) = q(s,p-1) - \sum_{j \in J} m_{in}(s,j,p) \qquad \forall\, s \in S,\ S = feed,\ j \in J,\ p \in P,\ p > 1 \quad (3.4)$$

$$q(s,p) = q(s,p-1) + \sum_{j \in J} m_{out}(s,j,p) - d(s,p)$$

$$\forall\, s \in S,\ S = product,\ j \in J,\ p \in P, p > 1 \quad (3.5)$$

$$q(s,p_1) = Q_s^0(s) - \sum_{j \in J} m_{in}(s,j,p_1) \qquad \forall\, s \in S,\ j \in J,\ p_1 \in P \quad (3.6)$$

The material balances are shown by constraints (3.2), (3.3), (3.4), (3.5) and (3.6). Constraint (3.2) is a mass balance over a processing unit. It simply states that the mass that enters unit j at time point $p-1$ must exit that unit at the next time point. Constraint (3.3), is a balance over a dedicated intermediate storage unit and only applies to intermediate products. This constraint states that the amount of state s that is stored in the dedicated intermediate storage unit is the difference between that which enters and exits for processing and the amount of state s that was already present at the previous time point. Constraint (3.4) applies where mass is only used, such as with feed states. The amount of state delivered to the customer is determined by constraint (3.5), where a state is only produced not used. Constraint (3.6) is similar to constraint (3.3), however, it applies at the beginning of the time horizon of interest. This constraint takes care of the possibility that there is state stored in a unit prior to the start of scheduling, such as feeds or in the case where rescheduling is required.

Duration Constraint

$$t_{out}(s_{out}, j, p) = t_{in}(s_{in}, j, p-1) + \tau(s)y(s_{in}, j, p-1)$$
$$\forall\, j \in J,\ p \in P,\ p > 1,\ s_{out} \in S_{out}, s_{in} \in S_{in} \quad (3.7)$$

The model is based on a fixed duration of tasks as shown in constraint (3.7). This constraint states that the time at which the output state from unit j exits, is the time at which the input state entered the unit at the previous time point plus the duration of the task. The binary variable ensures that the constraint holds whenever the unit is used at the precise time, i.e. $p-1$.

Sequence Constraints

$$t_{in}(s_{in}, j, p) \geq \sum_{s_{in} \in S_{in}} \sum_{s_{out} \in S_{out}} \sum_{j \in J} \sum_{p' \leq p} \Big(t_{out}(s_{out}, j, p') - t_{in}(s_{in}, j, p'-1) \Big)$$
$$\forall\, j \in J,\ p \in P,\ p > 1 \quad (3.8)$$

$$t_{in}(s_{in}, j, p) \geq t_{out}(s_{out}, j, p) \qquad \forall\, j \in J,\ p \in P,\ s_{out} \in S_{out},\ s_{in} \in S_{in} \quad (3.9)$$

$$t_{in}(s_{in}, j, p) \geq t_{out}(s_{out}, j', p) \qquad \forall j, j' \in J, p \in P, s_{out} = s_{in}, s_{out} \in S_{out}, s_{in} \in S_{in} \quad (3.10)$$

Constraint (3.8) reduces the search space by ensuring that the time at which a state s can be processed in unit j at time point p is at least after the sum of the durations of all previous tasks that have taken place in the unit. Constraint (3.9) ensures that the

processing of state s_{in} into unit j can only take place after the previous batch has been processed. Constraint (3.10) stipulates that state s_{in} can only be processed in unit j after it has been produced from unit j', where units j and j' are consecutive stages in the recipe.

Feasibility Constraints

$$\sum_{s \in S_{in}} y(s,j,p) \leq 1 \qquad \forall\, j \in J,\ p \in P \tag{3.11}$$

This constraint ensures that only one task can take place in a unit at a particular time point.

Time Horizon Constraints

$$t_{in}(s,j,p) \leq H \qquad \forall\, j \in J,\ p \in P, s \in S_{in,j} \tag{3.12}$$

$$t_{out}(s,j,p) \leq H \qquad \forall\, j \in J,\ p \in P, s \in S_{out,j} \tag{3.13}$$

Constraints (3.12) and (3.13) ensure that all the tasks take place within the time horizon of interest.

Storage Constraints

$$q(s,p) \leq Q_s^{max} \qquad \forall\, s \in S,\ p \in P \tag{3.14}$$

This constraint ensures that the maximum capacity of the intermediate storage units is not exceeded.

3.1.3 Extension to PIS policy

The model in the above form does not take into account the possibility of using latent storage, i.e. PIS operational policy. There are a number of additional constraints needed to fully capture this operational policy.

Capacity Constraints

$$V_j^{min} y^{lt}(s,j,p) \leq \sum_{j' \in J} m_{in}^{lt}(s,j',j,p) \leq V_j^{max} y^{lt}(s,j,p) \qquad \forall\, s \in S, j \in J,\ p \in P \quad (3.15)$$

Constraint (3.15) states that the mass entering a process unit for latent storage must be between the minimum and maximum capacities of the unit. Furthermore, it ensures that mass can only enter the unit if the binary variable associated with latent storage is active for unit j at time point p.

Material balances

$$\sum_{j' \in J} m_{in}^{lt}(s,j',j,p-1) = \sum_{j' \in J} m_{out}^{lt}(s,j,j',p) \qquad \forall\, s \in S, j \in J,\ p \in P,\ p > 1 \quad (3.16)$$

$$m_{in}(s,j,p) = \sum_{j' \in J} m_{out}^{lt}(s,j',j,p) + m_{out}^{s}(s,j,p) \qquad \forall\, s \in S, j \in J,\ p \in P \quad (3.17)$$

$$m_{out}(s,j,p) = \sum_{j' \in J} m_{in}^{lt}(s,j,j',p) + m_{in}^{s}(s,j,p) \qquad \forall\, s \in S, j \in J,\ p \in P \quad (3.18)$$

The mass balance over a process unit which is being used as latent storage is given by constraint (3.16). It should be noted that the input and output states remain the same. Constraints (3.17) and (3.18) are the inlet and outlet mass balances for mass which is to be used for, or produced from processing, respectively. Mass which enters a unit for processing comes from dedicated storage and/or latent storage as stated in constraint

(3.17). Similarly, mass which exits a unit is either moved to dedicated storage or latent storage as stated in constraint (3.18).

Duration constraints

$$t^{lt}_{out}(s,j,p) = t^{lt}_{in}(s,j,p-1) + w(s,j,p) \qquad \forall\, s \in S,\ j \in J,\ p \in P \tag{3.19}$$

$$W^{L}(s,j)y^{lt}(s,j,p) \leq w(s,j,p) \leq W^{U}y^{lt}(s,j,p) \qquad \forall\, s \in S,\ j \in J,\ p \in P \tag{3.20}$$

The duration constraint for a latent storage, constraint (3.19), is similar to constraint (3.7) except that the residence time is a variable in the latter case. In the case of latent storage the actual duration is a variable that can vary between the lower and upper limits specified for state s in unit j, as shown in constraint (3.20).

Sequence Constraints

$$t^{lt}_{in}(s,j,p) \geq t^{lt}_{out}(s',j,p) \qquad \forall\, s,s' \in S,\ j \in J,\ p \in P \tag{3.21}$$

$$t^{lt}_{in}(s,j,p) \geq t_{out}(s',j,p) \qquad \forall\, s,s' \in S,\ j \in J,\ p \in P \tag{3.22}$$

$$t_{in}(s,j,p) \geq t^{lt}_{out}(s',j,p) \qquad \forall\, s,s' \in S,\ j \in J,\ p \in P \tag{3.23}$$

$$t^{lt}_{out}(s,j,p) \leq t_{in}(s,j',p) + H\Big(2 - y^{lt}(s,j,p-1) - y(s,j',p)\Big) \qquad \forall\, s \in S_{in,j'},\ j,j' \in J,\ p \in P,\ p > 1 \tag{3.24}$$

$$t^{lt}_{out}(s,j,p) \geq t_{in}(s,j',p) - H\Big(2 - y^{lt}(s,j,p-1) - y(s,j',p)\Big)$$
$$\forall\, s \in S_{in,j'},\ j,j' \in J,\ p \in P,\ p > 1 \quad (3.25)$$

$$t^{lt}_{in}(s,j,p) \leq t_{out}(s,j',p) + H\Big(2 - y^{lt}(s,j,p) - y(s',j',p-1)\Big)$$
$$\forall\, s \in S_{out,j'},\ s' \in S_{in,j'},\ j,j' \in J,\ p \in P,\ p > 1 \quad (3.26)$$

$$t^{lt}_{in}(s,j,p) \geq t_{out}(s,j',p) - H\Big(2 - y^{lt}(s,j,p-1) - y(s',j',p-1)\Big)$$
$$\forall\ s \in S_{out,j'},\ s' \in S_{in,j'}, s \in S,\ j,j' \in J,\ p \in P,\ p > 1 \quad (3.27)$$

$$t_{in}(s,j',p) \geq t_{out}(s,j,p) - H\Big(2 - y(s,j',p) - y(s',j',p-1)\Big)$$
$$\forall\, s \in S,\ j,j' \in J,\ p \in P,\ p > 1 \quad (3.28)$$

$$t_{in}(s,j',p) \leq t_{out}(s,j,p) + H\Big(2 - y(s,j',p) - y(s',j',p-1)\Big)$$
$$\forall\, s \in S,\ j,j' \in J,\ p \in P,\ p > 1 \quad (3.29)$$

Constraints (3.21), (3.22) and (3.23) ensure that a state can only be processed or stored in unit j when the unit is available. It is assumed that after a batch has been stored in a process unit then it must follow the next processing step in its recipe. Constraints

(3.24) and (3.25) ensure that the time at which a state leaves a unit after latent storage coincides with the time that the state enters a unit which is capable of processing that state. Constraints (3.26) and (3.27), are similar to constraints (3.24) and (3.25), however these apply to a state moving from processing to latent storage. If mass is moved from processing in unit j to processing in unit j', the time at which the mass is produced must coincide with the time at which it is used, as shown by constraints (3.28) and (3.29).

Time Horizon Constraints

$$t^{lt}_{in}(s,j,p) \leq H \qquad \forall\, j \in J,\ p \in P, s \in S \tag{3.30}$$

$$t^{lt}_{out}(s,j,p) \leq H \qquad \forall\, j \in J,\ p \in P, s \in S \tag{3.31}$$

These constraints ensure that all storage activities take place within the time horizon of interest.

Feasibility Constraints

$$\sum_{s \in S_{in}} y^{lt}(s,j,p) + \sum_{s' \in S_{in}} y(s',j,p) \leq 1 \qquad \forall\, j \in J,\ p \in P \tag{3.32}$$

$$\sum_{j \in J} y^{lt}(s,j,p) \leq 1 \qquad \forall\, s \in S,\ p \in P \tag{3.33}$$

To ensure that a unit is only used for either processing or storage at a particular time point, constraint (3.32) is required. Constraint (3.33), ensures that a batch can not be split. Constraint (3.11) is redundant in the presence of constraint (3.32).

3.1.4 Necessary modifications to the basic scheduling model

In order to ensure the completeness of the model that takes the PIS operational policy into account, the basic scheduling model developed by Majozi & Zhu (2001) has to be modified as follows.

$$q(s,p) = q(s,p-1) + \sum_{j \in J_s^{out}} m_{in}^s(s,j,p) - \sum_{j' \in J_s^{in}} m_{out}^s(s,j',p) \qquad \forall\, s \in S,\ j \in J,\ p \in P \quad (3.34)$$

$$q(s,p) = q(s,p-1) - \sum_{j \in J_s^{in}} m_{out}^s(s,j,p) \qquad \forall\, s \in S,\ S = feed,\ j \in J,\ p \in P,\ p > 1 \quad (3.35)$$

$$q(s,p) = q(s,p-1) + \sum_{j \in J_s^{out}} m_{in}^s(s,j,p) - d(s,p) \qquad \forall\, s \in S,\ S = product,\ j \in J,\ p \in P, p > 1 \quad (3.36)$$

$$q(s,p_1) = Q_s^0(s) - \sum_{j \in J_s^{in}} m_{out}^s(s,j,p_1) \qquad \forall\, s \in S,\ j \in J,\ p_1 \in P \quad (3.37)$$

The balance over a dedicated intermediate storage unit has to be modified because of the possibility of latent storage. Constraint (3.34), provides the link for the inlet and outlet mass balance between units, as shown in constraints (3.17) and (3.18). Constraints (3.35), (3.36) and (3.37), are similar to constraints (3.4), (3.5) and (3.6), however they apply to the case where the PIS operational policy is taken into account.

$$t_{in}(s_{in},j,p) \geq \sum_{s}\sum_{j}\sum_{p'\leq p}\Big(t_{out}(s_{out},j,p') - t_{in}(s_{in},j,p'-1) + t^{ls}_{out}(s',j,p') - t^{ls}_{in}(s',p'-1)\Big)$$
$$\forall\, j \in J,\ p \in P,\ p>1, p'>1,\ s_{out} \in S_{out},\ s_{in} \in S_{in}, s' \in S \quad (3.38)$$

Constraint (3.8) has to be modified to include the possibility of using a unit as latent storage, as shown by constraint (3.38).

3.1.5 Objective Functions

The main goal of the project is the minimization of plant size via the exploitation of latent storage. In order to achieve this goal two cases are considered. The goal of the first case is to check the advantages gained in terms of throughput (constraint 3.39) when there is zero intermediate storage (constraint 3.40), while in the second case the goal is the minimization of intermediate storage (constraint 3.43) while maintaining the optimal throughput (constraint 3.41). In this case the optimal throughput is defined as that which is achieved when the model is solved with infinite intermediate storage. Both cases investigate the effect of using latent storage.

Case 1

$$Z = max \sum_{s\in S}\sum_{p\in P} d(s,p) \quad (3.39)$$

$$\text{while } q(s,p) = 0 \qquad \forall s \in S, p \in P \quad (3.40)$$

Case 2

Step 1:

$$Z = max \sum_{s \in S} \sum_{p \in P} d(s,p) \tag{3.41}$$

$$\text{while } q(s,p) \geq 0 \qquad \forall s \in S, p \in P \tag{3.42}$$

Step 2:

$$Z = min \sum_{s \in S} \sum_{p \in P} q(s,p) \tag{3.43}$$

$$\text{while } d(s,p) = \text{production goal} \qquad \forall s \in S, p \in P \tag{3.44}$$

3.1.6 Illustrative example

In order to illustrate these cases an example taken from the papers of Ierapetritou & Floudas (1998) and Majozi & Zhu (2001) will be used. The flowsheet for the example is given in Figure 3.5 with the SSN representation shown in Figure 3.6. The data for the example is shown in Table 3.2, the time horizon of interest for this example has been altered from 12 hours presented in Ierapetritou & Floudas (1998) and Majozi & Zhu (2001) to 24 hours for illustrative purposes.

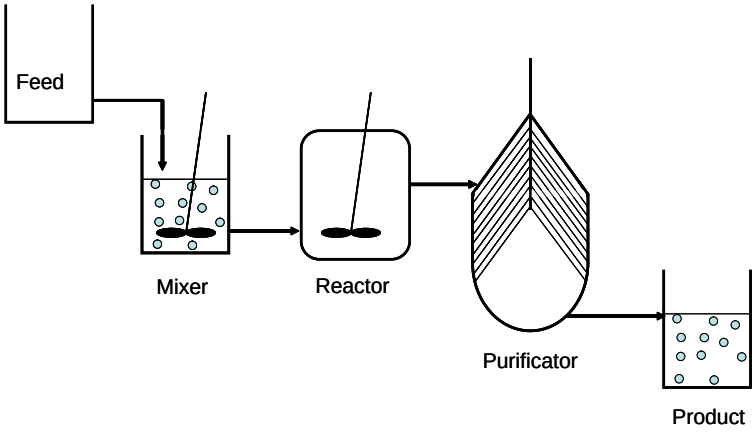

Figure 3.5: Flowsheet for the literature example

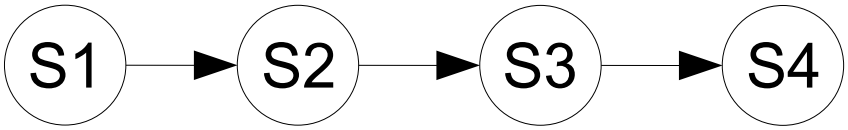

Figure 3.6: SSN for the literature example

Table 3.2: Data for illustrative example

Unit	Capacity	Suitability	Processing time (h)
1	100	Mixing	4.5
2	75	Reaction	3
3	50	Purification	1.5
State	Storage capacity	Initial amount	Price
1	Unlimited	Unlimited	0.0
2	100	0.0	0.0
3	100	0.0	0.0
4	Unlimited	0.0	1.0

Case 1

Case 1 involves solving the model using the first case where the objective is to find the maximum throughput with zero intermediate storage without using the PIS operational policy and then resolving the model with the use of the PIS operational policy to compare the results. The optimal throughput achieved without using the PIS operational policy is 200, the schedule is shown in Figure 3.7. The stair step nature of the schedule is expected due to the NIS operational policy. When the PIS operational policy is introduced the optimal throughput increases from 200 to 300 units as shown in Figure 3.8

The way this improvement is achieved is clearly seen in Figure 3.8. A portion of a batch is stored in a unit while the remaining batch is sent to processing. In this case half of the batch processed in the mixer is taken for storage in the purificator while the remaining mass is processed in the reactor. Once the reaction has proceeded to completion the mass that was stored in the purificator is moved to the reactor for processing. Further latent storage is required at 13.5 and 18 hours, in this case the mass is stored in the reactor after processing and then moved to the purificator for processing. It should be noted that there is a cycle in this schedule. A cycle occurs when a unit fills and empties at the same time.

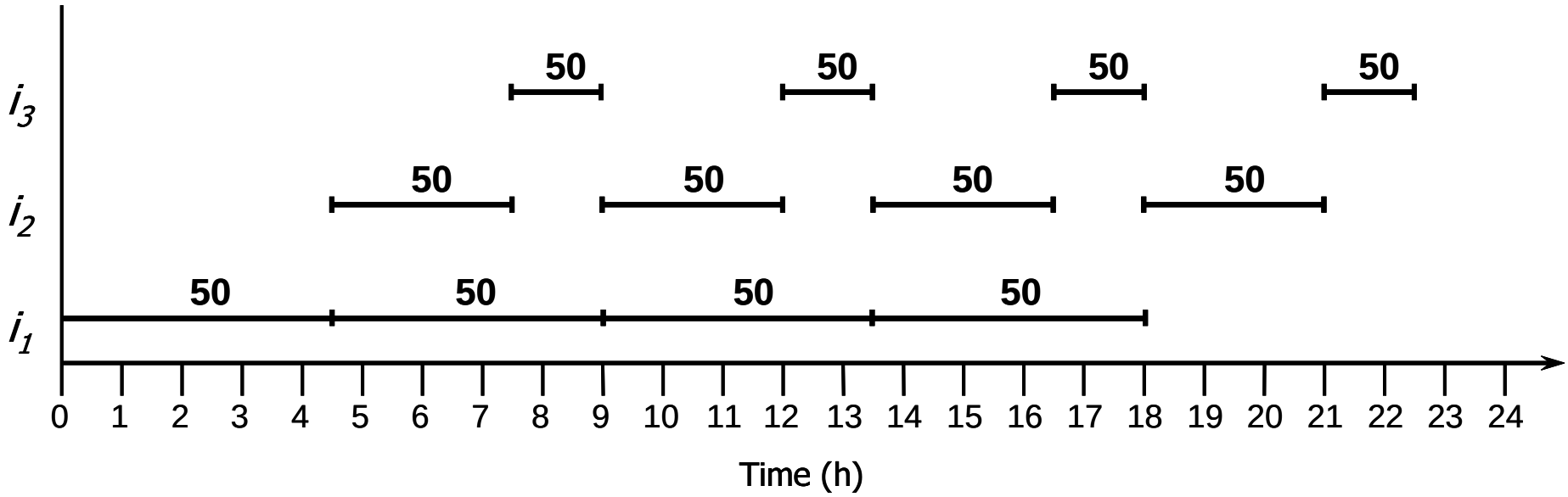

Figure 3.7: Literature example without using the PIS operational policy

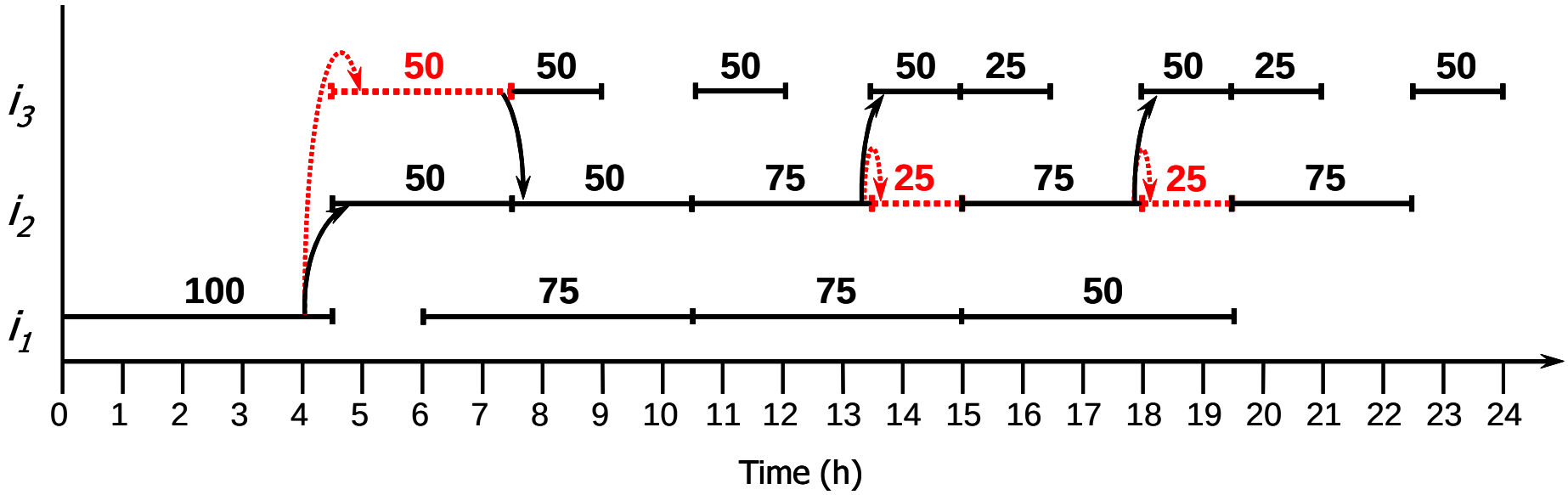

Figure 3.8: Literature example using the PIS operational policy

The model was solved using GAMS and the CPLEX solver version 9.1.2. The computational results for case 1 are shown in are shown in Table 3.3. From these results it is clear to see the potential benefits for using the PIS operational policy, with a 50% increase in throughput.

Table 3.3: Results from the first case

	Without PIS policy	With PIS policy
Number of time points	7	10
Objective function value	200	300
Number of binary variables	18	87
Solution time (CPU s)	0.062	1.718
Number of variables	561	921
Number of constraints	1619	2592

Case 2

The purpose of this case is to determine the minimum amount of intermediate storage available while maintaining the optimal throughput, where the optimal throughput is defined as that which is achieved when the model is solved with infinite intermediate storage. In this example the optimal throughput was 350 units. The schedule for this case is shown in Figure 3.9. Without using the PIS operational policy the minimum amount of intermediate storage required was 300 units. The schedule for this case is shown in Figure 3.10. In the case where the PIS operational policy was used a minimum of 200 units of dedicated intermediate storage was required. The schedule for this example is shown in Figure 3.11.

The model was solved on an Intel Core 2 CPU, T7200 2GHz processor with 1 GB of RAM, using GAMS and the CPLEX solver version 9.1.2. The computational results for the second case are shown in Table 3.4.

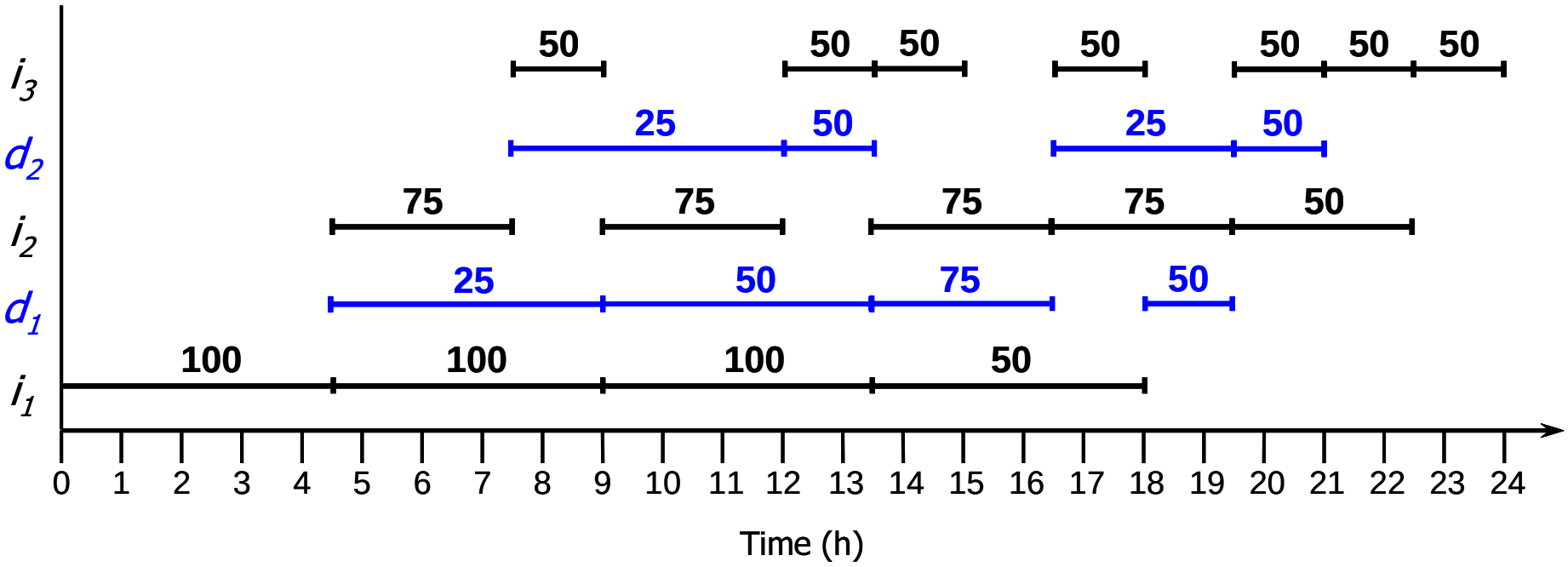

Figure 3.9: Literature example with infinite intermediate storage

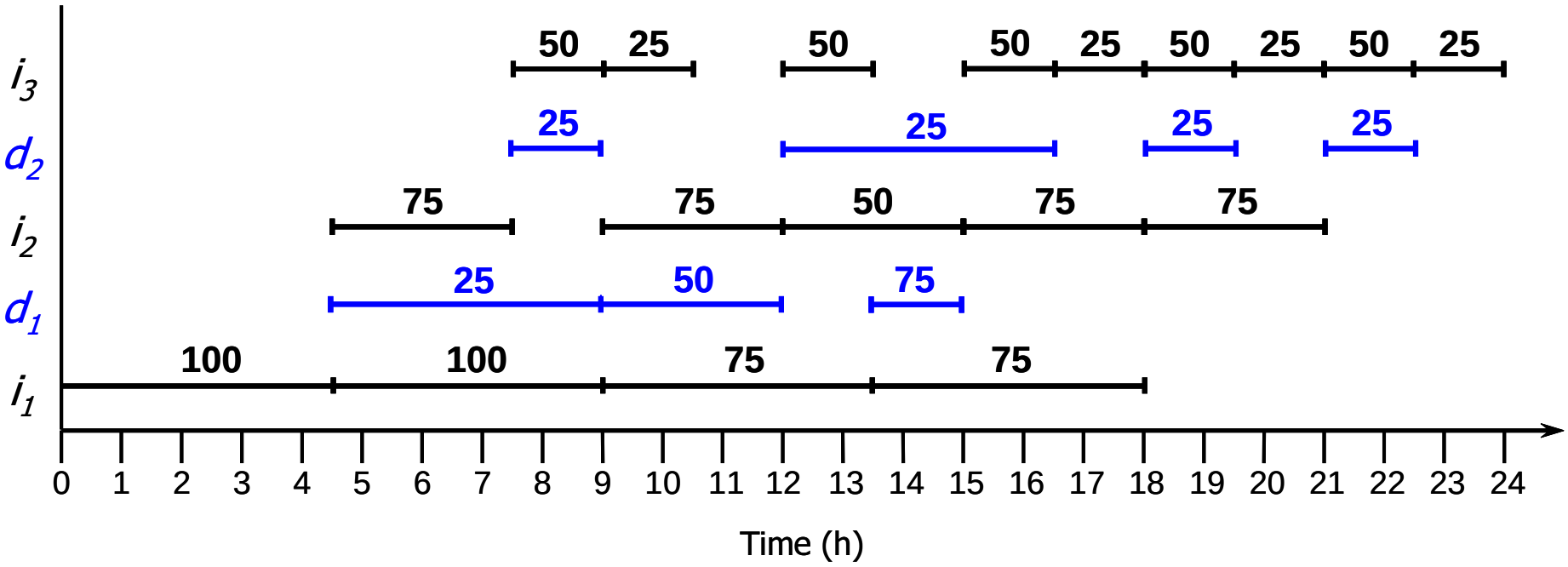

Figure 3.10: Literature example without using the PIS operational policy

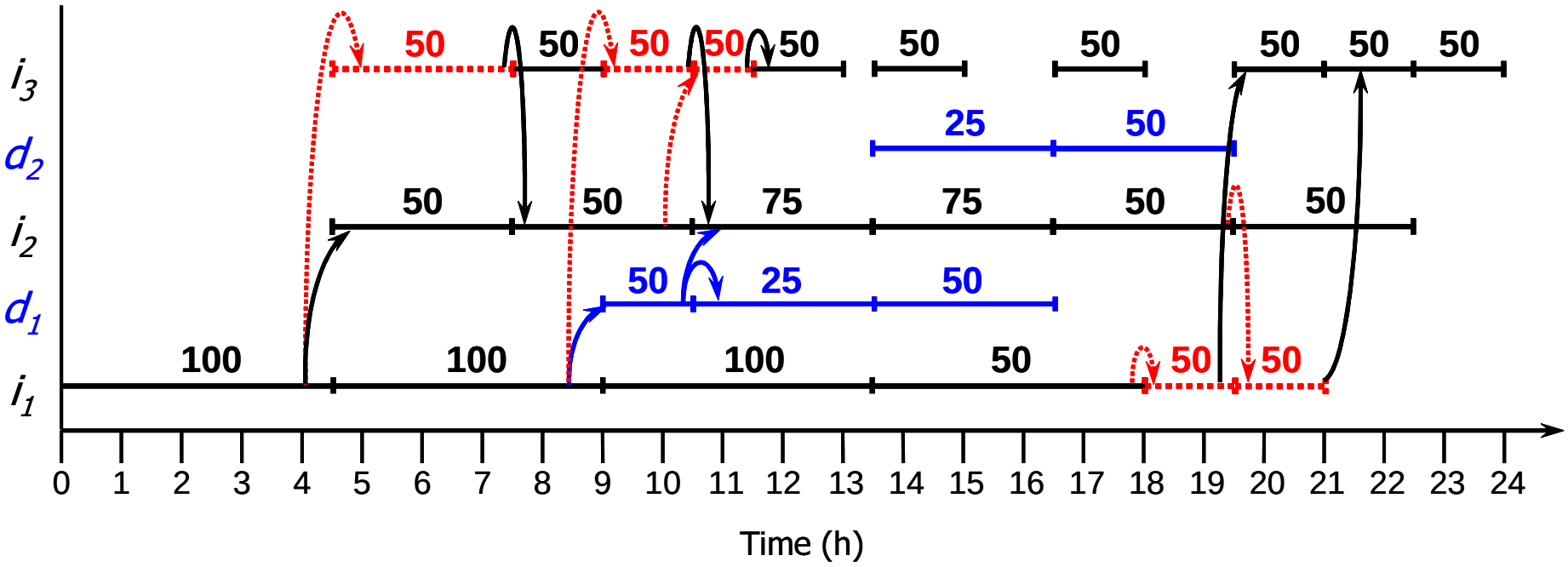

Figure 3.11: Literature example without using the PIS operational policy

Table 3.4: Results from the second case

	Infinite storage	Without PIS policy	With PIS policy
Number of time points	10	13	11
Objective function value	350	300	200
Number of binary variables	27	36	96
Solution time (CPU s)	0.156	5.734	10.125
Number of variables	801	605	1014
Number of constraints	2432	2605	2893

3.2 Design implications

This section furthers the model developed in the previous section to include the possibility of design.

3.2.1 Problem Statement

The problem considered in this subsection can be stated as follows,

Given:

(i) the production recipes, i.e. processing times for each task in a suitable unit as well as their sequence,

(ii) the availability and suitability of process vessels,

(iii) the ***potential*** number of process units in a stage, and range of capacity of potential process vessels,

(iv) production requirement, and

(v) the time horizon of interest,

determine,

the optimal number of units in a particular stage so as to minimise the capital cost

3.2.2 Necessary modifications to case 1

Constraints (3.1), (3.2), (3.7), (3.9), (3.10), (3.12), (3.13) (3.15) to (3.31) and (3.34) to (3.38) still apply to the model, however, further modifications of the model in section 3.1 are required to take into account the possibility of design.

Capacity constraints

$$V_j^{min} e(j) \leq c(j) \leq V_j^{max} e(j) \qquad \forall\, j \in J,\, p \in P \tag{3.45}$$

$$m_{in}^{lt}(s, j, j', p) \leq c(j') \qquad \forall\, s \in S,\, j, j' \in J,\, p \in P \tag{3.46}$$

$$m_{in}(s, j, p) \leq c(j) \qquad \forall\, s \in S,\, j \in J,\, p \in P \tag{3.47}$$

$$c(j) \leq c(j') + V_j^{max}(2 - e(j) - e(j')) \qquad \forall j, j' \in J \tag{3.48}$$

$$c(j) \geq c(j') - V_j^{max}(2 - e(j) - e(j')) \qquad \forall j, j' \in J \tag{3.49}$$

Constraint (3.45) ensures that the capacity of unit j is between the minimum and maximum permissible range, furthermore, it ensures that for a unit to have a capacity it must exist. Constraint (3.46) ensures that the mass entering unit j' for latent storage does not exceed the capacity of the unit. Constraint (3.47) is similar to constraint (3.46), however, it applies to unit j which is processing state s at time point p. It is further assumed that units in the same stage all have the same capacity, as shown by constraints (3.48) and (3.49), where units j and j' are units in the same stage.

Feasibility Constraint

$$\sum_{s \in S} y(s,j,p) + \sum_{s' \in S} y^{lt}(s',j,p) \leq e(j) \qquad \forall\, j \in J,\ p \in P \tag{3.50}$$

Constraint (3.50) is similar to constraint (3.32), however, it ensures that a unit can only be used for either processing or latent storage if that unit exists.

3.2.3 Objective function

$$\sum_{j \in J} \left(Ae(j) + B\left[c(j)\right]^{\alpha} \right) \tag{3.51}$$

The objective function in this case is the minimization of capital cost. Due to the non-linearity of this equation the model becomes an MINLP model.

3.2.4 Illustrative example

This example is similar to the previous example in section 3.1.6, however, there is another reactor in the reaction stage as shown in Figure 3.12. The SSN remains the same and is shown here in Figure 3.13. The data for this example is shown in Tables 3.5 and 3.6.

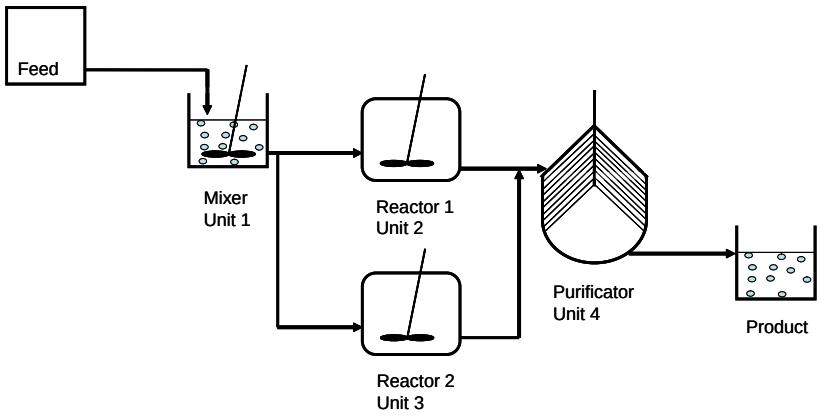

Figure 3.12: Flowsheet for the literature example

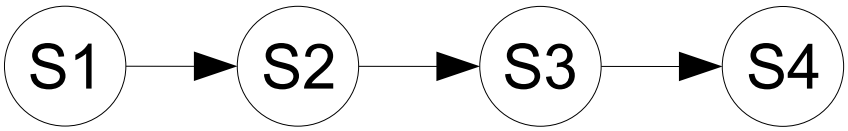

Figure 3.13: SSN for the literature example

Table 3.5: Design data for illustrative example

Unit	Capacity range	Suitability	Processing time (h)	Capital Cost
1	25 - 100	Mixing	4.5	$V^{0.68}$
2, 3	25 - 75	Reaction	3	$V^{0.6}$
4	25 - 50	Purification	1.5	$V^{0.7}$

Results

The model was solved using GAMS DICOPT, with CLPEX as the MIP solver and CONOPT as the NLP solver. The computational results are shown in Table 3.7. The resulting plant requires only one reactor as shown in Figure 3.14. The optimal capacites of the remaining units are 75 units for the mixer (U_1), 75 units for the reactor (U_2) and 37.5 units for the purificator (U_3). The resulting schedule for the optimal plant is shown in Figure 3.15, where the numbers above the bars are the amount of each state processed and the dotted lines represent the storage of a state in a process unit.

3.3 Conclusions

MILP and MINLP models are developed to take into account the PIS operational policy for testing and design, respectively. The MILP model is used to determine the effectiveness of the PIS operational policy by, firstly, solving the model with zero intermediate storage with and without the use of latent storage. In the illustrative example a 50% in-

Table 3.6: Design data for illustrative example

State	Storage capacity	Initial amount	Price
1	Unlimited	Unlimited	0.0
2	0	0.0	0.0
3	0	0.0	0.0
4	Unlimited	0.0	1.0

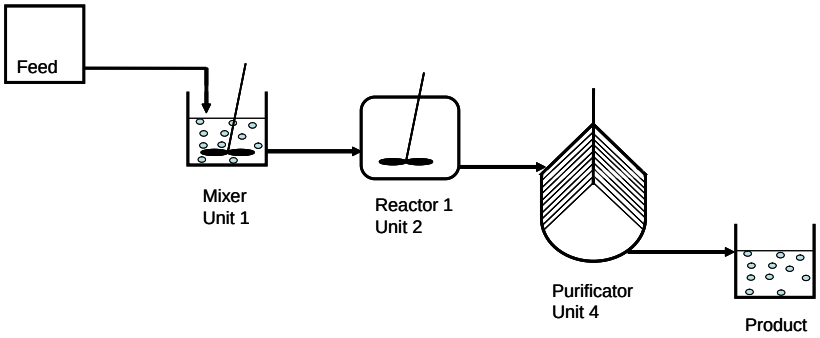

Figure 3.14: Resulting design from the model

Table 3.7: Computational results of the design literature example

Model property Model	Results
Number of time points	11
Number of constraints	3864
Number of variables	1616
Number of binary variables	132
MINLP solution	44.82
CPU time (s)	35.858
Number of major iterations	3

crease in the throughput was achieved. Secondly, the minimum amount of interemediate storage is determined with and without the PIS operational policy In both cases the production goal was set to that which was achieved when the model was solved with infinite intermediate storage. In the illustrative example a 33.33 % reduction in the amount of intermediate storage is achieved. The design model is an MINLP model due to the nonlinear capital cost objective function. This model is applied to an illustrative problem and results in the flowsheet as well as determining the capacities of the required units.

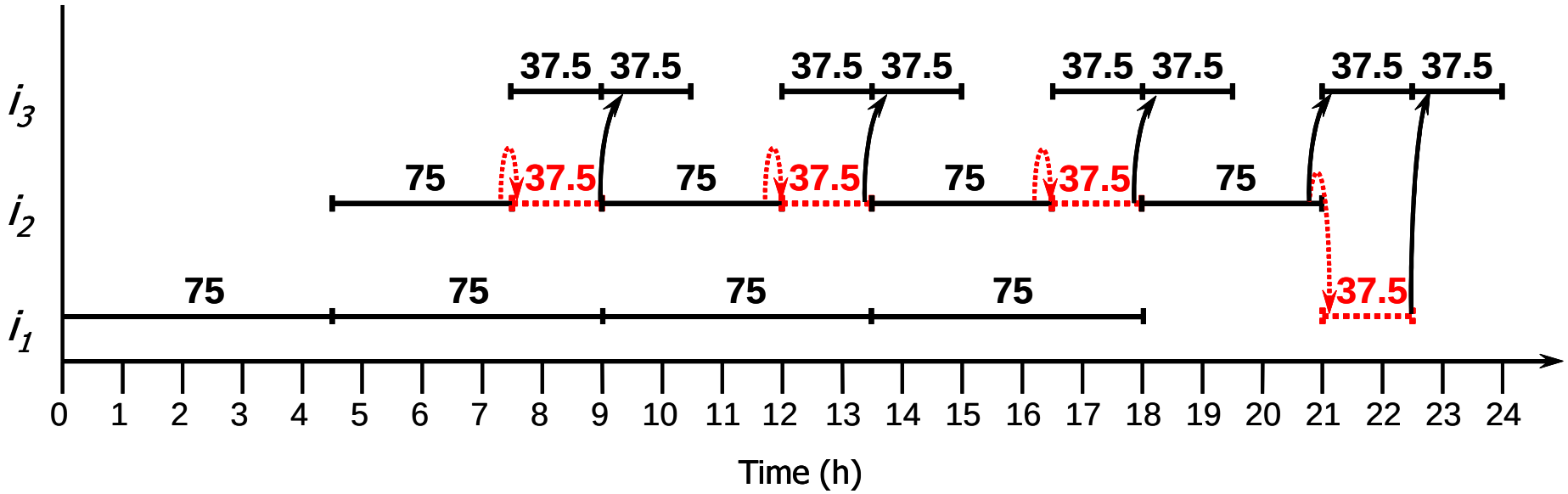

Figure 3.15: Schedule for the optimal design

CHAPTER 4

Industrial Application

The industrial case study presented in this section is taken from the petrochemicals industry. The project is in the design phase and as such the design model will be used to determine the design which leads to the minimum capital cost, while using the PIS operational policy. For secrecy reasons the example has been modified and the names of the raw materials and products have been changed to the generic form.

The flowsheet for the industrial case study is shown in Figure 4.1, this case study is used to illustrate the application of the design model. The SSN for the case study is shown in Figure 4.2.

The process has four stages, separated in Figure 4.1 by dashed lines. The first stage involves the reaction between raw 1 and raw 2. This reaction can take place in either of the two reactors (R1_1 and R1_2) in stage 1. The intermediate produced in this reaction is then transfered to either of the four reactors in the second step (R2_1, R2_2, R2_3 and R2_4) where a further reactant, raw 3 is added as well as the solvent. In this stage the hot solvent is added to the reactor, so parts of the reaction mixture from the previous

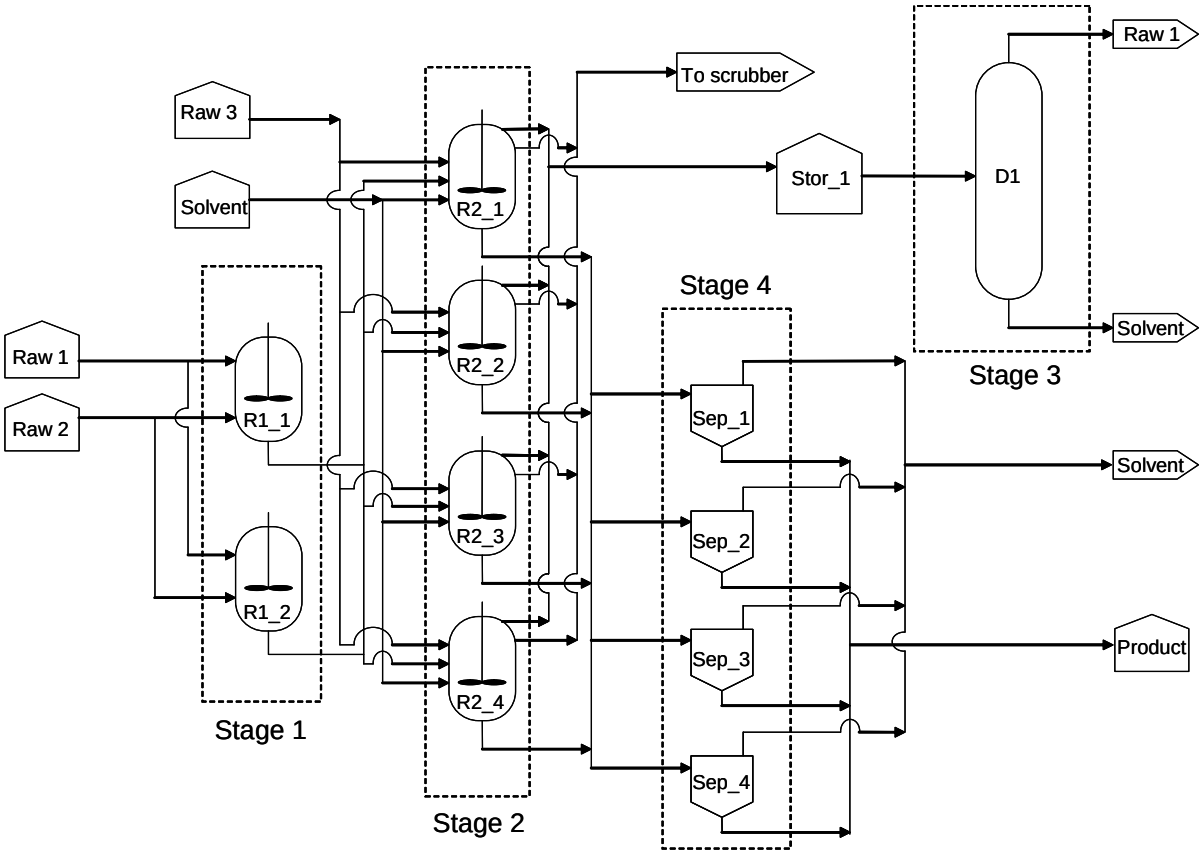

Figure 4.1: Flowsheet for the industrial case study

reaction are flashed off and sent to the scrubber. After 3 hours of drying raw 3, is added and the reaction proceeds. During the reaction, parts of the reaction mixture are vented and transported to storage tank, Stor_1, before distillation in unit D1. In the distillation stage the raw material, raw 1, is separated from the solvent. Following the separation both the raw 1 and the solvent are recycled back to storage for reuse. The remaining reaction mixture is then transfered to either of the four settlers (Sep_1, Sep_2, Sep_3 and Sep_4) where the product, prod, is separated from the solvent, solv. The solvent is then recycled back to the solvent storage tank to be reused. All of the units in stages 1, 2 and 4 can be used for latent storage, while, the distillation column, unit D1, in stage 4 cannot be used as storage for contamination reasons. Furthermore, only intermediate states can make use of latent storage. The time horizon of interest for the case study is 48 hours for illustrative purposes. The production goal for the plant was 109.9t of product.

The data for the case study is shown in Tables 4.1, 4.2, 4.3 and 4.4.

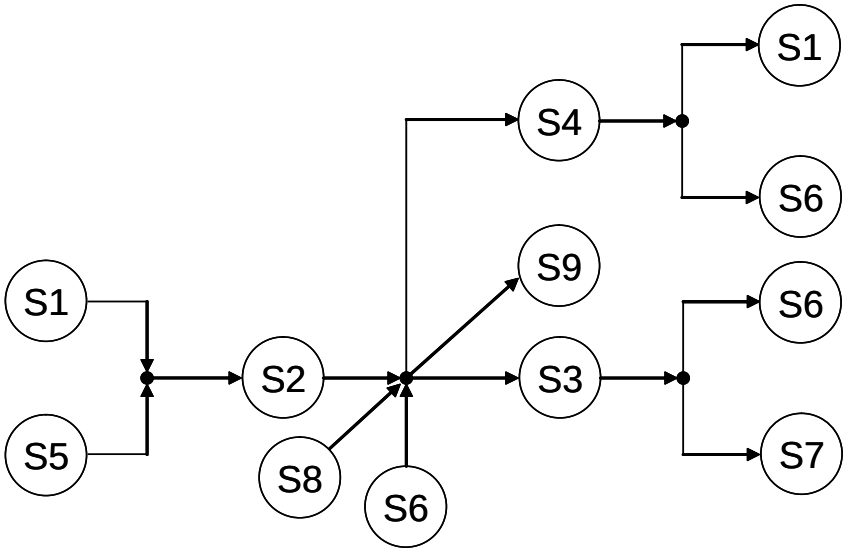

Figure 4.2: SSN for the industrial case study

Table 4.1: Data for industrial case study

Unit	Max. Capacity (t)	Suitability	Processing time (h)
R1_1, R1_2	25	Reaction 1	5
R2_1, R2_2, R2_3 and R2_4	83	Reaction 2	8
D1	100	Purification	1
Sep_1, Sep_2, Sep_3 and Sep_4	41	Separation	4

Table 4.2: Capital cost data for industrial case study

Unit	Capital cost
R1_1, R1_2	$89.1(c(j)/\mathrm{V}_j^{max})^{0.6}$
R2_1, R2_2, R2_3 and R2_4	$169.5(c(j)/\mathrm{V}_j^{max})^{0.6}$
D1	$41.4(c(j)/\mathrm{V}_j^{max})^{0.6}$
Sep_1, Sep_2, Sep_3 and Sep_4	$109(c(j)/\mathrm{V}_j^{max})^{0.6}$

Table 4.3: Storage and initial amount of state for the case study

State	Description	Storage capacity	Initial amount
S1	Raw 1	500	400
S2	Stage 2 feed	0	0.0
S3	Stage 4 feed	0	0.0
S4	Stage 3 feed	25	0.0
S5	Raw 2	400	400
S6	Solvent	1000	100
S7	Product	600	0.0
S8	Raw 3	Unlimited	Unlimited
S9	Vent to scrubber	Unlimited	0.0

Table 4.4: Feed and output ratios

State	units	ton/ton
S1/S5	R1_1, R1_2	0.9
S2/S8	R2_1, R2_2, R2_3 and R2_4	7
S2/S6	R2_1, R2_2, R2_3 and R2_4	0.5
S3/S9	R2_1, R2_2, R2_3 and R2_4	4
S3/S4	R2_1, R2_2, R2_3 and R2_4	15
S6/S7	Sep_1, Sep_2, Sep_3 and Sep_4	3.5
S1/S6	D1	0.02

4.1 Computational results

The model was solved on an Intel Core 2 CPU, T7200 2GHz processor with 1 GB of RAM. The computational results are shown in Table 4.5. The model was solved using GAMS DICOPT using CONOPT as the NLP solver and CPLEX as the MIP solver. The resultant flowsheet is shown in Figure 4.3, as can be seen from this flowsheet the design calls for fewer units in stages 2 and 4. The schedule is shown in Figure 4.4. As can be seen from the schedule latent storage is utilised during the time horizon of interest and the there are no cycles in this schedule. Table 4.6 details the required unit capacities, which are lower than the original design.

Table 4.5: Results from the industrial case study

Model property Model	Results
Number of time points	8
Objective function value	1727.9
Number of binary variables	228
Solution time (CPU s)	12082
Number of variables	4777
Number of constraints	8183
Number of major iterations	3

Table 4.6: Unit capacity results from the industrial case study

Unit	Capacity
R1_1, R1_2	25
R2_2, R2_3 and R2_4	75.138
Sep_1, Sep_2 and Sep_3	32.700
D1	1.54

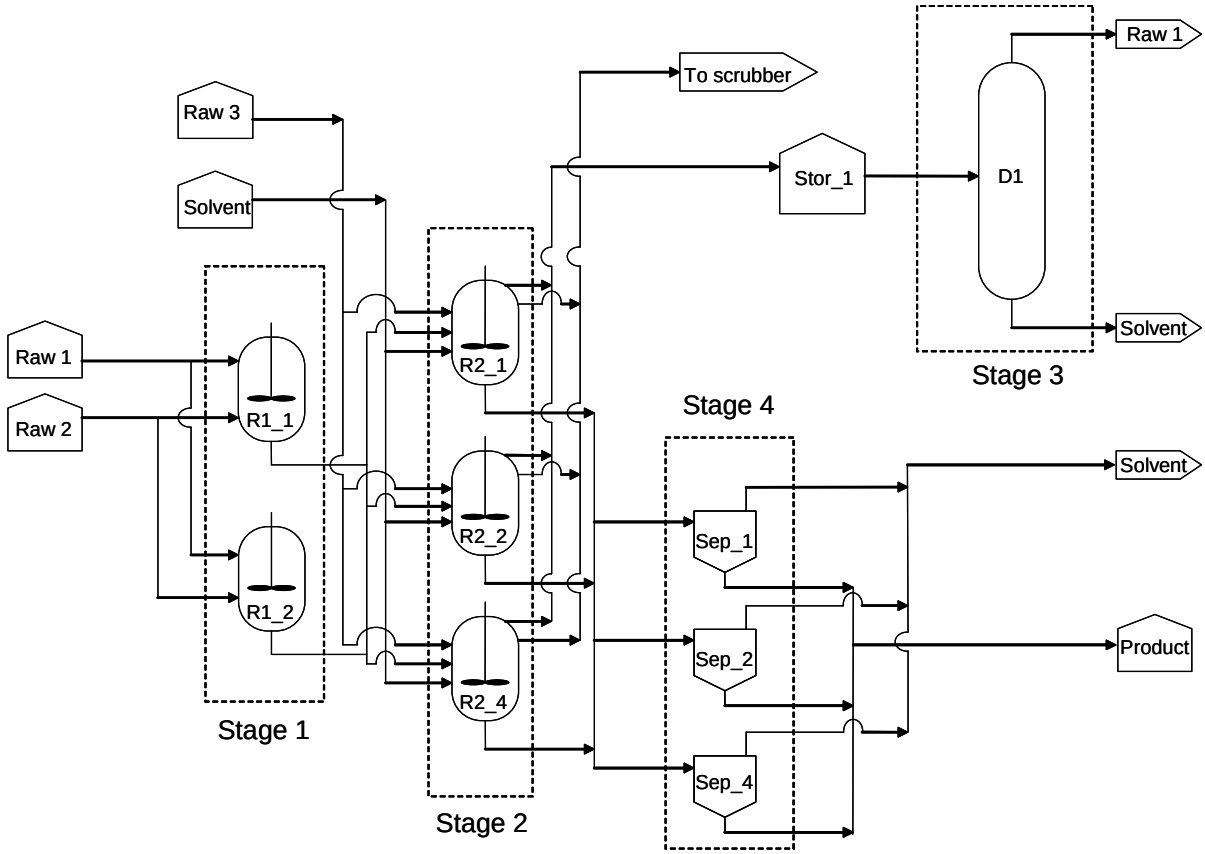

Figure 4.3: Resultant flowsheet for the industrial case study

4.2 Conclusions

The model is successfully applied to the case study resulting in fewer units required to meet the demand. Furthermore, the units that are required have a lower capacity than the original design called for. The model also makes effective use of the latent storage available during the time horizon of interest. It is clear from Table 4.5 that the solution time for this case study is long. The reasons for this are that there is a large degree of complexity due to the possible use of latent storage. The use of latent storage also has a significant contribution on the overall number of binary variables, which can lead to increases in solution times. Due to the nonlinearity of the objective function global optimality is not assured.

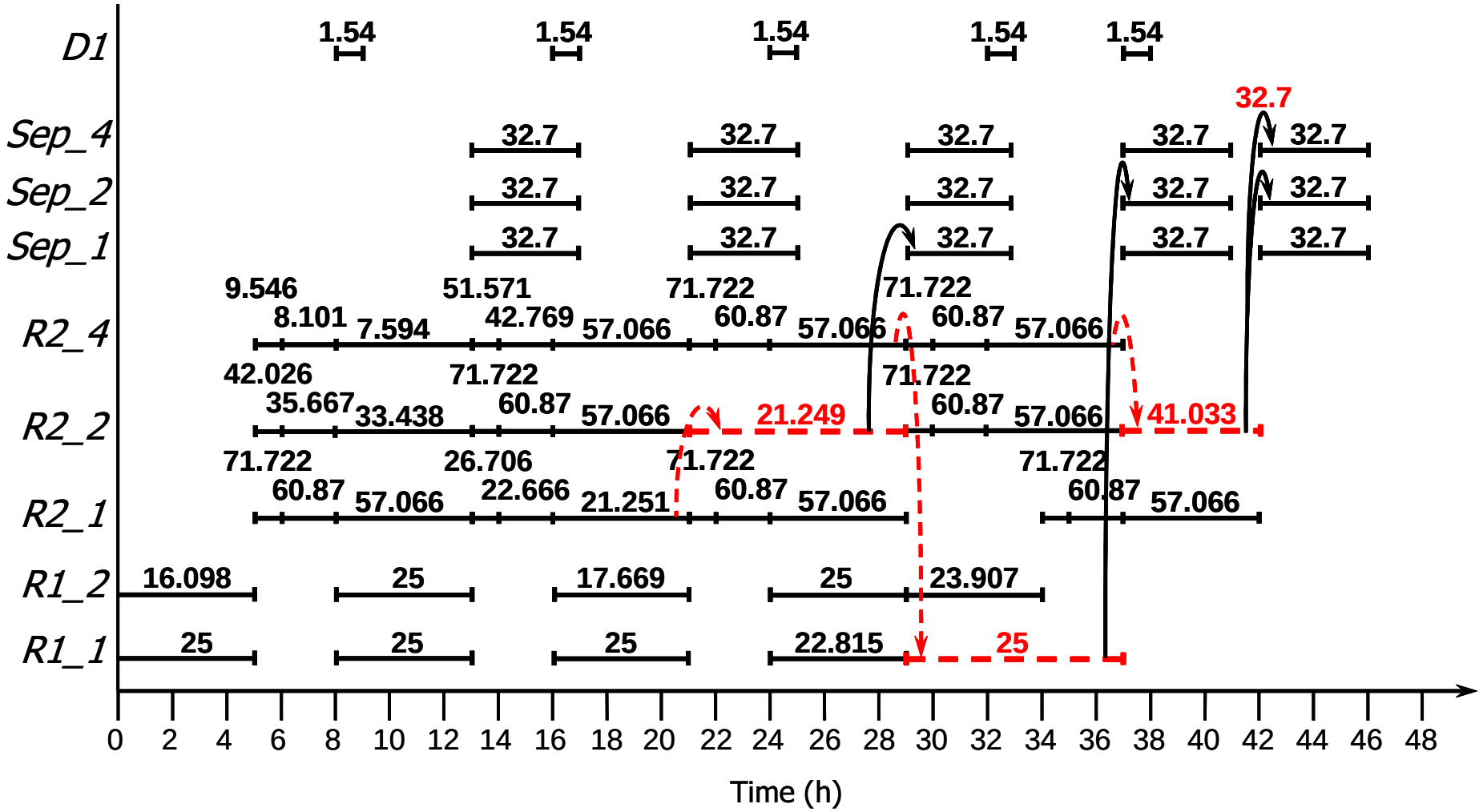

Figure 4.4: Schedule for the optimal design

CHAPTER 5

Discussion and Conclusions

The literature survey was conducted to determine the best model to use as a basis for the PIS operational policy, while the methodology development was focused on the development of the model and its application to literature examples. Using the design model developed in the methodology development the industrial case study was solved. This section details the findings from all of three elements of the adopted approach.

From the literature survey, i.e. Chapter 2, it was determined that the best framework and model to base the models on, was that of Majozi & Zhu (2001). The reasons for this are that, firstly, the models that exploit the structure of the SSN result in fewer binary variables than those derived from other mathematical methods, because the SSN only takes states into account while tasks are implicitly incorporated. Secondly, the model developed by Majozi & Zhu (2001) is based on the non-uniform discretization of the time horizon, thus resulting in fewer binary variables. Thirdly, the method is a mathematical programming, implying that it would be able to handle tasks of varying duration, such as the use of latent storage. Fourthly, the model developed by Majozi & Zhu (2001) is a

MILP, thus solutions from this model are globally optimal. From this starting point the model was developed as detailed in the methodology section, i.e. Chapter 3.

Two distinctive models were developed in order to investigate the effectiveness of PIS operational policy. The first model is separated into two parts. The first part is used to determine the optimal throughput when there is zero intermediate storage available. Two situations were studied, firstly the model was solved without the use of the PIS operational policy. Secondly, the model was solved with the PIS operational policy. In the simple example shown in this section a 50% increase in the throughput was achieved when the PIS operational policy was used. In both cases the models developed were a MILP, thus garenteeing global optimality.

The second part of the first model was used to determine the minimum amount of intermediate storage required to achieve the same throughput achieved when there is infinite storage available. This part required a three step algorithm. Firstly, the optimal throughput was determined where there was infinite storage available. Secondly, the model was solved with the production goal set to that achieved in the first step and then the model was solved with the objective of minimizing the amount of intermediate storage without the use of the PIS operational policy. The third step of the algorithm is similar to the second, however, the PIS operational policy is used. In the literature example an optimal throughput of 350 units was achieved in the first step. Using this as the production goal a 33.33% reduction in the amount of intermediate storage required was achieved when the model was solved without the PIS operational policy compared to the case with the PIS operational policy. Once again the models derived were MILP models, thus guaranteeing global optimality.

The second model developed in the methodology development section, presented in Chapter 3, is a design model based on the PIS operational policy. The model developed in this section is a MINLP model due to the capital cost objective function. The model is applied to a literature example and an improved design is achieved when compared to the flowsheet. The design model is then applied to an industrial case study to determine its effectiveness.

The industrial case study used in the industrial application section of Chapter 4 comes from the petrochemicals industry. The data for the case study are subject to a secrecy agreement and as such the names and details of the case study are altered.

The model is successfully applied to the case study resulting in lower capacity and fewer units than the original design while achieving the required production goal. The model also makes effective use of the latent storage available during the time horizon of interest. However, the solution time for this case study is long, due to the large number of binary variables and a complex non-linear objective function which leads to a high degree of complexity. Furthermore, the possible use of latent storage results in a large number of possible combinations which also contribute to the overall complexity of the model.

CHAPTER 6

Recommendations

Through the development of the model a number of potential improvements were identified. This section details these improvements and the potential directions for the application of these improvements. Further changes are suggested to the model to remove the possibility of filling and emptying a unit at the same time, while also proposing a new algorithm for the reduction of the number of transfers, where cross contamination is a major concern.

6.1 Cycling

The model developed in this thesis does not preclude the possibility of cycles occurring in the schedules, as shown in Figure 6.1. As can be seen the unit is emptying and filling at the same time which is physically impossible. This is a current problem in the scheduling of batch processes, where only the S-graph approach (Sanmartí *et al.*, 1998) has been proven to preclude these cycles from a schedule. However, S-graph cannot readily handle tasks with variable duration, such as latent storage.

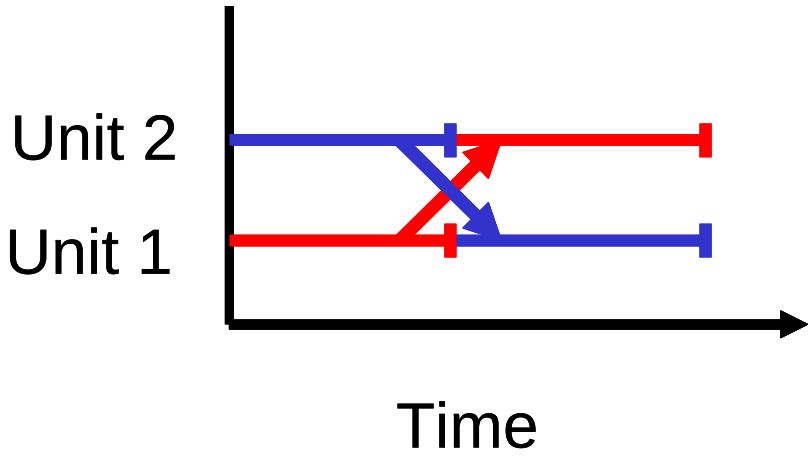

Figure 6.1: Cycling

In the methodology development chapter (section 3.1.6) a cycle appeared in Figure 3.8, however, this cycle can be removed through an iterative solution method where the cycles are removed manually. In this particular case the transfer of mass from the mixer to the purificator for storage at the particular time point is prevented. The resulting schedule is shown in Figure 6.2. Although on this occasion this method worked, this will not always be the case. In some case preventing the transfer of mass to latent storage will effect the resulting objective function. If this is the case global optimality can no longer be assured, in fact, very little is known about the resulting solution apart from its feasibility. This method is a manual and iterative method and is, therefore, not general.

In order to remove the possibility of cycles from the model, tracing of the mass is required. In the current model all mass that exits a unit, and moves for processing, enters a dedicated intermediate storage unit, even if this is for a zero duration, as shown by constraint 3.18. One has to note that when storage occurs all of the same state enters the same storage vessel regardless of where it was produced, this is shown by constraint 3.34. As such, when a state leaves a unit for processing the information about its source is lost. It is similar to adding drops of water to a glass, once in the glass its source cannot be identified, it is just a glass of water.

The reason that tracking of mass is so important to the removal of cycles is that once the mass can be tracked a binary variable can be initialised as shown by constraint 6.1.

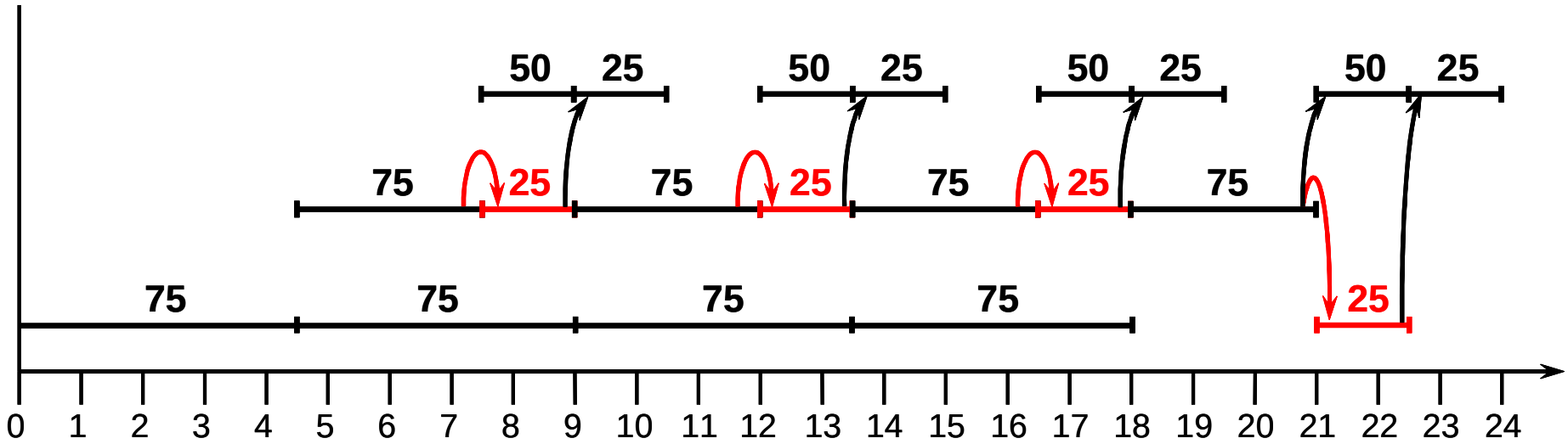

Figure 6.2: Literature example using the PIS operational policy without cycles

This binary variable simply states that mass has moved from unit j to unit j' at time point p. Similar constraints can be applied to the case of latent storage, however, existing variables can be used to achieve this, as shown by constraints 6.3 and 6.4. Using this binary variable it is possible to remove the possibility of moving from j to j' at the same time point as moving from j' to j, thus removing cycles as shown in constraint 6.2.

$$m_{in}(s,j,j',p) \geq V_{j'}^{min} x_{tr}(j,j',p) \qquad \forall\, j,j' \in J,\ p \in P \tag{6.1}$$

$$x_{tr}(j,j',p) + x_{tr}(j',j,p) \leq 1 \qquad \forall\, j,j' \in J,\ p \in P \tag{6.2}$$

$$m_{in}^{lt}(s,j,j',p) \geq V_{j'}^{min} x_{tr}(j,j',p) \qquad \forall\, j,j' \in J,\ p \in P \tag{6.3}$$

$$m_{out}^{lt}(s,j,j',p) \geq V_{j'}^{min} x_{tr}(j,j',p) \qquad \forall\, j,j' \in J,\ p \in P \tag{6.4}$$

6.1.1 Minimising the number of transfers

By defining a transfer binary variable the practical concern of the number of times a unit is used for storage or processing can be addressed without affecting the throughput. This can be done by using a two step algorithm. Firstly, the model is solved using the PIS operational policy to determine the optimal throughput. Secondly, the production goal is set to that which was achieved in the first step and setting the objective function to minimise the number of transfers as shown by equation 6.5.

$$min \sum_{p} \sum_{j} \sum_{j'} x_{tr}(j,j',p) \tag{6.5}$$

6.2 Complexity of the model

Using the PIS operational policy can lead to very complex models which, in general, lead to longer solution times. The reason for this complexity is due to the possible number of interactions for each unit, as shown in Figure 6.3. Due to this complexity the solution times for large scale problems can be long, hence the model was based on the SSN and non-uniform discretization of the time horizon model developed by Majozi & Zhu (2001). However, this model is a short-term scheduling model, therefore in cases where the time horizon of interest is long the model may become intractable. It is therefore recommended that in the case of medium and long-term scheduling that the model be redeveloped to test these cases.

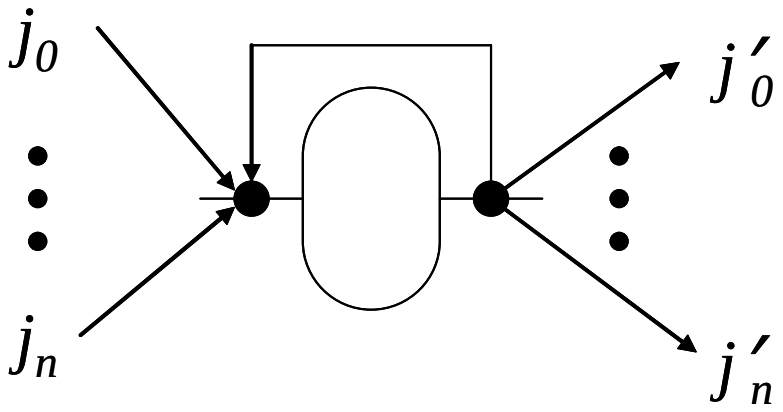

Figure 6.3: Complexity of unit interactions

BIBLIOGRAPHY

Cott, B. J. and Macchietto, S. (1989)

"Minimizing the effects of batch process variability using online schedule modification", *Computers and Chemical Engineering, 13* (1/2), 150–113.

Glover, F. December (1975)

"Improved linear integer programming formulation of nonlinear problems", *Management Science, 22* (4), 455–460.

Grossmann, I. and Sargent, W. (1979)

"Optimum design of multipurpose chemical plants", *Industrial and Engineering Chemistry Process Design Development, 18* (2), 343–348.

Harjunkoski, I. and Grossmann, I. E. (2002)

"Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods", *Computers and Chemical Engineering, 26*, 1533–1552.

Huang, W. and Chen, B. (2005)

"Scheduling of batch plants: Constraint-based approach and performance investigation", *International Journal Production Economics, Article in press.*

Huang, W. and Chung, P. W. H. (2000)

"Scheduling of pipeless batch plants using constraint satisfaction techniques",

Computers and Chemical Engineering, 24, 377–383.

Huang, W. and Chung, P. W. H. (2005)

"Integrating routing and scheduling for pipeless batch plants",

Computers and Chemical Engineering, 29, 1069–1081.

Ierapetritou, M. and Floudas, C. (1998)

"Effective continuous-time formulation for short-term scheduling. 1. Multipurpose batch processes",

Industrial and Engineering Chemistry Research, 37 (11), 3037–3051.

Jain, V. and Grossmann, I. E. (2001)

"Algorithms for hybrid MILP/CP models for a class of optimization problems",

INFORMS Journal on Computing, 13 (4), 258–276.

Kanakamedala, K. B.; Reklaitis, G. V. and Venkatasubramanian, V. (1994)

"Reactive schedule modification in multipurpose batch chemical plants",

Industrial and Engineering Chemistry Research, 33 (1), 77–90.

Karimi, I. A. and Reklaitis, G. V. (1985)a

"Deterministic variability analysis for intermediate storage in noncontinuous processes, Part I: Allowability conditions",

American Institute of Chemical Engineers Journal, 31 (9), 1516–1527.

Karimi, I. A. and Reklaitis, G. V. (1985)b

"Deterministic variability analysis for intermediate storage in noncontinuous processes, Part II: Storage sizing for serial systems",

American Institute of Chemical Engineers Journal, 31 (9), 1528–1537.

Kondili, E.; Pantelides, C. and Sargent, W. (1993)

"A general algorithm for short-term scheduling of batch operations - I. MILP formulation",

Computers and Chemical Engineering, 17 (2), 211–227.

Ku, H. M. and Karimi, I. A. (1990)

"Completion time algorithms for serial multiproduct batch processes with shared storage",

Computers and Chemical Engineering, 14 (1), 49–69.

Majozi, T. and Friedler, F. (2006)

"Maximization of throughput in a multipurpose batch plant under a fixed time horizon: S-graph approach",

Industrial and Engineering Chemistry Research, 45 (20), 6713–6720.

Majozi, T. and Zhu, X. (2001)

"A novel continuous-time MILP formulation for multipurpose batch plants.1. Short-Term Scheduling",

Industrial and Engineering Chemistry Research, 40 (23), 5935–5949.

Maravelias, C. and Grossmann, I. E. (2004)

"A hybrid MILP/CP decomposition approach for the continuous time scheduling of multipurpose batch plants",

Computers and Chemical Engineering, 28, 1921–1949.

Méndez and, C. A. and Cerdá (2003)

"Dynamic scheduling in multiproduct batch plants",

Computers and Chemical Engineering, 27, 1247–1259.

Méndez and, C. A. and Cerdá (2004)

"An MILP framework for batch reactive scheduling with limited resources",

Computers and Chemical Engineering, 28, 1059–1068.

Niwa, T. (1993)

"Pipeless plants boost batch processing",

Chemical Engineering, 100 (6), 102–108.

Pantelides, C. C.; Realff, M. J. and Shah, N. (1995)

"Short–term scheduling of pipeless batch plants",

Chemical Engineering Research and Development, 73 (A4), 431–444.

Ravemark, D. and Rippin, W. (1998)

"Optimal design of a multi-product batch plant",

Computers and Chemical Engineering, 22 (1), 177–183.

Realff, M. J.; Shah, N. and Pantelides, C. C. (1996)

"Simultaneous design, layout and scheduling of pipeless batch plants",

Computers and Chemical Engineering, 20 (6/7), 869–883.

Roe, B.; Papageorgiou, L. G. and Shah, N. (2005)

"A hybrid MILP/CLP algorithm for multipurpose batch process scheduling",

Computers and Chemical Engineering, 29, 1277–1291.

Roslöf, J.; Harjunkoski, I.; Björkqvist, J.; Karlsson, S. and Westlund, T. (2001)

"An MILP–based reordering algorithm for complex industrial scheduling and rescheduling",

Computers and Chemical Engineering, 25, 821–828.

Sanmartí, E.; Espuña, A. and Puigjaner, L. (1997)

"Batch production and preventative maintenance scheduling under uncertainty",

Computers and Chemical Engineering, 21 (10), 1157–1168.

Sanmartí, E.; Friedler, F. and Puigjaner, L. (1998)

"Combinatorial technique for short term scheduling of multipurpose batch plants based on the schedule-graph representation",

Computers and Chemical Engineering, 22 (Suppl.), S847–S850.

Schilling, G. and Pantelides, C. (1996)

"A simple continuous-time process scheduling formulation and novel solution algorithm",

Computers and Chemical Engineering, 20 (suppl.), S1221–S1226.

Shah, N.; Pantelides, C. and Sargent, R. (1993)

"A general algorithm for short-term scheduling of batch operations - II. Computational issues",

Computers and Chemical Engineering, 17 (2), 229–244.

Sparrow, R.; Forder, G. and Rippin, D. (1975)

"The choice of equipment sizes for multiproduct batch plants. Heuristics vs. Branch and Bound",

Industrial and Engineering Chemistry Process Design Development, 14 (3), 197–203.

Suhami, I. and Mah, R. (1982)

"Optimum design of multipurpose batch plants",

Industrial and Engineering Chemistry Process Design Development, 21 (1), 94–100.

Weide Jr., W. and Reklaitis, G. V. (1987)

"Determination of completion times for serial multiproduct processes– 3. Mixed intermediate storage",

Computers and Chemical Engineering, 11 (4), 357–368.

Zanetti, R. (1992)

"The 'pipeless' batch plant",

Chemical Engineering, 99 (6), 5.