UNIVERSITY OF THE
WITWATERSRAND,
JOHANNESBURG

A Computationally Efficient Method for the Scheduling of Complex Batch Processes

Joel Croft

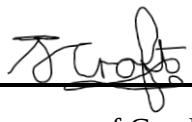
Master of Science in Engineering by research:

"A dissertation submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science in Engineering."

Johannesburg, 2022

Declaration

I declare that this dissertation is my original and unaided work. It is being submitted for the degree of Master of Science in Chemical Engineering to the University of the Witwatersrand, Johannesburg. This is the first submission of the dissertation to any university.



Signature of Candidate

__10_ day of __August__ year ___2022_____

Acknowledgements

This research project would not have been possible without the support and assistance I received from the following parties.

First, I would like to thank my parents who encouraged, supported and advised me during this period. Your input aided me in overcoming many obstacles as well as getting unstuck from many a rut.

Second, I would like to thank my supervisor, Professor Thokozani Majozi, for your unparalleled guidance, experience and wisdom. You steered me in the exploration and establishment of a direction, challenging my conclusions and refining the quality of this work.

Third, I would like to express sincere gratitude to the National Research Foundation for its financial support and to the University of the Witwatersrand for technical assistance and provision of software facilities, all of which enabled this research.

Abstract

This dissertation presents an improved continuous time, unit-specific event-based Mixed-Integer Linear Programming formulation for the optimal scheduling of general network-represented batch plants, based on the State-Task Network representation. The formulation draws on and combines the strengths of previous works in order to incorporate rigorous conditional sequencing, pre- and post-processing unit wait and task splitting while ensuring the integrity of the Finite Intermediate Storage policy. Task splitting is simulated without requiring potential problem truncation and nested iteration which may result from the utilization of a splitting parameter Δn and three-index binary and continuous variables to represent the start and end events of tasks. Additionally, the proposed formulation allows for the fractional extraction of produced states from their producing unit at multiple events, thereby increasing the flexibility of resulting schedules. Computational performance is compared against reimplementations of four recent task splitting formulations through solution of 22 example problems using the GAMS CPLEX solver in order to demonstrate the effectiveness of the proposed approach and highlight its advantages. It is shown how the proposed formulation is the most reliable due to its ability to converge in all of the considered problem instances and not get trapped at suboptimal solutions due to iterative procedures relating to task splitting. The proposed model performed the fastest during solution in 16 of these examples, while in the other six (Examples 4.1 and 10.4 to 10.8.), the solution time was very comparable to the other formulations investigated. The proposed model demonstrated a best-case CPU time reduction of more than 90%.

Contents

List of Figures

List of Tables

Nomenclature

Indices

p, p'	Event points
s	Material states
j, j'	Processing units
$s_{in,j}, s'_{in,j}, s''_{in,j}$	Effective states representing tasks

Sets

P	Event points
S	Material states
S^R	Raw material states
S^I	Intermediate material states
S^P	Product material states
S^C	Consumed states – raw materials and intermediates
S^M	Produced states – intermediates and products
S^E	End point states – raw materials and products
S^{FIS}	Intermediate states which have finite dedicated storage facilities
S^{ZW}	Intermediate states which must be consumed immediately upon production due to material instability
S^{NIS}	Stable intermediate states which have no dedicated storage facilities
S^{UW}	Intermediate states which are stable and need not be consumed immediately upon production ($S^{FIS} \cup S^{NIS}$)
S^{NS}	Intermediate states which have no dedicated storage facilities ($S^{ZW} \cup S^{NIS}$)
S_J^c	Set of all states s which are consumed by specified task $s_{in,j}$
S_J^p	Set of all states s which are produced by specified task $s_{in,j}$

Parameters

M	Largest expected value for the time horizon
H	Time horizon of interest (becomes a variable in case of makespan minimization)
$V^U(s_{in,j})$	Maximum capacity of a unit for a particular task
$V^L(s_{in,j})$	Minimum capacity of a unit for a particular task
$a(s_{in,j})$	Fixed processing time term for a task
$b(s_{in,j})$	Batch size-dependent processing time coefficient for a task
$q^U(s)$	Maximum storage capacity for an intermediate state $s \in S^I$
$q^0(s)$	Initial inventory of intermediate state $s \in S^I$ at the start of the time horizon
$\rho_c^L(s, s_{in,j})$	Minimum fraction of consumed state $s \in S^C$ in material consumed by task $s_{in,j} \in S_{in,J'}^c$
$\rho_c^U(s, s_{in,j})$	Maximum fraction of consumed state $s \in S^C$ in material consumed by task $s_{in,j} \in S_{in,J'}^c$ (also used when the consumption ratio is constant)
$\rho_p^L(s, s_{in,j})$	Minimum fraction of produced state $s \in S^M$ in material produced by task $s_{in,j} \in S_{in,J'}^p$
$\rho_p^U(s, s_{in,j})$	Maximum fraction of produced state $s \in S^M$ in material produced by task $s_{in,j} \in S_{in,J'}^p$ (also used when the production ratio is constant)
$S_{Pr}(s)$	Sell value of product state $s \in S^P$ (can be extended to other states if applicable)
$d(s)$	Demand profile for product state $s \in S^P$ (can be extended to other states if applicable)

Continuous Variables

$m_u(s_{in,j}, p)$	Total quantity of material charged to a unit to begin processing in task $s_{in,j}$ at event p
$m_c^V(s, s_{in,j}, p)$	Quantity of consumed state $s \in S^C$ charged to a unit to begin processing at event p for a task $s_{in,j} \in S_{in,J'}^c \cap S_{in,J'}^{c,V}$ which can consume s in a variable fraction
$m_p^V(s, s_{in,j}, p)$	Quantity of produced state $s \in S^M$ produced by task $s_{in,j} \in S_{in,J'}^p \cap S_{in,J'}^{p,V}$, which can produce s in a variable fraction
$t_{in}(s_{in,j}, p)$	Time at which processing of task $s_{in,j}$ begins at event p
$t_{out}(s_{in,j}, p)$	Time at which processing of task $s_{in,j}$ ends at event p

J	Processing units j	$q_T(s)$	Total quantity of raw material or product $s \in S^E$ respectively consumed or produced over the time horizon of interest
$S_{in,J}$	Tasks described as effective state s entering unit j for processing	$q(s,p)$	Excess quantity of FIS state $s \in S^{FIS}$ stored at event p
$S_{in,J}^{J}$	Set of all possible tasks $s_{in,j}$ which can be performed in specified unit j	$b_1(s_{in,j}, s'_{in,j}, s, p)$	Quantity of material $s \in S^I$ transferred to consuming task $s'_{in,j} \in S^c_{in,J'}$ at event p from producing task $s_{in,j} \in S^p_{in,J'}$ at event $p-1$, due to lack of available material in storage for consumption task $s'_{in,j}$
$S_{in,J'}^{J}$	Set of all possible tasks $s_{in,j}$ which can be performed in specified unit j'	$b_2(s_{in,j}, s'_{in,j}, s, p)$	Quantity of material $s \in S^{FIS}$ consumed by task $s'_{in,j} \in S^c_{in,J'}$ at event p, which is produced by task $s_{in,j} \in S^p_{in,J'}$ at event $p-1$, in order to prevent FIS violations
$C_{in,J}$	Set of all possible tasks $s_{in,j}$ in all possible units j which consume specified state s	$u(s, s_{in,j}, p)$	Quantity of material $s \in S^I$ produced by task $s_{in,j} \in S^p_{in,J'}$ at event $p-1$ or earlier which is temporarily stored in its producing unit at event p
$P_{in,J}$	Set of all possible tasks $s_{in,j}$ in all possible units j which produce specified state s	**Binary Variables**	
$P_{in,J}^{ZW}$	Set of all possible tasks $s_{in,j}$ in all possible units j which produce at least one ZW state	$y(s_{in,j}, p)$	Indicates whether or not task $s_{in,j}$ begins production at event p
$C_{in,J}^{V}$	Set of all tasks $s_{in,j}$ which consume states in a variable ratios	$z(s_{in,j}, s'_{in,j}, s^I, p)$	Indicates whether or not state $s \in S^I$ is transferred to consuming task $s'_{in,j} \in S^c_{in,J'}$ at event p from producing task $s_{in,j} \in S^p_{in,J'}$ at event $p-1$ due to lack of available material in storage for consumption task $s'_{in,j}$
$P_{in,J}^{V}$	Set of all tasks $s_{in,j}$ which produce states in a variable ratios	$v(s_{in,j}, s'_{in,j}, s, p)$	Indicates whether or not state $s \in S^{FIS}$ is directly transferred to unit j' for consumption by task $s'_{in,j} \in S^c_{in,J'} \cap S'_{in,J'}$ at event p after being produced by task $s_{in,j} \in S^p_{in,J'} \cap S'_{in,J}$ in unit $j \neq j'$ at event $p-1$, in order to prevent FIS violations
		$x(s_{in,j}, s'_{in,j}, s, p)$	Indicates whether or not state $s \in S^{FIS}$ is consumed by task $s'_{in,j} \in S^c_{in,J'} \cap S'_{in,J'}$ in unit j' at event p at a time prior to production of the same state in task $s_{in}j \in S^p_{in,J'} \cap S'_{in,J}$ in unit $j \neq j'$ at event $p-1$ in order to prevent FIS violations
		$w(s_{in,j}, p)$	Indicates whether or not states $s \in S^{ZW}$ produced by task $s_{in}j \in S^{p,ZW}_{in,J'}$ continue to be stored at event p in the unit that produced them at event $p-1$ or earlier

Term	Abbreviation
Common intermediate storage	CIS
Cost units	c.u.
Finite intermediate storage	FIS
Finite wait	FW
Hours	h
Infeasible sets	IS
Integer program	IP
Linear programming	LP
Mass unit	m.u.
Mixed-integer linear program	MILP
Mixed-integer nonlinear program	MINLP
Mixed-integer program	MIP
No intermediate storage	NIS
No storage	NS
Nonlinear program	NLP
Process intermediate storage	PIS
Rakovitis et al. (2019)	R 2019
Resource-task network	RTN
Seconds	s
Shaik and Vooradi (2017)	S&V 2017
State-equipment network	SEN
State-sequence network	SSN
State-task network	STN
Unlimited intermediate storage	UIS
Unlimited wait	UW
Vooradi and Shaik (2012)	V&S 2012
Vooradi and Shaik (2013)	V&S 2013
Zero wait	ZW

Chapter 1

Introduction

1.1. Background

The chemical industry is dedicated to the production of a wide array of solid, liquid and gaseous materials and products, such as metals, ceramics, plastics, petroleum and other oil products, solvents, agrochemicals, pharmaceuticals, specialty chemicals and food products. Such products are used on a day-to-day basis by a large portion of the global population as well as downstream manufacturing sectors, making the industry largely impactful in terms of its economic, environmental and social impact.

In a recent report, it was estimated that the chemical industry directly contributed $1.1 trillion to the global GDP in 2017, making it the fifth largest manufacturing sector, with 8.3% of the global manufacturing sector's economic value (Oxford Economics, 2019). The chemical sector in the report was limited to the basic chemicals, fertilizer, plastics and synthetic rubbers; pesticides and agrochemicals; paints, varnishes, inks and mastics; soaps, detergents, cleaning, polishing, perfume and toilet preparations; explosives and pyrotechnic, glue, essential oil; and man-made fibre sub-sectors.

The manufacture of such chemical products is broadly conducted in two major categories: continuous processes and batch processes, although hybrid classifications exist. In the former, products, which are often of lower relative value, are produced in great scale due to a very large and stable demand. Perhaps the best example of this is the petroleum industry, where crude oil is consumed continuously to produce a variety of fuel products. Continuous processes require transient start-up and shut-down

procedures, however they are mainly designed for steady state operation, which occurs for the vast majority of the active time of the processing plant.

Batch processes, on the other hand, produce relatively smaller quantities of high-value products. These products are often characterised as having a lower demand, however they may have more stringent restrictions such as consistency in product quality. Alternatively, they may be produced in smaller quantities due to fluctuating demands in unstable markets.

Due to the current competitive state of global economies, as well as the scale of the chemical industry, the latter has been identified as a prime target for study, analysis and improvement in terms of productivity and intensification as well as time and waste reduction. In this regard, the optimization of chemical production plants has become the focus of much research.

Historically, optimization studies have targeted continuous plants (Majozi, 2010). This is largely because of an understanding that the scale of raw material and utility consumption as well as the generation of waste is somewhat larger in such plants. However, this understanding has been changing as the demand for flexible processing has increased. It is now understood that optimization studies of batch plants can also offer significant savings through careful planning, scheduling and utilization of resources. However, methodologies for batch process synthesis, design, scheduling and resource integration are far less mature than their continuous counterparts, due to a pertinent feature of batch plants: discrete tasks distributed in time.

Unlike in continuous processes, the production of wasteful by-products and the consumption of raw material or utilities occurs discretely in time and therefore any analysis must consider such discrete consumption and production. Therefore, studies of

batch production often include, in some form or another, the idea of scheduling, in addition to dynamic considerations of batch reaction and separation unit operations. Another important consideration, is that of intermediate storage, which, while also present in continuous production plants, is perhaps more important in batch plants in order to debottleneck processes.

The added dimension of time renders most optimization studies of batch processes impossible via graphical and insight-based methods. Simplifying assumptions can be made by approximating continuous circumstances, or sequential procedures can be implemented whereby the time dimension is fixed beforehand, removing it from consideration in the optimization studies. However, this approach may result in the exclusion of the globally optimal solution (Papageorgiou et al., 1994). Instead, mathematical optimization and the use of high-performance computation is required for a complete study. Superstructure representation is often employed, in which all possibilities and combinations of the problem to be solved are contained. The optimal solution is then a subset of this superstructure and many techniques exist for isolating the optimal configuration.

In many cases, resulting optimization studies may be too complex to solve practically, due to large computational time requirements for problem solution, and further assumptions are made to simplify the problem. Alternatively, near-optimal solutions may be satisfactory. As the state of batch optimization is still in its infancy, it is still quite academic and many improvements are required in formulation, both in terms of computational efficiency and practicality. However, the application to real industrial problems is still possible in a number of cases.

1.2. Motivation

The complexity arising in batch scheduling models has been a major limitation in its development and application. This is a limitation which the current work aims to diminish.

Furthermore, a model is only as good as the simplifying assumptions it makes. Models which represent physical phenomena seldom characterize their full extent. They include simplifications which reduce the search space as a trade-off for facilitating the acquisition of usable results. The same is true for batch scheduling. Due to the decisions which are required in batch scheduling, such as assignment and distribution of resources, a highly combinatorial problem often results, for which the solution algorithm has been shown to be exponential. Due to this, a model seldom represents all possibilities which may be achievable in practice and focuses instead on excluding all infeasible solutions. Therefore, the current work also aims at improving the scope of batch scheduling in order to include more possible interactions in an attempt to obtain better solutions.

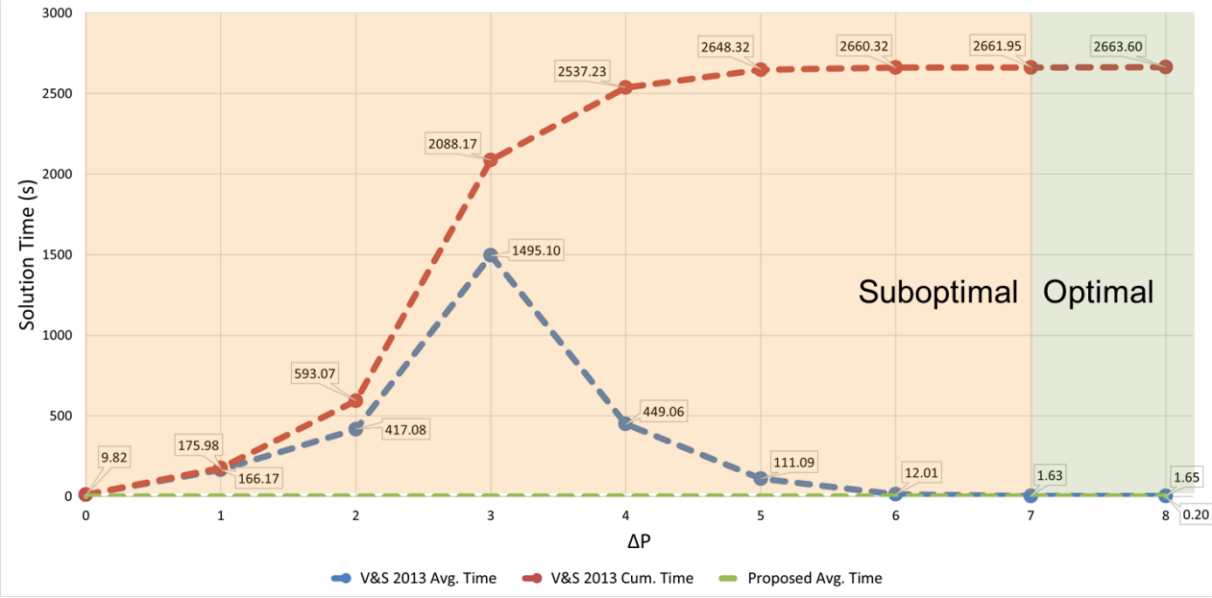


Figure 1-1. Importance of Improvement in the Modelling of Task Splitting

The current work is aimed at overcoming the limitations in previous techniques for implementing task splitting, while accurately handling the Finite Intermediate Storage (FIS) policy, rigorous conditional sequencing and pre- and post-processing unit wait. The importance of this is highlighted in Figure 1-1, which demonstrates the excessive compounded computational time which may be required in certain examples in order to obtain the globally optimal solution. The figure contains the solution time results for the solution of Example 9 (presented in Section 4.2.8), solved with the formulation by Vooradi and Shaik (2013), with varying values of ΔP along with the cumulative solution time after each iteration. For comparison, it is possible to achieve the same solution quality with the proposed model, without uncertainty regarding ΔP and in significantly lower computational time.

Furthermore, it was observed that the proposed method of handling task splitting facilitates fractional extraction without the requirement of a binary variable for non-ZW intermediates. Fractional extraction is the ability to fractionally discharge produced intermediates from the unit that produced them at a number of event points and times. This was first published by Rakovitis et al. (2019). The concept is demonstrated in Figure 1-2, where the obviously maximal production of 10 units of product is not identifiable in most unit-specific event-based formulations due to insufficient dedicated storage availability for the intermediate. The example is that of Example 1 (presented in Section 4.2.1).

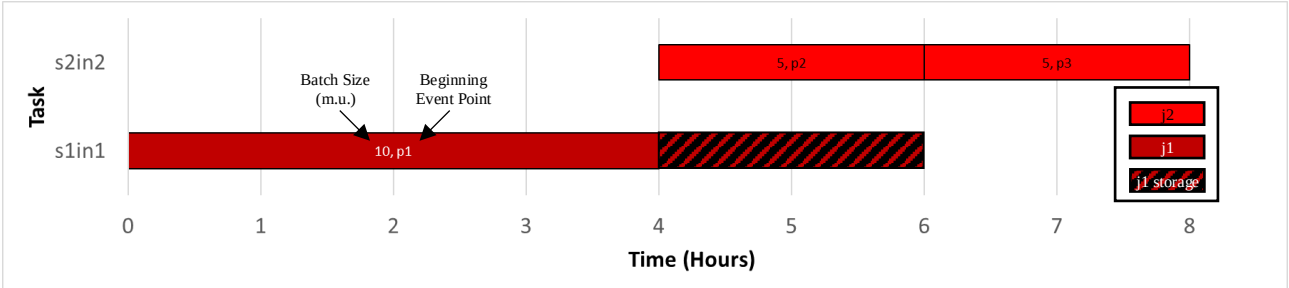


Figure 1-2. Optimal Schedule for Example 1 (Proposed Model)

1.3. Scope

Scheduling may refer to short or long-term scheduling and may be cyclic. Cyclic scheduling occurs when a number of tasks are repeated continuously and in the exact same manner (Draper et al., 1999). The type of scheduling considered in the current work is short-term scheduling, where a single cycle is considered over a relatively short horizon spanning a number of hours or days. This may reflect dynamic production demands occurring on a daily basis, or where a particular schedule is to be used frequently among other schedules for separate production demands.

Furthermore, batch scheduling is a highly problem-specific study. Many unique characteristics may exist in any given batch plant. Production demands may include numerous, temporally-distributed deadlines or the consideration of utility integration during production may be desired. Additionally, the changeover and setup time requirement, as well as the cleaning requirement, between consecutive tasks occurring in a batch processing unit may depend on the order of these consecutive tasks. This is known as sequence-dependent changeover. These extended features are not considered in the current work. Therefore, the proposed work aims at addressing the form of scheduling which involves meeting a demand for a number of products by a single given due date, minimizing the makespan required for such a given demand or maximizing production throughput in a given time horizon.

1.4. Objectives

The objective of the current work is to investigate a method for scheduling a generic batch production plant while improving on the scope of what is realistically possible to achieve while simultaneously improving the problem formulation for faster computational performance. The idea is to provide an improved scheduling basis which

can be used independently or which provides a better foundation for the inclusion of synthesis, design, resource integration and other problem-specific features than those currently available. The expected results are therefore improved solutions for the same underlying problems obtained in faster time, as compared with published literature.

1.5. Problem Statement

This works aims to provide an improved method for determining an optimal operating schedule for batch production processes in the chemical industry, via mathematical optimization techniques, in order to address better resource utilisation and waste minimization in such processes. The scheduling problem addressed by this work can be described as follows:

Given:

i The process recipe, detailing the required path from raw materials to products via any necessary intermediates,

ii the available units, their associated task suitability and maximum and minimum capacities,

iii the processing time requirements for each task, as a linear function of the batch size processed,

iv the initial inventory levels and maximum storage capacity for each state,

v the economic data of the states involved, such as raw material costs and value of the final products and

vi the time horizon of interest or the required demand profile

Determine:

i the optimal assignment of tasks to units and their sequences

ii the batch sizes of each task,

iii their start and end times as well as their durations

which maximize the economic criteria of the process or minimize the time required to meet the given demand.

1.6. Structure of the Dissertation

The dissertation is organised as follows:

Chapter 1 provides the background and motivation of the investigation into batch schedule optimization. Chapter 2 contains an overview of concepts and terminology used in the field of batch process scheduling, followed by a survey of historical model development. In this chapter, opportunity for model improvement is identified. Chapter 3 introduces an improved mathematical model which includes an improved approach to batch scheduling through removing a dimension of iteration during the solution of a given problem. A detailed model is presented and explained. Chapter 4 introduces a number of challenging or benchmark case studies commonly studied in the literature and contains comparative results from the proposed model. This section serves as validation for the proposed model and provides a demonstration of its advantages as well as a summary of the results obtained during the study in graphical format. Chapter 5 draws conclusions from the outcomes of the model analysis and discussion.

Chapter 2

Literature Review

The following chapter presents some background into the field of batch scheduling optimization. Section 2.1 begins with an overview of optimization and the most common and well understood principles and methods used to perform optimization, which are not necessarily specific to batch process scheduling. Important characteristics and the development of historical models, as well as discussion of mathematical programming techniques specifically designed for the field of short-term batch scheduling, are then presented in Section 2.2.

2.1. Optimization Principles for Batch Scheduling

Optimization is a study performed on a system in order to improve its performance. The system is usually described by a number of variables which represent the various physical characteristics of the system. These variables are related by a number of equations and inequalities. When the number of independent equations match the number of variables, the system is well defined and the values of all the variables are set. A study carried out in this fashion is called simulation, whereby it is of interest to view how the system behaves under set conditions. Otherwise there is room for variance in one or more variables, resulting in a number of degrees of freedom in the system. Changing these variables will affect the output of the system. At least one degree of freedom is required for optimization. Additionally, the system should have some performance criteria by which it can be measured, such as throughput, profit or time required. This serves as an objective function for optimization.

2.1.1 Mathematical Programming

A very well-structured approach to optimization is that of mathematical programming. This involves describing a physical system with purely mathematical constraints, often algebraic. The simplest form of mathematical programming is linear programming (LP), in which all the equations and inequalities as well as the objective function are linear and the variables are defined over continuous ranges. A simple two-dimensional example is shown in Figure 2-1. The system consists of two variables and one constant: x, y and c respectively. The red, green and blue lines represent the problem constraints which communicate the relationship between the system variables as well as their bounds. The constraints indicate the feasible region (shaded triangle) wherein all feasible solutions lie. The black dot is the objective function, which is a function of a number of the system variables. The objective is to either maximize or minimize *Z*. In LP, the best objective value always lies at a vertex of the feasible region.

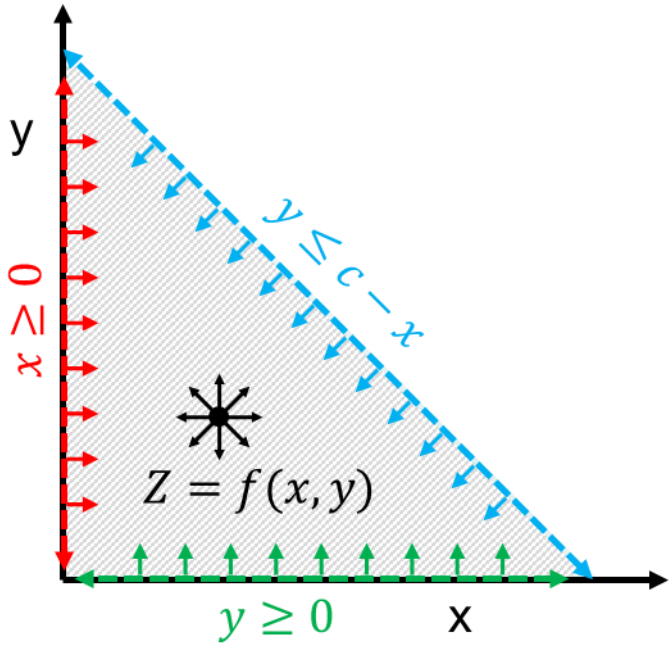


Figure 2-1. Linear Programming

In reality, problems are often far more complex and can be defined over thousands of variables. This means the problem is defined over thousands of dimensions and is not

possible to represent graphically. However, the principles of the LP remain the same and the problem can be solved mathematically, with the help of powerful computing.

The standard form of the LP is given by:

$$Min\ c^T x\ s.t. \begin{cases} Ax = b \\ Cx \leq d \end{cases}$$

Where *x* is a vector of the problem variables, *c*, *b* and *d* are vectors of known coefficients and *A* and *C* are matrices of known coefficients.

2.1.2 Simplex Method

The simplex method is the most commonly used method to solve LP problems. In this method, the problem constraints and objective function are organised into a matrix representing the search space or feasible region. The objective function is then evaluated at the vertices of the resulting polytope sequentially, by moving from vertex to vertex according to the greatest rate of improvement in the objective function value. This is done via matrix row operations until convergence criteria for optimality and feasibility are met. Two simplex algorithms exist, namely: the primal simplex and dual simplex algorithms. In the former, successive iterations are all feasible and successively less suboptimal, whereas in the latter, successive iterations are successively less infeasible and less superoptimal.

Other algorithms exist for solving LPs. The documentation for the CPLEX solver in the GAMS user manual (28.2.0 - August 19, 2019) mentions the network optimizer, the barrier algorithm and the sifting algorithm, however they rarely outperform the simplex method.

Often, in engineering and other optimization problems, discrete decisions need to be made as part of a system, such as the assignment of resources to processing tasks or the number of stages to include in a membrane water purification system. These discrete variables are integers and therefore are not defined continuously in the search space for the solution. They can, however, be included in mathematical programming studies, although special solution techniques will be required. If all the variables of a problem must assume integer values, the mathematical program is classified as an Integer Program (IP). If only some of the variables are integers, while others are real or continuous, the classification is a Mixed-Integer Program (MIP). Figure 2-2 provides a visual representation of LP, IP and MIP as well as how such restrictions can affect the optimal solution of a problem. When the problem is linear, integer variables result in a Mixed-Integer Linear Program (MILP). In order to solve mathematical programs involving discrete variables, the branch and bound technique is used.

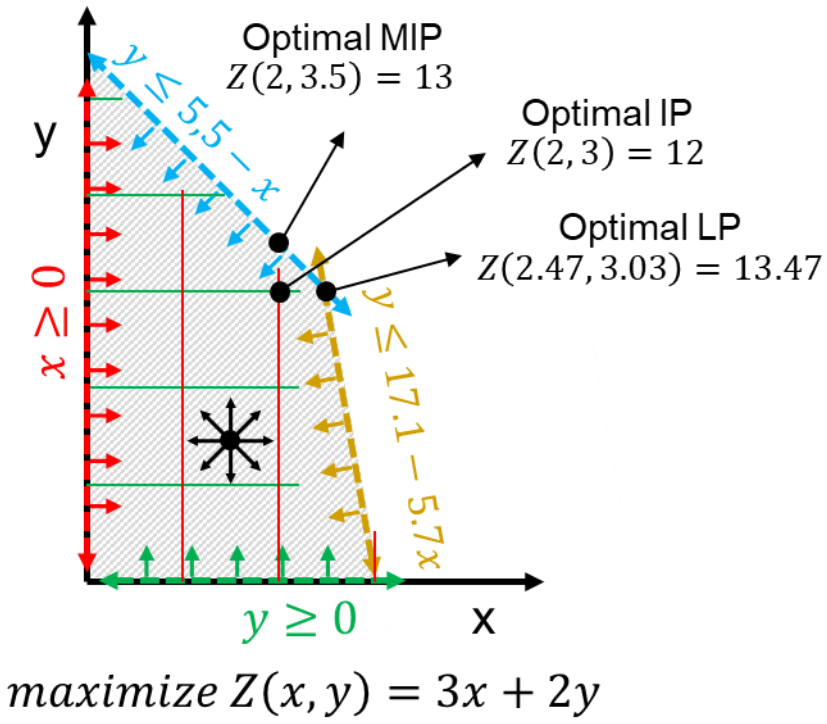


Figure 2-2. LP, IP and MIP

2.1.3 Branch and Bound Technique

The branch and bound technique maps out all combinations of integer variables as nodes in a search tree and systematically searches through the tree for the optimal solution. The algorithm generally exhibits an exponential relationship between the computational time required to find the optimal solution and the number of integer variables. In the worst case scenario, all combinations of the integer variables will need to be considered, however, as the search progresses, more and more information is obtained about the upper and lower bounds of the objective and this information is used to cut unnecessary branches from the tree, significantly reducing computational time requirements. The algorithm is displayed graphically in Figure 2-3.

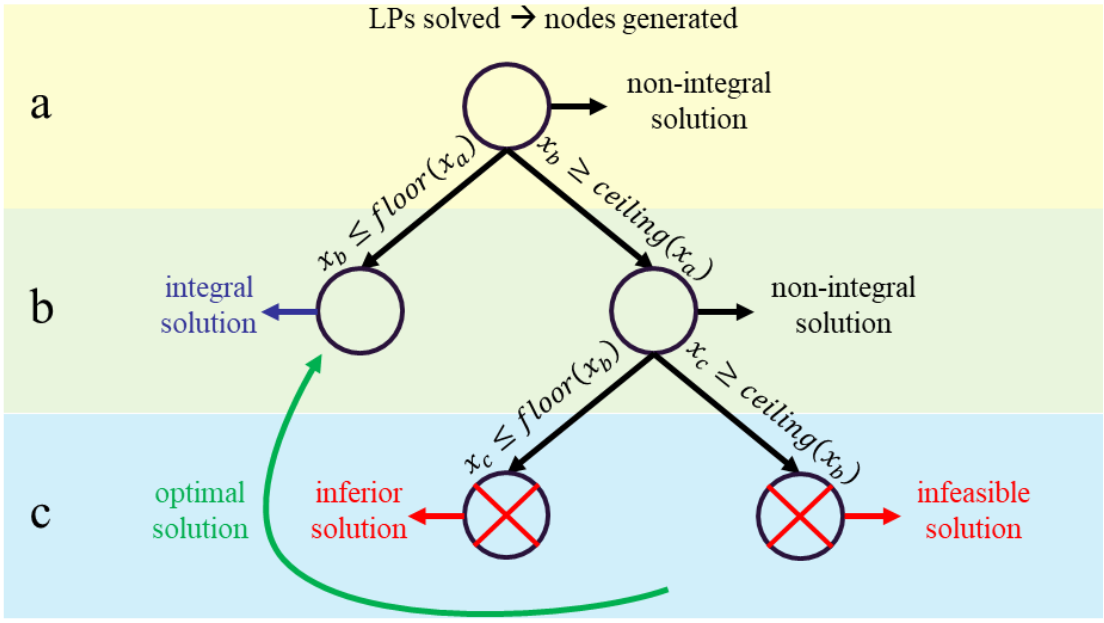


Figure 2-3. Branch and Bound Algorithm

Initially, the problem is relaxed and solved without any integer restrictions on the variables. In the case of a MILP, the solution at the first node is that of a LP. In the case of a minimization problem, this solution provides a hard *lower* bound on the optimal solution.

If the solution obtained at the first node is such that all integer variables assume integer values, then the problem is solved and is said to have a zero integrality gap. The integrality gap is the absolute difference between the value of the objective for the relaxed problem, containing only continuous or real-valued variables, and for the actual problem, containing integer variables. Usually, integer variables will take fractional values at the first node and therefore branching occurs for each of these variables. For each non-integral variable, a subproblem is generated with an additional constraint stating that the variable of interest must be less than or equal to its previous value rounded down and another subproblem is generated where the variable must be greater than or equal to its previous value rounded up. This way, child nodes are generated, at which the current problem is again relaxed and solved without integer variables as a LP.

The process continues until a feasible (integral) solution is first obtained, in which all the constraints of the original problem are satisfied. This solution provides an upper bound on the optimal solution to the problem. From this point, when the relaxed solution at any node is inferior to (greater than) the current best solution, it and all its descendant nodes can be removed from the search tree. This is because the relaxed solution at a node is always better than or equal to any solutions occurring in its descendant nodes. Additionally, whenever imposing the additional bounds on the integer variables results in an infeasible LP, the current node can progress no further and is removed from the search space. The upper bound or best current feasible integer solution is updated whenever a better one is found.

Once the entire tree has been pruned and searched, the upper bound and lower bound will have converged and the globally optimal solution will have been found, assuming that the problem has at least one feasible solution. Note that in the case of

nonconvexities appearing in nonlinear mathematical programs, global optimality cannot be guaranteed.

While the branch and bound algorithm generally processes nodes based on integer variables, some novel techniques were employed in batch schedule optimisation, such as in a solution procedure developed by Schilling and Pantelides (1996), where the algorithm branches on both binary and continuous variables.

The branch and bound algorithm can occur with a breadth-first search or a depth-first search. In the former, each level is explored before descendent nodes, whereas in the latter, all descendent nodes are exhausted before moving across to the next node on a particular level. However, the extent to which backtracking occurs can also vary between these two extremes in certain optimization suites, such as with the GAMS CPLEX solver.

When the system constraints involve nonlinear relationships, a Nonlinear Program (NLP) results. If integer variables are also present, a Mixed-Integer Nonlinear Program (MINLP) results. Some batch scheduling formulations, especially the earlier ones, resulted in MINLPs (Zhang & Sargent, 1996; Mockus & Reklaitis, 1997). These are discussed in more detail in Section 2.2. These problems are generally more difficult to solve, especially if nonconvexities exist in the nonlinear constraints, however, the current state of batch scheduling formulations includes mainly MILP problems (Ierapetritou & Floudas, 1998; Seid & Majozi, 2012; Lee & Maravelias, 2017).

The size of a LP is defined by the number of variables and constraints that comprise it i.e. the number of rows and columns respectively in its matrix. The current practical limit on the size of a LP is considered to be of the order 10^6 rows and columns. In order to reduce the dimensionality of both integer and continuous variables as well as

constraints, pre-processing can be performed whereby the problem structure and inherent relationships can be exploited. This technique can also have the effect of improving the problem's bounds (Biegler & Grossmann, 2004).

Despite this, however, sometimes a problem may be too large to practically solve via mathematical modelling. In these cases, approximations may be made to reduce the problem size and speed up computation at the expense of accuracy or the exclusion of a set of solutions, some of which may potentially be optimal. An alternative is to use stochastic modelling. This involves the generation of random solutions and implementing an algorithm which randomly iterates upon the variables in an attempt to improve the solution. This continues until a stopping criterion, such as a maximum number of iterations or some objective improvement criterion in consecutive iterations, is met. In stochastic optimization, there is no guarantee that the returned solution will be optimal for the problem and performing the procedure twice will likely yield different results for many problems.

2.1.4 Genetic Algorithm

A popular stochastic optimization method is that of genetic algorithms. This algorithm simulates biological evolution as it iterates. A set of initial solutions is randomly generated and the *fitness* of each is measured using a fitness function. This determines the quality of the solutions and is to be minimized or maximized like an objective function. The *population* of solutions then undergoes changes in their variables in order to obtain a new set of solutions. This can occur via selection, mutation and crossover. Selection involves retaining the best solutions from one *generation* or iteration to the next. Mutation involves making random changes to the underlying problem variable values in order to produce different results. Crossover involves producing solutions for the next iteration by combining characteristics i.e. values of certain variables from two

parent solutions. Crossover attempts to retain strong *genes* from a population while varying the other variables and mutation allows for random variance which can help prevent solutions from becoming trapped in local optima. The algorithm proceeds over a number of generations until an upper limit on the number of generations is reached or until a set number of consecutive iterations produces too small an improvement. The algorithm has been used successfully in batch scheduling applications, among others, such as in Woolway and Majozi (2018).

2.2. Batch Scheduling

Batch scheduling involves the planning and assignment of limited shared resources to processing tasks such that a set of products can be generated from a set of raw materials. Tasks require a number of items in order for processing to occur. These include, among others, a unit, the input materials and resources, such as utilities or manpower. Such items are often limited in availability and must be shared amongst a number of competing tasks, without the consumption levels violating the upper and lower availability bounds. Furthermore, each task is associated with a specific processing time, which separates the consumption and generation of items along the time dimension. Some important considerations in performing a batch scheduling optimization study are the time representation, the storage policy for any intermediates required in the process and the layout of the batch plant and interconnectivity of its tasks. These are discussed in more detail below.

Batch processes differ from continuous processes in that tasks occur in discrete intervals and the profiles of material quantity, temperatures, flowrates and other process variables fluctuate along a time horizon of interest. The fundamental difference between batch and continuous processes is summarized in Figure 2-4, which has been adapted from Majozi (2010). Figure 2-4a depicts a continuous process, in which all the units are

active constantly along the time axis. This is representative of the steady state condition applicable to continuous processes which allows the suppression of the time dimension. However, Figure 2-4b represents a batch process in which no steady state condition is possible due to the discrete nature of the processing tasks. In a scheduling study, careful placement of tasks in time must occur in order to ensure that intermediates are not consumed or produced in infeasible quantities with respect to their relative producing and consuming tasks.

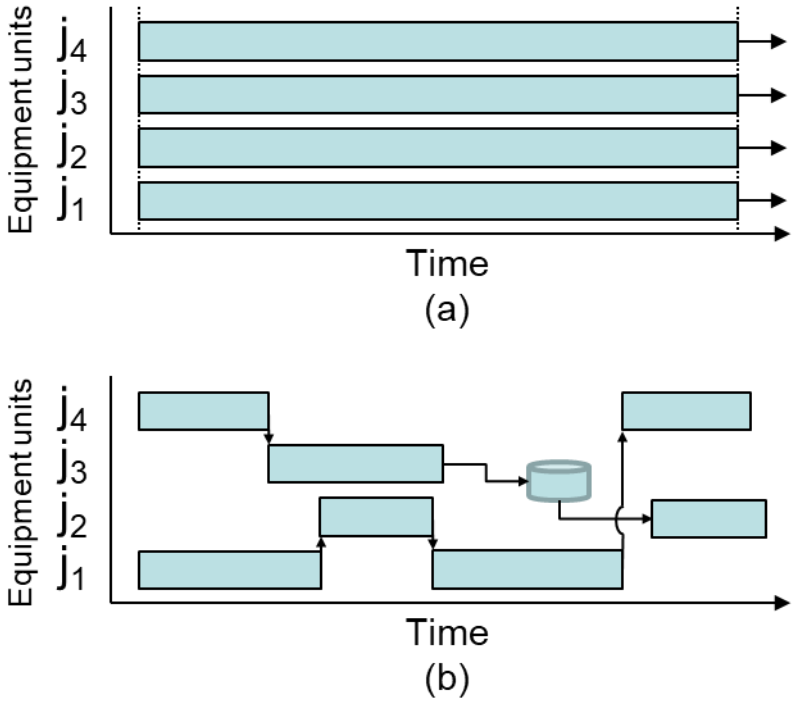


Figure 2-4. Continuous and Batch Processes

Typically, batch processes involve a number of processing units which are each suited to one or more specific tasks responsible for the transformation of materials. Raw materials are the starting materials. These are processed to produce products or intermediates. Intermediates may be stored in dedicated storage vessels in order to debottleneck the process by allowing units to be emptied in a timely fashion and to begin processing subsequent tasks. Intermediates are then further processed in other tasks in a series of steps until final products are produced. Sometimes, for a particular

batch scheduling problem, some intermediates exist at the start of the time horizon of interest, such as when there are residual intermediates from a previous cycle.

Storage policies refer to how excess material states should be managed within the batch processing plant. These almost invariably concern how much storage is available for the materials, although this can also be affected by unique qualities of the materials themselves. A number of storage policies exist.

When unlimited intermediate storage (UIS) is considered for a particular state, it means that the processing facility, or the study thereof, is not concerned with limiting the excess available quantity of the state i.e. infinite storage availability is assumed. This may occur when a storage vessel for a state is to be designed after information regarding the maximum requirement for storage is determined. Therefore, the UIS policy is more of a theoretical policy.

FIS for a state implies that a restriction is placed on the maximum excess availability of the state at any given instant in time in order to prevent a given upper bound from being exceeded. This occurs when a dedicated storage vessel for the state already exists or when the maximum capacity thereof is known prior to scheduling. The FIS policy is desirable in practice since the availability of dedicated intermediate storage often facilitates significant debottlenecking of a process.

The no intermediate storage (NIS) policy is applied when any produced intermediates must be consumed immediately or be temporarily stored in a processing unit prior to consumption, since no dedicated storage vessel exists for the state. NIS is essentially a special case of the FIS policy, where the upper bound on the quantity of stored material is zero. The NIS and FIS policies represent the more practical storage policies in chemical batch plants and, according to Sanmarti, Friedler and Puigjaner (1998), the

NIS policy is very common. The NIS policy is often implemented under duress due to a lack of physical space in a batch plant to facilitate storage vessels.

The zero wait (ZW) policy applies to unstable intermediates which must be consumed immediately upon production, without delay, in order to prevent degradation of the material which may render it unusable. This implies that no storage is possible for these materials.

Other classifications exist which are used less commonly. The common intermediate storage (CIS) policy refers to the sharing of a storage vessel among a number of different states. This is uncommon and is more appropriately considered an academic policy due to the strict regulations on product contamination and integrity existing in many batch production industries, such as those of food and pharmaceuticals. Process intermediate storage (PIS) refers to the utilization of processing units as temporary storage for a set of states (Pattinson & Majozi, 2010). Finite wait (FW) refers to an unstable intermediate which can be stored for some maximum amount of time before it must be consumed. Unlimited wait (UW) refers to materials which may be stored indefinitely without degrading. States adhering to either the FIS or NIS policies also adhere to the UW policy. The no storage (NS) policy applies to all states for which no storage exists, however the description does not differentiate states following the NIS and ZW policies.

Where FIS, NIS, ZW etc. *states* are mentioned in the remainder of this work, this refers to the states adhering to the mentioned storage *policy* for a given problem.

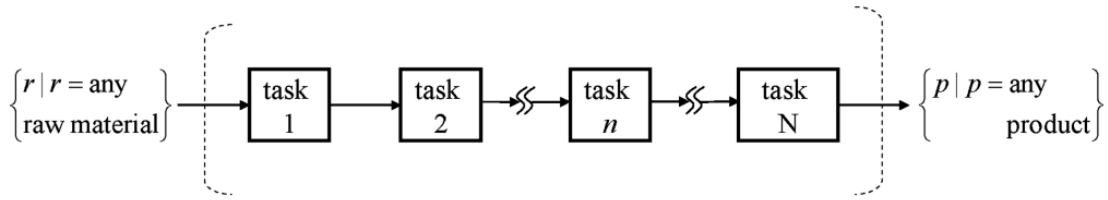
As mentioned above, batch processes involve the transformation of raw materials through any necessary intermediates to produce final products. This network of predetermined processing steps is known as a batch recipe. Batch recipes can be

classified according to their characteristics which ultimately determine the structure of the plant. The type of recipe also informs the method used to (optimally) schedule production. In this regard, a batch plant may be described as sequential or network-represented.

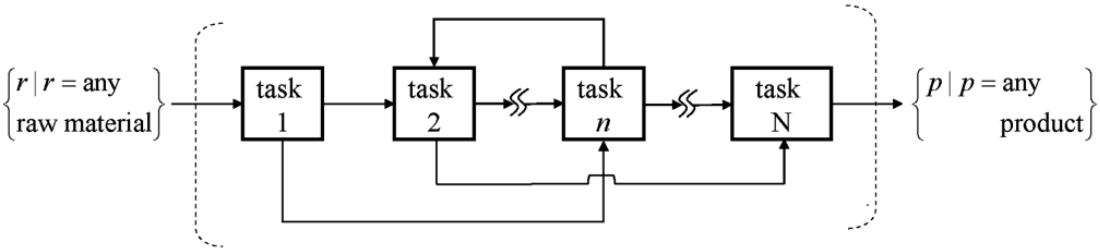
Sequential batch plants retain the unique identity of each intermediate produced as a batch progresses along the ordered set of processing stages. This often occurs in the automotive industry where intermediates are physical artefacts for which the subsequent processing stage is easily identifiable. However, it can also occur in manufacturing or chemical processes which handle fluids and in which the integrity of produced intermediates must be ensured due to strict quality control requirements.

Network-represented processes allow batch mixing and splitting and therefore careful consideration of intermediate quantities must be observed. Storage vessels may be used to aggregate intermediates of the same type and therefore the unique identity and quantity of the intermediates produced by specific instances of specific stages is not recognisable. This often occurs when intermediates may be processed by a number of different subsequent stages to ultimately produce different products.

A batch process recipe may also be described as single-stage, multistage or multiproduct. Single-stage processes do not include any intermediates and simply convert raw materials into final products via a single operation which may occur in a single processing unit or multiple processing units of a similar type. A single-stage process is neither sequential nor network-represented. This type of recipe is represented in Figure 2-6a (Lee & Maravelias, 2017).



(a) Multiproduct batch plant



(b) Multipurpose batch plant

Figure 2-5. Multiproduct and Multipurpose Batch Recipes

A multistage process requires a number of sequential operations which are performed in the same processing unit or a number of different ones. There may be one or more units per stage, allowing parallel processing. However, each product follows the same route through the process, i.e. the same order of stages and the same set of machines which are assigned to each stage. Multistage processes are sequential processes and are also referred to as flow shop processes or multiproduct processes. Each stage produces the exact quantity of intermediate needed to perform the next stage and therefore no mixing from different batches is required. This type of recipe is represented in Figure 2-6b (Lee & Maravelias, 2017) and Figure 2-5a (Majozi, 2010).

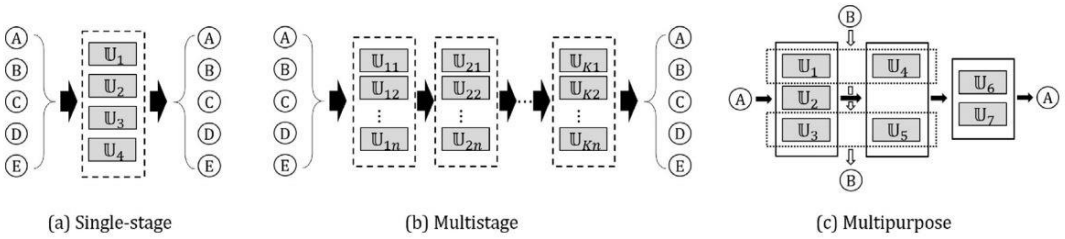


Figure 2-6. Batch Recipe Classification

Multipurpose processes require a number of product-specific stages to be conducted in a specified sequence in order to produce products i.e. each product may have a unique number of stages which may occur in the same units but in a different order and the same processing unit may be used more than once for the same batch. Sequential multipurpose processes are referred to as job shop processes, however multipurpose processes may also be network-represented processes. This type of recipe is represented in Figure 2-6c (Lee & Maravelias, 2017) and Figure 2-5b (Majozi, 2010).

Network-represented processes are the most complex out of these classifications due to the flexibility of merging and splitting batches through the use of inter-batch transfer and storage vessels. A large number of processes in the batch chemical industry fall into this category. In order to model these types of processes, a representation of the product recipes aids in providing a basis for mathematical formulations.

In order to model the assignment and activation of tasks for a multipurpose batch plant, with batch mixing and splitting and variable assignment of tasks to units, a framework of event points is required. Event points mark points along a time grid where one or more events, such as the beginning or ending of processing tasks, occur. Binary variables are used to model the decision regarding the activation of a task at a particular event point. As shown in Figure 2-7, a bar or rectangle represents an active task $s_{in,j}$, the unit in which the active task is being processed is represented by j and p represents the ordered event at or in which the processing is occurring. Events written along the abscissa indicate the individual beginning and ending events of the task, while events

written inside bars indicate that the task both begins and ends at the mentioned event. These distinct structures are sometimes termed *event point* or *slot* based structures, however, the terms are often used somewhat freely. The difference lies in which numbered event relates an end time variable to the start time variable for a given task, however both structures require a variable for or to determine each time. Specifically, the event point structure may require an additional numbered event over the slot structure for the same problem and therefore most contemporary formulations actually use the slot structure. However, these formulations often refer to event points or are referred to as event-based formulations. Event points have been used in three major classes of time representation: discrete time representation, global, continuous time event-based representation and unit-specific, continuous time event-based representation. These are discussed in more detail below.

Various Gantt charts, similar to Figure 2-7, are provided in the remainder of this chapter to demonstrate concepts used in batch schedule optimization. The letters A, B and C occurring in some of them represent a task producing one or more intermediates, a task consuming one or more of these intermediates and unrelated tasks respectively.

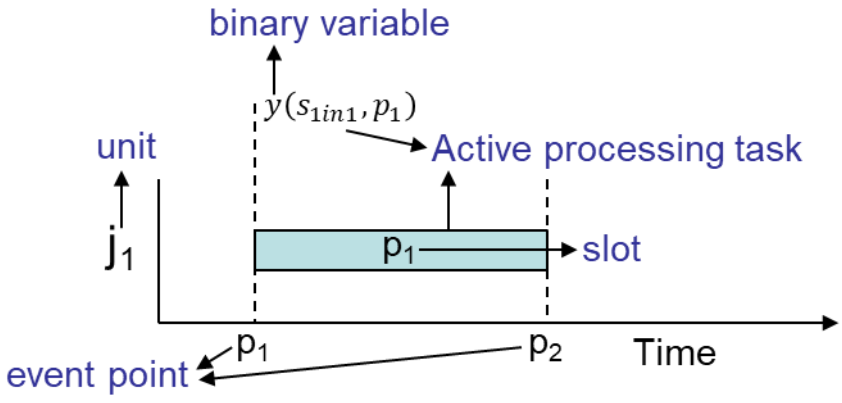


Figure 2-7. Generic Schedule Gantt Chart

In the optimization of a batch schedule, the aim is to determine the best possible configuration of tasks in time in order to maximize product throughput in a given

horizon, or minimize the time required to meet a given production demand. This has to be achieved while considering a number of physical constraints such that the resulting solution may actually be implemented. Some of the primary physical constraints include that a material state cannot be consumed in a quantity greater than that which is available at that time; that the excess available material state cannot exceed the capacity assigned for it and that a processing unit cannot perform more than one task at any given time. However, secondary constraints exist, such as those used to represent possible transfers of materials between units and storage vessels. This can be a difficult objective to achieve when just a few different products, units and resources exist for a given batch plant. The problem requires decisions to be made regarding the assignment of materials, units and resources to tasks, resulting in a highly combinatorial search. This is why the scheduling of batch plants is often considered in the short term.

The question then concerns the most accurate method of modelling physically feasible interactions while obeying the primary constraints and yet having a model which can actually return a useable solution in feasible time. As such, models have developed by including features to better represent reality, by facilitating interactions which were previously ignored or by solving flaws in representations of such features. Simplifying assumptions are, of course, also required. For instance, while batch reaction material balances and kinetics are often modelled with differential and nonlinear equations, such detail is often omitted in batch scheduling due to the hardships they impose on the solution procedure. Instead, many combinatorial batch scheduling problems assume that every processing task has a duration which is constant or at most a linear function of the batch size which it processes. This maintains the linearity of the scheduling problem. In practicality, it is desirable to design a tool that will provide accurate results using rigorous methods of solution, that requires the shortest time possible. Speed is of

the essence, for example, in reactive decision making during unexpected occurrences in plant operation.

The first attempts at improving the operating capacity or efficiency of batch chemical production plants were concerned with multiproduct plants. The earliest efforts at process scheduling optimization occurred in the late 1950s, however the concept only began to gain traction in the chemical industry with the advent of the STN and scheduling formulation by Kondili et al. (1993).

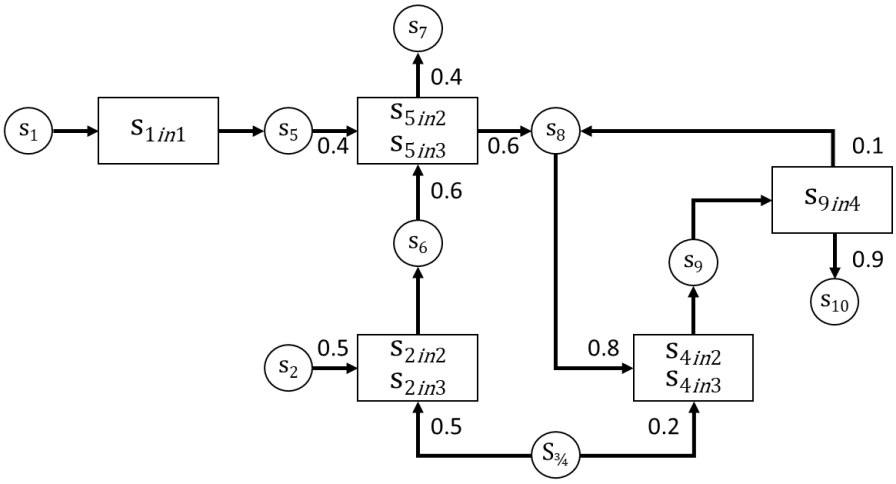


Figure 2-8. STN for a Network-Represented Batch Process

The state-task network (STN) is a bipartite graph used for the ambiguity free representation of processes (Kondili et al., 1993). There are two types of nodes present in the STN. Circular nodes represent different states, which are unique or distinct materials present in the process. Rectangular nodes represent tasks which act to convert a set of states into another set of states. Arcs display the precedence relationship between tasks, or the order which must be followed in order to produce products from raw materials, via any necessary intermediates. The STN includes data concerning the ratios of the different states required to begin a task as well as those produced from a task. It does not, however, include data concerning the capacity of units in the process or processing time data. A commonly studied, network-represented batch process is depicted in Figure 2-8.

Based on the STN, Kondili et al. (1993) developed a MILP for the determination of an optimal production schedule. The MILP was distinguished from its predecessors in its ability to address concepts present in multipurpose batch plants, including batch mixing and splitting, the assignment of units to tasks, variable batch sizes, using a unit for different tasks at different times and combinations of various storage policies. The model employs discrete-time representation.

In discrete, or *even* time representation, the time horizon is divided into even intervals of pre-specified duration. The ordinalities of the events are then exactly related to the times at which the events occur. A binary variable is defined for each task (in each applicable unit) at each potential event and activation of the task is represented by setting the binary variable to a value of 1. This class of time representation is displayed graphically in Figure 2-9.

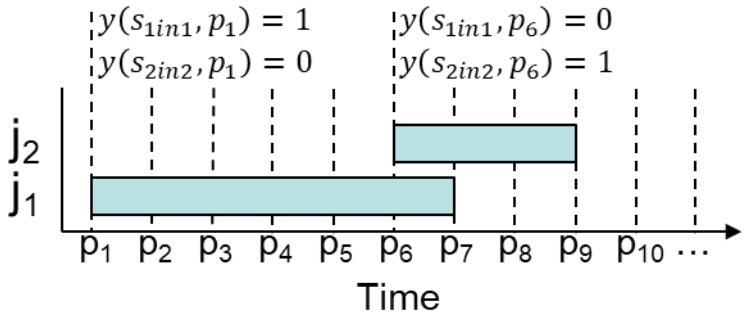


Figure 2-9. Discrete/Even Time Representation

The drawback of this approach is that a large number of binary variables are required in many cases. This impacts heavily on the duration of the solution procedure. Additionally, this technique does not allow for continuously variable processing time due to the predetermined event points at which events can start or finish. Therefore, a degree of inaccuracy may be present in the timing of tasks. However, a major advantage is the ease with which time-varying aspects can be accounted for, such as intermediate due dates, that is, product demands occurring before the end of the time horizon of interest, and changes in raw material or utility availability.

Pantelides (1994) introduced the resource-task network (RTN) framework for mathematical formulations and demonstrated how all process resources, such as material states, utilities, equipment and manpower can be addressed uniformly. The RTN expands on the STN by including a wider range of *states*, known as resources (Pantelides, 1994). These include material states of the STN as well as equipment units, utilities, availability of manpower etc. This is demonstrated in Figure 2-10 where equipment units are included as resources. The tasks are shown to consume the unit resource and regenerate it (upon completion of the task).

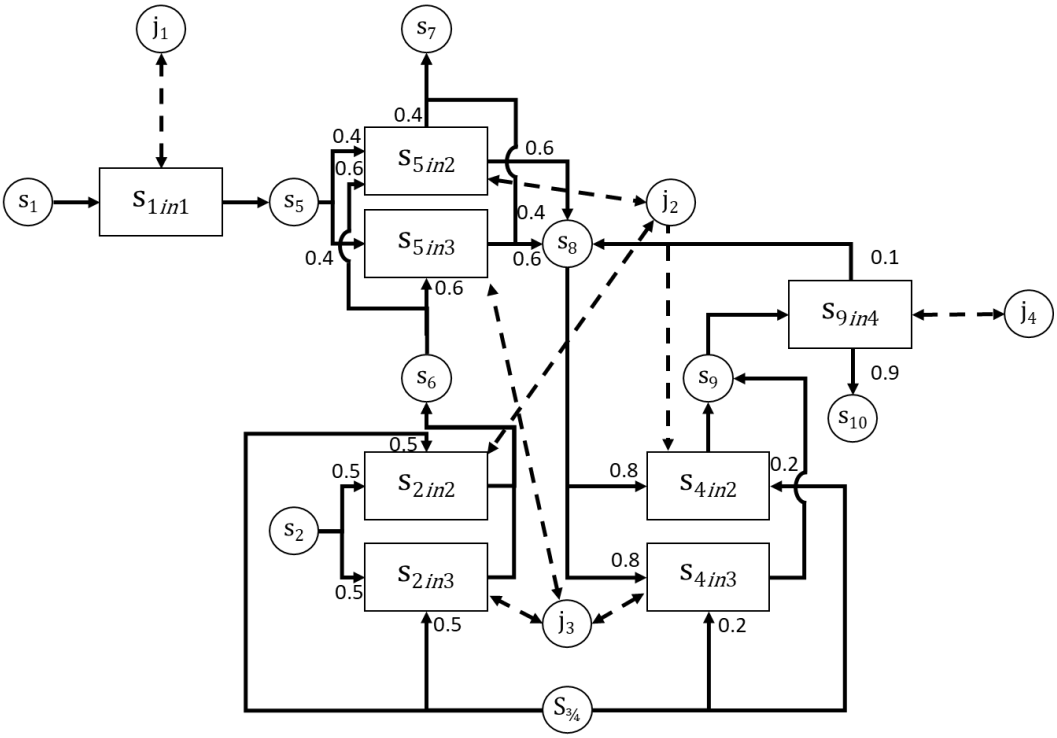


Figure 2-10. RTN for a Network-Represented Batch Process

The author also demonstrated how a wider array of process scheduling problems can be modelled by a single common formulation based on the RTN framework than is possible with one based on the STN framework. The latter often requires specialized constraints in order to address resource utilization and the many novel types of process scheduling problems. Finally, the RTN formulation proposed was shown to be computationally superior to equivalent STN formulations. This is due to its ability to

express tasks occurring in multiple units of the same type with a single binary variable. The result is both “smaller linear programming relaxations” and “reduced integer degeneracy in the solution of the MILP” (Pantelides, 1994). The formulation was still based on a discrete representation of time, however, resulting in the same issues of approximating the time dimension and the large problem size. The RTN is still used in more recent formulations, however it is not as popular as the STN.

An alternative framework was postulated by Smith and Pantelides (1995) which became known as the state-equipment network (SEN). The SEN is an approach to flowsheet and recipe representation (Smith & Pantelides, 1995). It assumes the material states and choice equipment units are known and these are represented as nodes on a diagram. Full connectivity between the equipment nodes exists. When used in batch scheduling, a sequence of tasks, or *operating modes*, are assigned to the units during optimization. Due to the use of detailed unit operation modelling associated with the SEN, no fixed mixing or splitting ratios of input and output states are pre-specified on the SEN. A simple representative example is given in Figure 2-11 (Nie et al., 2012).

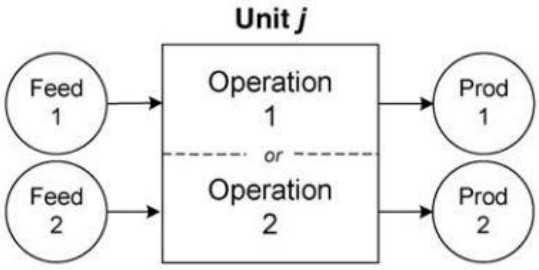


Figure 2-11. State-Equipment Network

As before, a superstructure is generated with state nodes and, this time, equipment nodes. Full connectivity between equipment units is assumed. The authors discussed how the SEN facilitates the inclusion of detailed unit operation modelling, which is advisable in order to obtain more accurate results which are more readily implemented and realised in practice. Originally, the SEN was intended for design and synthesis

optimization of continuous processes, however it was adapted to batch plant scheduling, such as in the formulation by Nie et al. (2012). This formulation implements rigorous detailed modelling of batch processes through dynamic optimization simultaneously with the scheduling aspect. The main difference in the scheduling module is that many variables, such as batch size and timing variables, are defined over units instead of tasks (like in the STN) which reduces the problem size when a unit may perform multiple tasks. Nonetheless, the popularity of the STN remained in future formulations due to the prioritization of tactical scheduling decisions over more accurate unit operation modelling as well as the lack of any significant advantage of the SEN in terms of generality and computational performance.

To overcome the limitations of the discrete-time representation, continuous or *uneven* time representations were developed. In continuous time representation, the exact positions of the events in time are not known beforehand, except that they are chronological, and the intervals are not necessarily equal. The spacing and positioning of the events are subject to optimization. Therefore, in this type of time representation, one or more continuous variables are declared for each event and used to describe the exact time at which the event occurs.

These techniques significantly reduce the number of binary variables required for scheduling and facilitate improved accuracy in the representation of time (Majozi, 2010). Redundant events are eliminated by allowing the assignment of events only to the start and end times of tasks as required. Continuous time variables match the ordered events to their placement in actual time, allowing the timings of tasks to vary continuously without being forced into predefined slots with predefined durations. However, these formulations require numerous big M constraints which increase the integrality gap of the model, thereby exacerbating the search for global optimality.

Continuous time formulations can be further categorised into global event point and unit-specific event point formulations. The latter is discussed in detail later, however, the former uses a common time grid for all units, thereby facilitating simple tracking of excess quantities of materials, units and resources. In *global* event-based time representation, sequencing constraints exist in order to ensure the chronological progression of each event and to allow tasks to continue over a number of adjacent events. Global continuous time event-based time representation is displayed in Figure 2-12.

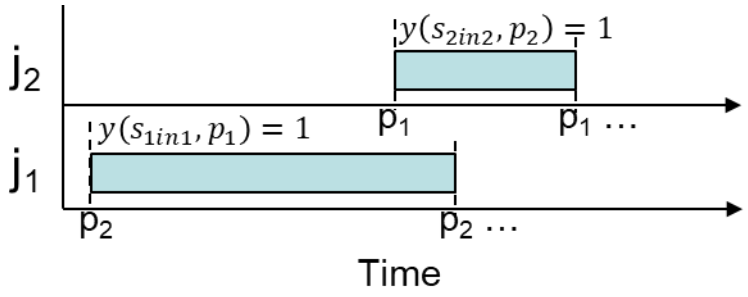


Figure 2-12. Global Event Continuous/Uneven Time Representation

Zhang and Sargent (1996) developed the first continuous time formulation for general network-represented processes based on the RTN. The formulation addressed batch as well as continuous processing and used monotonically increasing event points over a global time horizon. In order to address sequencing and timing of tasks, the formulation incorporated multi-dimensional binary variables as well as nonlinear, nonconvex products of variables. The authors also discussed solution techniques for the resulting large MINLPs.

Similarly, Mockus and Reklaitis (1997) developed a continuous time based formulation for general network-represented processes, however it was based on the STN. The authors included constraints to address limited renewable and non-renewable resources such as utilities and manpower. The resulting formulation contains bilinear terms, resulting again in large scale, nonconvex MINLP problems which can be linearized in

the case of simple objective functions. Otherwise, for problems involving the minimization of storage or utility costs, a modified outer approximation algorithm was proposed for the solution. The authors noted how the computational effort required for the solution of a particular problem is not always correlated to the problem size i.e. the number of binary and continuous variables and constraints, but that the problem structure and the values of the parameters also play an important role.

Schilling and Pantelides (1996) developed an RTN-based, continuous time, global event formulation for the scheduling of general network-represented batch plants. Instead of representing time using events spaced non-uniformly along the time horizon, the boundaries of which represent the start and end times of tasks, the formulation utilizes time slots of unknown duration. Additionally, the formulation requires that, in cases where multiple instances of a task begin at the same time, all of the instances must process the same quantity of material and therefore must be processed for the same duration. The formulation results in a MINLP with bilinear terms including products of binary and continuous variables, which can be readily linearized using the transformations in Glover (1975). The authors also proposed a novel solution procedure whereby branching occurs on the continuous variables for slot durations as well as binary ones in order to tighten the bounds on these variables.

An alternative to event or slot based formulations is that of precedence-based scheduling techniques. Perhaps the most noteable of these is the schedule-graph, or S-graph, developed by Sanmarti et al. (1998). The S-graph is a very computationally efficient method of representing and scheduling a class of multipurpose batch processes.

The framework is used in depicting network-represented processes and consists of nodes for process tasks and arcs indicating precedence relationships between the tasks.

There is one node for each stage of a product batch plus one additional node to indicate the completion of the batch. Each node contains information regarding the node identity and the unit in which the task occurs. The arcs may occur for two reasons: a recipe precedence relationship, existing due to the required order of processing a batch to correctly produce a product and/or equipment scheduling, which specifies the sequence of tasks occurring in a given unit. The arcs also contain information relating to the duration of tasks or required delay before units can be used. An example S-graph is displayed in Figure 2-13.

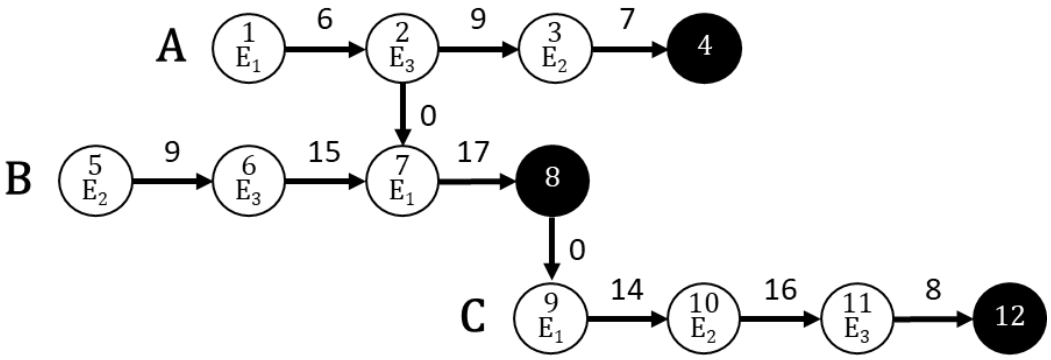


Figure 2-13. Schedule-Graph

Using the S-graph, Sanmarti et al. (1998) developed an algorithm which determines the minimum makespan required for a given product demand. A branch and bound technique is used where a lower bound for the makespan is computed using only recipe precedence relationships and a longest path algorithm. Tasks are then arbitrarily added to a given unit's sequence, thereby generating a branch. The longest path algorithm is implemented to determine the makespan at each node and an infeasibility detection algorithm is applied. Once an entire schedule is obtained, an upper bound is computed and suitable branches can be pruned. If unlimited waiting time for intermediates produced in tasks is impossible, a LP is solved to determine the makespan instead of the longest path algorithm. Once the entire tree has been explored, the optimal solution is determined. Due to the relative simplicity of the algorithms involved, compared to,

say those of solving MILPs, the algorithm is computationally efficient, especially when unlimited waiting times are assumed. However, in this approach, the applicability of the S-graph is limited to scheduling of processes with a prespecified number of batches for each product. It follows that profit maximization cannot be applied. Additionally, while the UIS and NIS policies are readily handled, the S-graph cannot be applied to processes requiring the FIS policy. Finally, constant processing times of each task must be specified a priori. Due to these limitations, event or slot based formulations are preferable over precedence based scheduling techniques for more general batch process scheduling.

Majozi and Friedler (2006) developed an algorithm based on the S-graph to facilitate varying numbers of product batches to be used in profit maximization studies. This was achieved by fixing the time horizon and searching combinations of the number of batches of each product against time-horizon feasibility criteria. A number of insights derived from the representation of the number of batches in a multidimentional plane facilitated extensive exclusion of suboptimal and infeasible possibilities from the subsequent schedule generation algorithm, thereby greatly improving computational performance over competing techniques. The methodology still requires fixed batch sizes and cannot be used for problems involving the FIS policy, however, approaches based on the S-graph constitute the only truly continuous time techniques due to their not requiring prespecification of the number of time or event points.

A further improvement to the representation of time in event-based, batch scheduling formulations was heralded by Ierapetritou and Floudas (1998), who proposed a STN-based, continuous time formulation for general network-represented processes. This formulation introduced the novel concept of a unique time grid for each unit in a given problem, allowing different tasks in different units which start at the same event to start

at different actual times. This means that the chronological ordering of two events occurring in different units does not imply anything about the relative timing of the two tasks. The authors also proposed the novel concept of decoupling task events from unit events. Additionally, the authors postulated the elimination of unnecessary binary and continuous variables by examining and pre-processing the STN for tasks which cannot take place at, or which do not provide value at certain events. The formulation drastically reduced the number of binary and continuous variables by eliminating unnecessary events at which certain tasks do not occur in certain units. This in turn reduced the computational time required to solve a given problem and afforded increased schedule flexibility. However, this asynchronicity requires complicated and stringent sequencing constraints in order to accurately monitor material and resource inventories. Unit-specific event-based time representation is displayed in Figure 2-14.

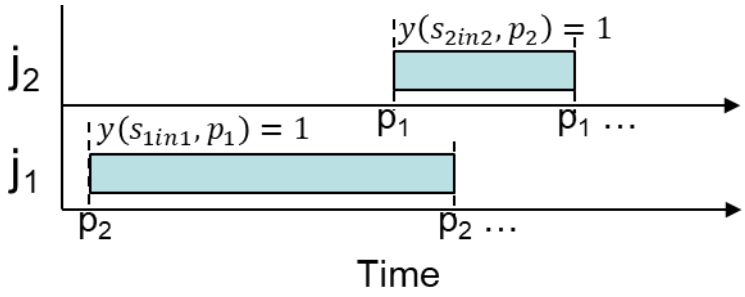


Figure 2-14. Unit-Specific Continuous/Uneven Time Representation

The additional sequencing considerations, which are required in unit-specific event-based formulations to manage production – consumption task couples, can result in difficulties regarding the FIS policy. In order to address this, the authors considered the addition of storage tasks and storage units to rigorously monitor the excess quantities of intermediate states.

Lin and Floudas (2001) proposed a continuous time, unit-specific event-based formulation, based on the STN which addressed the simultaneous synthesis, design and scheduling of multipurpose batch plants. The objective, among other possibilities, is an

economic minimization involving capital costs of dynamically chosen operating units offset by profits obtained via the production scheduling. The scheduling aspects of many of the constraints are based on those by Ierapetritou and Floudas (1998) and storage tasks are used to accurately model FIS considerations. However, the duration constraints for processing times and the cost functions for units include nonlinear terms in the form of powers of variables. The authors discuss how the resulting nonconvex MINLPs can be transformed into convex lower-bounding problems, thereby facilitating the ability of the model to determine globally optimal solutions in many cases.

Majozi and Zhu (2001) proposed an adapted STN, namely the SSN, in which only state nodes are required. This is due to the insight that the transformation of one state to another implicitly indicates the presence of a task and corresponding unit.

The state-sequence network (SSN) is also a method of recipe representation wherein only one type of node is required: the state node (Majozi & Zhu, 2001). This is due to the fact that two connected state nodes indicate a transition and imply the presence of a task and unit. The same batch process represented above using the STN and RTN are shown in Figure 2-15 with the SSN.

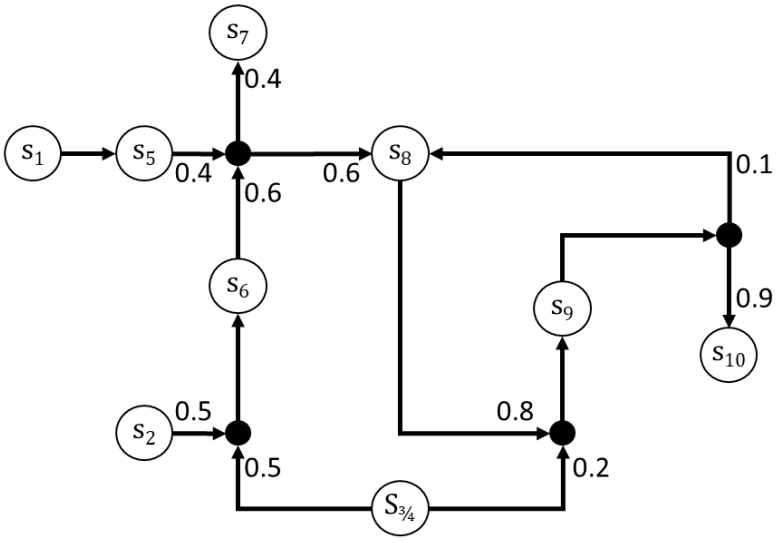


Figure 2-15. SSN for a Network-Represented Batch Process

The resulting continuous time, unit-specific event-based MILP formulation proposed requires only the definition of assignment binary variables based on an effective input state to a task and the event at which the task occurs – $y(s,p)$. An effective state is defined as *one* of the states required by the task and represents the subset of such states. However, this definition is effectively the same as a task, even for cases where a task can be performed in multiple units. This is because such tasks can be treated as separate tasks, resulting in a one to one task-unit ratio (Ierapetritou & Floudas, 1998). The result is that SSN-based formulations require the same number of binary variables as STN-based formulations and therefore the latter have remained more prevalent in recent batch scheduling models. The authors also discussed an alternative representation of task duration based on various influencing factors such as catalyst health, raw material purity and operator response time. Finally, the authors demonstrated how the use of aggregate modelling for multiple units of similar performance can be used to reduce the problem size without compromising accuracy in a number of examples.

Maravelias and Grossmann (2003) developed a continuous time, global event point formulation for the optimal scheduling of multipurpose batch plants based on the STN representation. This formulation, due to the use of the global event point approach, can easily facilitate consideration of resources. The authors discuss how global event point formulations are more general than unit-specific ones and can therefore find better solutions in certain examples, however they acknowledge the advantages that the latter afford in terms of model size and computational performance. Nevertheless, the solution times in the examples studied were of comparible values. The authors also introduce tightening constraints aimed at reducing the integrality gap associated with big M constraints and discuss their mechanisms in leading to improved solution

performance. The formulation addresses post-processing unit wait for tasks as well as sequence dependant changeover times and costs.

Post-processing unit wait, shown in Figure 2-16, is simply the ability of a unit, having performed a task, to continue to hold any produced intermediates subsequent to the completion of the task i.e. the unit is active for a duration exceeding the task's processing time. This introduces additional flexibility into resulting schedules by providing temporary storage for the produced intermediates and facilitating discharge at more convenient times. Note that this concept is not applicable to ZW intermediates. Storage vessels that can store one of a set of states are also addressed. Finally the authors discuss how makespan minimization problems are more computationally complex than profit maximization problems.

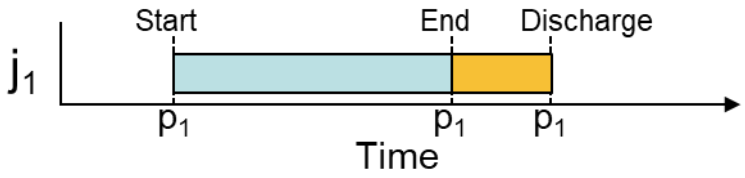


Figure 2-16. Post-processing Unit Wait

Floudas and Lin (2004) provide a review of developments in scheduling frameworks and formulations, which includes the major shift from discrete to continuous representations of the time horizon. Formulations by various authors are discussed while distinguishing and explaining key batch scheduling considerations such as sequential and general network-represented processes as well as global and unit-specific event-based formulations. The authors discuss how continuous time representations provide better computational performance by overcoming the challenges of explosive binary dimensions and approximations of the time horizon inherent in discrete-time representations. Furthermore, they note that while unit-specific event-based formulations tend to perform best in terms of computational time,

they require more complex sequencing constraints. However, continuous time formulations suffer from uncertainty regarding the globally optimal solution and a sufficient stopping criterion for the iterations through the number of event points considered. This is because better solutions may exist when more event points are considered.

Janak et al. (2004) developed a unit-specific event-based continuous time formulation based on the STN process representation. The authors introduced the concept of task splitting for unit-specific event-based formulations.

Task splitting, as demonstrated in Figure 2-17, refers to the assignment of an active task over a number of consecutive event points. In Janak et al. (2004), this increased the flexibility of resulting schedules by allowing tasks of different duration to begin at the same event. The concept is important in discrete time as well as global continuous time event-based formulations, due to the fact that all tasks beginning at a given event also begin at the same time. However, Janak et al. (2004) show how this is also important in unit-specific event-based formulations in order to accommodate accurate modelling of limited resource utilization.

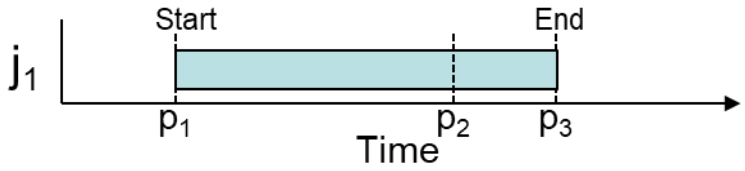


Figure 2-17. Task Splitting

The formulation is an extension of that in Ierapetritou and Floudas (1998) in that it addresses limited resources other than material and equipment, however in order to accomplish this, all tasks that consume a resource and that begin at a given event are forced to begin at the same time. The formulation uses two binary variables to indicate the start and end events of tasks, however it also utilises a continuous variable which determines whether or not a task is active at each event. This facilitates the splitting of

tasks over a number of event points while monitoring batch sizes and related variables. Additionally, storage tasks are used to accurately monitor the quantity of excess FIS states stored in dedicated storage units at a given point in time as they are produced and consumed by tasks at asynchronous events. As per resource consumption, all tasks beginning to consume a state at a given event must begin at the end time of the storage task for that state. The formulation is capable of effectively handling intermediate due dates by allowing the solver to select which tasks and which events are relevant for the production of a given order by a given time. Finally, the formulation contains many big M constraints as well as constraints written for every pair of events $(p, p'), p' \geq p$ which can be explosive in larger, more complex problems.

Janak and Floudas (2008) discuss how task splitting is required to address FIS scheduling, especially in certain examples containing recycle streams, in addition to addressing resource considerations. *Partial* task splitting, defined as only allowing tasks which produce or consume FIS states or recycled states, is implemented. The authors present a hybrid unit-specific event-based, continuous time formulation which combines the formulations of Ierapetritou and Floudas (1998) and Janak et al. (2004) in order to incorporate partial task splitting. The authors also introduce a number of techniques in the effort to reduce the large integrality gap inherent in unit-specific continuous time formulations. These include the pre-processing of a given STN in order to remove as many unnecessary binary and continuous variables as possible; the addition of tightening constraints and valid inequalities to bound the sums of key variables, the bounds of which are determined from the overall problem's relaxed solution as well as through the solution of supporting subproblems; and the implementation of a reformulation linearization technique which allows for tighter relaxations.

Shaik and Floudas (2008) presented the first unit-specific event-based continuous time formulation based on the RTN which supports the FIS policy without the need for considering storage as a separate task. This facilitated the elimination of binary and continuous variables as well as constraints which were previously necessary in order to address the FIS policy. This is accomplished by enforcing a zero-wait policy between production – consumption task pairs occurring at adjacent events for the same FIS state. The authors also propose a constraint to ensure that the starting time of consuming tasks is equal to the finishing time of producing tasks at the same event, in order to avoid real-time storage violations.

Shaik and Floudas (2009) proposed a novel formulation based on the STN for addressing both problems with and without resources in a unified way using three index binary and continuous variables to represent the start and end times of tasks. Again this is accomplished by enforcing that all tasks beginning or ending at a given event which produce or consume the same non-material resource must begin or end at the same time. The authors demonstrate the importance of allowing tasks to split over multiple events, even in unit-specific continuous time formulations without resource considerations, in order not to exclude globally optimal solutions in certain examples. A splitting parameter, ΔP in this work, is used to limit the maximum number of events over which a task is allowed to continue in order to control the problem size. However, this approach has a number of limitations. The splitting parameter has the ability to truncate the problem to the extent where potential globally optimal solutions may be excluded. Additionally, it requires that a nested iteration procedure be conducted through both ΔP and the maximum number of events considered for a given problem. This compounds the computational time and complexity of the solution procedure.

Pattinson and Majozi (2010) proposed some modifications to the formulation by Majozi and Zhu (2001) whereby the latent storage available in idle processing units could be utilized i.e. the PIS policy. Through application to a case study, the authors demonstrated that throughput could be improved by 50% when no dedicated storage vessels were present and that a 20% reduction in dedicated storage capacity could be achieved for a given throughput. Additionally, the concept was extended to the design of a multipurpose batch plant in a second case study. The concept clearly facilitates improvements in the scope of what is practically achievable in a batch production plant, however it has not featured in subsequent formulations. This is presumably due to the large increase in model size which results from the implementation of the PIS, with solution times increasing by a factor of 30 and 2 in the first case study compared to when PIS is not implemented. Furthermore, the design case study was reported to require a solution time of over three hours.

Vooradi and Shaik (2012) improved the formulation of Shaik and Floudas (2009) by introducing the concept of active task, thereby reducing the number of constraints and big M terms required. A number of improvements in the allocation, duration and sequencing constraints were also incorporated.

Seid and Majozi (2012) proposed a robust continuous time scheduling formulation based on the SSN. The formulation facilitated non-simultaneous material transfers or pre-processing unit wait, which is another important concept requiring explicit consideration and deliberate implementation in batch scheduling models.

Pre-processing unit wait is the ability to allow a task producing an intermediate (producing task) to transfer the intermediate to another unit in which it will subsequently be consumed, prior to the commencement of the consuming task. This allows debottlenecking of the producing unit and is especially effective when multiple intermediates, produced in different multipurpose units are required for a consuming task. This is demonstrated in Figure 2-18.

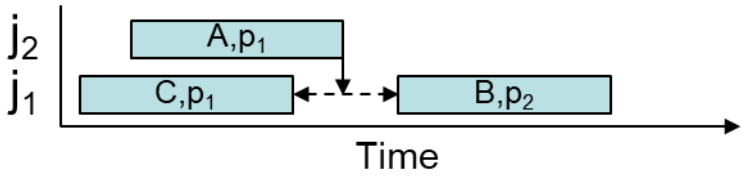


Figure 2-18. Pre-processing Unit Wait

Additionally, the formulation relaxed sequencing of production – consumption task pairs at adjacent events where the produced material is not directly required for the consuming task i.e. partial conditional sequencing was introduced.

Up to this point, batch scheduling formulations were predicated on *unconditional* sequencing, where all tasks which consume an intermediate must be sequenced in time to follow any tasks producing the intermediate (e.g. Ierapetritou & Floudas, 1998; Vooradi & Shaik, 2012). This is done in order to facilitate the FIS policy, however it excludes potential sequencing opportunities. Thus conditional sequencing, which sequences producing – consuming task pairs only when transfer occurs between them, was developed in order to obtain higher quality solutions. However, further complications in model formulation are required in order to achieve this. Figure 2-19 demonstrates a producing task and a consuming task (of the same intermediate) at adjacent events which do not respectively supply and consume the same specific batch of any intermediates and are therefore not sequenced accordingly. The figure also

demonstrates a pair which *is* sequenced due to the existence of such transfer. The former may occur if the feed material to the consuming task is available from other sources.

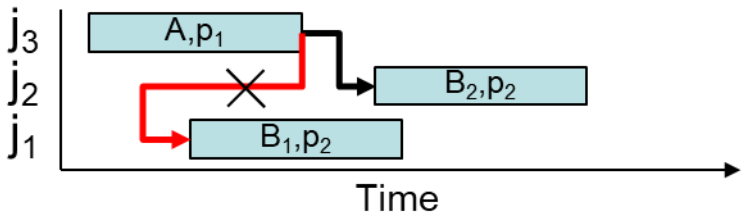


Figure 2-19. Conditional Sequencing

However, for the model in Seid and Majozi (2012), in cases where produced material is required in an adjacent consuming task, all consumption tasks at the following event are aligned with the producing task, hence the term *partial* conditional sequencing.

Sequencing was also relaxed where sufficient storage space is available for produced states, thereby not requiring the consuming task to begin immediately after production. These conditional sequencing aspects overcome drawbacks in previous methods of modelling the FIS policy, thereby facilitating more accurate solution of certain problems with fewer events, resulting in better objective values in shorter CPU times.

The model also simulated task-splitting by allowing units to continue to hold the materials they produced for subsequent events before eventual discharge at a later and more convenient event. However, the formulation lacked rigorous sequencing constraints to prevent real-time storage violations through the overlapping of pre- or post-processing unit wait with actual processing in a unit.

Vooradi and Shaik (2013) improved conditional sequencing by introducing rigorous conditional sequencing in which material flows between individual production – consumption task pairs are accurately monitored. This allowed only specific consumption tasks to be conditionally sequenced while others could be sequenced more flexibly. The model is structurally based on that of Vooradi and Shaik (2012) and Shaik

and Floudas (2009) before it. The formulation also addressed FIS violations that were present in the formulation by Seid and Majozi (2012). This formulation uses three-index variables and a splitting parameter similar to the formulation by Vooradi and Shaik (2012).

Shaik and Vooradi (2017) reformulated constraints in order to facilitate material transfer at the same event, unlike previous formulations which were predicated on material transfer at adjacent events. The model is also structurally based on the three-index formulation of Shaik and Floudas (2009) and uses a parameter to indicate the maximum number of events over which a task can split. The model is not capable of handling problems involving recycle streams however a hybrid approach was implemented in order to deal with states involved in recycle loops. The model provides substantial decreases in the required number of events to solve many multiproduct problems, however it is not capable of solving certain multipurpose problems involving the production and consumption of a state in the same unit to global optimality, due to the modelling of material transfer at the same event.

Similarly, Rakovitis et al. (2018) proposed a formulation where production-consumption task pairs are allowed to occur at the same event for tasks not involved in recycle loops. The constraints are much the same as those in Shaik and Vooradi (2017) except that the FIS policy is not adequately dealt with due to the lack of constraints regarding the maximum stored capacity of FIS intermediates. Additionally, a more comprehensive definition of recycling tasks is used.

The model was expanded by Rakovitis et al. (2019) to address conditional sequencing. Noteably, this formulation allows for intermediates to continue to wait in the units which produced them for a number of events while providing the necessary sequencing restrictions to allow the material to be extracted in fractions from the units.

Fractional extraction, demonstrated in Figure 2-20, refers to the ability of a unit to discharge one or multiple of its produced intermediates in fractions at different events and times. This can occur in multiple stages. The processing unit continues to hold the remaining portions of these intermediates until it is more convenient to discharge them to storage or to consuming tasks and therefore no subsequent tasks should occur in this unit until it has been completely emptied. Fractional extraction improves the flexibility of resulting schedules and addresses major drawbacks of previous formulations. However, in the model by Rakovitis et al. (2019), a binary variable is used for every unit at every event where intermediates may be temporarily stored. The formulation also uses the three index technique to address task splitting and does not include pre-processing unit wait or address ZW states.

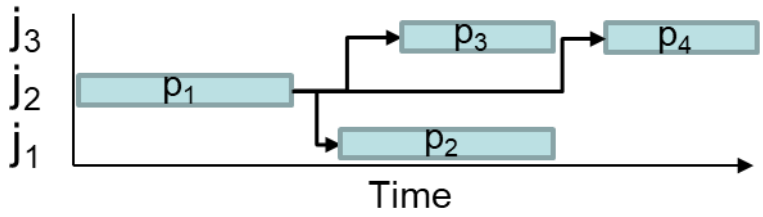


Figure 2-20. Fractional Extraction

The foundations for fractional extraction were present in the formulation by Seid and Majozi (2012), however it only become completely functional in the model by Rakovitis et al. (2019). It is developed further in this work.

Lee and Maravelias (2017) developed two MILPs for the scheduling of multipurpose batch processes belonging to the sequential environment. This is a class of problems in which batches are processed by one or multiple units in parallel through one or multiple product-specific stages and where batch mixing and splitting is not allowed. The authors addressed two drawbacks of previous formulations for this class of problems i.e. the simultaneous consideration of batching decisions and scheduling as well as the consideration of limited intermediate storage. The formulation is based on a discrete

representation of time which facilitates the ability of the model to address time-dependent availability of renewable resources. The formulation also allows the modelling of stage-dependent batch sizes however it assumes constant batch processing times for all tasks. A detailed comparison is performed comparing two types of models. In the first, all sub-batches for a given order are labelled explicitly and scheduled independently. In the second, feasible batch size intervals are identified to form bounds on the sizes of each sub-batch for a given order. This provides guidelines which should be used to model a given problem. It was shown how large scale problems, for which the time horizon is defined over hundreds of hours and for which hundreds of thousands of discrete variables are involved, can be solved for the minimization of cost, earliness or makespan in reasonable time.

Woolway and Majozi (2018) developed a novel scheduling framework by implementing a stochastic metaheuristic approach involving a coupled-chromosome genetic algorithm. The technique is aimed at addressing the scheduling of multipurpose batch facilities including tasks with variable processing times. It was shown to be very effective in drastically reducing the computational time requirements for large instances of two commonly studied examples to mere seconds. This is due to the fact that the algorithm does not scale exponentially with increased problem size. However, its stochastic nature did result in slightly inferior solutions compared to the MILP formulation it was tested against (Seid & Majozi, 2012).

Lee and Maravelias (2018) developed a scheduling algorithm for multipurpose batch processes belonging to the network environment which combines the relative strengths of continuous- and discrete-time formulations while overcoming their drawbacks. The algorithm consists of three stages: in the first stage, a discrete-time formulation is solved without too small a discretization interval in order to obtain an approximate solution.

In the second stage, a mapping algorithm is used to identify the activity of units and states present in the solution from the first stage for use in the third stage. This allows binary variables to be eliminated. In the third stage, resolution of timing variables is performed via the solving of a LP in order to improve the accuracy and quality of the original solution. The authors consider objective functions of profit maximization, cost minimization and makespan minimization. Discrete-time formulations allow for the straight forward modeling of time-varying resource availability and intermediate delivery and demand scenarios. This advantage is retained in the proposed methodology, while speedy solution times for large scale problems is facilitated. The algorithm assumes the UIS policy for intermediate states.

Puranik et al. (2018) present a systematic approach for the determination of infeasibility sources in the modelling of multipurpose batch processes. The aim is to identify constraints or groups of constraints which cause infeasibility in a model status. Specifically, constraints related to insufficient raw material, insufficient time horizon for processing or insufficient available capacity for processing intermediates, especially ZW intermediates form part of the investigation. An algorithm, referred to as the s-filter, which is an extension of the deletion filter for infeasibility analysis, is presented in order to isolate infeasible sets (IS) of constraints. Guidelines are given for pre-processing of the constraints performed for determining inviolable constraints, constraint weighting/ordering and constraint grouping, in order to facilitate the s-filter algorithm and improve the computational performance of the infeasibility analysis. The algorithm operates by successively eliminating groups of constraints and determining if the model is still infeasible. Unnecessary constraints (which do not render the infeasible model feasible) are eliminated until a small subset of constraints remains (the IS). The algorithm can be applied successively to determine multiple independent

infeasibilities. The authors discuss how the results can be meaningfully presented to plant operators with little to no optimization experience so that decisions can readily be made on how to correct the infeasibility.

2.3. Conclusions

The capability of a model to allow tasks to split, or to consume input states at a given event and produce output states at some later event, even for unit-specific event-based formulations, is very important as evidenced by a number of examples which cannot be solved to global optimality otherwise. The technique has been successfully achieved in a number of unit-specific event-based formulations using three-index binary variables, containing two indices for event points. For larger problems, defining binary variables for combinations of two event points $(p, p'), p' \geq p$ can lead to extremely large and often intractable problems. Therefore, in order to render more complex problems tractable, a limit on the maximum number of events over which a task can split is imposed. This is done through the use of a splitting parameter, referred to as ΔP in this work, which reflects this maximum number of events over which a task can split. It is not possible to know the optimum value of ΔP a priori, therefore it is determined iteratively. This iterative procedure is terminated, similar to the determination of the number of event points, by increasing its value until no change in the objective value is obtained for a given number of consecutive iterations.

While the issue of uncertainty regarding the globally optimal solution is ever present in event-based formulations, due to the potential of hidden superior solutions at a higher number of events, this problem is worsened through imposing the additional limit on the maximum number of events over which a task can split. Moreover, computational time requirements are also aggravated through the nested iteration present in these formulations. This effect has not been well investigated or discussed in the literature.

The present work is aimed at overcoming this limitation by simulating the concept of task splitting over any number of events without requiring increasing numbers of binary and continuous variables and iteration through ΔP. This is done while incorporating the concepts of rigorous conditional sequencing, pre- and post-processing unit wait and fractional extraction.

Chapter 3

Mathematical Model and Constraints

3.1. Introduction

The proposed model is capable of multipurpose batch scheduling for profit maximization or makespan minimization in general problems involving stream splitting and mixing, multiple units suitable for a given task, single units suitable for multiple different tasks, different storage policies such as FIS, ZW and NIS and any number of tasks with variable consumption and production ratios for any number of input and output streams. The model is capable of allowing pre- and post-processing unit wait and rigorous conditional sequencing. Task splitting is facilitated by allowing units to continue to hold material over multiple events, even for ZW states, without the need for separate indices to describe the start and end events of the task and without prefixing the maximum or iterating through the number of split events. Furthermore, the proposed formulation allows states to be fractionally extracted from a unit, thereby allowing a portion of produced states to wait for storage or transfer, or in the case of multiple products produced in a task, for some states to be removed while others continue to wait. The objective is for the proposed batch scheduling model to obtain better and more consistent solutions in faster time by allowing more flexibility and being more computationally efficient than existing models.

3.2. Assumptions

In the proposed formulation, tasks begin and end at the same event. Intermediates enter storage at the event after which they are produced, however products enter storage at the same event. This handling of intermediates is crucial for obtaining the globally

optimal solution in some multipurpose batch production setups where a state can be produced and consumed in the same unit, such as Example 5 appearing in Section 4.2.5. Additionally, it is assumed in this model and examples under study that raw materials are available as and when required and that there is unlimited storage available for product states. Simple constraints can readily be incorporated to restrict the raw material consumption or product storage if necessary. It is further assumed that profit maximization or makespan minimization occurs over a single cycle. Therefore, NS states, which are ZW and NIS states, cannot be consumed at the first event and ZW states cannot be produced at the last event. *All* variables and constraints relating to such tasks are eliminated in pre-processing before solving the model via rigorous exclusion. This means that, wherever possible, certain constraints are not written for certain states, units, tasks or events and other constraints are reduced to exclude unnecessary terms. If the model is to be applied to cyclic systems, these constraints can be relaxed. As such, *generalised* pre-processing is performed for every example under study. The elimination of *STN-specific* tasks relating to tasks which cannot be performed or which do not add value at certain events, as proposed by Janak and Floudas (2008), is not performed. Finally, the proposed model does not include specific consideration for UIS states as this storage policy is the simplest and easiest to handle. It has been addressed at length in the literature. If the proposed model is to be applied to case studies involving UIS states, simple amendments can be made to the constraints or UIS states can be treated as FIS states having a very large dedicated storage capacity.

3.3. Model Constraints

The proposed model is comprised of the following constraints, which are expressed below in written equations. The constraint and equation number are the same.

3.3.1 Allocation Constraints

Constraint 1 states that a maximum of one task can begin in a unit j at any given event p. This allows for situations in which *no* task begins in unit j at event p. The constraint is written for every unit j at every event p where *more than one* task is possible.

$$\sum_{s_{in,j} \in S^J_{in,J}} y(s_{in,j}, p) \leq 1, \quad \forall\, j \in J, p \in P \tag{1}$$

3.3.2 Capacity Constraints

Constraints 2 and 3 respectively limit batch sizes for every task to be between their maximum and minimum capacities if the task is active and equal to zero otherwise.

$$m_u(s_{in,j}, p) \leq V^U(s_{in,j})\, y(s_{in,j}, p), \quad \forall\, s_{in,j} \in S_{in,J}, p \in P \tag{2}$$

$$m_u(s_{in,j}, p) \geq V^L(s_{in,j})\, y(s_{in,j}, p), \quad \forall\, s_{in,j} \in S_{in,J}, p \in P \tag{3}$$

3.3.3 Material Balance Constraints

Constraints 4 and 5 are only written for tasks which consume states in variable ratios. The variable $m_c^V(s, s_{in,j}, p)$ is defined for each state consumed in these tasks and must lie between the minimum and maximum allowed ratio for the state if the task is active i.e. if $m_u(s_{in,j}, p)$ is positive. Otherwise it is equal to zero.

$$m_c^V(s, s_{in,j}, p) \leq \rho_c^U(s, s_{in,j})\, m_u(s_{in,j}, p), \quad \forall\, s \in (S^R \cup S^I), s_{in,j} \in (C_{in,J} \cap C^V_{in,J}), p \in P \tag{4}$$

$$m_c^V(s, s_{in,j}, p) \geq \rho_c^L(s, s_{in,j})\, m_u(s_{in,j}, p), \quad \forall s \in (S^R \cup S^I), s_{in,j} \in (C_{in,J} \cap C^V_{in,J}), p \in P \tag{5}$$

Constraints 6 and 7 are similar to 4 and 5 but apply to tasks which *produce* states in variable ratios. This occurs, for example, in the Westenberger-Kallrath problem (Kallrath, 2002).

$$m_p^V(s, s_{in,j}, p) \leq \rho_p^U(s, s_{in,j})\, m_u(s_{in,j}, p), \quad \forall\, s \in (S^I \cup S^P), s_{in,j} \in (P_{in,J} \cap P^V_{in,J}), p \in P \tag{6}$$

$$m_p^V(s, s_{in,j}, p) \geq \rho_p^L(s, s_{in,j})\, m_u(s_{in,j}, p), \quad \forall\, s \in (S^I \cup S^P), s_{in,j} \in (P_{in,J} \cap P^V_{in,J}), p \in P \tag{7}$$

Constraint 8 ensures that the total batch size for tasks which consume states in variable ratios is equal to the sum of its parts. Constraint 9 does the same for tasks which *produce* states in variable ratios.

$$m_u(s_{in,j},p) = \sum_{s \in S_J^C} m_c^V(s,s_{in,j},p), \quad \forall\, s_{in,j} \in C_{in,J}^V, p \in P \tag{8}$$

$$m_u(s_{in,j},p) = \sum_{s \in S_J^P} m_p^V(s,s_{in,j},p), \quad \forall\, s_{in,j} \in P_{in,J}^V, p \in P \tag{9}$$

Constraint 10 is written for all raw material states and calculates the total required quantity of each over the time horizon of interest. This quantity is contributed to by tasks which consume it in both variable and constant ratios. In the case of constant ratios, the upper bound for the ratio of the state consumed in the task is used.

$$q_T(s) = \sum_{p \in P} \left[\sum_{s_{in,j} \in \left(C_{in,J} \cap C_{in,J}^V\right)} m_c^V(s,s_{in,j},p) + \sum_{s_{in,j} \in \left(C_{in,J} \setminus C_{in,J}^V\right)} \rho_c^U(s,s_{in,j})\, m_u(s_{in,j},p) \right], \quad \forall\, s \in S^R \tag{10}$$

Constraint 11 is similar to Constraint 10, however it refers to the total quantity of product states produced over the time horizon of interest. The constraint assumes that there is no initial inventory of any product states, however this can be modified as necessary.

$$q_T(s) = \sum_{p \in P} \left[\sum_{s_{in,j} \in \left(P_{in,J} \cap P_{in,J}^V\right)} m_p^V(s,s_{in,j},p) + \sum_{s_{in,j} \in \left(P_{in,J} \setminus P_{in,J}^V\right)} \rho_p^U(s,s_{in,j})\, m_u(s_{in,j},p) \right], \quad \forall\, s \in S^P \tag{11}$$

Constraints 12 and 13 track the excess available quantity of FIS states at each event over the time horizon of interest. Constraint 12 is written at the first event and Constraint 13 is for subsequent events. The difference is that Constraint 12 considers

the initial inventory of the state and does not include terms for the production of the state, since produced FIS states only reflect in storage at the event after which they are produced.

$$q(s,p) = q^0(s) - \sum_{s_{in,j} \in \left(P_{in,J} \cap P^V_{in,J}\right)} m^V_c\left(s, s_{in,j}, p\right) - \sum_{s_{in,j} \in \left(P_{in,J} \setminus P^V_{in,J}\right)} \rho^U_c\left(s, s_{in,j}\right) m_u\left(s_{in,j}, p\right), \quad \forall\, s \in S^{FIS}, p \in P, p = 1 \tag{12}$$

$$\begin{aligned} q(s,p) = q(s,p-1) &+ \sum_{s_{in,j} \in \left(P_{in,J} \cap P^V_{in,J}\right)} m^V_p\left(s, s_{in,j}, p-1\right) \\ &+ \sum_{s_{in,j} \in \left(P_{in,J} \setminus P^V_{in,J}\right)} \rho^U_p\left(s, s_{in,j}\right) m_u\left(s_{in,j}, p-1\right) \\ &- \sum_{s_{in,j} \in \left(C_{in,J} \cap C^V_{in,J}\right)} m^V_c\left(s, s_{in,j}, p\right) \\ &- \sum_{s_{in,j} \in \left(C_{in,J} \setminus C^V_{in,J}\right)} \rho^U_c\left(s, s_{in,j}\right) m_u\left(s_{in,j}, p\right), \\ &\forall\, s \in S^{FIS}, p \in P, p > 1 \end{aligned} \tag{13}$$

Constraint 14 states that if a FIS state-consuming task, which requires more of a particular FIS state than is available in storage (excess state which was produced at $p-2$ or earlier), should begin at event p, then it must receive material produced at $p-1$ by another task or instance of a task i.e. transfer mechanism b_1. A positive value for b_1 activates the binary variable z, which is used in Constraint 40 to enforce the condition that the consuming task must begin after the producing task finishes, since the produced material is being used in the consuming task. This way, conditional sequencing is facilitated, allowing certain consuming tasks to be placed more flexibly when they are not consuming material produced by tasks at $p-1$. If exclusion of tasks which produce ZW states at the last event or tasks which consume NS states at the first event result in a zero-valued left-hand side or zero valued summation on the right-hand side, the

constraint can be excluded altogether for that FIS state at that event, since this reduces the constraint to a simple storage and material balance.

Note that b_1 exists for all production – consumption task pairs for UW states (FIS and NIS states). Its corresponding binary variable, z, only exists for production – consumption task pairs that occur in different units, since the sequencing, for which z exists, is taken care of in Constraint 39 for production – consumption tasks in the same unit. This is different for ZW states.

$$\sum_{s_{in,j} \in \left(C_{in,J} \cap C^V_{in,J}\right)} m^V_c(s, s_{in,j}, p) + \sum_{s_{in,j} \in \left(C_{in,J} \setminus C^V_{in,J}\right)} \rho^U_c(s, s_{in,j})\, m_u(s_{in,j}, p) \leq q(s, p-1) + \sum_{s_{in,j} \in P_{in,J}} \sum_{s'_{in,j} \in C_{in,J}} b_1(s_{in,j}, s'_{in,j}, s, p), \quad \forall\, s \in S^{FIS}, p \in P, p > 1 \tag{14}$$

Constraints 15a and 15b ensure that the tasks responsible for production of FIS states are active and that batch sizes are sufficient for transfer to consuming tasks via the mechanism b_1, if the transfer is indeed occurring. Otherwise the constraints are trivially satisfied. Constraint 15a is for tasks which produce states in variable ratios and Constraint 15b is for those producing states in constant ratios.

$$m^V_p(s, s_{in,j}, p) \geq \sum_{s'_{in,j} \in C_{in,J}} b_1(s_{in,j}, s'_{in,j}, s, p+1), \quad \forall\, s \in S^{FIS}, s_{in,j} \in \left(P_{in,J} \cap P^V_{in,J}\right), p \in P, p < P \tag{15a}$$

$$\rho^U_p(s, s_{in,j})\, m_u(s_{in,j}, p) \geq \sum_{s'_{in,j} \in C_{in,J}} b_1(s_{in,j}, s'_{in,j}, s, p+1), \quad \forall\, s \in S^{FIS}, s_{in,j} \in \left(P_{in,J} \setminus P^V_{in,J}\right), p \in P, p < P \tag{15b}$$

Constraints 16 – 19 carefully monitor the production, temporary in-unit storage, transfer and consumption of NS states. Constraints 16a and 16b deal with the production of NS states at the first event for variable production ratios and constant production ratios respectively.

$$m^V_p(s, s_{in,j}, p) = u(s, s_{in,j}, p+1) + \sum_{s'_{in,j} \in C_{in,J}} b_1(s_{in,j}, s'_{in,j}, s, p+1), \tag{16a}$$

$$\forall\, s \in S^{NS}, s_{in,j} \in \left(P_{in,J} \cap P_{in,J}^{V}\right), p \in P, p = 1$$

$$\rho_p^U\left(s, s_{in,j}\right) m_u\left(s_{in,j}, p\right) = u\left(s, s_{in,j}, p+1\right) + \sum_{s'_{in,j} \in C_{in,J}} b_1\left(s_{in,j}, s'_{in,j}, s, p+1\right), \tag{16b}$$
$$\forall\, s \in S^{NS}, s_{in,j} \in \left(P_{in,J} \setminus P_{in,J}^{V}\right), p \in P, p = 1$$

Constraints 17a and 17b deal with the production and temporary in-unit storage of NS states at intermediate events.

$$m_p^V\left(s, s_{in,j}, p\right) + u\left(s, s_{in,j}, p\right) = u\left(s, s_{in,j}, p+1\right) + \sum_{s'_{in,j} \in C_{in,J}} b_1\left(s_{in,j}, s'_{in,j}, s, p+1\right), \tag{17a}$$
$$\forall\, s \in S^{NS}, s_{in,j} \in \left(P_{in,J} \cap P_{in,J}^{V}\right), p \in P, 1 < p < P-1$$

$$\rho_p^U\left(s, s_{in,j}\right) m_u\left(s_{in,j}, p\right) + u\left(s, s_{in,j}, p\right) = u\left(s, s_{in,j}, p+1\right) + \sum_{s'_{in,j} \in C_{in,J}} b_1\left(s_{in,j}, s'_{in,j}, s, p+1\right), \tag{17b}$$
$$\forall\, s \in S^{NS}, s_{in,j} \in \left(P_{in,J} \setminus P_{in,J}^{V}\right), p \in P, 1 < p < P-1$$

At the last event, NIS states are allowed to wait in the unit at the end of production whereas for ZW states, there is no production or storage allowed. Constraints 18a and 18b deal with production and temporary in-unit storage of NIS states at the penultimate event, whereas Constraints 19a and 19b address ZW states at the penultimate event.

$$m_p^V\left(s, s_{in,j}, p\right) + u\left(s, s_{in,j}, p\right) = u\left(s, s_{in,j}, p+1\right) + \sum_{s'_{in,j} \in C_{in,J}} b_1\left(s_{in,j}, s'_{in,j}, s, p+1\right), \tag{18a}$$
$$\forall\, s \in S^{NIS}, s_{in,j} \in \left(P_{in,J} \cap P_{in,J}^{V}\right), p \in P, p = P-1$$

$$\rho_p^U\left(s, s_{in,j}\right) m_u\left(s_{in,j}, p\right) + u\left(s, s_{in,j}, p\right) = u\left(s, s_{in,j}, p+1\right) + \sum_{s'_{in,j} \in C_{in,J}} b_1\left(s_{in,j}, s'_{in,j}, s, p+1\right), \tag{18b}$$
$$\forall\, s \in S^{NIS}, s_{in,j} \in \left(P_{in,J} \setminus P_{in,J}^{V}\right), p \in P, p = P-1$$

$$m_p^V\left(s, s_{in,j}, p\right) + u\left(s, s_{in,j}, p\right) = \sum_{s'_{in,j} \in C_{in,J}} b_1\left(s_{in,j}, s'_{in,j}, s, p+1\right), \tag{19a}$$
$$\forall\, s \in S^{ZW}, s_{in,j} \in \left(P_{in,J} \cap P_{in,J}^{V}\right), p \in P, p = P-1$$

$$\rho_p^U\left(s, s_{in,j}\right) m_u\left(s_{in,j}, p\right) + u\left(s, s_{in,j}, p\right) = \sum_{s'_{in,j} \in C_{in,J}} b_1\left(s_{in,j}, s'_{in,j}, s, p+1\right), \tag{19b}$$
$$\forall\, s \in S^{ZW}, s_{in,j} \in \left(P_{in,J} \setminus P_{in,J}^{V}\right), p \in P, p = P-1$$

Note that temporary in-unit storage of ZW states occurs over event points only and is not allowed to extend in actual time, which would be akin to allowing post-processing unit wait for tasks which produce ZW states. This is enforced in Constraint 48.

Constraints 20a and 20b ensure that the tasks consuming FIS states are active and that batch sizes are large enough to accommodate what is transferred from producing tasks via the mechanism b_1, if the transfer is indeed occurring. Otherwise the constraints are trivially satisfied. Constraint 20a is for tasks which consume states in variable ratios and Constraint 20b is for those consuming states in constant ratios.

$$m_c^V(s, s'_{in,j}, p) \geq \sum_{\substack{s_{in,j} \in P_{in,J} \\ \in P, p > 1}} b_1(s_{in,j}, s'_{in,j}, s, p), \quad \forall\, s \in S^{FIS}, s'_{in,j} \in (C_{in,J} \cap C_{in,J}^V), p \tag{20a}$$

$$\rho_c^U(s, s'_{in,j})\, m_u(s'_{in,j}, p) \geq \sum_{\substack{s_{in,j} \in P_{in,J} \\ \in (C_{in,J} \setminus C_{in,J}^V), p \in P, p > 1}} b_1(s_{in,j}, s'_{in,j}, s, p), \quad \forall\, s \in S^{FIS}, s'_{in,j} \tag{20b}$$

Constraints 21a and 21b carefully monitor the consumption of NS states regarding the transfer from producing tasks and in-unit temporary storage.

$$m_c^V(s, s'_{in,j}, p) = \sum_{\substack{s_{in,j} \in P_{in,J} \\ \in P, p > 1}} b_1(s_{in,j}, s'_{in,j}, s, p), \quad \forall\, s \in S^{NS}, s'_{in,j} \in (C_{in,J} \cap C_{in,J}^V), p \tag{21a}$$

$$\rho_c^U(s, s'_{in,j})\, m_u(s'_{in,j}, p) = \sum_{\substack{s_{in,j} \in P_{in,J} \\ \in (C_{in,J} \setminus C_{in,J}^V), p \in P, p > 1}} b_1(s_{in,j}, s'_{in,j}, s, p), \quad \forall\, s \in S^{NS}, s'_{in,j} \tag{21b}$$

Constraint 22 deals with the FIS policy and prevents violations by controlling the upper limit on storage. Material continuing to wait, in a unit which produced it, at subsequent events is considered in this constraint. This effectively increases the maximum storage capacity. The term b_2 reflects the freeing up of storage space via one of two mechanisms. One mechanism is the transfer of produced material into a consuming unit directly, for pre-processing unit wait or immediate consumption, which is controlled via the binary variable v. This requires that the producing task at event $p-1$ ends after

any other possible task in the consuming unit at event $p-1$ so that the unit is free to receive material for consumption at event p. The other mechanism is that some of the FIS state is consumed at event p at a time prior to the end of production at event $p-1$, thereby ensuring that there is enough storage space for the produced material. This is controlled via the binary variable x. These two mechanisms work together to ensure that there are sufficient options available for flexible production and consumption interaction without any FIS violations.

If, at event 2, the summations on the left-hand side have a zero value due to exclusion of tasks which consume NS states at the first event, the constraint need not be written for the particular FIS state at event 2. However, if the double summation on the right-hand side has a zero value due to exclusion of tasks which produce ZW states at the last event, the constraint still needs to be written in order to prevent storage violations.

$$\begin{aligned} q(s,p-1) + & \sum_{s_{in,j} \in \left(P_{in,J} \cap P^{V}_{in,J}\right)} m^{V}_{p}\left(s, s_{in,j}, p-1\right) \\ & + \sum_{s_{in,j} \in \left(P_{in,J} \setminus P^{V}_{in,J}\right)} \rho^{U}_{p}\left(s, s_{in,j}\right) m_{u}\left(s_{in,j}, p-1\right) \\ & \leq Q^{U}(s) + \sum_{s_{in,j} \in P_{in,J}} u\left(s, s_{in,j}, p\right) \\ & + \sum_{s_{in,j} \in P_{in,J}} \left[\sum_{s'_{in,j} \in C_{in,J}} b_{2}\left(s_{in,j}, s'_{in,j}, s, p\right) \right], \\ & \forall\, s \in S^{FIS}, p \in P, p > 1 \end{aligned} \tag{22}$$

Constraints 23 – 24 are similar to 15a and 15b except that they apply to mechanism b_2. Note that in the case of b_2 transfer, some of the transferred material can come from temporary in-unit storage. Constraints 23a and 24a are for tasks which produce states in variable ratios and Constraints 23b and 24b are for those producing states in constant ratios. Constraints 23a and 23b are for the first event and Constraints 24a and 24b are for all intermediate events.

$$m_p^V(s, s_{in,j}, p) \geq \sum_{s'_{in,j} \in C_{in,J}} b_2(s_{in,j}, s'_{in,j}, s, p+1), \quad \forall\, s \in S^{FIS}, s_{in,j} \in (P_{in,J} \cap P_{in,J}^V), p \in P, p = 1 \tag{23a}$$

$$\rho_p^U(s, s_{in,j})\, m_u(s_{in,j}, p) \geq \sum_{s'_{in,j} \in C_{in,J}} b_2(s_{in,j}, s'_{in,j}, s, p+1), \quad \forall\, s \in S^{FIS}, s_{in,j} \in (P_{in,J} \setminus P_{in,J}^V), p \in P, p = 1 \tag{23b}$$

$$m_p^V(s, s_{in,j}, p) + u(s, s_{in,j}, p) \geq \sum_{s'_{in,j} \in C_{in,J}} b_2(s_{in,j}, s'_{in,j}, s, p+1), \quad \forall\, s \in S^{FIS}, s_{in,j} \in (P_{in,J} \cap P_{in,J}^V), p \in P, 1 < p < P \tag{24a}$$

$$\rho_p^U(s, s_{in,j})\, m_u(s_{in,j}, p) + u(s, s_{in,j}, p) \geq \sum_{s'_{in,j} \in C_{in,J}} b_2(s_{in,j}, s'_{in,j}, s, p+1), \quad \forall\, s \in S^{FIS}, s_{in,j} \in (P_{in,J} \setminus P_{in,J}^V), p \in P, 1 < p < P \tag{24b}$$

Constraints 25a and 25b are similar to Constraints 20a and 20b except that they apply to mechanism b_2. Constraint 25a is for tasks which consume states in variable ratios and Constraint 25b is for those consuming states in constant ratios.

$$m_c^V(s, s'_{in,j}, p) \geq \sum_{s_{in,j} \in P_{in,J}} b_2(s_{in,j}, s'_{in,j}, s, p), \quad \forall\, s \in S^{FIS}, s'_{in,j} \in (C_{in,J} \cap C_{in,J}^V), p \in P, p > 1 \tag{25a}$$

$$\rho_c^U(s, s'_{in,j})\, m_u(s'_{in,j}, p) \geq \sum_{s_{in,j} \in P_{in,J}} b_2(s_{in,j}, s'_{in,j}, s, p), \quad \forall\, s \in S^{FIS}, s'_{in,j} \in (C_{in,J} \setminus C_{in,J}^V), p \in P, p > 1 \tag{25b}$$

3.3.4 Storage Constraints

The variable u was introduced by Seid and Majozi (2012) in order to allow produced states to wait in the unit that produced them at subsequent events, thereby effectively increasing the maximum storage capacity. However necessary constraints which carefully control the quantity of stored material and sequencing of its discharge in order to prevent FIS violations and simultaneous processing and post-processing unit wait were lacking. The concept was also used by Rakovitis et al. (2019). This is updated and carefully managed in the current work whereby material in temporary in-unit storage is

only reduced via the b_2 mechanism. u can be used to allow production of a state exceeding maximum storage capacity for consumption at the subsequent event, however it can also be used to hold material over a number of events. In the latter case, production at event p results in temporary in-unit storage at event $p+1$ and consumption at some event $p' \geq p+2$. The actual quantity stored is precisely tracked until the unit is emptied. This ensures that if material is temporarily stored in a unit for a number of events, the start time of the following task in the unit is always appropriately late enough such that all of the stored material can be discharged to a consuming task. Constraints 26 – 31 and Constraint 48 enforce the necessary restrictions on the variable u.

Constraints 26 and 27 ensure that, at any given event, a unit is either providing temporary storage to produced states or processing states as part of a task and never both. Since u can only be reduced via transfer mechanism b_2, as per Constraints 28 – 30, Constraints 26 and 27 state that when any task begins in unit j, full transfer of any stored states to consuming tasks must have been completed. The constraints have been adapted from Seid and Majozi (2012) in that they include summations of all possible tasks producing states for temporary in-unit storage and are as such written for each unit as opposed to being written for each producing task individually. This reduces the number of constraints in the model. Constraint 26 is not written at the last event, since ZW materials cannot continue to wait in a unit at the last event. Constraint 27 is therefore written only for UW states at the last event.

$$\sum_{s \in S^I}\left[\sum_{s_{in,j} \in \left(P_{in,J} \cap S^J_{in,J}\right)} u(s, s_{in,j}, p)\right] \leq \max_{s_{in,j} \in S^J_{in,J}}\left[V^U(s_{in,j})\right]\left[1 - \sum_{s_{in,j} \in S^J_{in,J}} y(s_{in,j}, p)\right], \quad (26)$$

$$\forall\, j \in J, p \in P, 1 < p < P$$

$$\sum_{s \in S^{UW}} \left[\sum_{s_{in,j} \in \left(P_{in,J} \cap S^{J}_{in,J}\right)} u\left(s, s_{in,j}, p\right) \right] \leq \max_{s_{in,j} \in S^{J}_{in,J}} \left[V^{U}\left(s_{in,j}\right)\right] \left[1 - \sum_{s_{in,j} \in S^{J}_{in,J}} y\left(s_{in,j}, p\right) \right], \quad (27)$$

$$\forall\, j \in J, p \in P, p = P$$

Constraint 28 states that variable u must maintain its value from previous events unless discharge occurs via the b_2 mechanism.

$$u\left(s, s_{in,j}, p+1\right) \geq u\left(s, s_{in,j}, p\right) - \sum_{s'_{in,j} \in C_{in,J}} b_2\left(s_{in,j}, s'_{in,j}, s, p+1\right), \quad (28)$$

$$\forall\, s \in S^{FIS}, s_{in,j} \in\ P_{in,J}, p \in P, 1 < p < P$$

Constraints 29 and 30 ensure that material in temporary in-unit storage can only occur at the event after production or if it was already stored in the unit at the previous event. Together, Constraints 28 – 30 ensure rigorous upper and lower bounds for the variable u to exactly monitor its value over production and consumption via transfer.

$$u\left(s, s_{in,j}, p+1\right) \leq m^{V}_{p}\left(s, s_{in,j}, p\right) - \sum_{s'_{in,j} \in C_{in,J}} b_2\left(s_{in,j}, s'_{in,j}, s, p+1\right), \quad (29a)$$

$$\forall\, s \in S^{FIS}, s_{in,j} \in \left(P_{in,J} \cap P^{V}_{in,J}\right), p \in P, p = 1$$

$$u\left(s, s_{in,j}, p+1\right) \leq \rho^{U}_{p}\left(s, s_{in,j}\right) m_u\left(s_{in,j}, p\right) - \sum_{s'_{in,j} \in C_{in,J}} b_2\left(s_{in,j}, s'_{in,j}, s, p+1\right), \quad (29b)$$

$$\forall\, s \in S^{FIS}, s_{in,j} \in \left(P_{in,J} \setminus P^{V}_{in,J}\right), p \in P, p = 1$$

$$u\left(s, s_{in,j}, p+1\right) \leq m^{V}_{p}\left(s, s_{in,j}, p\right) + u\left(s, s_{in,j}, p\right) - \sum_{s'_{in,j} \in C_{in,J}} b_2\left(s_{in,j}, s'_{in,j}, s, p+1\right), \quad (30a)$$

$$\forall\, s \in S^{FIS}, s_{in,j} \in \left(P_{in,J} \cap P^{V}_{in,J}\right), p \in P, 1 < p < P$$

$$u\left(s, s_{in,j}, p+1\right) \leq \rho^{U}_{p}\left(s, s_{in,j}\right) m_u\left(s_{in,j}, p\right) + u\left(s, s_{in,j}, p\right) - \sum_{s'_{in,j} \in C_{in,J}} b_2\left(s_{in,j}, s'_{in,j}, s, p+1\right), \quad (30b)$$

$$\forall\, s \in S^{FIS}, s_{in,j} \in \left(P_{in,J} \setminus P^{V}_{in,J}\right), p \in P, 1 < p < P$$

3.3.5 Linking Constraints

Constraints 28 – 30 are written for FIS states only because strict control for the variable u for NS states is worked into the material balances in Constraints 16 – 19. However, for ZW states, additional control is required to ensure that temporary in-unit storage does not result in post-processing unit wait. This is done using the variable w in Constraints 31 and 48.

$$u(s, s_{in,j}, p) \leq \rho_p^U(s, s_{in,j})\, V^U(s_{in,j})\, w(s_{in,j}, p), \quad \forall\, s \in S^{ZW}, s_{in,j} \in P_{in,J}, p \in P, 1 < p < P \qquad (31)$$

It is noted here that the variables b_1, b_2, z, v, x, u and w do not exist at the first event for any tasks and states, since they represent production at the previous event to which they are written.

Constraint 32 links the continuous variable b_1 to its corresponding binary variable z. This is necessary in order to provide rigorous conditional sequencing based on whether b_1 takes a zero or positive value. The coefficient for binary variable z is the minimum of the maximum quantity of produced state in the producing task and the maximum quantity of consumed state in the consuming task, since the quantity transferred is limited by the smaller of these values. In Vooradi and Shaik (2013), binary variables z and v, for transfer of NIS states, both enforce that consumption of NIS states occurs at a later time relative to its production at the previous event, as per *their* Constraints 20 and 39 respectively. Additionally, the condition that the start time of consumption must be less than or equal to the finish time of production, thereby ensuring equality of these times, is only enforced for variable z, as per their Constraint 35. For variable v, it is enforced that the finish time of production at event $p-1$ must lie between the finish time of all tasks occuring in the consuming unit at $p-1$ and the start time of the consuming task in the consuming unit at event p, as per their Constraint 28. This case

is a general case of the condition for variable z and to distinguish them for NIS states is not necessary. Therefore in this formulation, only one binary variable z is used to describe the sequencing for production – consumption of NIS states, and it takes on the more general role of that for binary variable v for NIS states in Vooradi and Shaik (2013).

$$b_1(s_{in,j}, s'_{in,j}, s, p) \leq \min\left[\rho_p^U(s, s_{in,j})\, V^U(s_{in,j}), \rho_c^U(s, s'_{in,j})\, V^U(s'_{in,j})\right] \times z(s_{in,j}, s'_{in,j}, s, p), \quad (32)$$

$$\forall\, j, j' \in J, j \neq j', s \in S^{UW}, s_{in,j} \in \left(P_{in,J} \cap S^J_{in,J}\right),\ s'_{in,j} \in \left(C_{in,J} \cap S^J_{in,J'}\right), p \in P, p > 1$$

Constraint 33 is written for ZW states and is similar to Constraint 32 except that it *must* be written even for production – consumption task pairs occurring in the same unit since this unique sequencing is not applied elsewhere.

$$b_1(s_{in,j}, s'_{in,j}, s, p) \leq \min\left[\rho_p^U(s, s_{in,j})\, V^U(s_{in,j}), \rho_c^U(s, s'_{in,j})\, V^U(s'_{in,j})\right] \times z(s_{in,j}, s'_{in,j}, s, p), \quad (33)$$

$$\forall\, s \in S^{ZW}, s_{in,j} \in P_{in,J}, s'_{in,j} \in C_{in,J}, \qquad p \in P, p > 1$$

Constraint 34 is similar to Constraints 32 and 33 except that it is written for transfer mechanism b_2.

$$b_2(s_{in,j}, s'_{in,j}, s, p) \leq \min\left[\rho_p^U(s, s_{in,j})\, V^U(s_{in,j}), \rho_c^U(s, s'_{in,j})\, V^U(s'_{in,j})\right] \times \left[x(s_{in,j}, s'_{in,j}, s, p) + v(s_{in,j}, s'_{in,j}, s, p)\right], \quad (34)$$

$$\forall\, j, j' \in J, j \neq j', s \in S^{FIS}, s_{in,j} \in \left(P_{in,J} \cap S^J_{in,J}\right), s'_{in,j} \in \left(C_{in,J} \cap S^J_{in,J'}\right), p \in P, p > 1$$

3.3.6 Duration Constraints

For tasks which produce only UW states, there is no need to include variable t_{in} at the first event, since the tasks can always begin at the start of the time horizon of interest. The material can then simply wait in the unit (post-processing unit wait at the same event) until whenever it should be discharged to storage or transferred to a consuming task. Similarly, there is no need to include variable t_{out} for such tasks at the last event,

as long as there is sufficient time between the start of the task and the end of the time horizon of interest. This way it is possible to exclude unnecessary continuous variables and reduce the model size. It is necessary to include t_{in} for tasks which produce at least one ZW state at the first event because such tasks cannot accommodate post-processing unit wait.

$$t_{out}(s_{in,j},p) \geq a(s_{in,j})\, y(s_{in,j},p) + b(s_{in,j})\, m_u(s_{in,j},p),\ \ \forall\, s_{in,j} \in (S_{in,J} \setminus P_{in,J}^{ZW}), p \in P, p = 1 \quad (35)$$

$$t_{out}(s_{in,j},p) \geq t_{in}(s_{in,j},p) + a(s_{in,j})\, y(s_{in,j},p) + b(s_{in,j})\, m_u(s_{in,j},p), \quad (36)$$

$$\forall\, s_{in}j \in (S_{in,J} \setminus P_{in,J}^{ZW}), p \in P, 1 < p < P$$

Constraint 37 doubles as an upper-bounding constraint for variables t_{out}, and by extension, t_{in}.

$$H \geq t_{in}(s_{in,j},p) + a(s_{in,j})\, y(s_{in,j},p) + b(s_{in,j})\, m_u(s_{in,j},p),\ \ \forall\, s_{in}j \in (S_{in,J} \setminus P_{in,J}^{ZW}), p \in P, p = P \quad (37)$$

Constraint 38 is excluded at the last event, since no tasks which produce ZW states are permitted at the last event. Note that no upper-bounding for start and end times are explicitly required for ZW-producing tasks, since all produced ZW material must be directly transferred and consumed by the last event, as per Constraints 19a and 19b ($u(s, s_{in,j}, p)$ does not exist at the last event for ZW states). Therefore, the upper-bounding applied to the consuming tasks will take care of this.

$$t_{out}(s_{in,j},p) = t_{in}(s_{in,j},p) + a(s_{in,j})\, y(s_{in,j},p) + b(s_{in,j})\, m_u(s_{in,j},p),\ \ \forall\, s_{in,j} \in P_{in,J}^{ZW}, p \in P, p < P \quad (38)$$

3.3.7 Sequence Constraints

Constraint 39 is written for all tasks occurring in a given unit. It combines the traditionally separated approaches for *same task in same unit* and *different task in same unit* sequencing constraints.

$$t_{in}(s'_{in,j}, p+1) \geq t_{out}(s_{in,j}, p), \quad \forall\, j \in J, s_{in,j}, s'_{in,j} \in S^J_{in,J}, p \in P, p < P \tag{39}$$

Constraint 40 is written for intermediate state production – consumption task pairs, that is, it represents the *different task in a different unit* constraints. It is only written for tasks which occur in different units because sequencing at adjacent events in the same unit is taken care of by Constraint 39. In Constraint 40, sequencing is only enforced if transfer actually occurs between tasks $s_{in,j}$ and $s'_{in,j}$. In other words, binary variable z must be activated by a positive valued b_1, which is done in Constraints 32 and 33.

$$t_{in}(s'_{in,j}, p) \geq t_{out}(s_{in,j}, p-1) - M[1 - z(s_{in,j}, s'_{in,j}, s, p)], \tag{40}$$

$$\forall\, j, j' \in J, j \neq j', s \in S^I, s_{in,j} \in (P_{in,J} \cap S^J_{in,J}), s'_{in,j} \in \left(C_{in,J} \cap S^J_{in,J'}\right), p \in P, p > 1$$

Constraint 41 is similar to Constraint 40 except that it is only written for production – consumption pairs involving ZW states and that the inequality is reversed. This is to ensure that ZW states are consumed immediately upon production in terms of time (even though it may be a few events later). A third difference is that it is written even when the production and consumption tasks occur in the same unit.

$$t_{in}(s'_{in,j}, p) \leq t_{out}(s_{in,j}, p-1) + M[1 - z(s_{in,j}, s'_{in,j}, s, p)], \tag{41}$$

$$\forall\, s \in S^{ZW}, s_{in,j} \in P_{in,J}, s'_{in,j} \in C_{in,J}, p \in P, p > 1$$

Constraint 42 addresses sequencing of production – consumption pairs transferring material via the mechanism b_2, whereby produced material is directly transferred to the consuming unit. The constraint deals with three separate tasks. $s_{in,j}$ is the producing task, occurring in unit j at event $p-1$. $s'_{in,j}$ is the consuming task, occurring in unit j' at event p and $s''_{in,j}$ is any other auxiliary task which may be occurring in the consuming unit j' at event $p-1$. The constraint states that the producing task must finish after the end of the auxiliary task, such that material can be transferred to the receiving unit

without any pre-processing unit wait overlapping in time with the processing of the auxiliary task. Units j and j' must be different.

$$t_{out}(s''_{in,j}, p-1) \leq t_{out}(s_{in,j}, p-1) + M[1 - v(s_{in,j}, s'_{in,j}, s, p)], \tag{42}$$

$$\forall\, j, j' \in J, s \in S^{FIS}, s_{in,j} \in (P_{in,J} \cap S^{J}_{in,j}), s'_{in,j} \in \left(C_{in,J} \cap S^{J}_{in,j'}\right), s''_{in,j} \in S^{J}_{in,j'}, p \in P, p > 1$$

Constraint 43 is similar to Constraint 42 except that it is applied to NIS states for which the binary variables z and v are combined. It is emphasized here that production of NIS states may begin at event $p' \leq p-1$, however when they are transferred from the producing unit to the consuming unit for consumption at event p, the time will be that of the finishing time of the producing task at event $p-1$.

$$t_{out}(s''_{in,j}, p-1) \leq t_{out}(s_{in,j}, p-1) + M[1 - z(s_{in,j}, s'_{in,j}, s, p)], \tag{43}$$

$$\forall\, j, j' \in J, j \neq j', s \in S^{NIS}, s_{in,j} \in (P_{in,J} \cap S^{J}_{in,j}), s'_{in,j} \in (C_{in,J} \cap S^{J}_{in,J}), s''_{in,j} \in S^{J}_{in,J}, p \in P, p > 1$$

Constraint 44 addresses sequencing of production – consumption pairs via the mechanism b_2, whereby a state is consumed at event p prior to production at event $p-1$ in order to ensure that sufficient storage space is available for the produced state.

$$t_{in}(s'_{in,j}, p) \leq t_{out}(s_{in,j}, p-1) + M[1 - x(s_{in,j}, s'_{in,j}, s, p)], \tag{44}$$

$$\forall\, j, j' \in J, j \neq j', s \in S^{FIS}, s_{in,j} \in (P_{in,J} \cap S^{J}_{in,j}), s'_{in,j} \in (C_{in,J} \cap S^{J}_{in,J}), p \in P, p > 1$$

Constraint 45 states that if production – consumption tasks are coupled via mechanism b_2, only one form is allowed i.e. direct transfer *or* prior consumption. However, this mechanism need not occur between the producing and consuming tasks at all.

$$x(s_{in,j}, s'_{in,j}, s, p) + v(s_{in,j}, s'_{in,j}, s, p) \leq 1, \tag{45}$$

$$\forall\, j, j' \in J, j \neq j', s \in S^{FIS}, s_{in,j} \in (P_{in,J} \cap S^{J}_{in,j}), s'_{in,j} \in (C_{in,J} \cap S^{J}_{in,J}), p \in P, p > 1$$

The proposed model operates, as is the case in Seid and Majozi (2012) and Vooradi and Shaik (2013), by allowing produced FIS states to freely enter and leave storage as is

deemed optimal without the rigorous tracking of flows, except at adjacent events. Therefore, in order to prevent FIS violations, it is enforced, perhaps in a limiting way, that any consuming task beginning at event p *must* begin at a time greater than or equal to any possible producing tasks at event $p-2$ (or earlier), as long as the producing task did indeed begin at event $p-2$. Constraint 46 enforces this.

$$t_{in}(s'_{in,j},p) \geq t_{out}(s_{in,j},p-2) - M[1-y(s_{in,j},p-2)], \tag{46}$$

$$\forall\, j,j' \in J, j \neq j', s \in S^{FIS}, s_{in,j} \in (P_{in,J} \cap S^{J}_{in,J}), s'_{in,j} \in (C_{in,J} \cap S^{J}_{in,J}), p \in P, p > 2$$

Constraint 47, originally proposed by Shaik and Floudas (2008) and used by Shaik and Floudas (2009), Vooradi and Shaik (2012), Vooradi and Shaik (2013) and Shaik and Vooradi (2017), is incorporated in the proposed model to prevent FIS violations arising as a result of production – consumption task couples occurring at the same event. The constraint is not written at the first event if either $s_{in,j}$ or $s'_{in,j}$ consume NS states. The constraint is amended here in that it is additionally not written at the first event for tasks $s'_{in,j}$ which do *not* produce ZW states, since there is no need for the variable t_{in} at the first event for such tasks.

$$t_{in}(s'_{in,j},p) \leq t_{out}(s_{in,j},p) + M[1-y(s_{in,j},p)], \tag{47}$$

$$\forall\, j,j' \in J, j \neq j', s \in S^{FIS}, s_{in,j} \in (P_{in,J} \cap S^{J}_{in,J}), s'_{in,j} \in (C_{in,J} \cap S^{J}_{in,J}), p \in P, p < P$$

Constraint 48 states that if temporary in-unit storage for ZW states occurs at event $p+1$, there must be no delay between the start time of the task at event $p+1$ and the end time at event p. This prevents post-processing unit wait for units holding ZW states. However, if ZW and UW states are produced simultaneously, then once the ZW states have been discharged, there is no need for binary variable w to be active and the UW states may participate in post-processing unit wait at subsequent events.

$$t_{in}(s_{in,j}, p+1) - t_{out}(s_{in,j}, p) \leq M[1 - w(s_{in,j}, p+1)], \quad \forall\, s_{in}j \in P_{in,J}^{ZW}, p \in P, p < P-1 \qquad (48)$$

3.3.8 Tightening Constraint

Constraint 49 was introduced by Maravelias and Grossmann (2003) in order to tighten the bounds on the allowed number and duration of tasks in each unit over the time horizon of interest. The constraint contributes towards reducing the computational time required to solve a given problem, however it is not strictly necessary for finding feasible solutions.

$$\sum_{p \in P} \sum_{s_{in,j} \in S_{in,J}^{J}} [a(s_{in,j})\, y(s_{in,j}, p) + b(s_{in,j})\, m_u(s_{in,j}, p)] \leq H, \quad \forall\, j \in J \qquad (49)$$

3.3.9 Objective Function

In the case of profit maximization, the parameter M appearing throughout the model should be set to the value of the time horizon of interest, or simply replaced with the parameter H. The objective function is expressed in Constraint 50, which is the sum of the selling price of each product multiplied by its final stored quantity, for all product states. Other financial aspects can readily be included in Constraint 50. For example, the cost price of raw materials used can be subtracted. Operating costs for certain tasks can be included by multiplying them by the binary variable y for task activation.

$$\text{maximize} \sum_{s \in S^P} S_{Pr}(s)\, q_T(s) \qquad (50)$$

In the case of makespan minimization, the parameter M appearing throughout the model should be set to the maximum expected value of the time horizon, H (which becomes a variable). The smaller the value of M, the tighter the formulation and hence the easier the problem is to solve, however it must be made large enough so that no potential solutions are excluded from the solve procedure. Additionally, with makespan minimization, a demand profile needs to be included which is to be met in the minimum

time. This is expressed in Constraint 51. The objective function is expressed in Constraint 52.

$$q_T(s) \geq d(s), \quad \forall\, s \in S^P \tag{51}$$

$$\text{minimize}\, H \tag{52}$$

Chapter 4

Model Validation and Analysis

4.1. Data Details and Sources

The proposed model was applied to 22 instances between 10 examples in order to compare computational performance and determine whether the proposed model is indeed capable of increasing scope and reducing computational time requirements in batch optimization studies. The examples were taken from Vooradi and Shaik (2012), Vooradi and Shaik (2013) and Rakovitis et al. (2019). The models by Vooradi and Shaik (2012), Vooradi and Shaik (2013), Shaik and Vooradi (2017) and Rakovitis et al. (2019), hereafter referred to as V&S 2012, V&S 2013, S&V 2017 and R 2019 respectively, were reproduced in order to accurately measure computational performance using the same hardware and software against the proposed model. The main motivating examples are Example 1, in which fractional extraction is demonstrated and Examples 7, 8 and 9, in which the limitations of the three-index variable technique for task splitting are demonstrated.

Each example is introduced along with its STN in Sections 4.2.1 to 4.3.2. Data for the examples can be found in Table 4-1 for state information and Table 4-2 for task information. Note that a task is represented as an effective state and a unit i.e. $s_{in,j}$ symbolises a group of states, represented by s, entering a suitable unit, represented by j, for processing as per the recipe depicted in the relevant STN. Example 10 is subdivided into 10.1 – 10.8, 10.10 and 10.12 – 10.14 to reflect different demand scenarios which were taken from Vooradi and Shaik (2012). Example 4 is subdivided into Example 4.1 and 4.2 due to the different objective functions considered for them. Examples 1 – 4.1 and 5 – 9 are solved for maximum profit, while Example 4.2 and all

of Example 10 are solved for the minimum makespan. All examples include intermediate states which are subjected to the FIS, NIS and/or ZW policies.

Each example was solved three times for each of the four models and each variation of Δp and number of event points considered, except for solutions which had not converged after the given maximum resource limit of 10 000 seconds. This was done in order to obtain an average CPU time and rule out the possibility of freezing, crashing or other random behaviour of the solver, even though the models are all deterministic.

For V&S 2012, post-processing unit wait is allowed and their Constraints 12b and 23a were used for each example for the tighter bounds. In order to address task splitting, each V&S model was solved at one iteration beyond what was required to obtain what is known to be the globally optimal solution in order to verify it and provide a concrete stopping criterion for iteration. The solution times for each iteration were then summed to yield the total solution time required at a given number of event points. For the three-index models, the number of event points were determined by their authors through nested iteration, successively increasing the number of splitting events and task events, until the solution no longer improved. The values published by the authors were used as a guideline in this work, however, one or two task events on either side of the optimal number were investigated. The number of splitting events was investigated from zero until one greater than their proposed optimal number. This investigation is not portrayed in the original works.

The system used to solve the examples included an Intel® Core™ i7-8700 CPU at 3.20 GHz and 16 GB RAM. The operating system was 64 bit Windows 10 and the solution software was GAMS v24.8.5. The default CPLEX solver was used with default settings except that the relative gap for termination was set to zero and node files were allowed to be stored on disk.

Results for the examples considered are given in Table 4-3 to Table 4-18. The best bound for unconverged solutions is included as an end note where applicable. Note that the numbers of binary and continuous variables as well as constraints differ slightly from those originally reported in the respective papers by Vooradi and Shaik. This is due to the omission of necessary details by the authors regarding the exact exclusion criteria. A perfect recreation was attempted; however, it was unsuccessful. Furthermore, the general exclusion criteria used in this work differs from their works in that STN-specific pre-processing was not included.

Table 4-1. State Information for All Examples

State	Type	Q^0 (m.u. [1])	Q^U (m.u. [1])	Cost/m.u.[1]
Example 1				
s_1	S^R	∞	∞	
s_2	S^{NIS}	0	0	
s_3	S^P	0	∞	10
Example 2				
s_1	S^R	∞	∞	
s_2	S^R	∞	∞	
s_3	S^R	∞	∞	
s_4	S^R	∞	∞	
s_5	S^R	∞	∞	
s_6	S^{FIS}	0	10	
s_7	S^P	0	∞	1
s_8	S^P	0	∞	1
s_9	S^P	0	∞	1
s_{10}	S^P	0	∞	1
s_{11}	S^P	0	∞	1
Example 3				
s_1	S^R	∞	∞	
s_2	S^{FIS}	0	200	
s_3	S^{FIS}	0	250	
s_4	S^P	0	∞	5
Example 4				
s_1	S^R	∞	∞	
s_2	S^R	∞	∞	
s_3	S^R	∞	∞	
s_4	S^R	∞	∞	
s_5	S^{FIS}	0	100	
s_6	S^{FIS}	0	150	
s_7	S^P	0	∞	10
s_8	S^{FIS}	0	200	
s_9	S^{FIS}	0	200	
s_{10}	S^P	0	∞	10
Example 5				
s_1	S^R	∞	∞	
s_2	S^R	∞	∞	
s_3	S^{FIS}	0	100	
s_4	S^{FIS}	0	100	
s_5	S^{FIS}	0	300	
s_6	S^{FIS}	50	150	
s_7	S^{FIS}	50	150	
s_8	S^R	∞	∞	
s_9	S^{FIS}	0	150	
s_{10}	S^{FIS}	0	150	
s_{11}	S^R	∞	∞	
s_{12}	S^P	0	∞	5
s_{13}	S^P	0	∞	5

[1] m.u.: mass unit

State	Type	Q^0 (m.u. [1])	Q^U (m.u. [1])	Cost/m.u.[1]
Example 6				
s_1	S^R	∞	∞	
s_2	S^{FIS}	0	6	
s_3	S^{FIS}	0	4	
s_4	S^P	0	∞	1
Example 7				
s_1	S^R	∞	∞	
s_2	S^R	∞	∞	
s_3	S^R	∞	∞	
s_4	S^{FIS}	0	60	
s_5	S^{FIS}	0	60	
s_6	S^P	0	∞	1
s_7	S^P	0	∞	1
Example 8				
s_1	S^R	∞	∞	
s_2	S^{FIS}	0	10	
s_3	S^{FIS}	0	15	
s_4	S^{FIS}	0	10	
s_5	S^{FIS}	0	15	
s_6	S^P	0	∞	10
Example 9				
s_1	S^R	∞	∞	
s_2	S^{FIS}	0	10	
s_3	S^{FIS}	0	17.5	
s_4	S^{FIS}	0	10	
s_5	S^{FIS}	0	18	
s_6	S^P	0	∞	10
Example 10				
s_0	S^R	∞	∞	
s_1	S^{FIS}	20	30	
s_2	S^{FIS}	20	30	
s_3	S^{FIS}	0	15	
s_4	S^{FIS}	20	30	
s_5	S^{ZW}	0	0	
s_6	S^{FIS}	0	10	
s_7	S^{FIS}	0	10	
s_8	S^{FIS}	0	10	
s_9	S^{ZW}	0	0	
s_{10}	S^{ZW}	0	0	
s_{11}	S^{FIS}	0	10	
s_{12}	S^{ZW}	0	0	
s_{13}	S^{FIS}	0	10	
s_{14}	S^P	0	∞	/
s_{15}	S^P	0	∞	/
s_{16}	S^P	0	∞	/
s_{17}	S^P	0	∞	/
s_{18}	S^P	0	∞	/

Table 4-2. Task Information for All Examples

Task	Formula	Unit	a (h)	b (h/m.u. [1])	V^L (m.u. [1])	V^U (m.u. [1])
Example 1						
s_{1in1}	$s_1 \rightarrow s_2$	1	4	0	0	10
s_{2in2}	$s_2 \rightarrow s_3$	2	2	0	0	5
Example 2						
s_{1in1}	$s_1 \rightarrow s_6$	1	2	0	0	3
s_{1in2}	$s_1 \rightarrow s_6$	2	2	0	0	9
s_{2in3}	$s_2 \rightarrow s_7$	3	3	0	0	1
s_{3in4}	$s_3 \rightarrow s_8$	4	3	0	0	1
s_{4in1}	$s_4 \rightarrow s_9$	1	3	0	0	1
s_{5in2}	$s_5 \rightarrow s_{10}$	2	3	0	0	1
s_{6in3}	$s_6 \rightarrow s_{11}$	3	2	0	0	3
s_{6in4}	$s_6 \rightarrow s_{11}$	4	2	0	0	9
Example 3						
s_{1in1}	$s_1 \rightarrow s_2$	1	1.333	0.01333	0	100
s_{1in2}	$s_1 \rightarrow s_2$	2	1.333	0.01333	0	150
s_{2in3}	$s_2 \rightarrow s_3$	3	1	0.005	0	200
s_{3in4}	$s_3 \rightarrow s_4$	4	0.667	0.00445	0	150
s_{3in5}	$s_3 \rightarrow s_4$	5	0.667	0.00445	0	150
Example 4						
s_{1in1}	$s_1 \rightarrow s_2$	1	0.667	0.00667	0	100
s_{2in2}	$s_2 + s_3 \rightarrow s_6$	2	1.334	0.02664	0	50
s_{2in3}	$s_2 + s_3 \rightarrow s_6$	3	1.334	0.01665	0	80
s_{5in2}	$s_5 + s_6 \rightarrow s_7 + s_8$	2	1.334	0.02664	0	50
s_{5in3}	$s_5 + s_6 \rightarrow s_7 + s_8$	3	1.334	0.01665	0	80
s_{4in2}	$s_4 + s_8 \rightarrow s_9$	2	0.667	0.01332	0	50
s_{4in3}	$s_4 + s_8 \rightarrow s_9$	3	0.667	0.008325	0	80
s_{9in4}	$s_9 \rightarrow s_8 + s_{10}$	4	1.3342	0.00666	0	200
Example 5						
s_{1in1}	$s_1 \rightarrow s_3$	1	0.667	0.00667	0	100
s_{2in2}	$s_2 \rightarrow s_4$	2	1.333	0.01333	0	100
s_{2in3}	$s_2 \rightarrow s_4$	3	1.333	0.00889	0	150
s_{3in2}	$s_3 + s_4 \rightarrow s_5$	2	0.667	0.00667	0	100
s_{3in3}	$s_3 + s_4 \rightarrow s_5$	3	0.667	0.00445	0	150
s_{5in4}	$s_5 \rightarrow s_4 + s_6 + s_7$	4	2	0.00667	0	300
s_{6in1}	$s_6 + s_8 \rightarrow s_9$	1	1	0.01	0	100
s_{7in5}	$s_7 + s_{10} + s_{11} \rightarrow s_{12}$	5	1.333	0.00667	20	200
s_{7in6}	$s_7 + s_{10} + s_{11} \rightarrow s_{12}$	6	1.333	0.00667	20	200
s_{9in2}	$s_9 \rightarrow s_{10} + s_{13}$	2	1.333	0.0133	0	100
s_{9in3}	$s_9 \rightarrow s_{10} + s_{13}$	3	1.333	0.00889	0	150
Example 6						
s_{1in1}	$s_1 \rightarrow s_2$	1	1	0	0	10
s_{2in2}	$s_2 \rightarrow s_3$	2	3	0	0	4
s_{2in3}	$s_2 \rightarrow s_3$	3	1	0	0	2
s_{3in4}	$s_3 \rightarrow s_4$	4	2	0	0	10

Task	Formula	Unit	a (h)	b (h/m.u. [1])	V^L (m.u. [1])	V^U (m.u. [1])
Example 7						
s_{1in1}	$s_1 + s_2 + s_3 \rightarrow s_4$	1	1.5	0	0	150
s_{4in2}	$s_4 \rightarrow s_5$	2	4.5	0	0	60
s_{4in3}	$s_4 \rightarrow s_5$	3	1.5	0	0	30
s_{4in4}	$s_4 \rightarrow s_5$	4	1.5	0	0	30
s_{5in5}	$s_5 \rightarrow s_6$	5	3	0	0	150
Example 8						
s_{1in1}	$s_1 \rightarrow s_2 + s_3$	1	1.666	0.03335	0	40
s_{2in2}	$s_2 \rightarrow s_4$	2	2.333	0.08335	0	20
s_{3in3}	$s_3 \rightarrow s_5$	3	0.667	0.0666	0	5
s_{4in4}	$s_4 + s_5 \rightarrow s_6$	4	2.667	0.008325	0	40
Example 9						
s_{1in1}	$s_1 \rightarrow s_2 + s_3$	1	1.666	0.03335	0	40
s_{2in2}	$s_2 \rightarrow s_4$	2	2.333	0.08335	0	20
s_{3in3}	$s_3 \rightarrow s_5$	3	0.333	0.0668	0	2.5
s_{4in4}	$s_4 + s_5 \rightarrow s_6$	4	2.667	0.008325	0	40
Example 10						
s_{0in1}	$s_0 \rightarrow s_1$	1	2	0	3	10
s_{1in2}	$s_1 \rightarrow s_2 + s_3$	2	4	0	5	20
s_{2ain4}	$s_2 \rightarrow s_5$	4	4	0	4	10
s_{2bin4}	$s_2 \rightarrow s_6$	4	4	0	4	10
s_{2in5}	$s_2 \rightarrow s_9$	5	6	0	4	10
s_{3in3}	$s_3 \rightarrow s_1 + s_4$	3	2	0	4	10
s_{4ain4}	$s_4 \rightarrow s_7$	4	4	0	4	10
s_{4bin4}	$s_4 \rightarrow s_8$	4	4	0	4	10
s_{4in5}	$s_4 \rightarrow s_{10}$	5	6	0	4	10
s_{5in8}	$s_5 + s_{11} \rightarrow s_{16}$	8	4	0	4	12
s_{6in6}	$s_6 \rightarrow s_{11}$	6	4	0	3	7
s_{6in7}	$s_6 \rightarrow s_{11}$	7	5	0	3	7
s_{7in6}	$s_7 \rightarrow s_{12}$	6	5	0	3	7
s_{7in7}	$s_7 \rightarrow s_{12}$	7	6	0	3	7
s_{8in6}	$s_8 \rightarrow s_{13}$	6	6	0	3	7
s_{8in7}	$s_8 \rightarrow s_{13}$	7	6	0	3	7
s_{9in8}	$s_9 \rightarrow s_{14}$	8	4	0	4	12
s_{9in9}	$s_9 \rightarrow s_{14}$	9	6	0	4	12
s_{10in8}	$s_{10} \rightarrow s_{15}$	8	4	0	4	12
s_{10in9}	$s_{10} \rightarrow s_{15}$	9	6	0	4	12
s_{12in8}	$s_{12} \rightarrow s_{17}$	8	6	0	4	12
s_{12in9}	$s_{12} \rightarrow s_{17}$	9	6	0	4	12
s_{13in8}	$s_{13} \rightarrow s_{18}$	8	6	0	4	12
s_{13in9}	$s_{13} \rightarrow s_{18}$	9	6	0	4	12

[1] m.u.: mass unit

4.2. Maximization of Profit

Examples 1, 2, 3, 4.1, 5, 6, 7, 8 and 9 are all solved for maximum profit. Computation results for these examples may be found in Table 4-3 to Table 4-11 respectively.

4.2.1 Example 1

Example 1 is adapted from Rakovitis et al. (2019) in order to demonstrate fractional extraction, which allows produced intermediates to be fractionally extracted from a processing unit at various event points. In this sequential batch process, the STN of which is presented in Figure 4-1, there are three states: one raw material, one intermediate and one product. Additionally, there are two tasks occurring in their own dedicated units. State 2 is subject to the NIS policy and therefore any state 2 produced by task s_{1in1} must be directly transferred to and consumed by task s_{2in2}. Task s_{1in1} can produce enough state 2 to supply two consecutive batches of task s_{2in2} of maximum volume, however since there is no dedicated storage vessel for state 2, material must continue to be stored in the producing vessel in order to achieve this. As such, a partial amount of the state 2 produced at event 1 must be transferred to task s_{2in2} for consumption at event 2 and the remaining quantity must be directly transferred for consumption at event 3. The proposed model as well as R 2019 allow this interaction whereas the other three models do not. This feature introduces flexibility into the model and can allow for better objective values to be obtained.

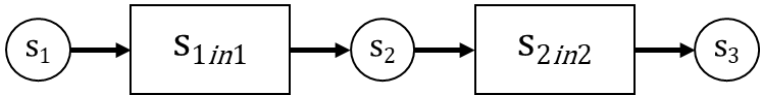


Figure 4-1. STN for Example 1

Table 4-3 contains the results of Example 1. Using 2, 2 and 1 event(s) for V&S 2012, V&S 2013 and S&V 2017 respectively all three models achieved the suboptimal solution of 50 cost units (c.u.). Increasing the number of events did not improve the objective value, whereas the proposed model and R 2019 found the globally optimal solution of 100 c.u with three and two events respectively. The results, however, for 3, 3 and 2 events are also displayed as these represent the minimum number of events which would be necessary for the models to find the globally optimal solution for the

other models. This is because three consecutive tasks occur, and by modelling production – consumption tasks to occur at adjacent events, the number of events required would be three for V&S 2012 and V&S 2013. By modelling production – consumption tasks to occur at the same event, as per S&V 2017, two events would be required. The above results are obtained with the splitting parameter of $\Delta P = 0$. The solution times for all four models are comparable at about 0.1 s. The proposed model requires fewer binary variables (7) than V&S 2013 (9), the fewest number of continuous variables (19 vs 26, 28, 20 and 33) and more constraints (34) than only S&V 2017 (30). R 2019 had the highest number of continuous variables and constraints (33 and 64 respectively). The V&S, S&V and R models are also solved with the splitting parameter of $\Delta P = 1$ in order to provide a stopping criterion and verify that the solution found is the best possible. In this case, the total combined solution times for the other models are effectively doubled, requiring a total of ~0.2 s each. The Gantt chart for the optimal solution of Example 1 is presented in Figure 1-2.

Table 4-3. Computational Results for Example 1

Model	H	Profit	Avg. CPU Time (s)	Total CPU Time (s)	Event Points	Δp	Binary Variables	Continuous Variables	Constraints	Nodes	RMIP
V&S 2012	8.0	50.0	0.1	0.1	2	0	3	17	35	0	50.0
V&S 2012	8.0	50.0	0.1		3	0	5	26	57	0	100.0
V&S 2012	8.0	50.0	0.1	0.2	3	1	8	29	72	0	100.0
V&S 2013	8.0	50.0	0.1	0.1	2	0	5	18	36	0	50.0
V&S 2013	8.0	50.0	0.1		3	0	9	28	60	0	100.0
V&S 2013	8.0	50.0	0.1	0.2	3	1	12	31	75	7	100.0
S&V 2017	8.0	50.0	0.1	0.1	1	0	2	11	16	0	50.0
S&V 2017	8.0	50.0	0.1		2	0	4	20	30	0	100.0
S&V 2017	8.0	50.0	0.1	0.2	2	1	6	22	48	5	100.0
R 2019	8.0	100.0	0.1		2	0	6	33	64	0	100.0
R 2019	8.0	100.0	0.1	0.2	2	1	8	35	70	0	100.0
This Work	8.0	100.0	0.1	0.1	3	/	7	19	34	0	100.0

4.2.2 Example 2

Example 2 is Case Study 2 in Vooradi and Shaik (2013). Its STN is presented in Figure 4-2. The case study is used in this work to benchmark the proposed model on its ability to correctly handle the FIS policy and to test that it does not allow overlapping of pre-processing unit wait with actual task processing. The process involves five sequential sub-processes, each producing their own single product from their own raw material. However, the six different tasks that can occur share four units, resulting in limitations on equipment resources. The objective is to maximize profit over a 5 hour time horizon.

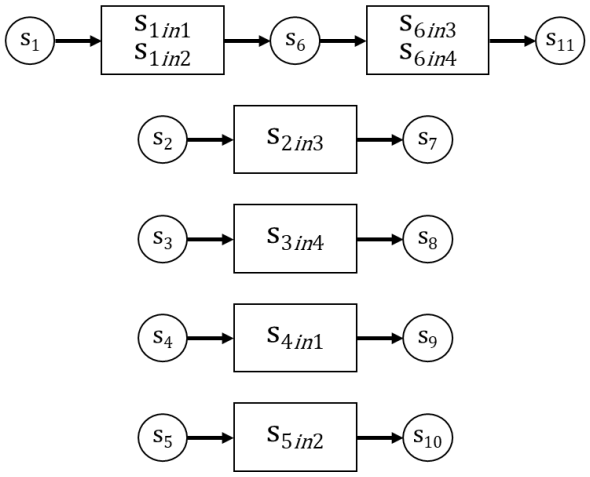


Figure 4-2. STN for Example 2

The results in Table 4-4 demonstrate how all five models were able to correctly obtain the optimal solution of 15 c.u without the need for task splitting. The proposed model, V&S 2013, S&V 2017 and R 2019 require only two event points whereas V&S 2012 requires three. All four models had comparable solution times (~0.1 s). The number of binary variables for the rigorous-conditional sequencing formulations of R 2019 were the highest (44), followed by the proposed work and V&S 2013 (28), however the proposed model required the fewest continuous variables (55) and the second fewest constraints (131) after S&V 2017 (128). R 2019 is again the largest model in these metrics. Again the other models were solved with $\Delta P = 1$ in order to verify the optimal

solution. As per Example 1, the total combined solution times are doubled, making the proposed formulation solve in the fastest total time.

Table 4-4. Computational Results for Example 2

Model	H	Profit	Avg. CPU Time (s)	Total CPU Time (s)	Event Points	Δp	Binary Variables	Continuous Variables	Constraints	Nodes	RMIP
V&S 2012	5.0	15.0	0.1		3	0	24	111	253	48	22.2
V&S 2012	5.0	15.0	0.1	0.2	3	1	40	127	333	136	23.0
V&S 2013	5.0	15.0	0.1		2	0	28	84	167	0	16.0
V&S 2013	5.0	15.0	0.1	0.2	2	1	36	92	207	0	16.0
S&V 2017	5.0	15.0	0.1		2	0	16	76	128	15	22.6
S&V 2017	5.0	15.0	0.1	0.2	2	1	24	84	184	22	22.7
R 2019	5.0	15.0	0.1		2	0	44	163	308	29	24.0
R 2019	5.0	15.0	0.1	0.2	2	1	52	171	332	103	24.0
This Work	5.0	15.0	0.1	0.1	2	/	28	55	131	0	16.0

4.2.3 Example 3

Example 3 is taken from Vooradi and Shaik (2012). The STN for the example is shown in Figure 4-3. The process is sequential with three different tasks occurring in five units to produce a single product from a single raw material. The objective is to maximize profit over a 16 hour time horizon.

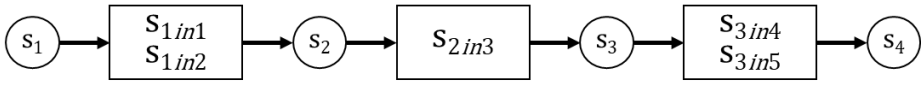


Figure 4-3. STN for Example 3

In Table 4-5, it is shown that all five models obtain the optimal objective of 5038.1 c.u. without any need for task splitting. The proposed model is the second slowest with a CPU time of 41.6 s at nine events, being ~12 seconds faster than the slowest model of S&V 2017, which also required nine events. The proposed model solved ~25 seconds slower than the fastest model, V&S 2012, which required 11 events. Compared to its rigorous conditional sequencing counterparts, the proposed model required the same number of binary variables (141) as V&S 2013, one more than R 2019, but fewer continuous variables (234 vs 237 and 351) and constraints (646 vs 710 and 1012). V&S 2012 required 55 binary variables, 211 continuous variables and 627 constraints,

whereas for S&V 2017 these values were 45, 173 and 342 respectively. In verifying the optimality of the results with a splitting parameter of $\Delta P = 1$, a drastic increase in solution time was observed. V&S 2012 required a total time of over 20 minutes, V&S 2013 required over six minutes, S&V 2017 required almost two hours and R 2019 required eight and a half minutes in order to solve. After this verification, the proposed model performed considerably faster than any of the V&S models while obtaining the globally optimal solution without the need for any iteration beyond the number of event points.

Table 4-5. Computational Results for Example 3

Model	H	Profit	Avg. CPU Time (s)	Total CPU Time (s)	Event Points	Δp	Binary Variables	Continuous Variables	Constraints	Nodes	RMIP
V&S 2012	16.0	5038.1	16.3		11	0	55	211	627	80172	6236.0
V&S 2012	16.0	5038.1	1308.6	1324.9	11	1	105	261	877	5293975	6601.7
V&S 2013	16.0	5038.1	33.1		9	0	141	237	710	134361	6601.7
V&S 2013	16.0	5038.1	343.2	376.3	9	1	181	277	910	1122780	6601.7
S&V 2017	16.0	5038.1	53.8		9	0	45	173	342	233343	6601.7
S&V 2017	16.0	5038.1	6935.5	6989.4	9	1	85	213	632	11693039	6601.7
R 2019	16.0	5038.1	30.6		7	0	140	351	1012	138405	6601.7
R 2019	16.0	5038.1	486.2	516.8	7	1	170	551	1102	2061760	6601.7
This Work	16.0	5038.1	41.6	41.6	9	/	141	234	646	175992	6601.7

4.2.4 Example 4.1

Example 4 has been extensively researched in literature by multiple authors. It features a complex multipurpose batch process with batch splitting and mixing as well as multiple tasks which can occur in a single processing unit and tasks which can be performed in multiple processing units. The STN is shown in Figure 4-4. The process involves nine different states, five different tasks and four units. In this particular representation, states 3 and 4, which are the same state, are differentiated in order to facilitate task differentiation. Three reaction tasks compete for two reactors, resulting in the requirement to efficiently manage limited equipment resources. The problem also includes a recycle loop involving states 8 and 9. Example 4.1 is solved for maximum

profit over a 16 hour time horizon. Example 4.2 is solved for the minimum makespan in Section 4.3.1.

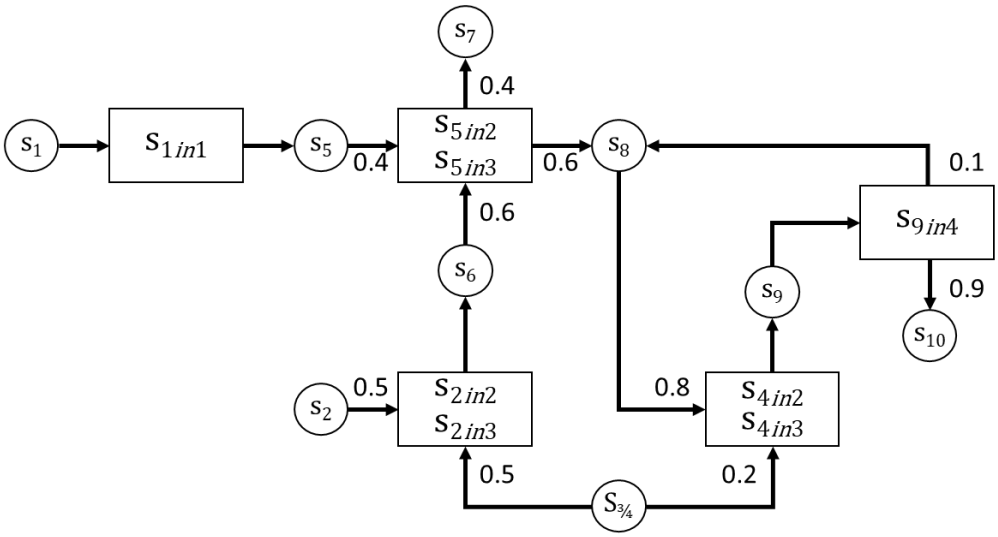


Figure 4-4. STN for Example 4

For Example 4.1, as shown in Table 4-6, all the models found the optimal objective of 3738.4 c.u. All the models except S&V 2017 required eight event points and no task splitting. S&V 2017 required one more event than the other models as well as a splitting parameter of $\Delta P = 1$ in order to obtain the optimal objective value, with a total solution time of over 25 minutes. This difficulty in the solution is attributed to the modelling of production – consumption tasks at the same in combination with unconditional sequencing. Since many production – consumption tasks can occur in the same unit, produced material must enter storage at the event it is produced before it can be consumed at the next event in the unit that produced it. This can cause conflict with the capacity of the storage unit and can impede the solution procedure or even exclude (potentially optimal) solutions in certain problems, such as in Example 5 in Section 4.2.5. The proposed model, requiring 21.5 s to solve, outperformed V&S 2013 by ~5 s but was slower than V&S 2012 and R 2019 by ~17 s and ~2 s respectively. The proposed model required the same number of binary variables as V&S 2013 (274) whereas V&S 2012 required only 64. S&V 2017 and R 2019 required 136 and 392 at

the minimum requirements for the optimal solution. The proposed model required more continuous variables (466) and fewer constraints (1440) than V&S 2013 (464 and 1473) whereas V&S 2012 required about half of these numbers. The size of the R 2019 model was around double the proposed model. After verifying the optimality with $\Delta P = 1$, V&S 2012 still outperformed the proposed model by ~5 seconds whereas V&S 2013 took three times as long as the proposed model. S&V 2017 was not verified at $\Delta P = 2$ due to the lengthy solution time at $\Delta P = 1$. R 2019 required just over a minute in total.

Table 4-6. Computational Results for Example 4.1

Model	H	Profit	Avg. CPU Time (s)	Total CPU Time (s)	Event Points	Δp	Binary Variables	Continuous Variables	Constraints	Nodes	RMIP
V&S 2012	16.0	3738.4	4.7		8	0	64	268	875	12354	4291.7
V&S 2012	16.0	3738.4	12.1	16.9	8	1	120	324	1155	29675	4291.7
V&S 2013	16.0	3738.4	26.3		8	0	274	464	1473	50103	4291.7
V&S 2013	16.0	3738.4	38.1	64.4	8	1	330	520	1753	60677	4291.7
S&V 2017	16.0	3724.4	11.2		8	0	64	268	687	19207	4316.9
S&V 2017	16.0	3724.4	42.3	53.5	8	1	120	324	1047	70500	4316.9
S&V 2017	16.0	3724.4	77.6		9	0	72	301	776	160515	4450.7
S&V 2017	16.0	3738.4	1461.3	1538.9	9	1	136	365	1186	1818306	4450.7
R 2019	16.0	3738.4	19.4		8	0	392	942	2563	29421	4291.7
R 2019	16.0	3738.4	48.4	67.8	8	1	448	998	2731	51650	4291.7
This Work	16.0	3738.4	21.5	21.5	8	/	274	466	1440	40988	4291.7

4.2.5 Example 5

Example 5 is a larger version of Example 4 with much of the same features. The STN is displayed in Figure 4-5. It involves 13 states, four of which are raw materials, seven are intermediates and two are products. States 4 and 5 are involved in a recycle loop. There are seven different tasks competing for six units. This example is solved for maximum profit over a 16 hour time horizon.

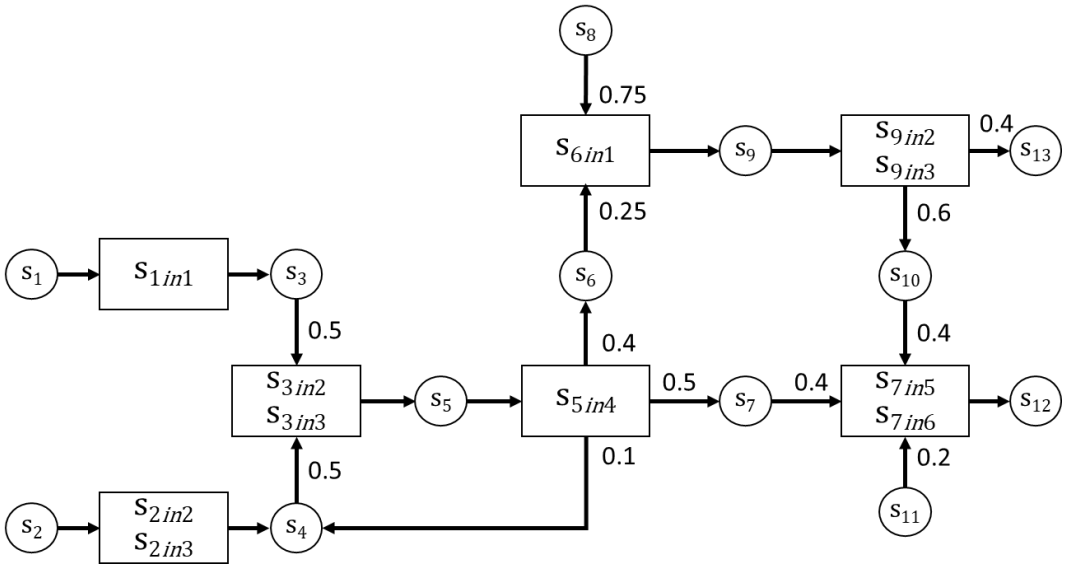


Figure 4-5. STN for Example 5

Examples 5 – 9 display some of the benefits of the proposed model, due to the requirement of task splitting. Table 4-7 contains the results for Example 5 in which V&S 2012 required a splitting parameter of $\Delta P = 1$ in order to obtain the optimal solution of 4262.8 c.u. For this example, S&V 2017 and R 2019 were unable to find the optimal solution for up to 11 and 9 events respectively in a resource limit of 10 000 s. This is due to a major drawback in modelling production – consumption tasks to occur at the same event as discussed in Section 3.2. With 10 events and $\Delta P = 1$ as well as with 11 events and $\Delta P = 0$, the solver was unable to converge for S&V 2017. As discussed in R 2019, the R 2019 model is able to obtain the optimal solution when considering all tasks as recycling tasks, however, this is not concomitant with the authors' definition of the term and it is therefore not possible to know where this can be applied on an ad hoc basis. For the minimum requirements to determine the optimal solution, as per Example 4.1, the proposed model required the same number of binary variables as V&S 2013 (569), but more continuous variables (826 vs 807) and fewer constraints (2774 vs 2843). The model size was significantly smaller for V&S 2012 (231, 621 and 2329), however this did not help in speedy convergence. The proposed

model performed the fastest for any single run by a minimum of ~11 s, however when compounded times for iterations of ΔP were accounted for in providing a stopping criterion for iteration and verifying optimal solutions, the proposed model was significantly faster than the V&S models by a minimum of ~223 s, despite being a larger model in many respects.

Table 4-7. Computational Results for Example 5

Model	H	Profit	Avg. CPU Time (s)	Total CPU Time (s)	Event Points	Δp	Binary Variables	Continuous Variables	Constraints	Nodes	RMIP
V&S 2012	16.0	4245.8	113.8		11	0	121	511	1779	112929	5644.6
V&S 2012	16.0	4262.8	689.8		11	1	231	621	2329	457946	5644.6
V&S 2012	16.0	4262.8	2989.5	3793.0	11	2	330	720	2725	1596394	5644.6
V&S 2013	16.0	4262.8	120.4		10	0	569	807	2843	134965	5225.9
V&S 2013	16.0	4262.8	205.3	325.7	10	1	668	906	3338	193584	5225.9
S&V 2017	16.0	4026.3	1258.5		9	0	99	419	1153	1248956	5640.6
S&V 2017	16.0	4193.6	3690.6	4949.1	9	1	187	507	1719	1639888	5640.6
S&V 2017	16.0	4205.0	2224.5		10	0	110	465	1285	1230625	5654.1
S&V 2017	16.0	4225.7	10000.0[1]	/	10	1	209	564	1920	3844937	5654.1
S&V 2017	16.0	4227.3	10000.0[2]	/	11	0	121	511	1417	4557616	5654.1
R 2019	16.0	4241.4	10000.0[3]		9	0	626	1429	4335	1523114	5640.6
R 2019	16.0	4241.4	10000.0[4]	/	9	1	714	1517	4599	1098569	5640.6
This Work	16.0	4262.8	102.5	102.5	10	/	569	826	2774	106616	5225.9

Best bound: [1]4242.8, [2]4561.8, [3]4249.0, [4]4278.8

4.2.6 Example 6

Example 6 is a simple sequential process involving four states: one raw material, two intermediates and one product as shown in the STN in Figure 4-6. There are three different tasks occurring in four units. The objective is to maximize profit over a 6 hour time horizon.

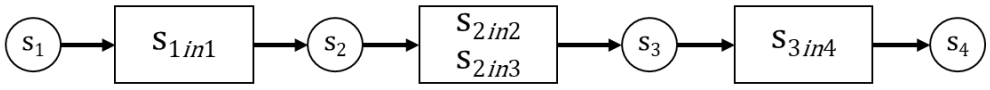


Figure 4-6. STN for Example 6

In Table 4-8, it is shown that all five models obtained the optimal solution of 10 c.u. with V&S 2012 and S&V 2017 both requiring $\Delta P = 1$. The proposed model performed the fastest over any single run with five events, tied with S&V 2017 with 3 events and $\Delta P = 0$, however the latter resulted in a suboptimal solution of 8 c.u. At the minimum

requirements to obtain the optimal solution, the proposed model had the same number of binary variables as V&S 2013 (68) and more than R 2019 (57), however both continuous variables (109 vs 114 and 130) and constraints (310 vs 343 and 377) were fewer. V&S 2012 required fewer binary (36) and continuous (98) variables, however it required more constraints (319) than the proposed model. S&V 2017 was the smallest model at 20 binary and 58 continuous variables and 165 constraints. When comparing the total CPU time including all necessary iterations, the proposed model required less than half the time of the other models. Nonetheless, all five models performed comparably with solution times under a third of a second.

Table 4-8. Computational Results for Example 6

Model	H	Profit	Avg. CPU Time (s)	Total CPU Time (s)	Event Points	Δp	Binary Variables	Continuous Variables	Constraints	Nodes	RMIP
V&S 2012	6.0	8.0	0.1		5	0	20	82	239	13	12.6
V&S 2012	6.0	10.0	0.1		5	1	36	98	319	5	14.0
V&S 2012	6.0	10.0	0.1	0.3	5	2	48	110	367	14	14.0
V&S 2013	6.0	10.0	0.1		5	0	68	114	343	17	14.0
V&S 2013	6.0	10.0	0.1	0.2	5	1	84	130	423	45	14.0
S&V 2017	6.0	8.0	0.1		3	0	12	50	101	39	14.0
S&V 2017	6.0	10.0	0.1		3	1	20	58	165	43	14.0
S&V 2017	6.0	10.0	0.1	0.3	3	2	24	62	181	5	14.0
R 2019	6.0	10.0	0.1		3	0	57	130	377	18	14.0
R 2019	6.0	10.0	0.1	0.3	3	1	65	138	401	80	14.0
This Work	6.0	10.0	0.1	0.1	5	/	68	109	310	9	14.0

4.2.7 Example 7

Example 7 is also a simple sequential process involving three raw materials, two intermediates and two products as shown in the STN in Figure 4-7. There are three different tasks occurring in five units. The objective is to maximize profit over a 9 hour time horizon.

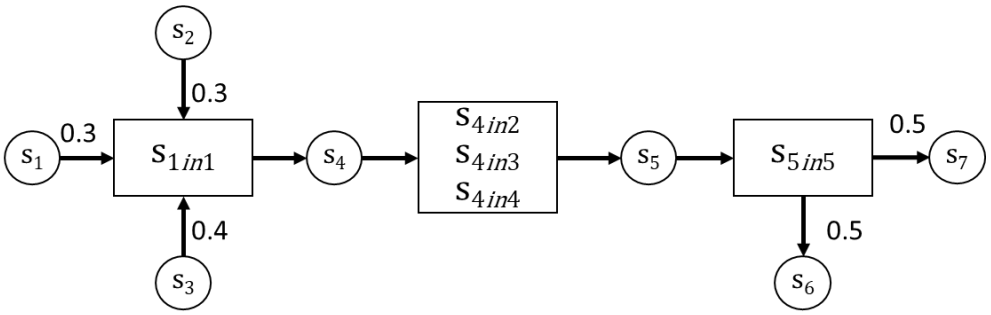


Figure 4-7. STN for Example 7

Table 4-9 shows that again all five models obtained the optimal objective value of 210 c.u. V&S 2013 required $\Delta P = 1$ with five events while V&S 2012 and S&V 2017 both required $\Delta P = 2$ with five and three events respectively. Note that for both V&S 2012 and S&V 2017, the same suboptimal objective value of 180 c.u. is obtained for two consecutive iterations i.e. for $\Delta P = 0$ and $\Delta P = 1$. This demonstrates a major drawback of formulations relying on the parameter ΔP since there is always uncertainty regarding the globally optimal solution and the sufficiency of a stopping criterion for the iterations on the parameter. If the stopping criterion for this example had been obtaining the same objective value for two consecutive iterations, solution would have terminated with the suboptimal objective of 180 c.u for these two models. All the models had comparable solution times of ~0.1 s except R 2019 which required double this. At the minimum requirements for the optimal solution, the proposed model required fewer binary variables (97), continuous variables (145), constraints (420) and nodes (50) than V&S 2013, which required 117, 182, 567 and 91 respectively. S&V 2017 required far fewer binary variables (30), continuous variables (85) and constraints (239), however it required more nodes (64). V&S 2012 required fewer binary variables (60) but more continuous variables (149) and constraints (478). The number of required nodes (31) was also lower. R 2019 required fewer binary variables (81) but was larger in the other metrics, as has been the trend in the other examples. V&S 2012 was solved at $\Delta P = 3$, V&S 2013 at $\Delta P = 2$ and R 2019 at $\Delta P = 1$ in order to verify the optimal

solution and provide a stopping criterion for iteration. S&V 2017 was not verified at $\Delta P = 3$ since it required a total of three events. When the total time for the iterations was considered, the proposed model required less than a third of the time of the other models which came to 0.4, 0.5, 0.3 and 0.5 s for V&S 2012, V&S 2013, S&V 2017 and R 2019 respectively, although the difference is negligible due to the order of magnitude of the solution times.

Table 4-9. Computational Results for Example 7

Model	H	Profit	Avg. CPU Time (s)	Total CPU Time (s)	Event Points	Δp	Binary Variables	Continuous Variables	Constraints	Nodes	RMIP
V&S 2012	9.0	180.0	0.1		5	0	25	114	318	10	266.7
V&S 2012	9.0	180.0	0.1		5	1	45	134	418	27	300.0
V&S 2012	9.0	210.0	0.1		5	2	60	149	478	31	300.0
V&S 2012	9.0	210.0	0.1	0.4	5	3	70	159	518	23	300.0
V&S 2013	9.0	180.0	0.1		5	0	97	162	467	75	300.0
V&S 2013	9.0	210.0	0.1		5	1	117	182	567	91	300.0
V&S 2013	9.0	210.0	0.2	0.5	5	2	132	197	627	74	300.0
S&V 2017	9.0	180.0	0.1		3	0	15	70	139	15	300.0
S&V 2017	9.0	180.0	0.1		3	1	25	80	219	71	300.0
S&V 2017	9.0	210.0	0.1	0.3	3	2	30	85	239	64	300.0
R 2019	9.0	210.0	0.2		3	0	81	220	550	115	300.0
R 2019	9.0	210.0	0.2	0.5	3	1	91	230	580	112	300.0
This Work	9.0	210.0	0.1	0.1	5	/	97	145	420	50	300.0

4.2.8 Examples 8 and 9

Examples 8 and 9 are respectively Examples 7 and 8 in Vooradi and Shaik (2012). These examples assertively demonstrate the necessity of allowing tasks to split over multiple events. They also reiterate the drawbacks of facilitating task splitting via three-index variables using the ΔP parameter i.e. the iteration stopping criterion and compounded solution time. The STN for Examples 8 and 9 is presented in Figure 4-8. The process involves one raw material, four intermediates and one product. Four different tasks occur in dedicated units. The objective is maximization of profit over a 10 hour time horizon. Note that the data given in Table 4-1 and Table 4-2 differ for the two examples.

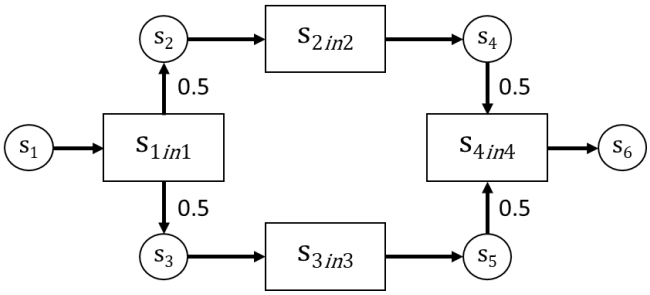


Figure 4-8. STN for Examples 8 and 9

Table 4-10 contains the computational results for Example 8. In this example, V&S 2012, V&S 2013 and S&V 2017 require a task to split over three events in order to obtain the optimal solution of 400 c.u, while the requirement for R 2019 is four. Note that, similarly to Example 7, all four models obtain the same suboptimal solution, of 200.1 c.u. in this case, for two consecutive iterations i.e. $\Delta P = 0$ and $\Delta P = 1$. If the stopping criterion for iteration had been two consecutive solutions of the same value, the solution procedure would have terminated at $\Delta P = 1$, thereby excluding the globally optimal solution. At the minimum requirements for the optimal solution, the proposed model had fewer binary variables (84), continuous variables (151) and constraints (446) than V&S 2013 and R 2019, which had 132, 198 and 697, and 135, 306 and 931 respectively. This is due to the large value of ΔP required by those models. The proposed model and S&V 2013 required six events, while R 2019 required five. S&V 2017 is altogether a smaller model, requiring only four events. V&S 2012, while requiring fewer binary variables (72), required more continuous variables (158) and constraints (525) at six events. Nonetheless, the proposed model required up to ~1 s less than the other models. Furthermore, when the compounded solution times were taken into account, the other models' CPU time requirements ranged from eight to 478 times that of the proposed model.

Table 4-10. Computational Results for Example 8

Model	H	Profit	Avg. CPU Time (s)	Total CPU Time (s)	Event Points	Δp	Binary Variables	Continuous Variables	Constraints	Nodes	RMIP
V&S 2012	10.0	200.1	0.2		6	0	24	110	313	141	400.0
V&S 2012	10.0	200.1	0.3		6	1	44	130	413	1453	400.0
V&S 2012	10.0	300.0	0.2		6	2	60	146	477	386	400.0
V&S 2012	10.0	400.0	0.1		6	3	72	158	525	0	400.0
V&S 2012	10.0	400.0	0.1	0.8	6	4	80	166	557	34	400.0
V&S 2013	10.0	200.1	0.4		6	0	84	150	485	1207	400.0
V&S 2013	10.0	200.1	1.1		6	1	104	170	585	3962	400.0
V&S 2013	10.0	300.0	0.6		6	2	120	186	649	1341	400.0
V&S 2013	10.0	400.0	0.1		6	3	132	198	697	0	400.0
V&S 2013	10.0	400.0	0.2	2.4	6	4	140	206	729	52	400.0
S&V 2017	10.0	200.1	0.2		4	0	16	74	149	295	400.0
S&V 2017	10.0	200.1	0.3		4	1	28	86	241	1598	400.0
S&V 2017	10.0	300.0	0.2		4	2	36	94	273	315	400.0
S&V 2017	10.0	400.0	0.1		4	3	40	98	289	72	400.0
S&V 2017	10.0	400.0	0.1	1.0	4	4	40	98	289	72	400.0
R 2019	10.0	200.1	3.0		5	0	95	266	811	6010	400.0
R 2019	10.0	200.1	27.2		5	1	111	282	859	69732	400.0
R 2019	10.0	300.0	13.3		5	2	123	294	895	21188	400.0
R 2019	10.0	350.0	3.3		5	3	131	302	919	4977	400.0
R 2019	10.0	400.0	1.0	47.8	5	4	135	306	931	1492	400.0
This Work	10.0	400.0	0.1	0.1	6	/	84	151	446	0	400.0

For Example 9, the results are shown in Table 4-11. All of the above observations are present on a larger scale as the optimal solution requires a task to split over *seven* events to obtain the optimal solution of 400 c.u. R 2019 required eight splitting events. Note that V&S 2012 obtains the same suboptimal solution of 200.133 c.u. for two consecutive iterations, at $\Delta P = 2$ and $\Delta P = 3$. V&S 2013 obtains this solution for four consecutive iterations, from $\Delta P = 0$ to $\Delta P = 3$ and S&V 2017 returns this solution for three consecutive iterations, from $\Delta P = 1$ to $\Delta P = 3$. R 2019 also obtained this solution for four consecutive iterations, from $\Delta P = 0$ to $\Delta P = 3$. This would surely meet any reasonable stopping criterion and prevent the determination of the optimal solution. V&S 2012, V&S 2013 and the proposed model required 10 events, while S&V 2017 and R 2019 required eight and nine events respectively to find the optimal solution. At the minimum requirements for this solution, due to the large number of splitting events, S&V 2017 is only marginally smaller than the proposed model with 144 binary variables and 258 continuous variables, compared to 148 and 263

respectively. The proposed model required fewer constraints (794 vs 841) and nodes (62 vs 333). V&S 2012, V&S 2013 and R 2019 were all larger models on all accounts. The proposed model had the shortest single run solution time of ~0.2 s along with V&S 2012 at $\Delta P = 7$. The other solution times range from 0.3 to 6852.7 s. R 2019 was unable to converge within 10 000 s for $\Delta P = 2$ and $\Delta P = 3$. When compounded iteration times are accounted for, V&S 2012, V&S 2013 and S&V 2017 required totals of 264.0, 2663.6 and 432.7 seconds to solve, respectively. These solution times are three to four orders of magnitude higher than that of the proposed model. Note that S&V 2017 was not solved at $\Delta P = 8$ since it required a total of eight events. Similarly, R 2019 was not solved at $\Delta P = 9$ since it required a total of nine events. The Gantt Chart displaying the optimal schedule for Example 9 can be found in Figure 4-9. The chart displays the batch sizes of each task as well as the event at which the task occurs. It can be seen that Task s_{1in1} produces 20 units of state 2 at event 1 which is more than the allowed storage. Therefore, unit j_1 continues to hold the material until event 9 where it can be discharged to unit j_2 for consumption, imitating task splitting. However, unit j_1 also discharges produced state 3 to unit j_3 at event 2.

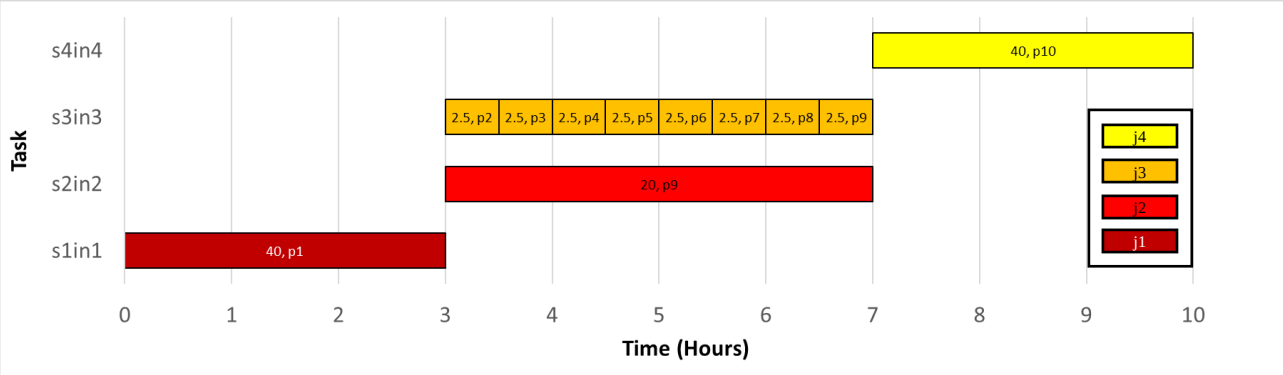


Figure 4-9. Optimal Schedule for Example 9 (Proposed Model)

Table 4-11. Computational Results for Example 9

Model	H	Profit	Avg. CPU Time (s)	Total CPU Time (s)	Event Points	Δ*p*	Binary Variables	Continuous Variables	Constraints	Nodes	RMIP
V&S 2012	10.0	200.1	1.5		10	0	40	182	529	5881	400.0
V&S 2012	10.0	200.1	43.4		10	1	76	218	709	183662	400.0
V&S 2012	10.0	200.1	99.0		10	2	108	250	837	278379	400.0
V&S 2012	10.0	200.1	72.7		10	3	136	278	949	149327	400.0
V&S 2012	10.0	250.0	22.5		10	4	160	302	1045	39344	400.0
V&S 2012	10.0	300.0	21.4		10	5	180	322	1125	31683	400.0
V&S 2012	10.0	350.0	2.9		10	6	196	338	1189	2198	400.0
V&S 2012	10.0	400.0	0.2		10	7	208	350	1237	91	400.0
V&S 2012	10.0	400.0	0.3	264.0	10	8	216	358	1269	274	400.0
V&S 2013	10.0	200.1	9.8		10	0	148	254	845	24191	400.0
V&S 2013	10.0	200.1	166.2		10	1	184	290	1025	345637	400.0
V&S 2013	10.0	200.1	417.1		10	2	216	322	1153	964619	400.0
V&S 2013	10.0	200.1	1495.1		10	3	244	350	1265	2572875	400.0
V&S 2013	10.0	250.0	449.1		10	4	268	374	1361	524677	400.0
V&S 2013	10.0	300.0	111.1		10	5	288	394	1441	101603	400.0
V&S 2013	10.0	350.0	12.0		10	6	304	410	1505	14036	400.0
V&S 2013	10.0	400.0	1.6		10	7	316	422	1553	1563	400.0
V&S 2013	10.0	400.0	1.7	2663.6	10	8	324	430	1585	1533	400.0
S&V 2017	10.0	200.1	2.6		8	0	32	146	301	10972	400.0
S&V 2017	10.0	200.1	65.4		8	1	60	174	505	137305	400.0
S&V 2017	10.0	200.1	125.9		8	2	84	198	601	206271	400.0
S&V 2017	10.0	200.1	153.9		8	3	104	218	681	276276	400.0
S&V 2017	10.0	250.0	45.8		8	4	120	234	745	75469	400.0
S&V 2017	10.0	300.0	35.8		8	5	132	246	793	35826	400.0
S&V 2017	10.0	350.0	3.0		8	6	140	254	825	2911	400.0
S&V 2017	10.0	400.0	0.3	432.7	8	7	144	258	841	333	400.0
R 2019	10.0	200.1	120.0		9	0	171	478	1491	241563	450.0
R 2019	10.0	200.1	4558.1		9	1	203	510	1587	5815447	450.0
R 2019	10.0	200.1	10000.0[1]		9	2	231	538	1671	8524578	450.0
R 2019	10.0	200.1	10000.0[2]		9	3	255	562	1743	4335637	450.0
R 2019	10.0	250.0	6852.7		9	4	275	582	1803	2847939	450.0
R 2019	10.0	300.0	6369.3		9	5	291	598	1851	2450455	450.0
R 2019	10.0	350.0	2774.0		9	6	303	610	1887	999775	450.0
R 2019	10.0	380.0	60.6		9	7	311	618	1911	41426	450.0
R 2019	10.0	400.0	4.1	/	9	8	315	622	1923	2572	450.0
This Work	10.0	400.0	0.2	0.2	10	/	148	263	794	62	400.0

Best bound: [1]200.1, [2]300.0

4.3. Minimization of Makespan

Table 4-12 to Table 4-18 contain the computational results for Examples 4.2, 10.1 – 10.8, 10.10 and 10.12 – 10.14, which are solved for minimum makespan.

4.3.1 Example 4.2

Example 4.2 uses the same data as Example 4.1 except that it is not concerned with the selling price of products. The big M value used in all necessary constraints is 100. The demand profile is 500 units for state 7 and 400 units for state 10.

Table 4-12 contains the results for Example 4.2. Note that none of the V&S models or R 2019 were able to converge by the resource limit of 10 000 s on the globally optimal solution for any value of ΔP considered. This brings into question which value of ΔP *should* be considered, since in practice the models would not be run two or three times for the duration of the given resource limit. The proposed model was able to converge on the globally optimal solution of 47.683 hours after just 425.1 seconds (~7 minutes), despite being the largest model for most of the runs, after R 2019, which is generally the largest model. The optimal schedule as determined by the proposed model is shown in Figure 4-10. The proposed model has the same number of binary variables (768) as V&S 2013 at $\Delta P = 0$, more continuous variables (1298 vs 1257) and fewer constraints (4093 vs 4113) however all of these values were lower than V&S 2013 at $\Delta P = 1$. On the other hand, they were all higher than those of V&S 2012 and S&V 2017 for any value of ΔP. The proposed model also had the worst relaxed solution at 46.5. S&V 2017 had the worst solutions of 48.8 at $\Delta P = 0$ and 48.9 at $\Delta P = 1$. In fact, the solution deteriorated when increasing the number of splitting events to one. These higher values are attributed to the sequencing of production – consumption tasks at the same event without the use of conditional sequencing as compensation. V&S 2012 and V&S 2013 both displayed improving objective values at increasing values of ΔP and V&S 2012 actually found the optimal solution at $\Delta P = 2$, although it did not converge on it. Nevertheless, it is not clear whether this solution would be obtained in practice due to the length of time required and the uncertainty regarding the optimal number of splitting events. Note that for S&V 2017 and R 2019, no solution was possible at fewer events, as the solver reported the demand profile to be infeasible at 20 events with $\Delta P = 0$ through $\Delta P = 2$.

Table 4-12. Computational Results for Example 4.2

Model	H	Average CPU Time (s)	Total CPU Time (s)	Event Points	Δ*p*	Binary Variables	Continuous Variables	Constraints	Nodes	RMIP
V&S 2012	47.7	10000.0[1]		21	0	168	697	2371	1419197	47.5
V&S 2012	47.7	10000.0[2]		21	1	328	857	3171	1120017	47.4
V&S 2012	47.7	10000.0[3]	/	21	2	480	1009	3779	2118671	47.4
V&S 2013	47.7	10000.0[4]		21	0	768	1257	4113	572195	47.5
V&S 2013	47.7	10000.0[5]	/	21	1	928	1417	4913	1723942	47.4
S&V 2017	48.8	10000.0[6]		21	0	168	697	1845	1351464	47.5
S&V 2017	48.9	10000.0[7]	/	21	1	328	857	2855	1285509	47.4
R 2019	47.7	10000.0[8]		21	0	1042	2476	6945	922920	47.3
R 2019	47.7	10000.0[9]	/	21	1	1202	2636	7425	673494	47.3
This Work	47.7	425.1	425.1	21	/	768	1298	4093	101150	46.5

Best bound: [1]47.5, [2]47.6, [3]47.4, [4]47.7, [5]47.4, [6]47.5, [7]47.4, [8]47.5, [9]47.4

Relative Gap: [1]0.45%, [2]0.19%, [3]0.64%, [4]0.01%, [5]0.65%, [6]2.68%, [7]3.20%, [8]0.45%, [9]0.63%

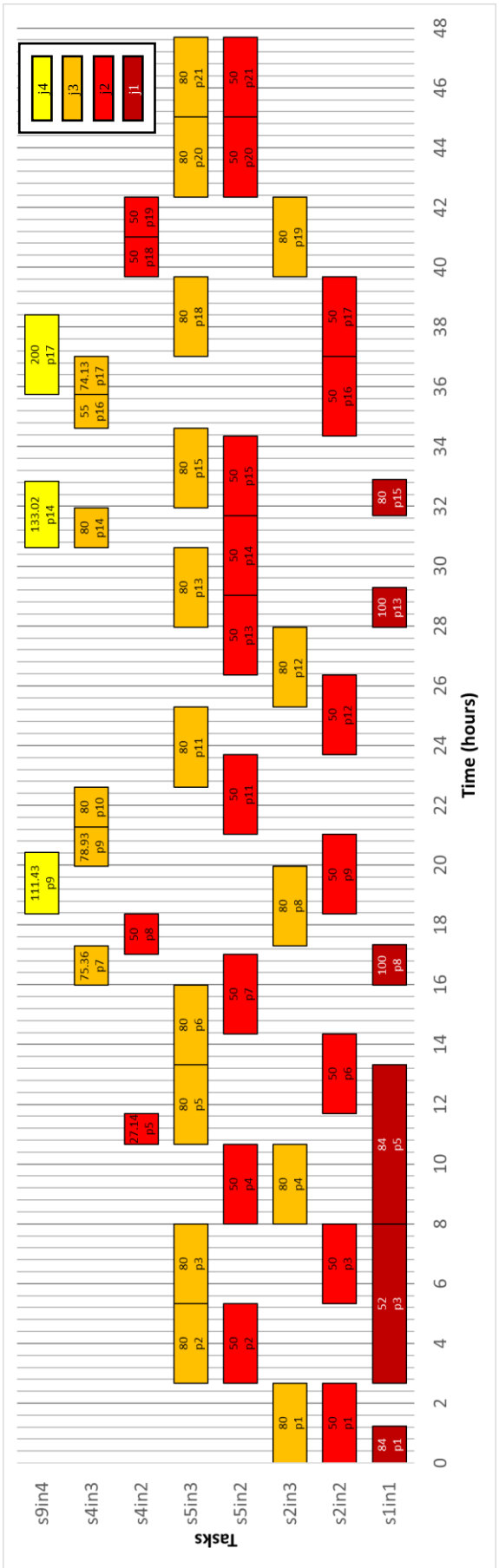


Figure 4-10. Optimal Schedule for Example 4.2 (Proposed Model)

4.3.2 Example 10 - Westenberger-Kallrath Problem

This example is a popular benchmark problem for scheduling formulations which was publicized by Kallrath (2002). The data for the problem is largely the same as that initially published, except that the task duration data is scaled for each task while maintaining the original ratios. Therefore, the data is more accurately as per that in Vooradi and Shaik (2012). The large process, for which the STN is depicted in Figure 4-11, involves 19 states: one raw material, 13 intermediates and five products. States 1 and 3 are involved in a recycle loop. There are 17 different tasks occurring in nine processing units with multiple tasks competing for some of the equipment resources. Notably, the problem involves four states which must adhere to the ZW policy: states 5, 9, 10 and 12. Note that results for R 2019 are not included for this examples as the model does not include handling of states which must adhere to the ZW policy. The problem is solved for the minimum makespan required to satisfy 12 different demand scenarios appearing in Vooradi and Shaik (2012). The demand patterns are given in Table 4-13 below, where patterns 9, 11 and 0 are excluded due to their excessive computational complexity. Tasks which were not required for a given demand profile were rigorously excluded along with their associated binary and continuous variables as well as any appropriate constraints. The big M value was taken as 100 for all scenarios. Note that for V&S 2013, modifications are required in many constraints relating to rigorous conditional sequencing in order to allow for variable production ratios in task s_{1in2}. The same substitutions are made for each of these cases as in V&S 2012, whereby the quantity of produced states 2 and 3 are expressed through a material balance around storage and excess quantities of these states.

Table 4-13. Demand Scenarios for Example 10

Demand Scenario	Product 14	Product 15	Product 16	Product 17	Product 18
1	20	20	20	0	0
2	20	20	0	20	0
3	20	20	0	0	20
4	20	0	20	20	0
5	20	0	20	0	20
6	20	0	0	20	20
7	0	20	20	20	0
8	0	20	20	0	20
10	0	0	20	20	20
12	30	20	20	10	10
13	10	20	30	20	10
14	18	18	18	18	18

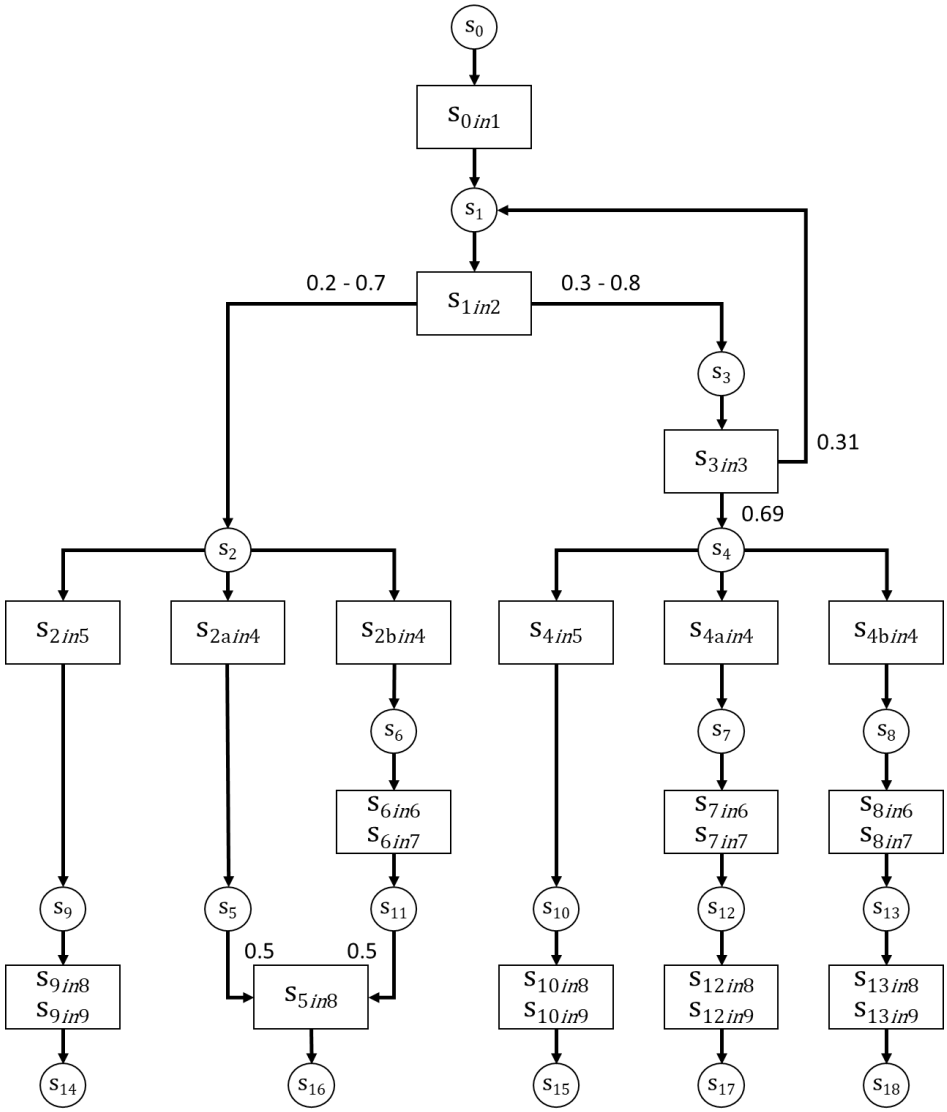


Figure 4-11. STN for Example 10

Table 4-14 and Table 4-15 contain the results for Examples 10.1 – 10.3 and 10.4 to 10.5 respectively, all of which are smaller problems requiring no task splitting. For Example 10.1, the proposed model is the largest in all aspects except for having fewer binary variables (214) than V&S 2013 at $\Delta P = 1$ (253) and fewer constraints (1061) than V&S 2013 at both $\Delta P = 0$ (1108) and $\Delta P = 1$ (1360). The proposed model solves in 0.2 s, which is slower than only V&S 2012 at $\Delta P = 0$ (0.1 s). However, when compounded time is considered for V&S models, the proposed model required about half the time of the other models. For Example 10.2, the proposed model is again the largest except for having fewer binary variables (238) than V&S 2013 at $\Delta P = 1$ (282) and fewer constraints (1171) than V&S 2012 at $\Delta P = 1$ (1238) and V&S 2013 at $\Delta P = 0$ (1216) and $\Delta P = 1$ (1536). The proposed model is slower (0.4 s) than only V&S 2012 at $\Delta P = 0$ (0.3 s). When total solution times are considered, the proposed model solved in half, a third and a tenth of the time required by V&S 2012, V&S 2013 and S&V 2017 respectively. For Example 10.3, the proposed model was the largest except for having fewer binary (280) and continuous (451) variables than V&S 2013 at $\Delta P = 1$ (336 and 508) and fewer constraints (1374) than V&S 2013 at $\Delta P = 0$ (1435) and $\Delta P = 1$ (1765). On any single run, the proposed model solved slower than V&S 2013 and S&V 2017 at $\Delta P = 0$ (0.2 s), however against the total compounded solution times, the proposed model required half the time of V&S 2012 and V&S 2013 and an ninth of the time for S&V 2017.

Table 4-14. Computational Results for Examples 10.1 to 10.3

Model	H	Average CPU Time (s)	Total CPU Time (s)	Event Points	Δ*p*	Binary Variables	Continuous Variables	Constraints	Nodes	RMIP
Example 10.1										
V&S 2012	28.0	0.1		5	0	62	253	777	17	24.0
V&S 2012	28.0	0.3	0.4	5	1	110	301	1029	25	24.0
V&S 2013	28.0	0.2		5	0	205	355	1108	31	24.0
V&S 2013	28.0	0.2	0.4	5	1	253	403	1360	74	24.0
S&V 2017	28.0	0.2		4	0	56	222	534	123	24.0
S&V 2017	28.0	0.3	0.6	4	1	98	264	820	222	24.0
This Work	28.0	0.2	0.2	5	/	214	490	1061	56	24.0
Example 10.2										
V&S 2012	28.0	0.3		6	0	74	296	918	122	24.0
V&S 2012	28.0	0.6	0.8	6	1	134	356	1238	880	24.0
V&S 2013	28.0	0.5		6	0	222	408	1216	575	24.0
V&S 2013	28.0	0.7	1.2	6	1	282	468	1536	560	24.0
S&V 2017	28.0	0.4		5	0	70	272	654	992	24.0
S&V 2017	28.0	3.6	4.1	5	1	126	328	1034	3485	24.0
This Work	28.0	0.4	0.4	6	/	238	520	1171	79	24.0
Example 10.3										
V&S 2012	28.0	0.3		6	0	78	308	988	17	24.0
V&S 2012	28.0	0.4	0.6	6	1	142	372	1318	154	24.0
V&S 2013	28.0	0.2		6	0	272	444	1435	54	24.0
V&S 2013	28.0	0.5	0.7	6	1	336	508	1765	129	24.0
S&V 2017	28.0	0.2		5	0	70	272	669	112	24.0
S&V 2017	28.0	2.6	2.8	5	1	126	328	1049	2639	24.0
This Work	28.0	0.3	0.3	6	/	280	451	1374	119	24.0

For Example 10.4, the globally optimal solution of 27 hours was obtained by all four models. S&V 2017 required five events and the other models required six. The proposed model required the most binary (321) and continuous (519) variables after V&S 2013 at $\Delta P = 1$ (376 and 575) as well as the most constraints (1610) after V&S 2013 at both $\Delta P = 0$ (1647) and $\Delta P = 1$ (2022). In this example, the proposed model required the third highest solution time for any single run (4.0 s), whereas the quickest model was V&S 2012 at $\Delta P = 0$ (0.5 s). It was faster than V&S 2013 at $\Delta P = 1$ (5.1 s) and S&V 2017 at $\Delta P = 1$ (5.7 s). When comparing the total solution times, the proposed model was slower than V&S 2012 by ~2.6 s but faster than the others by at least 2.4 s. Example 10.5 was solved with five event points by S&V 2017 and with six event points by the other models. The proposed model was the largest except for having fewer binary (363) and continuous (557) variables than V&S 2013 at $\Delta P = 1$ (430 and

615). As expected the proposed model also had fewer constraints (1810) than V&S 2013 at both $\Delta P = 0$ (1861) and $\Delta P = 1$ (2246). On a single run basis, the proposed model required more time (1.8 s) than V&S 2012 at $\Delta P = 0$ (0.6 s) and $\Delta P = 1$ (0.9 s) and S&V 2017 at $\Delta P = 0$ (0.5 s), however when an extra iteration is considered in order to verify the optimal solution, the proposed model was slower than V&S 2012 by only ~0.3 s while it was faster than the others by a minimum of ~3 s.

Table 4-15. Computational Results for Examples 10.4 and 10.5

Model	H	Average CPU Time (s)	Total CPU Time (s)	Event Points	Δ*p*	Binary Variables	Continuous Variables	Constraints	Nodes	RMIP
Example 10.4										
V&S 2012	27.0	0.5		6	0	87	347	1118	323	16.0
V&S 2012	27.0	0.9	1.4	6	1	158	418	1493	864	16.0
V&S 2013	27.0	3.0		6	0	305	504	1647	3198	16.0
V&S 2013	27.0	5.1	8.1	6	1	376	575	2022	5774	16.0
S&V 2017	27.0	0.7		5	0	80	312	798	935	16.0
S&V 2017	27.0	5.7	6.4	5	1	144	376	1218	4532	16.0
This Work	27.0	4.0	4.0	6	/	321	519	1610	3594	16.0
Example 10.5										
V&S 2012	26.0	0.6		6	0	91	359	1187	608	16.0
V&S 2012	26.0	0.9	1.5	6	1	166	434	1572	683	16.0
V&S 2013	26.0	3.1		6	0	355	540	1861	3563	16.0
V&S 2013	26.0	3.6	6.7	6	1	430	615	2246	3083	16.0
S&V 2017	26.0	0.5		5	0	80	312	813	885	16.0
S&V 2017	26.0	4.2	4.8	5	1	144	376	1233	3845	16.0
This Work	26.0	1.8	1.8	6	/	363	557	1810	3417	16.0

As shown in Table 4-16, Example 10.6 required that tasks in both V&S 2012 and S&V 2017 split over one event in order to obtain the optimal solution of 30 hours. S&V 2017 required six events and the other models required eight. The proposed model required fewer binary (466) and continuous (731) variables than V&S 2013 at $\Delta P = 1$ (553 and 806) as well as fewer constraints (2300) than V&S 2012 at $\Delta P = 2$ (2466) and V&S 2013 at both $\Delta P = 0$ (2341) and $\Delta P = 1$ (2887). It solved slower than V&S 2012 by ~223 s under the minimum requirements for the optimal solution however it was slower only by ~121 s when total solution time was accounted for. Under these conditions, it was faster than V&S 2013 and S&V 2017 by at least ~214 s. Note that V&S 2013 does

not scale well for this example when increasing the number of splitting events, as at $\Delta P = 1$, the solver required 1391.8 s to converge. Also shown in Table 4-16, Example 10.7 required no task splitting and was solved by S&V 2017 with six events and with seven events by the other models. The proposed model followed a familiar trend in terms of size being the largest except for having fewer binary (387) and continuous (622) variables than V&S 2013 (454 and 686) as well as fewer constraints (1944) than V&S 2013 at both $\Delta P = 0$ (1977) and $\Delta P = 1$ (2436). The proposed model was slower (14.1 s) than V&S 2012 at $\Delta P = 0$ (1.0 s), however the cumulative solution time for V&S 2012 was 6.2 s. Also in the case of compounded solution time it was twice as fast as V&S 2013 (28.7 s) and more than three times as fast as S&V 2017 (49.6 s). Again in Table 4-16, Example 10.8 displayed the poorest results for the proposed model in which the compounded solution times for the V&S models were better in the case of V&S 2012 (7.0 s) as well as S&V 2017 (1.5 s) against 14.9 s. V&S 2013 required 30.4 s. Again this example required no task splitting and the trends on binary and continuous variables as well as constraints were consistent with Example 10.7.

Table 4-16. Computational Results for Example 10.6 to 10.8

Model	H	Average CPU Time (s)	Total CPU Time (s)	Event Points	Δ*p*	Binary Variables	Continuous Variables	Constraints	Nodes	RMIP
Example 10.6										
V&S 2012	31.0	21.3		8	0	121	469	1564	9593	16.0
V&S 2012	30.0	68.7		8	1	226	574	2110	26263	16.0
V&S 2012	30.0	80.3	170.3	8	2	315	663	2466	26065	16.0
V&S 2013	30.0	241.3		8	0	448	701	2341	114177	16.0
V&S 2013	30.0	1391.8	1633.0	8	1	553	806	2887	267711	16.0
S&V 2017	31.0	43.1		6	0	96	368	956	25021	16.0
S&V 2017	30.0	119.7		6	1	176	448	1476	43286	16.0
S&V 2017	30.0	343.0	505.7	6	2	240	512	1732	95805	16.0
This Work	30.0	291.6	291.6	8	/	466	731	2300	103036	16.0
Example 10.7										
V&S 2012	28.0	1.0		7	0	103	409	1331	762	16.0
V&S 2012	28.0	5.1	6.2	7	1	190	496	1790	2254	16.0
V&S 2013	28.0	7.0		7	0	367	599	1977	5604	16.0
V&S 2013	28.0	21.7	28.7	7	1	454	686	2436	12068	16.0
S&V 2017	28.0	2.6		6	0	96	374	963	3124	16.0
S&V 2017	28.0	47.0	49.6	6	1	176	454	1483	21463	16.0
This Work	28.0	14.1	14.1	7	/	387	622	1944	10531	16.0
Example 10.8										
V&S 2012	28.0	1.6		7	0	107	421	1406	1135	16.0
V&S 2012	28.0	5.4	7.0	7	1	198	512	1873	2826	16.0
V&S 2013	28.0	8.2		7	0	425	639	2225	6559	16.0
V&S 2013	28.0	22.2	30.4	7	1	516	730	2692	15261	16.0
S&V 2017	28.0	0.4		5	0	80	312	813	507	16.0
S&V 2017	28.0	1.1	1.5	5	1	144	376	1233	886	16.0
This Work	28.0	14.9	14.9	7	/	435	665	2177	14265	16.0

The results for Examples 10.10 and 10.12 are displayed in Table 4-17. No task splitting was required and S&V 2017 obtained the globally optimal solution of 35 hours with seven events where as the other models required eight. The trends for model size are again consistent with Example 10.8 and 10.7. The proposed model solved slower (17.1 s) than the single runs for V&S 2012 at $\Delta P = 0$ and $\Delta P = 1$, V&S 2013 at $\Delta P = 0$ and S&V 2017 at $\Delta P = 0$ by a maximum of ~11 s, however against the compounded solution times the proposed model solved the fastest by up to ~135 s. Example 10.12 requires no task splitting and was solved with eight events by V&S 2012 and with seven events by the other three formulations. Again the proposed formulation followed a similar trend in terms of model size as above, except that it also required fewer constraints (3164) than V&S 2012 at $\Delta P = 1$ (3291). The proposed model also had the

shortest single run time (1.4 s), however it was significantly faster than V&S 2012 and S&V 2017, especially when compounded solution times for iterations necessary to verify the optimal objective value were considered. V&S 2013 performed well at 6.8 s, however V&S 2012 and S&V 2017 required a total of ~109 and ~2690 s respectively.

Table 4-17. Computational Results for Example 10.10 and 10.12

Model	H	Average CPU Time (s)	Total CPU Time (s)	Event Points	Δp	Binary Variables	Continuous Variables	Constraints	Nodes	RMIP
Example 10.10										
V&S 2012	35.0	6.3		8	0	138	536	1890	3024	24.0
V&S 2012	35.0	12.3	18.5	8	1	258	656	2511	4286	24.0
V&S 2013	35.0	11.0		8	0	563	831	3053	7463	24.0
V&S 2013	35.0	37.7	48.7	8	1	683	951	3674	19674	24.0
S&V 2017	35.0	16.3		7	0	126	485	1371	10845	24.0
S&V 2017	35.0	136.1	152.4	7	1	234	593	2058	39301	24.0
This Work	35.0	17.1	17.1	8	/	581	877	3016	14441	24.0
Example 10.12										
V&S 2012	34.0	19.9		8	0	180	694	2476	4900	30.0
V&S 2012	34.0	89.1	109.0	8	1	336	850	3291	12146	30.0
V&S 2013	34.0	2.4		7	0	573	899	3249	287	30.0
V&S 2013	34.0	4.4	6.8	7	1	705	1031	3939	386	30.0
S&V 2017	34.0	41.6		7	0	168	639	1843	14242	30.0
S&V 2017	34.0	2648.5	2690.1	7	1	312	783	2752	392539	30.0
This Work	34.0	1.4	1.4	7	/	598	914	3164	519	30.0

Table 4-18 contains the results for Example 10.13 which required no task splitting and was solved by S&V 2017 with eight events and by the other models with nine events. The proposed model was the largest on any single run except for having fewer binary (800) and continuous (1216) variables than V&S 2013 at $\Delta P = 1$ (945 and 1363) as well as fewer constraints (4256) than V&S 2013 at both $\Delta P = 0$ (4341) and $\Delta P = 1$ (5281). The proposed model was also the third fastest (30.7 s), after V&S 2013 at $\Delta P = 0$ (26.6 s) and V&S 2012 at $\Delta P = 0$ (30.6 s), however, after accounting for compounded solution times, required to verify the global optimal solution, the proposed model significantly outperformed the other models by up to ~255 s. Example 10.14 also required no task splitting although S&V 2017 was unable to converge or obtain the optimal solution of 36 hours with nine events, as shown in Table 4-18. In terms of

model size, the proposed formulation followed the common trend of being the largest with the exception of having fewer binary (800) and continuous (1216) variables than V&S 2013 at $\Delta P = 1$ (945 and 1363) as well as fewer constraints (4256) than V&S 2013 at both $\Delta P = 0$ (4341) and $\Delta P = 1$ (5281). These values are identical to those in Example 10.13. For a given single run, only V&S 2013 at $\Delta P = 0$ solved faster (547.9 s) than the proposed model (925.5 s), however when compounded solution time for iterations was accounted for, the proposed model was the fastest by a significant margin, as the other models did not scale well. V&S 2012 at $\Delta P = 1$ was unable to converge on the optimal solution after 10 000 s, having a best bound by this stage of ~30 hours. The total solution time for V&S 2013 was just under 5594 s (~1.5 hours) and, as mentioned above, S&V 2017 was unable to converge at either $\Delta P = 0$ or at $\Delta P = 1$. The optimal schedule for Example 10.14 as determined by the proposed model is shown in Figure 4-12.

Table 4-18. Computational Results for Examples 10.13 and 10.14

Model	H	Average CPU Time (s)	Total CPU Time (s)	Event Points	Δp	Binary Variables	Continuous Variables	Constraints	Nodes	RMIP
Example 10.13										
V&S 2012	36.0	30.6		9	0	204	785	2820	4857	24.0
V&S 2012	36.0	53.3	83.9	9	1	384	965	3760	7036	24.0
V&S 2013	36.0	26.6		9	0	765	1183	4341	3294	24.0
V&S 2013	36.0	79.5	106.1	9	1	945	1363	5281	7402	24.0
S&V 2017	36.0	155.0		8	0	192	730	2117	18921	24.0
S&V 2017	36.0	130.5	285.5	8	1	360	898	3173	17606	24.0
This Work	36.0	30.7	30.7	9	/	800	1216	4256	11886	24.0
Example 10.14										
V&S 2012	36.0	2305.6		10	0	228	876	3164	282640	21.6
V&S 2012	36.0	10000.0[1]	/	10	1	432	1080	4229	383929	21.6
V&S 2013	36.0	547.9		9	0	765	1183	4341	110734	21.6
V&S 2013	36.0	5045.9	5593.8	9	1	945	1363	5281	1267050	21.6
S&V 2017	37.0	10000.0[2]		9	0	216	821	2391	505518	21.6
S&V 2017	38.0	10000.0[3]	/	9	1	408	1013	3594	517330	21.6
This Work	36.0	925.5	925.5	9	/	800	1216	4256	74371	21.6

Best bound: [1]30.0, [2]28.6, [3]27.6

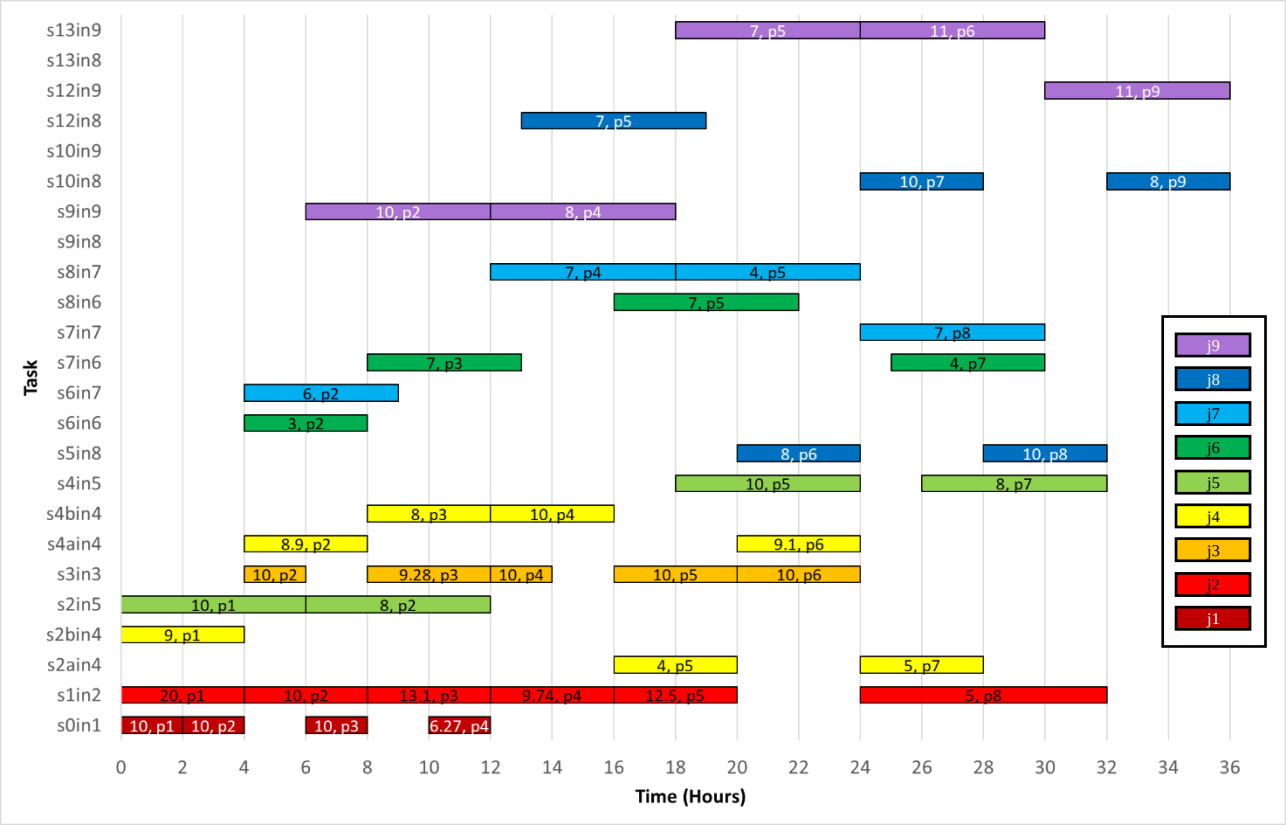


Figure 4-12. Optimal Schedule for Example 10.14 (Proposed Model)

4.4. Summary of Findings

Figure 4-14 and Figure 4-13 summarize the overall performance of the proposed model against the other four models considered through a number of metrics. Figure 4-14 displays the number of instances of a model performing the fastest with the optimal parameters and with total, cumulative time as well as the number of considered examples converging on the globally optimal solution.

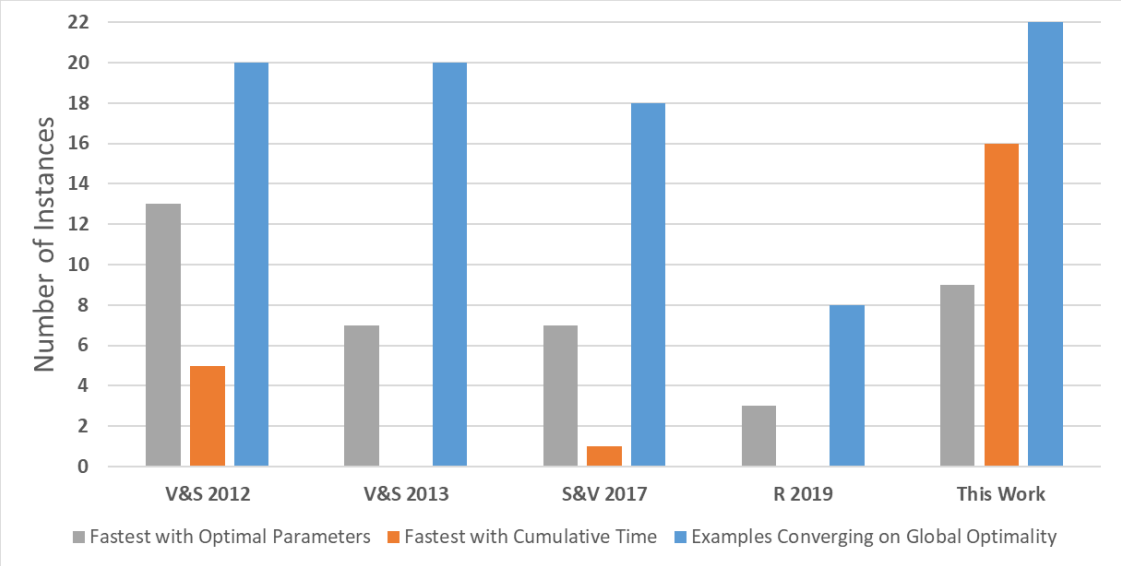


Figure 4-14. Model Computational Performance by Number of Instances

Figure 4-13 displays the maximum solution time for any one problem instance as well as the total solution time for all 22 instances. The figure also displays the greatest lead each model had over the next fastest model when it was fastest, and the greatest lag it had behind the fastest model, across all 22 problem instances. Note that the results for R 2019 are for the first 10 example instances only, which do not include ZW states. This heavily impacts the maximum and total solution time.

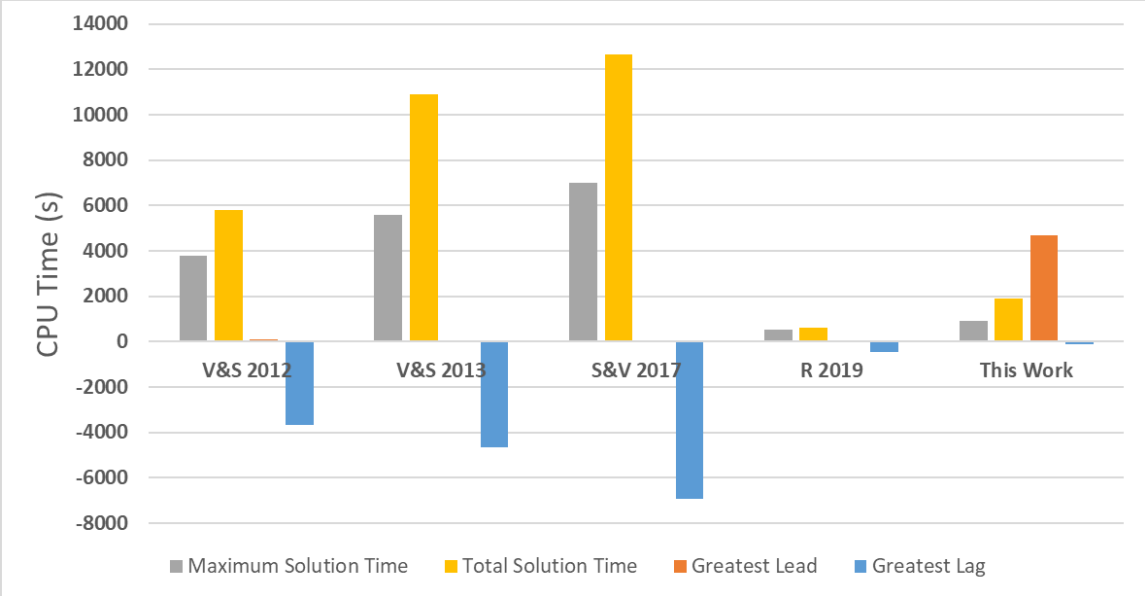


Figure 4-13. Model Computational Performance by CPU Time

The examples in which the proposed model was outperformed are Examples 4.1 and 10.4 to 10.8. These six examples were solved faster by V&S 2012 and only Example

10.8 was solved faster than the proposed model by S&V 2017. V&S 2013 and R 2019 did not outperform the proposed model in any of the examples considered. In all of these inconsistent examples, except Example 10.6, task splitting was not required by the other models in order to find the globally optimal solution. This infers a smaller number of binary variables for these problems when solved by V&S 2012 and S&V 2017, as the requirement of a nonzero splitting parameter expands the number of binary variables required by a model significantly. In these cases, the number of binary variables required by V&S 2012, at the maximum value of Δp investigated, was still less than half of that of the proposed model. This difference is perhaps the reason why the proposed model was outperformed in some cases.

Chapter 5

Conclusions and Recommendations

A new MILP formulation for the scheduling of multipurpose batch plants is presented which incorporates the techniques of rigorous conditional sequencing, pre- and post-processing unit wait and fractional extraction as well as facilitates task splitting without the need for iterative procedures on a task splitting parameter. Additionally, guidelines for rigorous exclusion of unnecessary constraints and variables are given in an attempt to diminish the model size.

It was shown how the proposed formulation overcomes the drawbacks of three-index based formulations by allowing tasks to effectively split over any number of events with a fixed model size for a given problem. This is done without the need for iteration or guesswork on the model parameters which compounds the problem size and complexity as well as the required computational time and may exclude potentially optimal solutions. The proposed formulation was able to determine the known best solutions for all the problem instances considered in as far as the optimal number of events over which a task should split is concerned.

It is acknowledged that the combined solution time of iteration for the determination of the optimal number of events is not shown in this work. Nonetheless a nested iterative procedure over both the number of events and the maximum number of events over which tasks can split is bound to exacerbate the problem and lead to higher solution times than if only a one-dimensional iterative procedure is required.

Furthermore, the proposed formulation was shown to be the most reliable in its ability to converge for all of the problems considered. The inability to converge in reasonable time (10 000 s) was demonstrated to occur in a number of examples solved by other

models. The proposed model converged on the optimal solution in no more than 925.5 seconds for any of the examples considered.

Of the 22 examples discussed, when comparing the solution time for fastest convergence to global optimality for a single run at the lowest number of total and splitting events, V&S 2012 performs the fastest or ties with the others in 13 examples, V&S 2013 in seven, S&V 2017 in seven, R 2019 in three and nine are solved in the fastest time or tied by the proposed model. However, for V&S 2012, five of these solutions were only obtainable at a number of splitting events greater than zero. Unless an algorithm is developed to determine a priori the optimal number of splitting events or it is guessed for a single run, iterations would result in a longer total solution time. Additionally, when one further iteration above those necessary to first obtain the optimal solution was performed on ΔP, in order to verify its optimality and provide a concrete stopping criterion for iteration, the proposed model solved the fastest in 16 of the examples, while V&S 2012 and S&V 2017 respectively solved five and one of the examples the fastest. V&S 2013 and R 2019 did not solve any examples the fastest in these circumstances. Of the six in which the proposed model was outperformed, it had the second fastest solution time in five. Furthermore, the proposed model and R 2019 are the only models which obtain the globally optimal solution in Example 1m while the proposed model is the only one which *converges* on the globally optimal solution within 10 000 seconds for Example 4.2.

The proposed model was outperformed in terms of CPU time by a maximum of ~378 seconds across all examples for any given single run. This occurred in Example 10.14. However, when one additional iteration on the number of splitting events was performed and the solution time summed for all necessary iterations, the proposed model outperformed the other models by a maximum of ~6948 seconds, just under two

hours, excluding the examples in which the V&S models did not converge. This corresponds to a CPU time reduction of 99.4% and occurred in Example 3.

It was shown that three-index formulations do not always scale well with increasing values of ΔP and the solution time can either increase exponentially or behave inconsistently with longer solution times at intermediate values of ΔP and smaller solution times at extreme values. It was also demonstrated how the size of a model is not always a direct indicator of performance, since for many of the examples considered, the proposed model was larger and yet it resulted in superior reliability, accuracy and speed. The proposed model solved every example in reasonable time (no more than 15.5 minutes), outperforming the other models in every example where task splitting was required, except in Example 10.6. In cases where the proposed model did not outperform the other models, it performed very comparably.

Finally, the proposed model allows for fractional extraction of states from a producing unit, further improving the flexibility and allowing, in some cases, better schedules to be determined. Together with the computational efficiency and steady convergence for all problems considered, the objectives of the work outlined in Section 1.4 have been satisfied.

The proposed model does not address resource considerations at the current time. It is possible to incorporate resource considerations by defining monotonically increasing continuous variables for the timing of resource utilization and enforcing that all tasks consuming a resource at a particular event begin and end at the same time as the resource times, as discussed in the literature. Alternatively, this can be incorporated efficiently by treating resources as states and modelling their regeneration at the event subsequent to the completion of the tasks which consumes them. A binary variable can be used to track when direct transfer to a consuming task at the subsequent event occurs

for the purposes of enforcing sequencing. This is as per the work by Vooradi and Shaik (2013).

References

Biegler, L. T., & Grossmann, I. E. (2004). Retrospective on optimization. *Computers and Chemical Engineering*, 1169-1192.

Draper, D. L., Jonsson, A. K., Clements, D. P., & Joslin, D. E. (1999). Cyclic Scheduling. *IJCAI'99 Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2* (pp. 1016-1021). Stockholm: Morgan Kaufmann Publishers Inc.

Floudas, C. A., & Lin, X. (2004). Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. *Computers and Chemical Engineering*, 2109-2129.

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

Ierapetritou, M. G., & Floudas, C. A. (1998). Effective continuous-time formulation for short-term scheduling: I. multipurpose batch processes. *Industrial and Engineering Chemistry Research*, 4341-4359.

Janak, S. L., & Floudas, C. A. (2008). Improving unit-specific event based continuous-time approaches for batch processes: Integrality gap and task splitting. *Computers and Chemical Engineering*, 913-955.

Janak, S. L., Lin, X., & Floudas, C. A. (2004). Enhanced continuous-time unit-specific event-based formulation for short-term scheduling of multipurpose batch processes: resource constraints and mixed storage policies. *Industrial and Engineering Chemistry Research*, 2516-2533.

Kallrath, J. (2002). Planning and scheduling in the process industry. *OR Spectrum*, 219-250.

Kondili, E., Pantelides, C. C., & Sargent, R. W. (1993). A general algorithm for short-term scheduling of batch operations - I. MILP formulation. *Computers and Chemical Engineering*, 211-227.

Lee, H., & Maravelias, C. T. (2017). Mixed-integer programming models for simultaneous batching and scheduling in multipurpose batch plants. *Computers and Chemical Engineering*, 621-644.

Lee, H., & Maravelias, C. T. (2018). Combining the advantages of discrete- and continuous-time scheduling models: Part 1. Framework and mathematical formulations. *Computers and Chemical Engineering*, 176-190.

Lin, X., & Floudas, C. A. (2001). Design, synthesis and scheduling of multipurpose batch plants via an effective continuous-time formulation. *Computers and Chemical Engineering*, 665-674.

Majozi, T. (2010). *Batch Chemical Process Integration.* Springer Science + Business Media.

Majozi, T., & Friedler, F. (2006). Maximization of Throughput in a Multipurpose Batch Plant under a Fixed Time Horizon: S-graph Approach. *Industrial and Engineering Chemistry Research*, 6713-6720.

Majozi, T., & Zhu, F. X. (2001). A novel continuous-time MILP formulation for multipurpose batch plants. 1. Short-term scheduling. *Industrial and Engineering Chemistry Research*, 5935-5949.

Maravelias, C. T., & Grossmann, I. E. (2003). New general continuous-time state-task network formulation for short-term scheduling of multipurpose batch plants. *Industrial and Engineering Chemistry Research*, 3056-3074.

Mockus, L., & Reklaitis, G. V. (1997). Mathematical programming formulation for scheduling of batch operations based on nonuniform time discretization. *Computers and Chemical Engineering*, 1147-1156.

Nie, Y., Biegler, L. T., & Wassick, J. M. (2012). Integrated scheduling and dynamic optimization of batch processes using state equipment networks. *AIChE Journal*.

Oxford Economics. (2019). *The Global Chemical Industry: Catalyzing Growth and Addressing Our World's Sustainability Challenges.* Washington DC: International Council of Chemical Associations (ICCA).

Pantelides, C. C. (1994). Unified frameworks for optimal process planning and scheduling. In D. Rippin, & J. Hale (Ed.), *Proceedings of the second international conference on foundations of computer-aided process operations* (pp. 253-274). Colorado: CACHE Publications.

Papageorgiou, L. G., Shah, N., & Pantelides, C. C. (1994). Optimal scheduling of heat-integrated multipurpose plants. *Industrial Engineering and Chemistry Research*, 3168-3186.

Pattinson, T., & Majozi, T. (2010). Introducing a new operational policy: the PIS operational policy. *Computers and Chemical Engineering*, 59-72.

Puranik, Y., Samudra, A., Sahinidis, N. V., Smith, A. B., & Sayyar-Rodsari, B. (2018). Infeasibility resolution for multi-purpose batch process scheduling. *Computers and Chemical Engineering*, 69-79.

Rakovitis, N., Li, J., & Zhang, N. (2018). A novel modelling approach to scheduling of multipurpose batch processes. *Proceedings of the 13th International Symposium on Process Systems Engineering* (pp. 1333-1338). San Diego: Elsevier.

Rakovitis, N., Li, J., & Zhang, N. (2019). An improved approach to scheduling multipurpose batch processes with conditional sequencing. *Proceedings of the 29th European Symposium on Computer Aided Process Engineering* (pp. 1387-1392). Eindhoven: Elsevier.

Sanmarti, E., Friedler, F., & Puigjaner, L. (1998). Combinatorial technique for short term scheduling of multipurpose batch plants based on schedule-graph representation. *Computers and Chemical Engineering*, S847-S850.

Schilling, G., & Pantelides, C. C. (1996). A simple continuous-time process scheduling formulation and a novel solution algorithm. *Computers and Chemical Engineering*, S1221-S1226.

Seid, E. R., & Majozi, T. (2012). A robust mathematical formulation for multipurpose batch plants. *Chemical Engineering Science*, 36-53.

Shaik, M. A., & Floudas, C. A. (2008). Unit-specific event-based continuous-time approach for short-term scheduling of batch plants using RTN framework. *Computers and Chemical Engineering*, 260-274.

Shaik, M. A., & Floudas, C. A. (2009). Novel unified modeling approach for short-term scheduling. *Industrial and Engineering Chemistry Research*, 2947-2964.

Shaik, M. A., & Vooradi, R. (2017). Short-term scheduling of batch plants: reformulation for handling material transfer at the same event. *Industrial and Engineering Chemistry Research*, 11175-11185.

Smith, E. M., & Pantelides, C. C. (1995). Design of reaction/separation networks using detailed models. *Computers and Chemical Engineering*, S83-S88.

Vooradi, R., & Shaik, M. A. (2012). Improved three-index unit-specific event-based model for short-term scheduling of batch plants. *Computers and Chemical Engineering*, 148-172.

Vooradi, R., & Shaik, M. A. (2013). Rigorous unit-specific event-based model for short-term scheduling of batch plants using conditional sequencing and unit-wait times. *Industrial and Engineering Chemistry Research*, 12950-12972.

Woolway, M., & Majozi, T. (2018). A novel metaheuristic framework for the scheduling of multipurpose batch plants. *Chemical Engineering Science*, 678-687.

Zhang, X., & Sargent, R. W. (1996). The optimal operation of mixed production facilities - a general formulation and some approaches for the solution. *Computers and Chemical Engineering*, 897-904.