processes

MDPI

Article

# Using Simulation for Scheduling and Rescheduling of Batch Processes

**Girish Joglekar**

Batch Process Technologies, Inc., 112 Eden Court, West Lafayette, IN 47906, USA; girish53@gmail.com

Received: 3 September 2017; Accepted: 26 October 2017; Published: 2 November 2017

**Abstract:** The problem of scheduling multiproduct and multipurpose batch processes has been studied for more than 30 years using math programming and heuristics. In most formulations, the manufacturing recipes are represented by simplified models using state task network (STN) or resource task network (RTN), transfers of materials are assumed to be instantaneous, constraints due to shared utilities are often ignored, and scheduling horizons are kept small due to the limits on the problem size that can be handled by the solvers. These limitations often result in schedules that are not actionable. A simulation model, on the other hand, can represent a manufacturing recipe to the smallest level of detail. In addition, a simulator can provide a variety of built-in capabilities that model the assignment decisions, coordination logic and plant operation rules. The simulation based schedules are more realistic, verifiable, easy to adapt for changing plant conditions and can be generated in a short period of time. An easy-to-use simulator based framework can be developed to support scheduling decisions made by operations personnel. In this paper, first the complexities of batch recipes and operations are discussed, followed by examples of using the BATCHES simulator for off-line scheduling studies and for day-to-day scheduling.

**Keywords:** batch process; scheduling; simulation; coordination control; rescheduling

## 1. Introduction

The problem of scheduling multiproduct and multipurpose batch processes has been studied extensively over the past 30 years [1]. Scheduling involves making decisions for the assignment of tasks to processing units, and the sequencing of various products through the processing facility. Typically, each product is made according to its unique recipe. In some methodologies, the underlying recipes are modeled using either a State-task network (STN) or resource-task network (RTN), and a mixed integer linear programming (MILP) formulation based on discrete time or continuous time representation is used for solving the optimization problem. In some problems where orders can be treated as individual batches, each order moves through various production stages as a discrete entity, and a sequential MIP formulation with order-indexed decision variables for assignments to equipment units and precedence decisions are used to solve the problem. In both the approaches, the manufacturing recipes are greatly simplified in order to keep the problem size to a manageable level so that it can be solved in a reasonable amount of time.

Some of the assumptions made in simplifying the recipes can have significant impact on the solutions generated from these formulations. For example, consider the simple STN shown in Figure 1. It consists of three states, S1, S2 and S3, two tasks T1 and T2, and two units U1 and U2, one for each task. The processing times of the two tasks are t1 and t2, respectively.

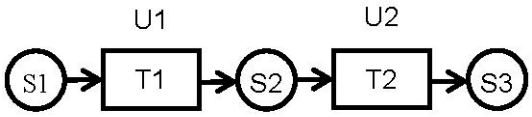



**Figure 1.** A simple state-task.

In a typical MILP formulation, material transfers S1 to T1, T1 to S2, S2 to T2 and T2 to S3 would be assumed to be instantaneous. The resulting schedule for making two batches of T1 is shown as a Gantt chart in Figure 2. Since the transfers are instantaneous, the end of T1 matches the start of T2.

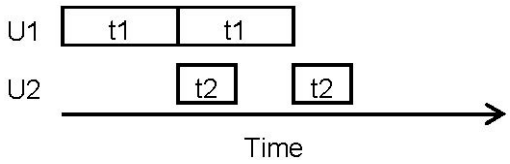



**Figure 2.** Schedule with zero transfer times.

However, in a real process the transfers are rarely instantaneous. The actual recipe of the two operations may be closer to the recipe network shown in Figure 3. As shown, each task may consist of 3 subtasks, Fill, Mix and Empty. Suppose that processing times t1 and t2 defined earlier are the durations of the Mix subtasks of the two tasks. The transfer of material into task T1 takes place during its Fill subtask, and the transfer from task T1 into T2 takes place during their Empty and Fill subtasks, respectively. As a result, both units, U1 and U2, must be active simultaneously during the transfer step. Suppose the recipe network in Figure 3 is used to drive a simulation model. As shown in the Gantt chart generated by the simulator, shown in Figure 4, the first batch on U2 starts at time f1 + t1, where f1 is the duration of the Fill subtask of task T1 (shown as solid black rectangle). Similarly, the start time of the second batch of T1 will be at (f1 + t1 + e1), where e1 is the duration of the Empty subtask of T1. The duration of the Fill subtask of T2 is the same as that of the Empty subtask of T1, namely e1. The duration of the Empty subtask of T2 is e2, shown as green rectangle. Therefore, the start times of the batches on U2 as predicted by the MILP solution would not be the same as the actual process. It is immaterial how the durations of the material transfers are accounted for, whether they are included in the task times t1 and t2, or treated as additions to t1 and t2. This simple example illustrates that regardless of how the task durations are interpreted, the 'schedule' on the actual process will not match the solution generated by the scheduler when transfer times are not ignored. As the transfer times become more significant, the reliability of the scheduler reduces further.

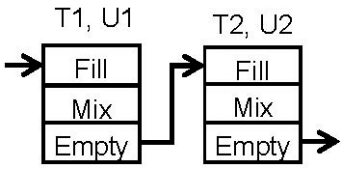



**Figure 3.** A simple recipe network.

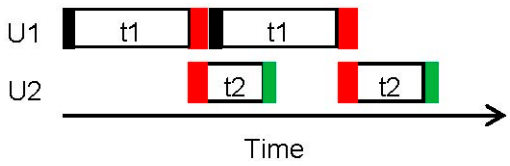



**Figure 4.** Schedule with non-zero transfer times.

Another important characteristic of batch processes is that the underlying recipes are inherently very complex structurally as well as logistically. For example, consider the recipe network shown in Figure 5 for the manufacture of product P1.

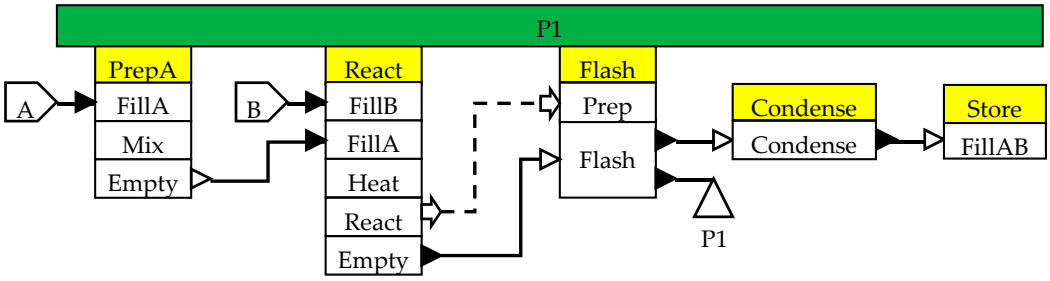



**Figure 5.** Recipe for manufacture of product P1.

A task in a recipe may consist of several subtasks which are executed in the specified sequence once that task starts in the assigned piece of equipment. For example, task React (yellow in color) in Figure 5 consists of 5 subtasks (white boxes) FillB, FillA, Heat, React and Empty. The flow of material between tasks takes place during specific subtasks. Thus, material from task PrepA is transferred into task React only when the pieces of equipment assigned to them are in Empty and FillA subtasks respectively. Some subtasks form a semi-continuous chain, for example, subtask Flash in Figure 5. The Flash subtask takes material from the Empty subtask of the React task, splits it into two streams, sending one stream to the Condense subtask, which in turn sends it to the FillAB subtask, while the second stream is removed as product P1. Thus, in order to execute the Flash subtask, units assigned to tasks React, Flash, Condense and Store must be in the appropriate subtasks. In MILP based formulations, subtask level details are rarely considered at their fullest level, and semi-continuous transfers across multiple tasks are often treated as single tasks.

In batch processes, the resource usage is typically at the subtask level. For example, the Heat subtask of the React task may require Low Pressure Steam, the React subtask may require Cooling Water, the Flash subtask may require High Pressure Steam, the Condense subtask may require Refrigeration, and so on.

The logistics of coordinating various operations so that the subtasks associated with material transfers are ready just in time are at the heart of scheduling complex recipe driven batch processes. Also, if resources are required in order to execute subtasks, and if there are constraints on the maximum availability of resources, then the resolution of the competition for resources becomes part of the logistics.

Several other factors, such as process variability, operating rules, multiple suitable equipment, equipment dependent batch size and cycle times, sequence dependent cleaning/setup, and so on, influence the operation of a batch process. If the underlying recipes are simplified for solving a scheduling problem, which is often the case in MILP based formulations, the resulting solutions are more likely to be 'broad brush' solutions than 'actionable' solutions. An 'actionable' solution is the information the operations personnel can use with certainty.

The paper by Joglekar [2] showed that using simplified recipe models that are identical to the ones used in MILP formulations, a simulator can generate high quality schedules in a fraction of time as compared to MILP based techniques. This paper demonstrates a simulation based methodology, which incorporates all the complexities of batch processes described above, for generating schedules that are very realistic or 'actionable', and verifiable. Simulation has not been extensively used in addressing scheduling problems, mainly because of the inherently myopic view that this technique takes in making assignment decisions. Simulation coupled with heuristics has been applied to discrete systems. However, in general the characteristics of batch processes are very complex compared to discrete systems. The complexities preclude the use of discrete event simulation based systems in

applications related to batch processes4. Simulation based framework applied to batch processes, as reported by Chu et al. [3] augments the underlying MP based formulations, and has similar limitations as stated earlier. The paper by Petrides et al. [4] discusses the roles played by simulators and finite capacity scheduling tools in optimizing biopharmaceutical batch processes, but the tools employed lack the ability to accurately model the underlying complexities.

Often, it is necessary to reschedule a batch process because of changes in external factors used in generating the original schedule, and due to the deviations in the predicted versus actual process trajectories. The rescheduling or reactive scheduling techniques often use RTN based MILP formulations [5,6] and therefore have the same limitations as discussed earlier. The simulation based methodology presented in this paper lends itself very easily to continuous monitoring and rescheduling. Although simulation based approach does not guarantee optimality, performance criteria based on equipment and resource utilization can be established to evaluate the quality of the results. Moreover, simulation predicts time series data for selected process variables and generates subtask based Gantt charts which can be used for tracking the underlying process over time. A supervisory control system could trigger rescheduling decisions based on deviations in the process trajectory.

The following are some of the commercially available simulators for batch processes: Batch Process Developer [7], SuperPro Designer and SchedulePro [8], gPROMS [9], BATCHES [10]. DynoChem [11] is designed for simulating single unit operations used in batch processes. None of these simulators, other than BATCHES, has the ability to dynamically assign a task to a piece of equipment, a key functionality required for solving scheduling problem. In addition, several discrete event simulators are available for discrete manufacturing systems. However, in general batch processes are very complex compared to discrete systems, which precludes the use of discrete simulators for applications related to batch processes.

In this study, the basic methodology of simulation based scheduling is illustrated with a simple process using the BATCHES simulator. First, the various aspects of modeling recipes and the time advance mechanism of the simulator are discussed. Next, the iterative process of doing 'what ifs ... ' to explore the parameter space is explained. At the end, the solution of a scheduling problem is presented.

## 2. Recipe Models

The need for a systematic framework for defining and managing process recipes was felt strongly by the process control and automation groups within the batch process industry. The basic concepts and terminology for batch process control were adopted in 1995 as the ANSI/ISA-88.01 standard (S88), with addition of more parts later, and an update in 2010 [12,13]. Over the past 20+ years, the S88 standards have been used extensively by the automation vendors. The manufacturing recipe information is at the core of the three kinds of control for batch processes: basic, procedural and coordination [14], and represents the ultimate level of detail that is needed to run the underlying process. A manufacturing recipe is typically implemented by a distributed control system (DCS), such as DeltaV. For example, an operation named REACT in a biopharmaceutical process consists of the following 13 phases [15]: Setup, Add A, Select Path, Receive Product, Dissolve, Add B, Adjust pH, Reaction, Hold, Transfer, CIP Setup, Rinse, and Caustic Wash. The categories of information associated with each phase are: Description, Formula Parameters, Report Parameters, Run Logic, Hold Logic, Abort Logic, Failure Conditions and Stop Logic. In addition, there is documentation of Alarm conditions, Interlocks and shutdowns. For this example, the functional specifications of just one unit procedure spanned 75 pages, with most of the information stored in the DCS system. Programming and maintaining a DCS require significant resources during commissioning and operation of a process. Since a simulation model is a surrogate of the process, if the same level of detail as the control recipe is incorporated into simulated recipes, the simulated results would be very accurate.

In a BATCHES simulation model, a recipe is represented by a top level graphical network like the one shown in Figure 5. The basic building blocks of a recipe network are: Recipe, Task, Subtask,

Raw material (pentagon), Sink (vertical triangle), material input (hollow or solid triangle on the left vertical edge of a subtask), material output (hollow or solid triangle on the right vertical edge of a subtask), flow line (solid line connecting material output to input), signal start (hollow arrow on the right vertical edge of a subtask), signal end (hollow arrow on the left vertical edge of a subtask), signal (dotted line connecting signal start and end). A recipe is simply a name given to a set of tasks. A task in a recipe network defines the set and sequence of elementary steps performed in the assigned unit, and is similar to a unit procedure in S88. A subtask is an elementary step that represents a specific physical/chemical change and is similar to a phase in S88. A sequence of subtasks defines a task, like a series of phases defines a procedure. A recipe network is like a general recipe in S88. A simulation model also consists of an equipment network which defines the set of units at a specific site. Each unit is a physical resource that is required for performing a task. Associated with each task is a list of equipment items that are suitable to perform that task. Thus, a specific combination of recipe and equipment network defines a site recipe in S88, and a specific batch of a task in a specific piece of equipment defines a master recipe. A batch is one instance of execution of a task on a unit. Typically, a recipe produces one material output stream that is considered primary and may produce additional streams that are considered secondary. A batch of material is the quantity of the primary stream produced when the associated task is executed on one of the suitable units. The size of each batch may or may not be the same, dictated by the combination of the recipe and the unit on which that task was performed. In order to produce the required amount of the primary stream, multiple batches are typically made.

The parameters associated with tasks and subtasks define the details of the associated recipe[6]. The important task parameters are: list of suitable equipment items, logic that determines how a piece of equipment is assigned to the task. The important subtask parameters are: dynamic model that best describes the physical/chemical changes taking place during that subtask, how a subtask ends, operator requirements, utility requirements, and state event conditions. The parameters associated with flow lines determine the amount transferred and the associated flowrate.

## 3. Model and Process Execution

From the discussion in the previous section, it is clear that a BATCHES recipe network can accurately represent the details of a product's manufacturing recipe, and is structurally very similar to the information that drives the batch control software. The main difference between simulation and procedural control is that in order to run a process through time the simulator has to make the key decisions of assigning units to tasks at different points in time, whereas the control software typically relies on an operator to make that decision. An operator, in turn, makes that decision based either on a predefined schedule or on the current process status and experiential knowledge. Once a task is initiated on a unit both the simulator and procedural control implement the recipes. Thus the assignment decisions made by the simulator during a simulation run generate the schedule, which can be given to the operators or the procedural control system as the blue print for running the process. In this section, the key concepts used by the simulator to make the assignment are discussed.

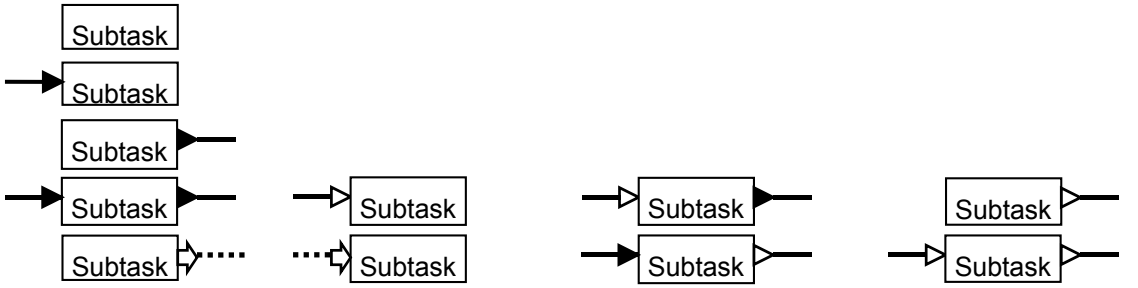
### *3.1. Time Advance Mechanism in BATCHES*

Time advance mechanism is the algorithm used by the simulator to march the specified process through time. The BATCHES simulator uses the time advance mechanism common in combined discrete and dynamic simulators [16,17]. In addition to the recipe driven batch and semi-continuous processes, the simulator also can be used effectively for discrete manufacturing systems. At any discontinuity (event), the simulation executive tries to assign a unit to a task or assign the required material and resources to start a subtask. After all assignments are completed at an event, the simulator integrates in time the state variables associated with all active subtask models until the next event occurs (time or state event). The following are the key decisions that determine the outcome of a simulation run: the assignment of a unit to initiate a task, the assignment of resources to start a

subtask and its associated flows, end a subtask and advance the unit to the next subtask, after ending the last subtask of a task end the task and make the associated unit available for next assignment. These decisions predict the process trajectory under the specified conditions, which in essence if the resulting schedule.

### *3.2. Subtask Types*

The type associated with a subtask plays a role in the unit and resource assignment decisions made by the simulator, and is determined by its graphical representation in the recipe network. Since the graphical representation is a modeling decision, a recipe network is a visualization of various subtask level interactions and the decision logic. There are four subtask types: Master, Slave, Chaining and Decoupling. The graphical coding of the subtask types is shown in Figure 6 [18].




(**a**) Master subtasks (**b**) Slave subtasks (**c**) Chaining subtasks (**d**) Decoupling subtasks

**Figure 6.** Graphical depictions of subtask types.

More precise information is needed to determine how flows are executed in a batch process. A material input to a subtask can be a hollow or solid triangle on its left vertical edge. A material input that pulls material from upstream is depicted as a solid triangle. If a material input accepts material pushed by an upstream output it is depicted as a hollow triangle, a passive input.

A material output from a subtask can be a hollow or solid triangle on its right vertical edge. A material output that pushes material downstream is depicted as a solid triangle. If a material output allows downstream input to pull material then it is depicted as a hollow triangle, a passive output.

Thus, the output and input on a material transfer line must be of opposite 'polarity', that is, either the output pushes material (solid triangle) and input receives material (hollow triangle) or the input pulls material (solid triangle) and the output allows material withdrawal (hollow triangle).

Based on its graphical representation, the following rules determine a subtask's type.

*Master subtask*: A subtask is a master subtask if it has

- no material inputs or output and no signal in or out, or
- only pulling input(s) and no material output(s), or
- only pushing output(s) and no material input(s), or
- pulling input(s) and pushing output(s), or
- only signal start(s).

The various depictions that make a subtask a master subtask are shown in Figure 6a.

*Slave subtask*: A subtask is a slave subtask if it has a passive input and no material output, or has a signal end and no material input or output. The various depictions that make a subtask a slave subtask are shown in Figure 6b.

*Chaining subtask*: A subtask is a chaining subtask if it has a passive input and a pushing output, or has a passive output and a pulling input. In the first case, the upstream subtask pushes material into a chaining subtask, and in turn the chaining subtask pushes material downstream (forward

chaining). In the second case, the downstream subtask pulls material from a chaining subtask, and in turn the chaining subtask pulls material from upstream (backward chaining). The two depictions that make a subtask a chaining subtask are shown in Figure 6c. A chaining subtask is part of a semi-continuous chain.

*Decoupling subtask*: A subtask is a decoupling subtask if it has a passive output and no material input, or a passive input and a passive output. A decoupling subtask allows the downstream subtask to pull material from it, and if it has passive input allows the upstream subtask to push material into it asynchronously. The depictions that make a subtask a decoupling subtask are shown in Figure 6d. A storage tank is a typical example of a decoupling subtask.

A subtask can have any number of material inputs and outputs, each connected to a different upstream or downstream subtask. All inputs 2 and above are solid triangles, that is, they pull upstream material. All outputs 2 and above are solid, that is, they push material downstream.

In a batch process, a semi-continuous chain rarely forms a single or nested recycle loop. If there is a continuous recycle loop, then exactly one subtask in the loop must be a decoupling subtask. A recipe network may form a batch recycle loop where material leaving a task during one subtask may return to the same task during another subtask, which would necessarily be at a different time.

When a task advances into a subtask, the following are the main steps in executing that subtask: wait until all the necessary conditions to start the subtask are satisfied, start all the subtasks controlled by the master subtask, implement the actions associated with the subtask as specified by the user, detect when the conditions for ending the subtask are satisfied, end the subtask, inform all other interacting subtask and advance the task to the next subtask. When the last subtask of a task is completed, the task is ended and the equipment assigned to it is released.

### *3.3. Task Types*

The type associated with a task plays a role in the assignment of a unit to that task. A task can be independent or dependent. A task is independent if its first subtask is of type master. Otherwise, it is a dependent task. Like subtasks, task type is determined by its graphical representation in the recipe network.

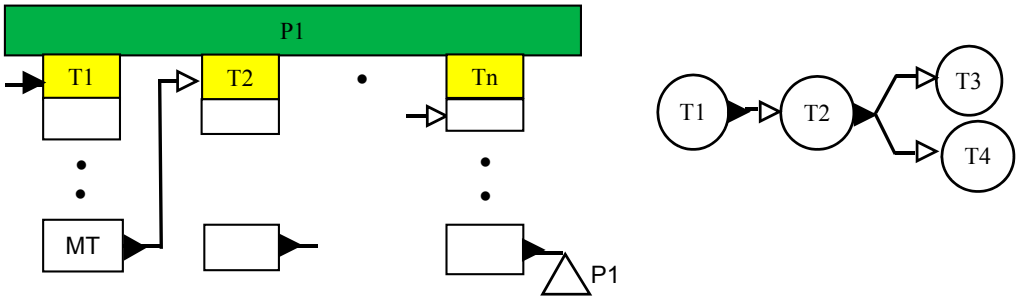
An independent task can be initiated through a sequence directive. In its simplest form, a sequence directive identifies the task to be initiated, the suggested start time for the first batch of the task, the number of batches of the task to be initiated, and the minimum time elapsed between consecutive initiations of the task (the intra-entry time). Note that the two time parameters define lower bounds. As described in the time advance mechanism, the simulation executive can only check if a task can be assigned to a unit at any discontinuity, it cannot guarantee assignment at a particular time. That is determined by the state of the process at that time. Similarly, the actual time elapsed between two consecutive batches of a task may be greater than or equal to the specified intra-entry time.

A dependent task is initiated when an upstream unit advances into a subtask that pushes material into its first subtask. The upstream subtask generates a request, and the queue processing mechanism initiates a task and the material is transferred downstream.

### *3.4. Recipe Network Topology*

Based on the structure of the graphical model of a recipe, several patterns can be identified that can be used to guide the simulator in making assignment decisions.

A recipe can be entirely front-end driven. Consider the recipe network shown in Figure 7a. It has only one independent task, T1.




(**a**) Simple flow (**b**) Multiple branches

**Figure 7.** Front-end driven recipe patterns with single independent task.

Once a batch of T1 is initiated, say through a processing directive, task T2 is started when a unit in task T1 reaches the MT subtask. Thus, T2 is 'scheduled' by T1. The same applies to all other tasks in a front-end driven recipe. A different pattern, shown in Figure 7b, follows the same principle that an upstream subtask triggers a downstream task but has multiple branches that are triggered. Same or different subtasks of T2 may trigger T3 and T4. Of course, these patterns could be arbitrarily complex at the subtask level.

The recipe pattern shown at the task level in Figure 8 can also be considered as front-end driven, but has multiple independent tasks.

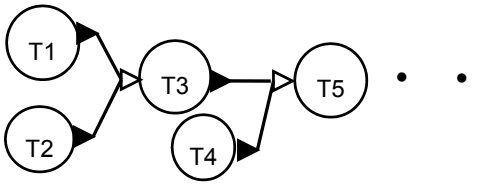



**Figure 8.** Front-end driven recipe patterns with multiple independent task.

Tasks T1, T2 and T4 are independent in this example presented in Figure 8.

Another common pattern in recipes consists of front-end driven segments that are joined by decoupling subtasks, such as a storage tank, as shown in Figure 9.

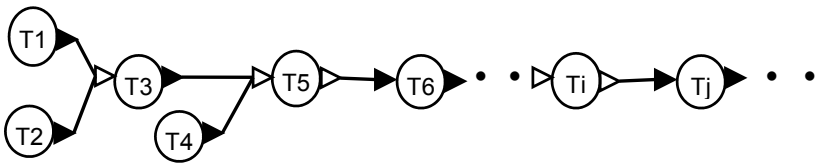
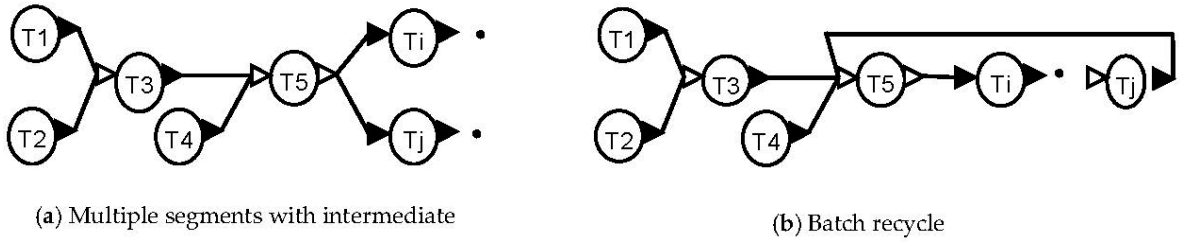



**Figure 9.** Front-end driven recipe patterns with decoupling subtask.

Tasks T5 and Ti are storage operations. Tasks upstream of T5 form a front-end driven segment, tasks between T5 and Ti form another front-end driven segment starting with T6, and the tasks downstream of Ti form a third front-end driven segment. Typically, the materials in tasks T5, Ti etc. are stable intermediates that can be stored for a long time. Additionally, each segment may represent a facility that is at different physical location.

In addition, various combinations of the front-end driven segments with or without intermediate storage are possible, including recycle of material. Some examples are shown in Figure 10.




(a) Multiple segments with intermediate (b) Batch recycle

**Figure 10.** Additional front-end driven recipe patterns with decoupling subtask.

In Figure 10a, multiple segments use the intermediate from storage task T5. This pattern would represent a process where multiple finished products are made from an intermediate. In Figure 10b, material from Tj is recycled to T5. This would represent a process where a chemical is recovered, such as a solvent, and recycled in a batch recycle mode.

Typically, the most important decision variable associated with each front-end driven segment of a recipe is the time between the successive batches of an independent task in that segment.

## 4. Simulation Based Scheduling Methodology

In this section, a user directed iterative approach to using simulation for generating a schedule is presented. The basic steps are shown in Figure 11.

The first step is to build a simulation model that has the appropriate level of detail. As a general rule, the more detailed and accurate the model of underlying process, the better the results predicted by the simulator. The model building phase often serves as a way to capture knowhow from various stakeholders and incorporate all significant operational level details into the model, the most important component being the recipe networks associated with the various products.

The next step is to develop a process signature for each recipe. This is achieved by initiating enough batches of the independent tasks to produce some amount of finished product(s). Typically initiating one batch each of independent tasks is sufficient to develop a preliminary process signature. If there are constraints on the process, it may be necessary to run the model for longer time so that all the constraints are manifested during the run. Developing process signature may require an exploratory approach where process complexities are added gradually, providing insights into process behavior at each stage. The following are some of the parameters that broadly characterize a process: effective batch cycle time, batch size of each stage, average throughput of each stage and lag times between any two subtasks in a recipe. At 'steady state', the average throughput of each stage is equal to the throughput of the slowest stage (the bottleneck).

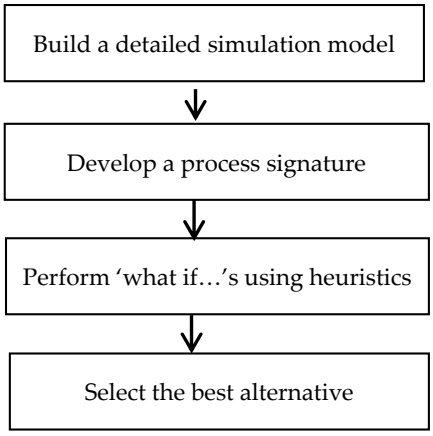



**Figure 11.** Steps in simulation based scheduling methodology.

The process signature is crucial in specifying the information that drives a simulation run. One of the driving mechanisms used for triggering independent tasks is 'sequence directive'. A sequence directive specifies the time at which to start the first batch, elapsed time between successive batches, and number of batches to initiate of the specified independent task. The guidelines described below for specifying the sequence directives constitute the heuristics used in running the process:

- For an independent task, the time lag between successive batches in a stage depends of the effective batch cycle time and the number of parallel units in that stage
- If a stage has multiple independent tasks, the task to start first can be identified from the recipe and the offset for the triggering of the first batch of other independent tasks can be determined from the process signature

During a typical simulation study, several simulation runs are made by changing the values of the desired input parameters. In this iterative approach, the results of a run are analyzed, and the values to be tried next are determined based on the analysis, trends established from prior runs and experiential knowledge.

## 5. Scheduling of a Specialty Chemical Process

The methodology of using simulation for scheduling is illustrated with a specialty chemical process. The recipe network used in this example is shown in Figure 12.

The manufacture of product P1 consists of 6 operations (tasks in yellow rectangles) REACT, PREPA, BFILTER, FLASH, COLDWCOND and SOLST. The chemical reaction between A and B takes place in a solvent SOL and in the presence of a catalyst CAT. After the reaction, the catalyst is separated using a batch filter. The filtrate is transferred to a batch flash unit where heat is added and the vapor is condensed in a condenser, and the condensate is stored in a solvent storage tank. Most of the solvent and reactants are recovered during the flash, and the remaining liquid is removed as the product P1. Some product is entrained in the condensate. For this study, the recipe was simplified to accumulate the recovered solvent. Therefore, there is only one subtask in the SOLST task. In the actual process, the recovered solvent would be recycled into the REACT and PREPA tasks. The key recipe details are given in Table 1.

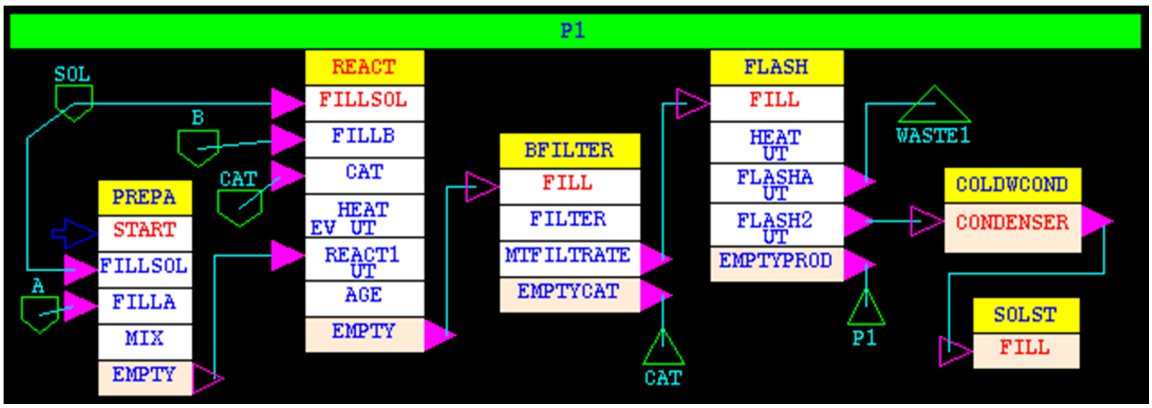



**Figure 12.** Recipe network of a specialty chemical process.

**Table 1.** Details of the recipe for P1.

| Task | Subtask | Duration (h) | Material | Utilities |
|---|---|---|---|---|
| REACT | FILLSOL | 2.0 | SOL, 760 kg | |
| | FILLB | 1.0 | B, 136.24 kg | |
| | CAT | 1.0 | CAT, 24 kg | |
| | HEAT | 3.33 | | STEAM, 50 MJ/h |
| | REACT1 | 5 | | COLD WATER, 40 MJ/h |
| | AGE | 0.5 | | |
| | EMPTY | 0.5 | To FILL | |
| PREPA | START | 0.0 | | |
| | FILLSOL | 1.0 | SOL, 760 kg | |
| | FILLA | 0.25 | A, 30 kg | |
| | MIX | 1.0 | | |
| | EMPTY | 0.0 | To REACT1 | |
| BFILTER | FILL | 0.5 | From EMPTY | |
| | FILTER | 0.427 | | |
| | MTFILTRATE | 0.5 | To FILL | |
| | EMPTYCAT | 0.5 | To Sink P1 | |
| FLASH | FILL | 0.5 | From MTFILTRATE | |
| | HEAT | 0.5 | | STEAM, 100 MJ/h |
| | FLASHA | 0.1 | | STEAM, 100 MJ/h |
| | FLASH2 | 8.2 | | STEAM, 100 MJ/h, |
| COLDWCOND | CONDENSER | 8.2 | From FLASH2 To FILL | COLDWATER, 100 MJ/h |

The recipe has one independent task, REACT. The PREPA task is initiated by a signal (signal input on subtask START) triggered 1.0 h after the HEAT subtask of the REACT task starts. Thus, the remainder of HEAT (2.33 h) is enough to cover the 2.25 h required for the first 3 subtasks of PREPA task, allowing the material to be just in time for feeding the reactor.

Two utilities, STEAM and COLDWATER, are required at the specified rates of consumption for the entire duration of the associated subtask. Each utility is constrained at the maximum rate of plant-wide consumption of 100 MJ/h.

One piece of equipment is suitable to perform each task.

The objective of the study is to determine the maximum amount of P1 that can be produced over a span of 168 h assuming that the plant is empty at the beginning.

The simulation runs were made on an HP 15 TouchSmart laptop, IntelCore i3-3110M@2.64 GHz processor, running the Xubuntu 16.04 64-bit operating system.

### *5.1. Process Signature*

To get the process signature, one batch of the REACT task was initiated.

The time required to process the material completely is 24.06 h, and the slowest task is REACT with batch cycle time of 13.33 h, which is also the process cycle time. The Gantt chart for this run is shown in Figure 13.

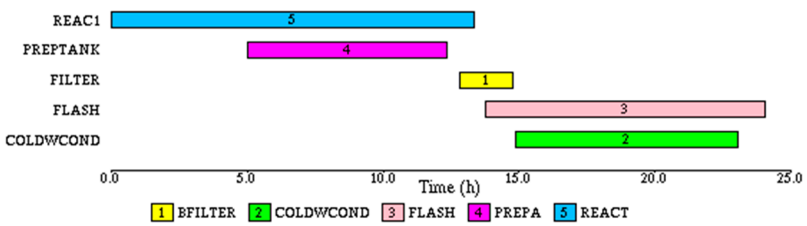
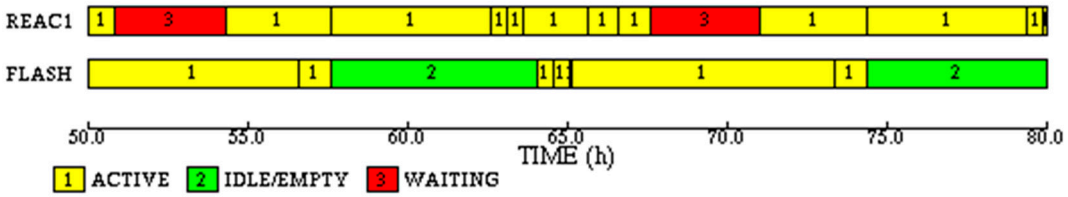



**Figure 13.** Gantt chart for the one-batch run.

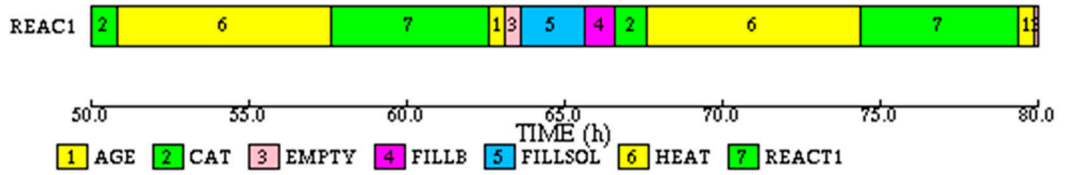
### *5.2. Base Case*

Since the cycle time is 13.33 h, for the given time horizon of 168 h it should be possible to make 12 complete batches. Accordingly, for this run the sequence directive was set to initiate 12 batches of the REACT task without any delay between successive batches.

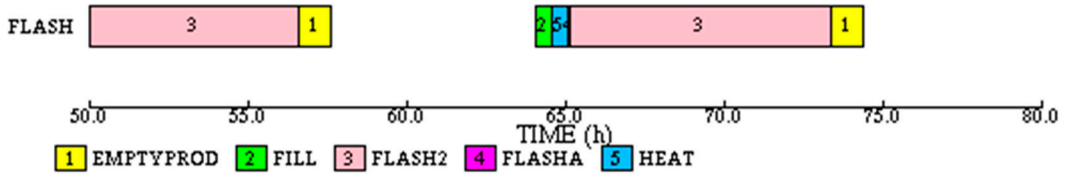
The simulation shows that the time required to make 12 batches is 233.7 h, much higher than expected. The reason for the increased makespan is the constraint on the STEAM utility. When HEAT, FLASHA and FLASH2 subtasks are running in FLASH, all of the available STEAM utility is consumed. Therefore, even if the reactor batches are started without any delay, the HEAT subtask of REACT task has to wait for STEAM to become available. In the status based Gantt chart for REAC1, shown in Figure 14a, the waiting times are shown in red. The subtask based Gantt chart in Figure 14b shows that the waiting occurs during the HEAT subtask. The waiting time on REAC1 can be eliminated by forcing idle time between successive batches on the reactor so that the end of FLASH2 is synchronized with the beginning of subtask HEAT in the reactor. This delay is approximately 19.06 h. Thus, the effective cycle time of the process is 19.06 h, thereby reducing the number of batches that can be made in the specified time.




(**a**) Status based Gantt chart for REAC1 and FLASH units




(**b**) Subtask based Gantt chart for REAC1




(**c**) Subtask based Gantt chart for FLASH

**Figure 14.** Subtask and status based Gantt charts for REAC1 and FLASH for base case.

Another simulation run was made with 19.06 h elapsed time between the starts of successive reactor batches. This eliminates the waiting times on the reactor. The makespan is still 233.7 h. The task based Gantt chart for this run is shown in Figure 15.

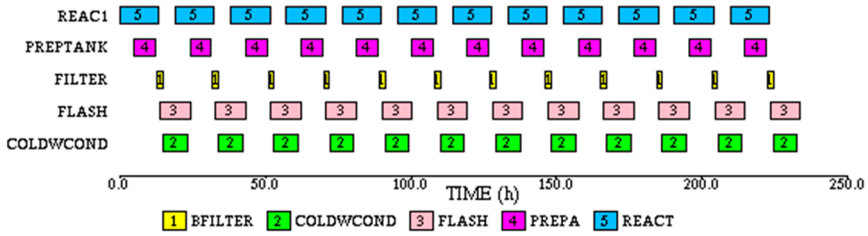



**Figure 15.** Gantt chart with 19.06 h elapsed time between reactor batches.

For this case 8 batches can be completely processed in 168 h. This case also represents the maximum number of batches that can be made under the given constraints. Thus, due to the constraints the process cycle time increases from 13.33 to 19.06 h, a 43% increase over the unconstrained case.

### *5.3. Accurate Heat Consumption Profile*

It should be noted that the timing of the competition for STEAM by two different subtasks creates a bottleneck, and also their relative positions in the recipe structure cannot be changed. A detailed study of the FLASH operation showed that the demand for STEAM reduces towards the end of FLASH2 subtask. This presents an opportunity to reduce the delays between successive REACT batches if the constraints are not violated. The consumption profile predicted by the dynamic model was approximated by the stepwise function given in Table 2.

A simulation run was made based on the new STEAM consumption profile and delay of 16.7 h between reactor batches. The Gantt chart for this case is shown in Figure 16.

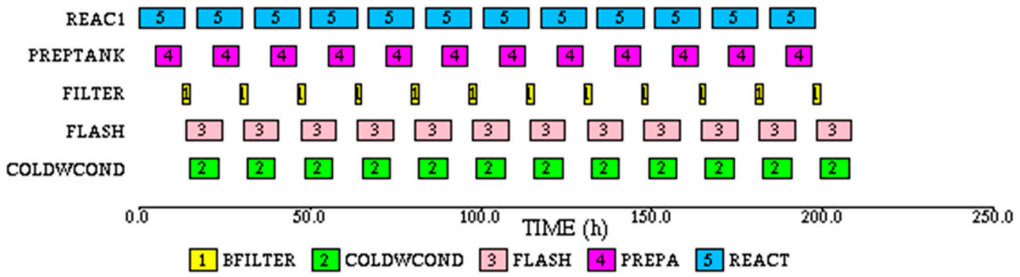



**Figure 16.** Gantt chart for the case with modified utility usage.

The STEAM consumption profile for the base case is shown in Figure 17a, and current case is shown in Figure 17b.

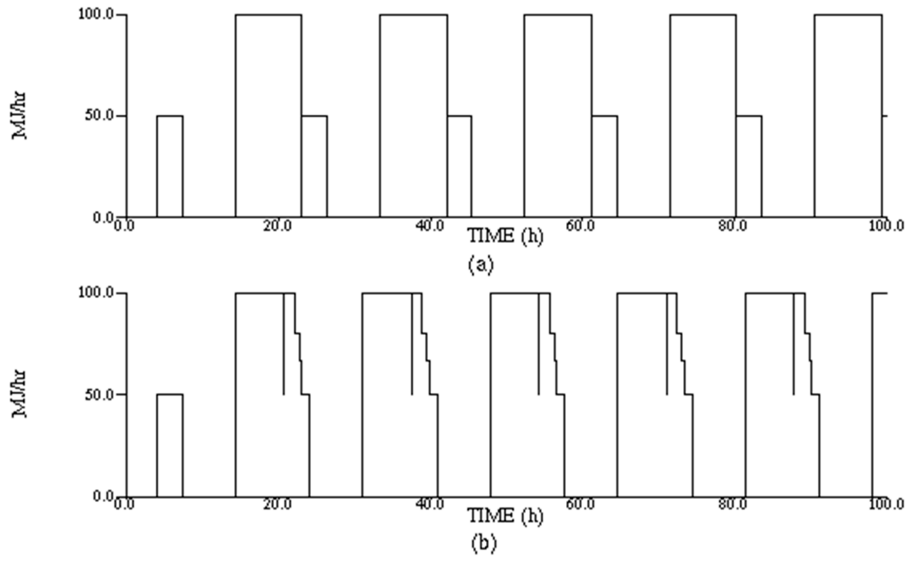



**Figure 17.** Steam consumption profiles for two cases: (**a**) the base case, (**b**) the current case.

The effective cycle time for this case is reduced to 16.7 h. Consequently, 9 reactor batches can be completely processed in 168 h, which is also the maximum possible throughput of the process. Thus, a more accurate modeling of the utility consumption reduces the effective cycle time or increases the throughput by 12.4%.

**Table 2.** Consumption profile for STEAM in FLASH2 subtask.

| Task | Subtask | Duration (h) | Material | Utilities |
|---|---|---|---|---|
| FLASH | FLASH2 | 8.2 | | STEAM, 100 MJ/h, 5.9 h<br>STEAM, 50 MJ/h, 1.3 h<br>STEAM, 30 MJ/h, 0.7 h<br>STEAM, 17 MJ/h, 0.3 h |

## 6. Use of Simulation in Rescheduling

A recipe network in a BATCHES simulation model very accurately represents the corresponding manufacturing recipe in an automation system used for running the process. Therefore, the results of a simulation run provide accurate and detailed information about the projected process milestones. For example, a subtask level Gantt chart, like the one shown in Figure 14, can be generated for each unit, defining the start and end of each subtask in that unit. A subtask Gantt chart provides a template that could be used for tracking the underlying process in real-time. Similarly, if detailed process dynamics models are used for any subtask, the predicted process variable trajectories also could be used for tracking the process performance. More importantly, checkpoints can be set up based on elapsed time or process state for computing process deviations. At a selected checkpoint, if the difference between the projected value and the actual value of the specified variables is too large, a new schedule could be generated.

The main causes of deviations between a simulated and a real process are process variability and inaccuracies in the process dynamics models. Process variabilities occur due to the randomness in the underlying phenomena, and are typically beyond control or inherent to the process. The inaccuracies in dynamic models arise due to the assumptions made during model formulations. When using simulation as an off-line tool for decision support, multiple simulation runs are made (replicates) with randomized parameters and average values for performance measures are computed. When using a simulation model as a process surrogate in real-time applications, such as rescheduling, fixed values for parameters are used for predicting the process trajectory, with the initial state of the model at each rescheduling point set to match the process state at that time.

The steps in using simulation for rescheduling are shown in Figure 18. At the beginning, the state of each unit is set to match the process state at t = 0. Typically at the beginning each unit is idle and empty.

Most of the automation systems have the functionality to generate a snapshot of the process at any given time. For each unit, the following information can be extracted from the process historian: current subtask, time at which the current subtask started, amount and composition of the material in the unit, temperature and pressure, names of units connected upstream and downstream at current time.

A process snapshot, which can be saved as a text file, is used to set up the initial state of all units in the model and the simulation is halted at a pre-determined end time (time increment b0), for example, 24 h after the start time. The results from the simulation run are made available in the desired format for tracking purposes. As the process marches in time, the status at any time t can be extracted and compared with that projected by the simulation. If the deviation is greater than the specified criteria, then the initial status of the model is set to the process snapshot at that time, and a new simulation run is made. The results of the new simulation are then used for tracking the process from that time forward.

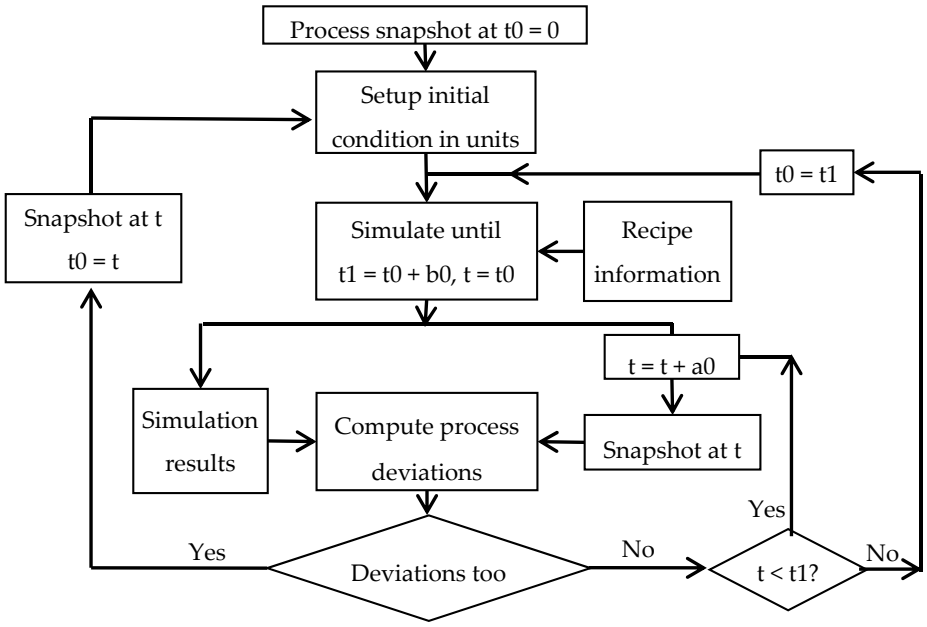



**Figure 18.** Steps in simulation based rescheduling.

*Rescheduling of the Specialty Chemical Process*

Simulation based rescheduling is illustrated with the specialty chemical process described earlier. Also, the recipe was modified to include more operational level details that make the model more realistic, and the need for rigorous methodology even more evident.

The HEAT subtasks of the REACT and FLASH tasks require more time for each batch because of fouling. The increases are 0.2 h and 0.1 h, respectively. After cleaning the associated units for 2.0 h after every third batch, the durations for these subtasks are equal to the nominal durations in Table 1. Also, the duration of the REACT1 subtask has variability, and is uniformly distributed between 4 and 6 h.

The variability in the REACT1 subtask and the drift in the HEAT subtask of the REACT task affect the start times of the FILTER and FLASH tasks. Similarly, the drift in the HEAT subtask of the FLASH task further affects the remaining subtasks of the task. Due to the resource constraint, there is a cascading effect on the following REACT batch. Therefore, a schedule generated by assuming fixed cycle time will not be very reliable and there is a need to develop a rescheduling strategy using the most up to date process status.

Based on the knowledge of the process recipe, the end of the REACT1 subtask of the REACT task is used at a check point for rescheduling the process. From practical standpoint, the AGE, EMPTY and FILTER subtasks together provide a time window for making a simulation run, and generating and disseminating the new schedule to the operations personnel.

The rescheduling strategy is as follows:

(1) Start the process at time 0.
(2) Take a process snapshot when the REACT1 subtask of the REACT task ends and note the current time.
(3) In the simulation model, initialize the units to match the snapshot, set the simulation start time to the current time, run the simulation until the REAC1 subtask of the next REACT batch ends.
(4) Trigger the REACT task 1.9 h after the start of FLASH2 subtask of the FLASH task so that the end the first segment of Steam use matches the beginning of HEAT.
(5) Generate the necessary reports for the operations personnel.
(6) Let the process run its course to the end of REACT1 subtask of the next REACT batch.
(7) Go to step 2.

Step 4 given above illustrates the modeling of coordination control in simulation, which is triggering of a task through the use of signals.

The recipe was modified to model the new features. The cleaning operations are modeled as separate tasks, RCLEAN for cleaning the REACT and FLCLEAN for cleaning FLASH, each with one subtask with the duration of 2.0 h. The RCLEAN task is triggered after every 3rd batch on REACT, and FLCLEAN is triggered after every 3rd batch on FLASH. The triggering was implemented through a special purpose FORTRAN subroutine. The modified recipe network is shown in Appendix B. Note that in the simulation used for predicting the process behavior, the duration of the REACT1 subtask is 5.0 h. Therefore, the end of REAC1 predicted by simulation will not match exactly with the end of REACT1 in the process because of the variability. The effect of variability is minimized by adjusting the start time for the simulation run for every cycle and regenerating the schedule one cycle at a time.

The composite Gantt chart for the actual process, simulated by sampling duration of the REACT1 subtask, is shown in Figure 19.

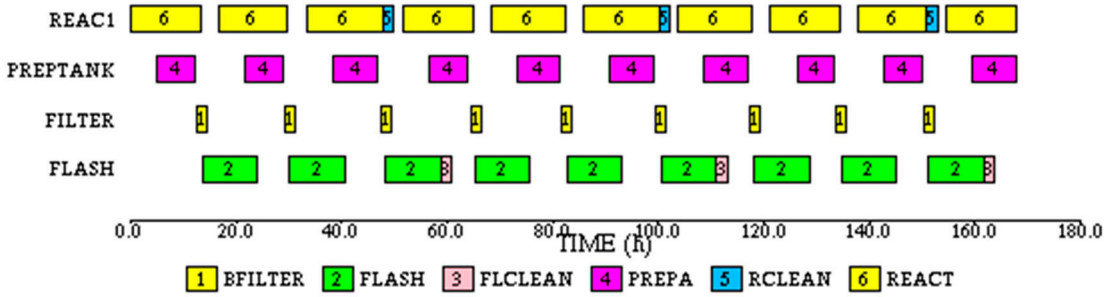



**Figure 19.** Composite Gantt chart for the specialty chemical process.

The subtask Gantt chart for REAC1 is shown in Figure 20. Note that the durations of the first three instances of HEAT subtask in REAC1 and FLASH tasks are different because of the drift, and durations of all instances of REACT1 subtask are different because of variability.

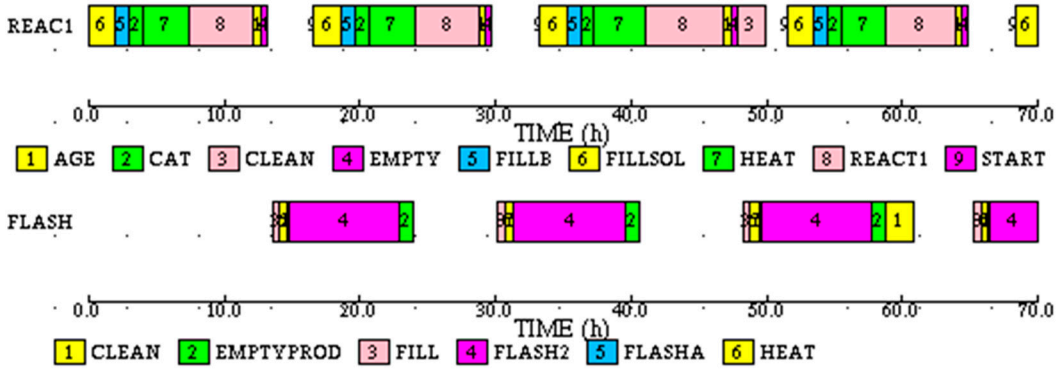



**Figure 20.** Subtask Gantt chart for REAC1 and FLASH for the specialty chemical process.

For all the simulation runs presented in this paper the execution times were very small, of the order of 100 milliseconds.

## 7. Conclusions

A recipe network in a BATCHES simulation model can accurately represent the complex recipes and operating rules typically encountered in batch process manufacturing. Based on the modeling of flow and execution controls, a recipe network can be divided into segments that are decoupled from operational standpoint. Each segment can be scheduled independently, assuming no net accumulation

in each segment. By using the advanced capabilities of the simulator for making assignment decisions, very reliable and verifiable schedules can be generated for the underlying process. For a recipe consisting of one segment, the best schedule was generated by manipulating one decision variable, namely, the time interval between successive batches of the independent task in the recipe. This methodology has the advantage of keeping the number of decision variables very small, which will facilitate its use in a two level optimization framework.

A rescheduling methodology for making day-to-day scheduling decisions using the simulator was presented for a recipe consisting of one segment. It generates schedules for shorter time horizons and is applied successively. It compensates for the variability in the process through the use of coordination controls incorporated in the recipe model. Such a methodology is feasible because the model can be initialized using a process snapshot generated by a DCS system, and the execution times for making simulation runs are very small.

**Conflicts of Interest:** The author is associated with Batch Process Technologies, Inc., which develops and licenses the BATCHES simulator. The simulator was used in modeling the processes studied in this paper.

## Appendix A

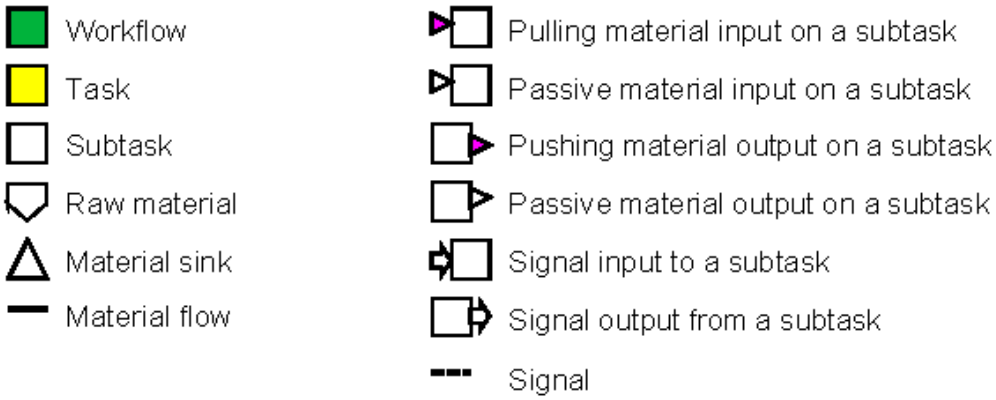



**Figure A1.** Recipe network Symbols.

## Appendix B

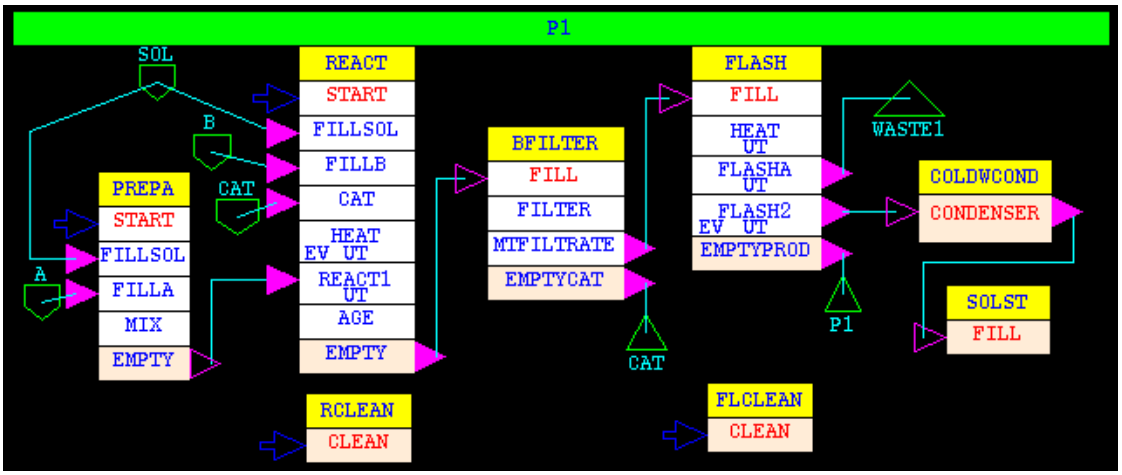



**Figure A2.** Recipe network for the modified recipe.

## References


1. Sundaramoorthy, A.; Maravelias, C.T. A General Framework for Process Scheduling. *AIChE J.* **2011**, *57*, 695–710. [CrossRef]
2. Joglekar, G.S. Incorporating Enhanced Decision-Making Capabilities into a Hybrid Simulator for Scheduling of Batch Processes. *Processes* **2016**, *4*, 30. [CrossRef]
3. Chu, Y.; You, F.; Wassick, J.M.; Agarwal, A. Integrated planning and scheduling under production uncertainties: Bi-level model formulation and hybrid solution method. *Comput. Chem. Eng.* **2015**, *72*, 255–272. [CrossRef]
4. Petrides, D.; Carmichael, D.; Siletti, C.; Koulouris, A. Biopharmaceutical Process Optimization with Simulation and Scheduling Tools. *Bioengineering* **2014**, *1*, 154–187. [CrossRef] [PubMed]
5. Nie, Y.; Biegler, L.T.; Wassick, J.M. Extended Discrete-Time Resource Task Network Formulation for the Reactive Scheduling of a Mixed Batch/Continuous Process. *Ind. Eng. Chem. Res.* **2014**, *53*, 17112–17123. [CrossRef]
6. Gupta, D.; Maravelias, C.T.; Wassick, J.M. From rescheduling to online scheduling. *Chem. Eng. Res. Des.* **2016**, *116*, 83–97. [CrossRef]
7. Batch Process Developer. Available online: www.aspentech.com/products/engineering/aspen-batch-process-developer (accessed on 15 August 2017).
8. Intelligen, Inc. Available online: www.intelligen.com (accessed on 15 August 2017).
9. gPROMS. Available online: www.psenterprise.com/gproms.html (accessed on 15 August 2017).
10. BATCHES. Available online: www.bptechs.com (accessed on 15 August 2017).
11. DynoChem. Available online: www.scale-up.com (accessed on 15 August 2017).
12. ANSI/ISA–88.00.01-2010. *Batch Control Part 1: Models and Terminology*; Instrument Society of America: Pittsburgh, PA, USA, 2010.
13. De Minicis, M.; Giordano, F.; Poli, F.; Schiraldi, M.M. Recipe Development Process Re-Design with ANSI/ISA-88 Batch Control Standard in the Pharmaceutical Industry. *Int. J. Eng. Bus. Manag.* **2014**, *6*, 16–27. [CrossRef]
14. ISAPublications InTech Magazine. Available online: https://www.isa.org/standards-and-publications/isa-publications/intech-magazine/2012/june/automation-basics-organizing-batch-process-control (accessed on 15 August 2017).
15. Watson, D.R.; Eli Lilly and Company, Indianapolis, IN, USA. Personal commumication, 2017.
16. BATCHES. *Users Manual*; Batch Process Technologies: West Lafayette, IN, USA, 2015.
17. Joglekar, G.S.; Reklaitis, G.V. A Simulator for Batch and Semi-continuous Processes. *Comput. Chem. Eng.* **1987**, *8*, 315–327. [CrossRef]
18. Joglekar, G.S.; Giridhar, A.; Reklaitis, G.V. A Workflow Modeling System for capturing data provenance. *Comput. Chem. Eng.* **2014**, *67*, 148–158. [CrossRef]