Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

Introducing a new operational policy: The PIS operational policy

Thomas Pattinson[a], Thokozani Majozi[a,b,c,*]

[a] Department of Chemical Engineering, University of Pretoria, Lynnwood Road, Pretoria 0002, South Africa
[b] Department of Computer Science, University of Pannonia, Egyetem u. 10, Veszprém H-8200, Hungary
[c] Modelling and Digital Systems Division, CSIR, South Africa

ARTICLE INFO

Article history:
Received 2 October 2007
Received in revised form 15 July 2009
Accepted 16 July 2009
Available online xxx

Keywords:
Batch process
Scheduling
PIS policy
Design
Latent storage
MILP

ABSTRACT

A novel operational policy, the Process Intermediate Storage operational policy, is introduced and used to synthesize, schedule and design multipurpose batch plants. The model is based on the State Sequence Network and non-uniform discretization of the time horizon of interest model developed by Majozi and Zhu [Majozi, T., 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]. Two cases are studied to determine the effectiveness of this operational policy. In the first case, which excludes any dedicated storage, the use of this operational policy results in 50% improvement in throughput. The second case is used to determine the minimum amount of intermediate storage while maintaining the throughput achieved with infinite intermediate storage. This results in 20% reduction in the amount of dedicated intermediate storage. The models developed for both cases are MILP models. An MINLP design model is then developed to exploit the attributes of the PIS operational policy.

1. Introduction

1.1. 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 & Reklaitis, 1987) and,
- Central Intermediate Storage (CIS) (Jung, Lee, & Lee, 1996; 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 Fig. 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

* Corresponding author at: Department of Chemical Engineering, University of Pretoria, Lynnwood Road, Pretoria 0002, South Africa. Tel.: +27 12 420 4130; fax: +27 86 633 5729.
E-mail addresses: thoko.majozi@up.ac.za, majozi@dcs.vein.hu (T. Majozi).

0098-1354/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compchemeng.2009.07.008

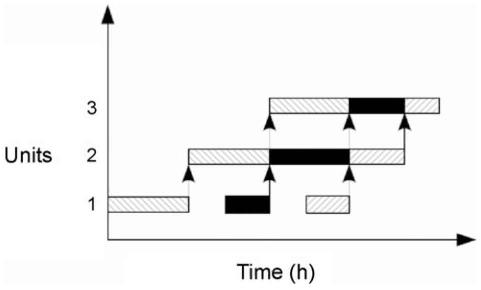

Fig. 1. Stair-step nature of schedules with the ZW operational policy.

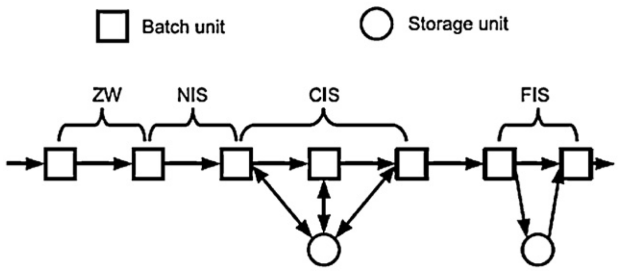

Fig. 2. A serial process with the MIS and CIS operational policies.

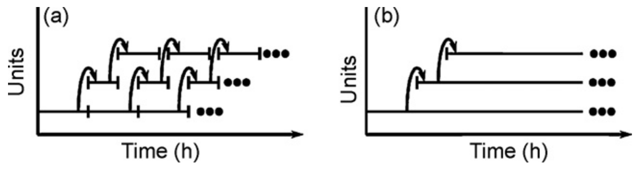

Fig. 3. Batch vs. continuous processing.

Weide and 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 Fig. 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 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 are clearly shown in Fig. 3. The discrete nature of batch processes can be clearly seen in Fig. 3a, where a task is processed in a unit and then once completed is transferred 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 Fig. 3b.

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, Forder, and Rippin (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, the 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 is 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 and 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 and 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 and 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, Pantelides, and Sargent (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 Fig. 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, Pantelides, & Sargent, 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.

Schilling and 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 Fig. 5. The formulation was derived from the Resource Task Network (RTN) representation. In this representation, the plant is modelled as resources and tasks. Resources

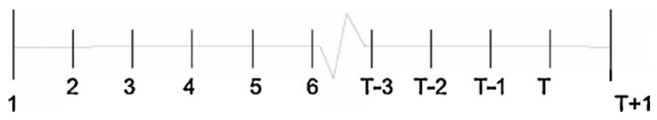

Fig. 4. Uniform discretization of the time horizon.

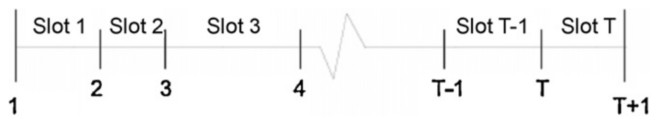

Fig. 5. Time points and slots used by Schilling and Pantelides (1996).

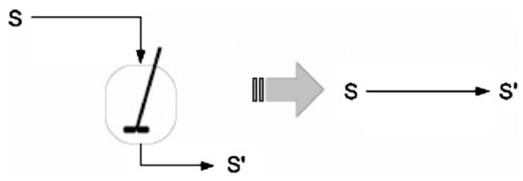

Fig. 6. Simple unit operation.

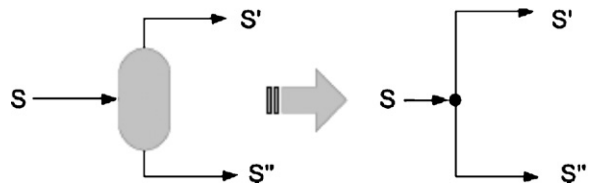

Fig. 7. Unit operation where states mix.

not only 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. Some of the detailed reviews on the state-of-the-art techniques in short-term scheduling have been presented by Mendez, Cerdá, Grossmann, Harjunkoski, and Fahl (2006) and Barbosa-Póvoa (2007).

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 and 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 and Zhu (2001) developed a scheduling representation and a continuous-time formulation, which gives the least number of binary variables and does not require simplification.

1.2. The State Sequence Network

Majozi and 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 Figs. 6–8. 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 and Pantelides (1996) and also used by Ierapetritou and Floudas (1998). The main difference between this model and the model proposed by Ierapetritou and 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.

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 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 Fig. 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 Fig. 9, where the SSN and STN are constructed from the given flowsheet, the attractiveness of the model proposed by Majozi and Zhu (2001) is illustrated. Applying the formulation of Ierapetritou and Floudas (1998) to the example in Fig. 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.

The above techniques 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 modelled. However, none of these operational policies exploit the latent storage found in

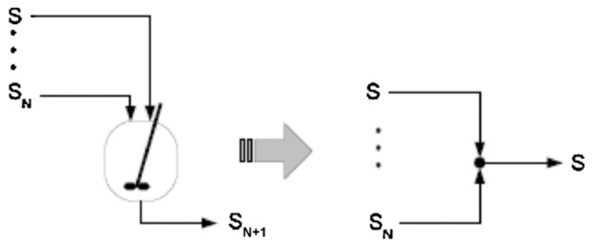

Fig. 8. Unit operation where states split.

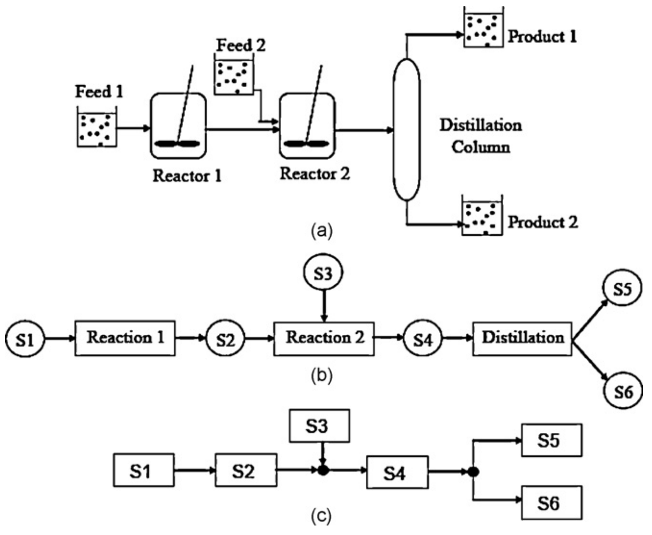

Fig. 9. (a) Flowsheet, (b) the STN and (c) SSN representation of (a).

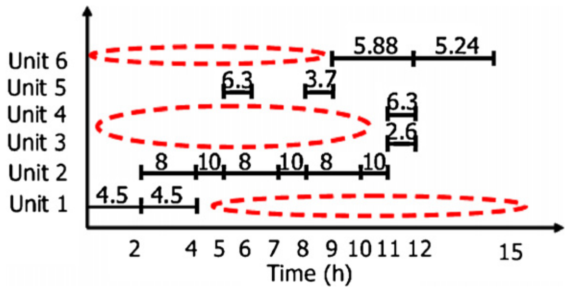

Fig. 10. General schedule.

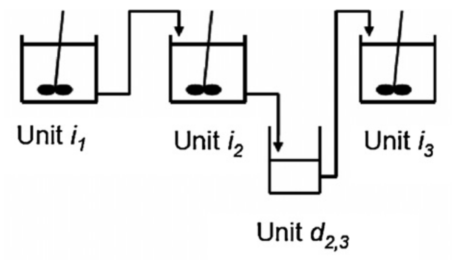

Fig. 11. Flowsheet for the simple example.

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.

2. The PIS operational policy

The models developed to take the PIS operational policy into account are detailed in this section. The models are based on the SSN and continuous-time model developed by Majozi and 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 and 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 paper 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 2 GHz processor with 1 GB of RAM, unless specifically stated. Worthy of mention, however, is the fact that the formulations based on other recipe representations like STN, RTN and mSTN would require modifications in order to be adapted to the PIS policy.

The PIS operational policy is novel and thus requires further explanation. When a batch operation is scheduled a Gantt chart is usually generated, such as in Fig. 10. From this figure it is simple to identify the latent storage potential of the units. For example, units 1, 3, 4 and 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.

Fig. 11 shows the flowsheet for the illustrative example. The data for the example are given in Table 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 h.

Table 1
Data for simple illustrative example.

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

When the PIS operational policy is not used, as in Fig. 12, 50 ton of dedicated intermediate storage unit, $d_{2,3}$, was required. The reason for this is that the capacity of the final unit is only 50 ton, due to this, the 100-ton batch produced from unit i_2 must be split into 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 100 ton. However, when compared to the schedule in Fig. 13, the 100-ton storage vessel is not needed. The reason for this is that 50 ton of the intermediate product produced from unit i_2 is moved to i_1 for storage, while the remaining 50 ton 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 100-ton storage vessel could have been saved.

In order to illustrate the uses of this novel operational policy, i.e. PIS operational policy, this paper 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.

2.1. Scheduling implications

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

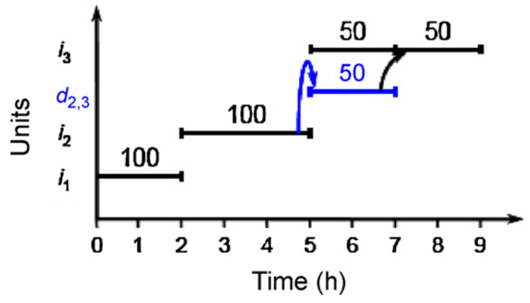

Fig. 12. Schedule of the simple example without using PIS.

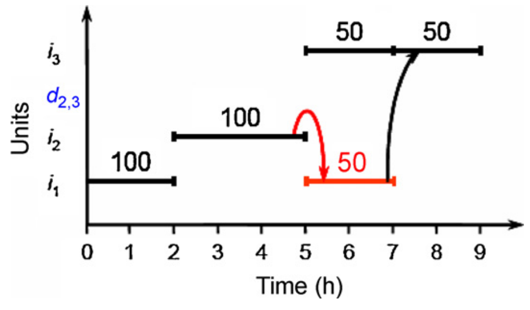

Fig. 13. Schedule of the simple example using PIS.

2.1.1. 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.

2.1.2. Mathematical model

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

Sets

P	$\{p \mid p = \text{time point}\}$
J	$\{j \mid j = \text{unit}\}$
S	$\{s \mid s = \text{is any state}\}$
S_{in}	$\{S_{in} \mid S_{in} = \text{is any input state}\}$
S_{out}	$\{S_{out} \mid S_{out} = \text{is any output state}\}$
S^*_{in}	$\{s^*_{in} \mid s^*_{in} = \text{effective state into unit}\} \subseteq S_{in}$

Binary variables

$y_{in}(s, j, p)$	decision variable describing the processing of states 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
$m^{lt}_{in}(s, j, j', p)$	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 p
$m^s_{out}(s, j, p)$	amount of state s fed from storage to unit j at time point p
$m^{lt}_{out}(s, j', j, p)$	amount of state s from latent store in j' fed to unit j at time point p
$c(j)$	capacity of unit j
$t_{out}(s, j, p)$	time at which state s is produced from unit j at time point p
$t_{in}(s, j, p)$	time at which state s is used in unit j at time point p
$t^{lt}_{in}(s, j, p)$	beginning of latent storage for state s in unit j at time point p
$t^{lt}_{out}(s, j, p)$	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

$V^{\min}_j$	minimum capacity of unit j
$V^{\max}_j$	maximum capacity of unit j
$Q^0_s(s)$	amount of state initially stored
$T(s)$	processing time of state s
$W^U(s, j)$	upper limit of the duration of the latent storage of states in unit j
$W^L(s, j)$	lower limit of the duration of the latent storage of states in unit j
H	time horizon of interest
A, B	unit-specific capital cost terms
α	power function for capital cost objective function
$Q^{\max}_s(s)$	maximum amount of state s stored within the time horizon of interest

2.1.3. Basic scheduling model

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

2.1.3.1. Capacity constraints.

$$V^{\min}_j y(s^*_{in}, j, p) \le \sum_{s \in S_{in}} m_{in}(s, j, p) \le V^{\max}_j y(s^*_{in}, j, p)$$
$$\forall\ s \in s_{in,j}, j \in J, p \in P \tag{1}$$

Constraint (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.

2.1.3.2. Material balances.

$$\sum_{s \in S_{in,j}} m_{in}(s, j, p-1) = \sum_{s \in S_{out,j}} m_{out}(s, j, p) \quad \forall\ j \in J, p \in P, p > 1 \tag{2}$$

$$q(s, p) = q(s, p-1) + \sum_{j \in J^{out}_s} m_{out}(s, j, p) - \sum_{j \in J^{in}_s} m_{in}(s, j, p)$$
$$\forall s \in S, S = intermediate, j \in J, p \in P, p > 1 \tag{3}$$

$$q(s, p) = q(s, p-1) + \sum_{j \in J^{in}_s} m_{in}(s, j, p)$$
$$\forall\ s \in S, S = \text{feed}, j \in J, p \in P, p > 1 \tag{4}$$

$$q(s, p) = q(s, p-1) + \sum_{j \in J^{out}_s} m_{out}(s, j, p) - d(s, p)$$
$$\forall s \in S, S = product, j \in J, p \in P, p > 1 \tag{5}$$

$$q(s, p_1) = Q^o_s(s) - \sum_{j \in J^{in}_s} m_{in}(s, j, p_1) \quad \forall s \in S, j \in J, p_1 \in P \tag{6}$$

The material balances are shown by constraints (2)–(6). Constraint (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) 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 (4) applies where mass is only used, such as with feed states. The amount of state delivered to the customer is determined by constraint (5), where a state is only produced not used. Constraint (6) is similar to constraint (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.

2.1.3.3. Duration constraint.

$$t_{out}(s_{out}, j, p) = t_{in}(s^*_{in}, j, p-1) + \tau(s) y(s^*_{in}, j, p-1)$$
$$\forall\ j \in J, p \in P, p > 1, s_{out} \in S_{out}, s_{in} \in S_{in} \quad (7)$$

The model is based on a fixed duration of tasks as shown in constraint (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. $p-1$.

2.1.3.4. Sequence constraints.

$$t_{in}(s^*_{in}, j, p) \geq \sum_{s_{in}} \sum_{s_{out}} \sum_{j \in J} \sum_{p' \in P} (t_{out}(s_{out}, j, p') - t_{in}(s_{in}, j, p-1))$$
$$\forall j \in J, p \in P, p > 1, s_{out} \in S_{out}, s_{in} \in S_{in} \quad (8)$$

$$t_{in}(s^*_{in}, j, p) \geq t_{out}(s_{out}, j, p) \quad \forall\ j \in J, p \in P, s_{out} \in S_{out}, s_{in} \in S_{in} \quad (9)$$

$$t_{in}(s_{in}, j, p) \geq t_{out}(s_{out}, j', p)$$
$$\forall j, j' \in J, p \in P, s_{out} = s_{in}, s_{out} \in S_{out}, s_{in} \in S_{in} \quad (10)$$

Constraint (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 (9) ensures that the processing of state s_{in} into unit *j* can only take place after the previous batch has been processed. Constraint (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.

2.1.3.5. Feasibility constraints.

$$\sum_{s \in S^*_{in,j}} y(s, j, p) \leq 1 \quad \forall j \in J, p \in P \quad (11)$$

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

2.1.3.6. Time horizon constraints.

$$t_{in}(s, j, p) \leq H \quad \forall j \in J, p \in P, s \in S_{in,j} \quad (12)$$

$$t_{out}(s, j, p) \leq H \quad \forall j \in J, p \in P, s \in S_{out,j} \quad (13)$$

Constraints (12) and (13) ensure that all the tasks take place within the time horizon of interest.

2.1.3.7. Storage constraints.

$$q(s, p) \leq Q^{\max}_s(s) \quad \forall s \in S, p \in P \quad (14)$$

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

2.1.4. 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.

2.1.4.1. Capacity constraints.

$$V^{\min}_j y^{lt}(s, j, p) \leq \sum_{j' \in J} m^{lt}_{in}(s, j', j, p) \leq V^{\max}_j y^{lt}(s, j, p)$$
$$\forall s \in S, j \in J, p \in P \quad (15)$$

Constraint (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*.

2.1.4.2. Material balances.

$$\sum_{j' \in J} m^{lt}_{in}(s, j', j, p-1) = \sum_{j' \in J} m^{lt}_{out}(s, j, j', p) \quad \forall s \in S, j \in J, p \in P, p > 1 \quad (16)$$

$$m_{in}(s, j, p) = \sum_{j' \in J} m^{lt}_{out}(s, j', j, p) + m^{s}_{out}(s, j, p) \quad \forall s \in S, j \in J, p \in P \quad (17)$$

$$m_{out}(s, j, p) = \sum_{j' \in J} m^{lt}_{in}(s, j', j, p) + m^{s}_{in}(s, j, p) \quad \forall s \in S, j \in J, p \in P \quad (18)$$

The mass balance over a process unit which is being used as latent storage is given by constraint (16). It should be noted that the input and output states remain the same. Constraints (17) and (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 (17). Similarly, mass which exits a unit is either moved to dedicated storage or latent storage as stated in constraint (18).

2.1.4.3. Duration constraints.

$$t^{lt}_{out}(s, j, p) = t^{lt}_{in}(s, j, p-1) + w(s, j, p) \quad \forall s \in S, j \in J, p \in P \quad (19)$$

$$W^L(s, j) y^{lt}(s, j, p) \leq \omega(s, j, p) \leq W^U(s, j) y^{lt}(s, j, p) \quad \forall s \in S, j \in J, p \in P \quad (20)$$

The duration constraint for a latent storage, constraint (19), is similar to constraint (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 (20).

2.1.4.4. Sequence constraints.

$$t^{lt}_{in}(s, j, p) \geq t^{lt}_{out}(s', j, p) \quad \forall s, s' \in S, j \in J, p \in P \quad (21)$$

$$t^{lt}_{in}(s, j, p) \geq t_{out}(s', j, p) \quad \forall s, s' \in S, j \in J, p \in P \quad (22)$$

$$t_{in}(s, j, p) \geq t^{lt}_{out}(s', j, p) \quad \forall s, s' \in S, j \in J, p \in P \quad (23)$$

$$t^{lt}_{out}(s, j, p) \leq t_{in}(s, j', p) + H(2 - y^{lt}(s, j, p-1) - y(s, j, p))$$
$$\forall s \in S_{in,j'}, j, j' \in J, p \in P, p > 1 \quad (24)$$

$$t^{lt}_{out}(s, j, p) \geq t_{in}(s, j', p) - H(2 - y^{lt}(s, j, p-1) - y(s, j, p))$$
$$\forall s \in S_{in,j'}, j, j' \in J, p \in P, p > 1 \quad (25)$$

$$t_{in}^{lt}(s,j,p) \le t_{out}(s,j',p) + H(2 - y^{lt}(s,j,p) - y(s',j',p-1))$$
$$\forall s \in S_{out,j'}, s' \in S_{in,j'}, j, j' \in J, p \in P, p > 1 \quad (26)$$

$$t_{in}^{lt}(s,j,p) \ge t_{out}(s,j',p) - H(2 - y^{lt}(s,j,p) - y(s',j',p-1))$$
$$\forall s \in S_{out,j'}, s' \in S_{in,j'}, s \in S, j, j' \in J, p \in P, p > 1 \quad (27)$$

$$t_{in}(s,j',p) \ge t_{out}(s,j,p) - H(2 - y(s,j',p) - y(s',j',p-1))$$
$$\forall s \in S, j, j' \in J, p \in P, p > 1 \quad (28)$$

$$t_{in}(s,j',p) \le t_{out}(s,j,p) + H(2 - y(s,j',p) - y(s',j',p-1))$$
$$\forall s \in S, j, j' \in J, p \in P, p > 1 \quad (29)$$

Constraints (21)–(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 (24) and (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 (26) and (27) are similar to constraints (24) and (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 (28) and (29).

2.1.4.5. *Time horizon constraints.*

$$t_{in}^{lt}(s,j,p) \le H \quad \forall j \in J, p \in P, s \in S \quad (30)$$

$$t_{out}^{lt}(s,j,p) \le H \quad \forall j \in J, p \in P, s \in S \quad (31)$$

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

2.1.4.6. *Feasibility constraints.*

$$\sum_{s \in S_{in}} y^{lt}(s,j,p) + \sum_{s' \in S_{in}} y(s',j,p) \le 1 \quad \forall j \in J, p \in P \quad (32)$$

$$\sum_{j \in J} y^{lt}(s,j,p) \le 1 \quad \forall s \in S, p \in P \quad (33)$$

To ensure that a unit is only used for either processing or storage at a particular time point, constraint (32) is required. Constraint (33) ensures that a batch cannot be split. Constraint (11) is redundant in the presence of constraint (32).

2.1.5. *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 and Zhu (2001) has to be modified as follows.

$$q(s,p) = q(s,p-1) + \sum_{j \in J_s^{out}} m_{in}^s(s,j,p) - \sum_{j' \in J_s^{in}} m_{out}^s(s,j',p)$$
$$\forall s \in S, j \in J, p \in P \quad (34)$$

$$q(s,p) = q(s,p-1) - \sum_{j' \in J_s^{in}} m_{out}^s(s,j',p)$$
$$\forall s \in S, S = feed, j \in J, p \in P, p > 1 \quad (35)$$

$$q(s,p) = q(s,p-1) + \sum_{j \in J_s^{out}} m_{in}^s(s,j,p) - d(s,p)$$
$$\forall s \in S, S = product, j \in J, p \in P, p > 1 \quad (36)$$

$$q(s,p_1) = Q_s^o(s) - \sum_{j' \in J_s^{in}} m_{out}^s(s,j,p_1) \quad \forall s \in S, j \in J, p_1 \in P \quad (37)$$

The balance over a dedicated intermediate storage unit has to be modified because of the possibility of latent storage. Constraint (34) provides the link for the inlet and outlet mass balance between units, as shown in constraints (17) and (18). Constraints (35)–(37) are similar to constraints (4)–(6), however they apply to the case where the PIS operational policy is taken into account.

$$t_{in}(s_{in},j,p) \ge \sum_{s}\sum_{j}\sum_{p' \le P}[t_{out}(s_{out},j,p')$$
$$-t_{in}(s_{in},j,p'-1) + t_{out}^{lt}(s',j,p') - t_{in}^{lt}(s',p'-1)]$$
$$\forall j \in J, p \in P, p > 1, p' > 1, s_{out,j} \in S_{out,j}, s_{in,j} \in S_{in,j}, s' \in S \quad (38)$$

Constraint (8) has to be modified to include the possibility of using a unit as latent storage, as shown by constraint (38).

2.1.6. *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 (39)) when there is zero intermediate storage (constraint (40)), while in the second case the goal is the minimization of intermediate storage (constraint (43)) while maintaining the optimal throughput (constraint (44)). 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.

2.1.6.1. *Case 1.*

$$Z = \max \sum_{s \in S}\sum_{p \in P} d(s,p) \quad (39)$$

$$\text{while } q(s,p) = 0 \quad \forall s \in S, p \in P \quad (40)$$

2.1.6.2. *Case 2.* Step 1:

$$Z = \max \sum_{s \in S}\sum_{p \in P} d(s,p) \quad (41)$$

$$\text{while } q(s,p) > 0 \quad \forall s \in S, p \in P \quad (42)$$

Step 2:

$$Z = \min \sum_{s \in S}\sum_{p \in P} d(s,p) \quad (43)$$

$$\text{while } d(s,p) = \text{production goal} \quad \forall s \in S, p \in P \quad (44)$$

with the SSN representation shown in Fig. 15

2.2. *Illustrative example*

In order to illustrate these cases an example taken from the papers of Ierapetritou and Floudas (1998) and Majozi and Zhu (2001) will be used. The flowsheet for the example is given in Fig. 14. The data for the example are shown in Table 2, the time horizon of interest for this example has been altered from 12 h presented in

Table 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

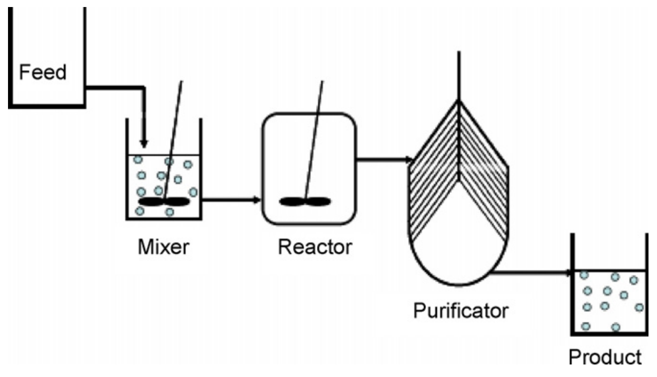

Fig. 14. Flowsheet for the literature example.

$$\begin{tikzcd} S1 \arrow[r] & S2 \arrow[r] & S3 \arrow[r] & S4 \end{tikzcd}$$

Fig. 15. SSN for the literature example.

Table 3
Results from the first case.

	Without PIS policy	With PIS policy
Number of time point	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

Ierapetritou and Floudas (1998) and Majozi and Zhu (2001) to 24 h for illustrative purposes.

2.2.1. 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 Fig. 16. The stair-step nature of the schedule is expected due to the NIS operational policy. When the PIS operational policy is introduced, i.e. addition of constraints (15)–(38) to constraints (1)–(14), the optimal throughput increases from 200 to 300 units as shown in Fig. 17.

The way this improvement is achieved is clearly seen in Fig. 17. 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 h, 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.

The model was solved using GAMS and the CPLEX solver version 9.1.2. The computational results for case 1 are shown in Table 3.

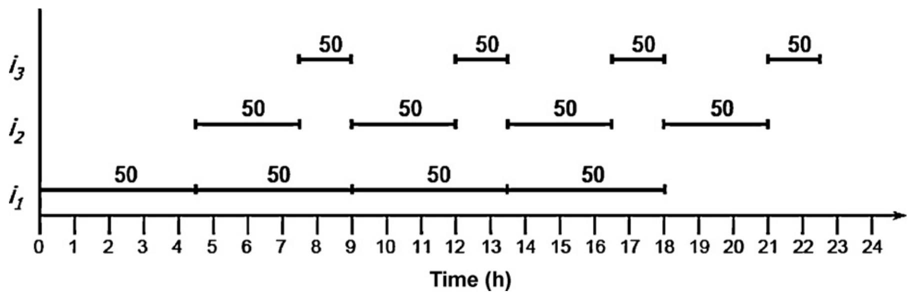

Fig. 16. Literature example without using the PIS operational policy.

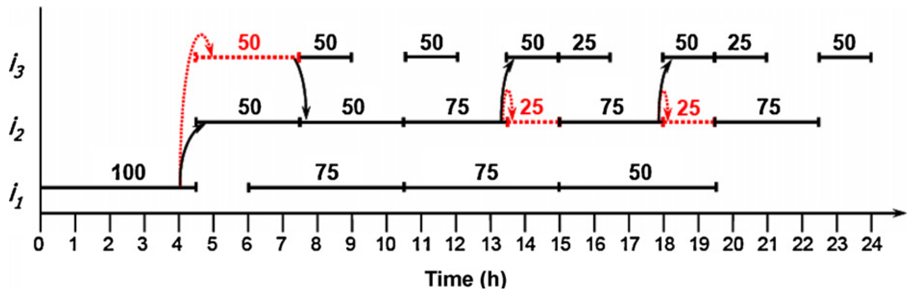

Fig. 17. Literature example using the PIS operational policy.

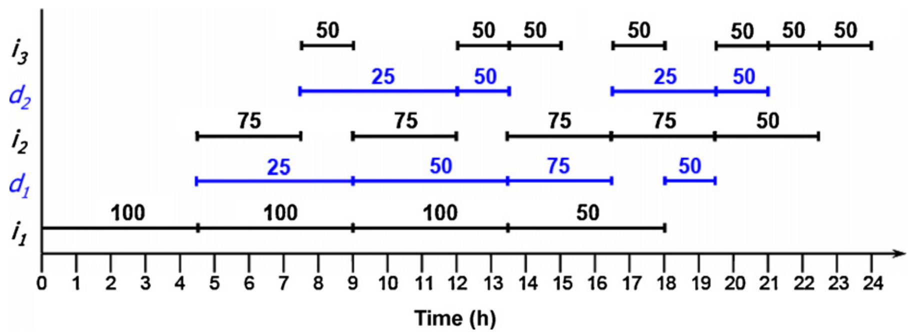

Fig. 18. Literature example with infinite intermediate storage.

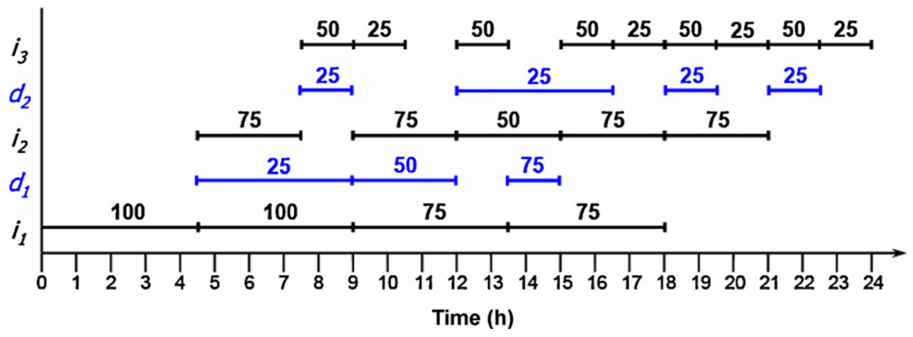

Fig. 19. Literature example without using the PIS operational policy.

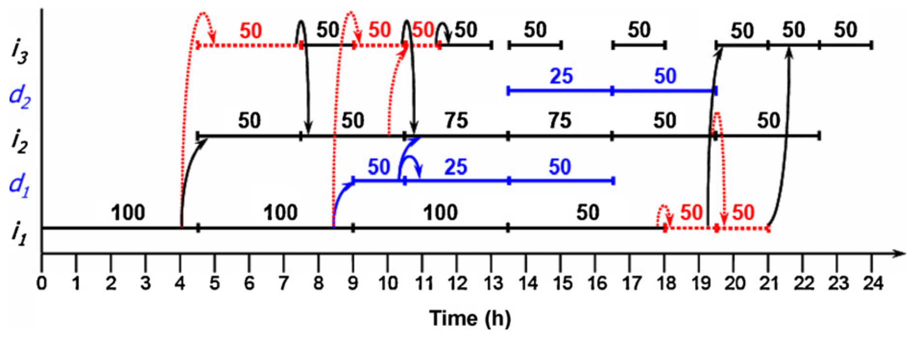

Fig. 20. Literature example using the PIS operational policy.

From these results it is clear to see the potential benefits for using the PIS operational policy, with a 50% increase in throughput.

2.2.2. *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 Fig. 18. Without using the PIS operational policy the minimum amount of intermediate storage required was 250 units. The schedule for this case is shown in Fig. 19. 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 Fig. 20.

The model was solved on an Intel Core 2 CPU, T7200 2 GHz 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 4.

Table 4
Results from the second case.

	Infinite storage	Without PIS	With PIS
Number of time point	10	13	11
Objective function value	350[a]	250[b]	200[c]
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

[a] Maximum throughput.
[b] Minimum storage without PIS.
[c] Minimum storage with PIS.

2.3. *Design implications*

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

2.3.1. *Problem statement*

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

Given:

Table 5
Design data for illustrative example.

Unit	Capacity range	Suitability	Processing time	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	$V^{0.7}$

(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.

2.3.2. Necessary modifications to case 1

Constraints (1), (2), (7), (9), (10), (12), (13) and (15)–(31) and (34)–(38) still apply to the model, however, further modifications of the model in Section 2.1 are required to take into account the possibility of design.

2.3.2.1. Capacity constraints.

$$V_j^{\min} e(j) \le c(j) \le V_j^{\max} e(j) \quad \forall j \in J, p \in P \tag{45}$$

$$m_{in}^{lt}(s, j, j', p) \le c(j') \quad \forall s \in S, j, j' \in J, p \in P \tag{46}$$

$$m_{in}(s, j, p) \le c(j) \quad \forall s \in S, j \in J, p \in P \tag{47}$$

$$c(j) \le c(j') + V_j^{\max}(2 - e(j) - e(j')) \quad \forall j, j' \in J \tag{48}$$

$$c(j) \ge c(j') - V_j^{\max}(2 - e(j) - e(j')) \quad \forall j, j' \in J \tag{49}$$

Constraint (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 (46) ensures that the mass entering unit j' for latent storage does not exceed the capacity of the unit. Constraint (47) is similar to constraint (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 (48) and (49), where units j and j' are units in the same stage.

2.3.2.2. Feasibility constraint.

$$\sum_{s \in S} y(s, j, p) + \sum_{s' \in S} y^{lt}(s', j, p) \le e(j) \quad \forall j \in J, p \in P \tag{50}$$

Constraint (50) is similar to constraint (32), however, it ensures that a unit can only be used for either processing or latent storage if that unit exists.

The objective function, which in this case is the minimization of capital cost, is shown in constraint (51). Due to the non-linearity of this constraint the model becomes an MINLP model.

$$\sum_{j \in J} (Ae(j) + B[c(j)]^{\alpha}) \tag{51}$$

2.4. Illustrative example

This example is similar to the previous example in Section 2.2, however, there is another reactor in the reaction stage as shown in Fig. 21. The SSN remains the same and is shown here in Fig. 22. The data for this example are shown in Tables 5 and 6.

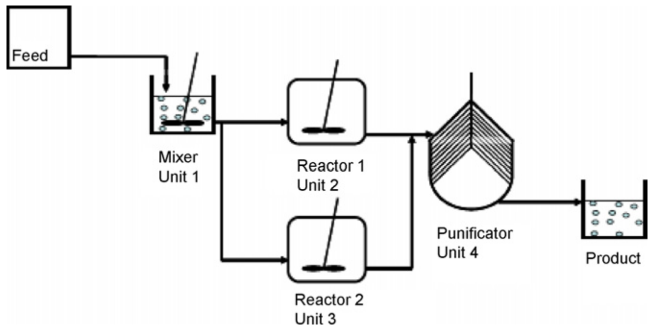

Fig. 21. Flowsheet for the literature example.

$$\begin{tikzcd} S1 \arrow[r] & S2 \arrow[r] & S3 \arrow[r] & S4 \end{tikzcd}$$

Fig. 22. SSN for the literature example.

Table 6
Storage data for illustrative example.

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

2.4.1. 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 7. The resulting plant requires only one reactor as shown in Fig. 23. The optimal capacities of the remaining units are 75 units for the mixer (U1), 75 units for the reactor (U2) and 37.5 units for the purificator (U3). The resulting schedule for the optimal plant is shown in Fig. 24, 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.

Table 7
Computational results of the design literature example.

	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

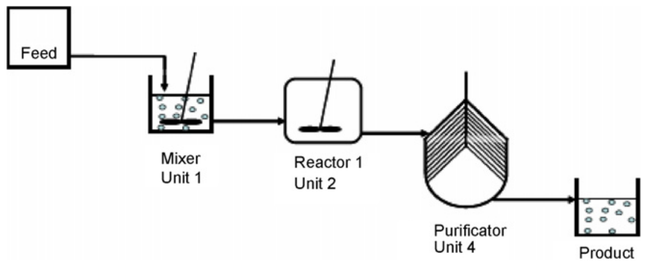

Fig. 23. Resulting design from the model.

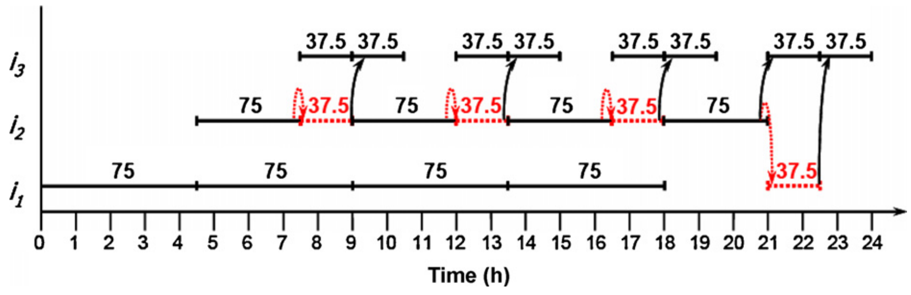

Fig. 24. Schedule for the optimal design.

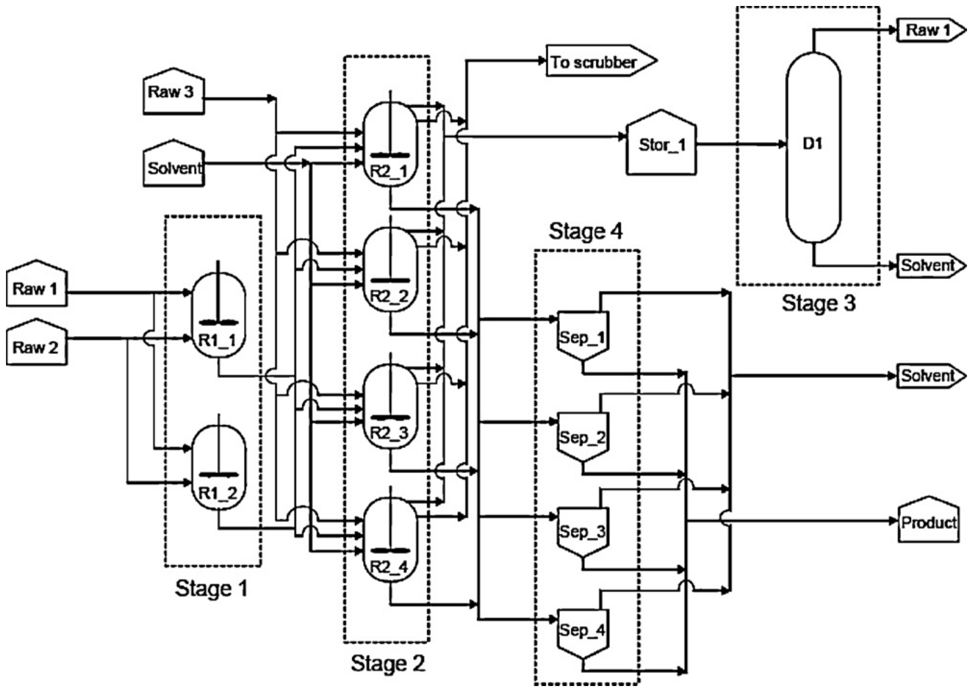

Fig. 25. Flowsheet for the industrial case study.

2.5. 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% increase in the throughput was achieved.

Secondly, the minimum amount of intermediate storage is determined with and without the PIS operational policy. In both the 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 20% reduction in the amount of intermediate storage is achieved. The design model is an MINLP model due to the non-linear 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.

3. 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 Fig. 25, this case study is used to illustrate the application of the design model. The SSN for the case study is shown in Fig. 26.

The process has four stages, separated in Fig. 25 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 trans-

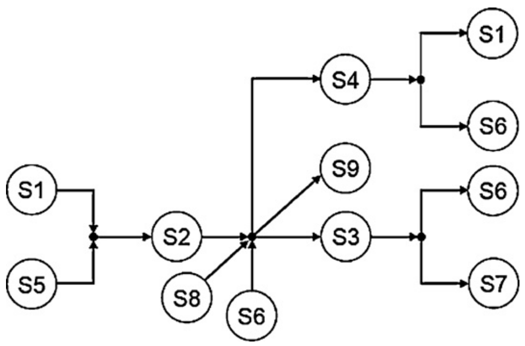

Fig. 26. SSN for the industrial case study.

Table 8
Data for industrial case study.

Unit	Max capacity (t)	Suitability	Duration (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 9
Capital cost data for industrial case study.

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

ferred 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 reaction are flashed off and sent to the scrubber. After 3 h 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 transferred 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 h for illustrative purposes. The production goal for the plant was 109.9 ton of product.

The data for the case study are shown in Tables 8–11.

Table 10
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 11
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

Table 12
Results from the industrial case study.

	Results
Number of time points	8
Objective function value	1727.9
Number of binary variables	228
Solution time (CPU, s)	12,082
Number of variables	4,777
Number of constraints	8,183
Number of major iterations	3

3.1. Computational results

The model was solved on an Intel Core 2 CPU, T7200 2 GHz processor with 1 GB of RAM. The computational results are shown in Table 12. 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 Fig. 27, as can be seen from this flowsheet the

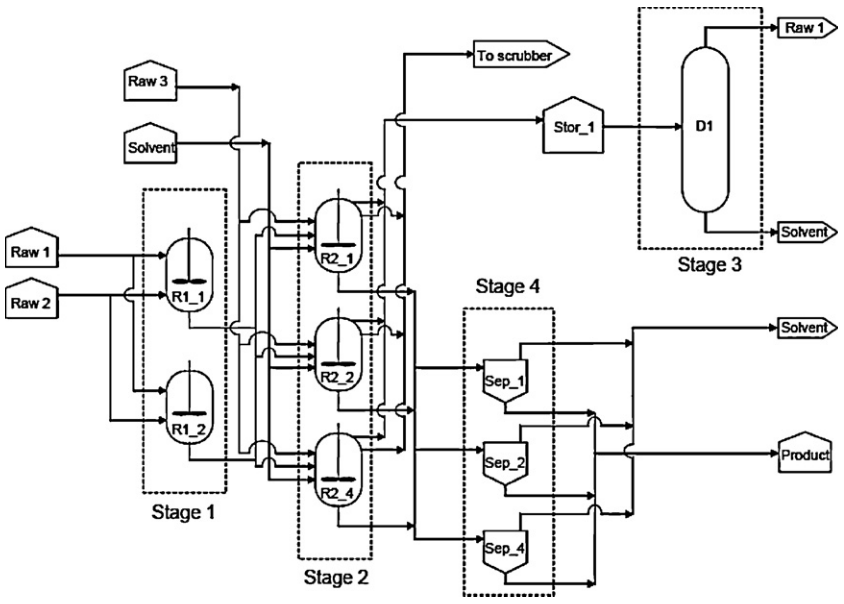

Fig. 27. Resultant flowsheet for the industrial case study.

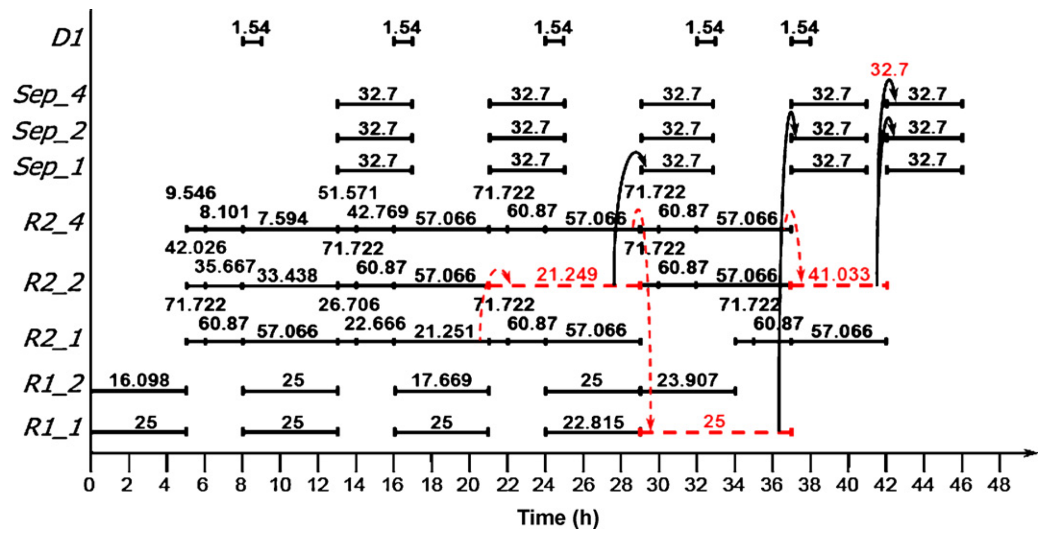

Fig. 28. Schedule for the optimal design.

Table 13
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

design calls for fewer units in stages 2 and 4. The schedule is shown in Fig. 28. As can be seen from the schedule latent storage is utilised during the time horizon of interest and there are no cycles in this schedule. Table 13 details the required unit capacities, which are lower than the original design.

3.2. Conclusions and discussion on the industrial application

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 12 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 non-linearity of the objective function global optimality is not assured.

4. Conclusions

The model developed was based on the framework and model developed by Majozi and Zhu (2001) for the following reasons. 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, this model is based on the non-uniform discretization of the time horizon, thus resulting in fewer binary variables. Thirdly, the model is a MILP, thus solutions are globally optimal. However, as aforementioned, mathematical formulations that are based on other recipe representations like STN, RTN and mSTN can also be adapted to the PIS policy. Future work will involve comparisons between the performance of the model presented in this manuscript and the models arising from the other network representations in the application of this policy.

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 guaranteeing 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 objective of minimizing the amount of intermediate storage without the use of the PIS operational policy, while keeping the optimal throughput from the first step fixed. 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 fixed objective, a 20% reduction in the amount of required intermediate storage was achieved in the PIS operational policy compared to the case without the PIS operational policy. Once again the models derived were MILP models, thus guaranteeing global optimality.

The second model presented in Section 2, 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 from the phenols production facility to determine its effectiveness. 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.

References

Barbosa-Póvoa, A. P. (2007). A critical review on the design and retrofit of batch plants. *Computers and Chemical Engineering*, *31*(7), 833–855.

Grossmann, I., & Sargent, W. (1979). Optimum design of multipurpose chemical plants. *Industrial and Engineering Chemistry Process Design Development*, *18*(2), 343–348.

Ierapetritou, M., & Floudas, C. (1998). Effective continuous-time formulation for short-term scheduling. 1. Multipurpose batch processes. *Industrial and Engineering Chemistry Research*, *37*(11), 3037–3051.

Jung, J. H., Lee, H. K., & Lee, I. B. (1996). Completion times algorithm of multi-product batch process for common intermediate storage policy (CIS) with nonzero transfer and setup times. *Computers and Chemical Engineering*, *20*(6–7), 845–852.

Kondili, E., Pantelides, C., & 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., & 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., & 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.

Mendez, C. A., Cerdá, J., Grossmann, I. E., Harjunkoski, I., & Fahl, M. (2006). State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Computers and Chemical Engineering*, *30*(6–7), 913–946.

Ravemark, D., & Rippin, W. (1998). Optimal design of a multi-product batch plant. *Computers and Chemical Engineering*, *22*(1), 177–183.

Schilling, G., & 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., & 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., & Rippin, D. (1975). The choice of equipment sizes for multi-product batch plants. Heuristics vs. branch and bound. *Industrial and Engineering Chemistry Process Design Development*, *14*(3), 197–203.

Suhami, I., & Mah, R. (1982). Optimum design of multipurpose batch plants. *Industrial and Engineering Chemistry Process Design Development*, *21*(1), 94–100.

Weide, W., Jr., & Reklaitis, G. V. (1987). Determination of completion times for serial multiproduct processes. 3. Mixed intermediate storage. *Computers and Chemical Engineering*, *11*(4), 357–368.