Scheduling of Batch Plants: Constraint-Based Approach and Performance Investigation

Wei Huang [a] and Bo Chen [b*]

[a] Department of Computing and Information Systems, University of Luton, Park Square, Luton, LU1 3JU, UK

[b] Warwick Business School, University of Warwick, Coventry, CV4 7AL, UK

Abstract:

Batch processing plants are attractive due to their suitability for the manufacturing of small-volume, high-value added products. Scheduling batch plants by using computer-aided systems is important for improving the plant productivity, since it harmonizes the entire plant operation efficiently to achieve production goals. However, the current scheduling approaches for batch plants are inadequate. This research develops a constraint-based model and system for batch-process scheduling and investigates their performance. The proposed constraint model analyzes and brings together many scheduling constraints, adds new constraints and categorizes them according to their functionality. A computer scheduling system, BPS, is developed in C++ to apply the model. A number of examples have been devised to study the performance of our constraint-based approach. It is found that the approach can schedule complex plants and solve large-size problems by finding feasible solutions satisfying all imposed constraints, which include some hard ones such as those of finite wait time. It is also identified that the first feasible solution can be found very quickly, but much more time, even exponentially more, is required to find the optimal solution particularly for complex and large-size problems. Feasibility and limitations of the proposed methodology are demonstrated by the results.

Keywords:
Scheduling; Constraint Satisfaction Techniques (CST); Batch Plants; Finite Wait; Storage Policy;

*Corresponding author. T: +44-24-76524755; F: +44-24-76524539; E: B.Chen@warwick.ac.uk

Scheduling of Batch Plants: Constraint-Based Approach and Performance Investigation

1. Introduction

Batch processes are the production processes in which products are produced in batches rather than in a continuous or discrete mode. Batch processing plants are attractive due to their suitability for the production of small-volume, high-value added products, which are becoming increasingly important with fast market changes. Batch manufacturing is typically used in the pharmaceutical, polymer, food and speciality chemical industries, because it provides the necessary flexibility to accommodate various production requirements using the same processing facility. In general, batch processes are characterized by (Liu, 1996):

- Manufacturing processes involving a set of operations that are executed independently and in batches.
- Sharing of resources (such as operator, steam, electricity or auxiliary equipment).
- Presence of intermediate storage to separate operations and mitigate the effects of process variations or upset.
- Multi-purpose equipment (e.g., a piece of equipment may be used for either processing or as a storage unit).
- Flexibility in configuration (since the equipment can often be connected in different ways).
- Decision dependent set-up costs are involved (e.g., cleaning of equipment before production of another scheduled product or batch).
- High quality specifications.

Batch processing plants often require multiple operations, such as mixing, blending and separating etc. in the case of chemical plants. The multi-stage nature of a processing network, comprised of a number of units in series, allows several different storage policies (Datta et al., 2001; Yu et al., 1998; Kim et al., 1996; Grau et al., 1996; Reklaitis, 1982). Storage units can hold intermediate materials so as to reduce idle time by freeing processing units to process other batch materials and thus increase utilization of the equipment.

Since plant flexibility is so important for improving productivity, pipeless batch plants have been developed and built to enhance production flexibility. In such plants, material processing takes place at a number of fixed processing stations, and materials are transferred from one processing stage to another in moveable vessels (MVs). The same vessel is normally used to transfer materials as well as to hold the materials being processed at each station (Realff et al., 1996). Niwa (1993) draws an analogy between a pipeless batch plant and a chemical laboratory. In the laboratory, a beaker or flask is a "mobile vessel", and the laboratory's stationary equipment consists of a number of "processing stations", such as weighing balances, mixers and Bunsen burners. To synthesize a product, the chemist generally uses a single flask, moving it to the appropriate processing station to carry out a specific operation: weighing, formulating, mixing or reacting. Pipeless batch plants have been built and used to produce a number of products such as lubricant oils, paints and inks (Niwa, 1993). Without the maze of a pipe network, pipeless batch plants permit a wide range of products to be handled with frequent changeover to meet market demands and seize opportunities efficiently. The concept has been successfully demonstrated with a number of production plants operating mainly in Japan (Niwa, 1993).

The manufacture of a diverse range of products in varying quantities poses a big challenge to process industries. The resulting competitive pressure calls for better use of existing facilities and provides an incentive for the application of computer-aided scheduling systems (Egli and Rippin, 1986). In batch plants, detailed requirements for the various products may be specified on a day-to-day basis. A production schedule must specify the sequence and manner in which the products are to be produced and specify the times at which the process operations are to be carried out. There is no doubt that the overall productivities and economic effectiveness of batch plants depend critically on the production schedule as it harmonizes the entire plant operation to achieve production goals.

While flexibility of batch plants improves productivity, it also makes plant scheduling a challenging task. In the past, the plant scheduler frequently communicated face-to-face with co-workers to gather the information necessary to perform a task. Producing a schedule manually is time consuming and cannot meet the requirement of fast market changes. As a result, manual approach has been replaced in many batch plants by computer-aided scheduling and operation.

2. Batch Process Scheduling Problem

Batch manufacturing process is a challenging domain for computer-aided scheduling systems. On the one hand, unlike other types of manufacturing, batch manufacturing process is neither continuous (there is no steady inflow of raw materials resulting in a steady product outflow) nor discrete (there is no manufacture or assembly of individual items). On the other hand, the plant environment is dynamic, e.g., equipment breaks down and new orders come in, and shared resources are required, which result in models of complicated tasks and resources (Goldman and Boddy, 1997).

In a broad sense, manufacturing processes, such as those of chemicals, food, pharmaceutical, polymer, mechanical parts, electrical products, and so on, can be in any of the mentioned three process modes: continuous, batch and discrete. In a narrow sense, manufacturing plants are usually referred to as plants for producing and/or assembling individual items such as mechanical and electrical products etc. in which discrete processes are mainly applied. These plants belong to discrete part industry (Applequist et al., 1997). Correspondingly, batch processes are mainly applied in plants where specialty chemicals, food, pharmaceutical and polymer are produced.

Existing models for scheduling problems of manufacturing plants, notably the job-shop model, are mainly developed for discrete processes. As described, e.g., by Chen et al. (1998), a classical job shop consists of a number of different machines and a number of jobs to be scheduled. A job is a sequence of operations (to be processed in that order), each of which needs to be processed on a specified machine for a specified period of time. One of the objectives mostly used in job-shop scheduling study is to minimize the makespan, which represents the total production time required.

Although the job-shop model can capture some basic characteristics of batch processes, it is too simple, even with some recent extensions (Chen et al., 1998), to describe batch processes accurately mainly in the following aspects (Goldman and Boddy, 1997):

- Single-use machines. Producing a product consists of a number of sequential steps, each of which requires some machines. In the job-shop model, schedulers are not free to choose machines, while they usually need to do so in many batch plants.

- Single operation sequence. A job is a simple sequence of operations in the job-shop model. However, in many batch processing recipes there are multiple and simultaneous activities. Precedence constraints also exist among operations of different batch activities in a batch plant.
- Simple resource handling. Job-shop model does not include the more complex resource types that appear in batch manufacturing processes, such as intermediate storage with finite capacity.

Besides, due to the extreme scheduling difficulty, set-up costs dependent on job sequence are rarely addressed in the literature, although problems of this kind do appear in batch process. For example, if a food-production plant needs to produce dark (such as chocolate) and light (such as vanilla) products, then the vessels would need to be cleaned between production of the batches and such cleaning would be much easier if the batch of light products is processed before the batch of dark products.

Therefore, a batch model needs to be more flexible and expressive. A typical batch process scheduling problem is characterized by the following specifications additional to the classical job-shop model (Datta et al., 2001; Reklaitis, 1982):

- Application polices of intermediate storage between processing stages.
- Constraints on the processing order between operations.
- Transfer times between different processing units.
- Multiple activities are allowed to take place simultaneously.
- The structure of processing network.

The quality of a batch schedule, which is critically affected by the structure of the processing network and the intermediate storage, can be measured by one or a combination of the following performance criteria:

- The makespan, which is the total time required to complete all jobs.
- The maximum or mean (over all jobs) flow time, where the flow time of a job is the time from its release to its completion.
- The maximum or mean (over all jobs) tardiness, where the tardiness of a job is the delay between its completion and its due time.

- The total changeover or set-up cost, where a set-up is incurred as a result of switching between operations of jobs.

The makespan, mean flow time and maximum tardiness are the most studied criteria in the literature, while the makespan and total set-up cost are the two most commonly used criteria in industry (Ku et al, 1987).

The multi-stage nature of a batch processing network allows several different storage and waiting options (Datta et al., 2001; Yu et al., 1998; Kim et al., 1996; Grau et al., 1996; Reklaitis, 1982):

- Unlimited intermediate storage (UIS)
- Finite intermediate storage (FIS)
- No intermediate storage (NIS)
- Unlimited wait (UW)
- Finite wait (FW)
- Zero wait (ZW)

In both the NIS and ZW modes, there is no storage between stages, while such storage is provided in UIS and FIS modes and may be provided in UW and FW modes. Intermediate materials can wait in storage or the current processing unit as long as necessary in UW mode, but such waiting time is limited in FW mode. In fact, NIS is a special case of FIS and ZW is a special case of FW. After materials are processed in a unit, they may be held in the unit temporarily under the UIS, FIS, UW, FW or NIS modes, but they must be transferred to the downstream unit immediately in the ZW mode. Some batch plants have a mixed intermediate storage (MIS) policy with a combination of the FIS, NIS and ZW modes. By holding intermediate materials to free processing units, storage units can reduce process idle time and thus increase utilization of the equipment. As the number of storage units increases, the idle time of processing units can be reduced greatly. For example, under UIS operation, unlimited storage units are available between any two processing units and the processing units can then be used efficiently to process as many materials as possible without worrying about where the intermediate materials can be stored. Therefore, with the same processing sequence, the UIS mode will result in minimum makespan (Ku et al., 1987).

In chemical batch plants, it is usual to have limited storage capability between stages and hence FIS policy is applied. The ZW or FW mode is used where unstable intermediate materials must be processed immediately or within a short time after the previous step has been completed. FW mode is not only necessary to limit the wait time of intermediate materials in a storage unit, but also necessary to limit the wait time in a processing unit when no storage is available but the created intermediate materials are unstable. Grau et al. (1996) proposed a simple approach to let intermediate materials wait in the processing unit temporarily for a limited time, however, it cannot be extended to the wait time of materials in storage. In addition, FW is also necessary when an activity needs to be completed within a fixed period of time after its previous activity. Compared with the UIS, FIS, NIS, ZW or MIS policy, FW mode is a neglected area in research literature.

3. Literature Review

During the last two decades, there has been considerable interest in the scheduling of batch processing plants, particularly chemical batch plants, for improving productivity and a large amount of research literature exists. A number of better known scheduling techniques, such as mixed integer linear programming (MILP) and constraint satisfaction techniques (CST), have been studied in the literature. While MILP is the most popular technique so far, CST is relatively more recent and promising. This section reviews these techniques and the scheduling applications of batch plants.

3.1 Mixed Integer Linear Programming (MILP)

In the MILP approach, a scheduling problem is formulated as a mixed integer linear program and solved by available optimization packages. In this approach, the scheduling performance criterion is formulated as a linear objective function and various scheduling constraints as linear equalities and/or inequalities. Based on a serial processing structure with no mixing or splitting of batches allowed, Rich and Prokopakis (1986) proposed an MILP formulation for the scheduling problems they studied. However, their MILP formulation is difficult to be generalized to other scheduling problems. Ku and Karimi (1988) used MILP for scheduling multi-product plants with intermediate storage. Tsirukis and Reklaitis (1991) proposed MILP formulation with resource constraints. In the context of the optimal short-term scheduling of batch operations, Kondili et al. (1993) proposed a general and rigorous algorithm to deal with complex processing networks and accommodate a wide variety of technical constraints. Their mathematical formulations led to a large-scale mixed integer linear program, which, for typical industrial problems, may involve several thousands of binary variables. Techniques that reduce search time for solutions to problems of such scale have been described by Shah et al. (1993). Kim et al. (1996) presented detailed completion time algorithms for various storage policies. Vin and Ierapetritou (2000) proposed their MILP approach to solve rescheduling problems of batch plants, and Pantelides et al. (1995) and Realff et al. (1996) studied scheduling problems of pipeless batch plants by modifying previous MILP approaches.

Since the MILP approach is based on linear relations in its formulation, simplification assumptions are necessary as scheduling problems are often nonlinear in practice. Although MILP models for more realistic systems have been proposed (Bok and Park, 1998; Lee et al., 2001), they require complicated mathematical formulations and hence demand heavy computation time for solutions. A main weakness of the MILP approach is that, as the complexity of a plant increases, the scheduling problem becomes very hard to formulate properly. Many case studies have indicated that MILP approach is suitable only for small-scale problems.

3.2 Constraint Satisfaction Techniques (CST)

Constraint Satisfaction Techniques are methods of expressing and solving constraint satisfaction problems (CSPs). A CSP is represented by constraints and constrained variables. Each variable has an associated domain, which includes a set of its potential values. A solution to a CSP is an assignment of values to the constrained variables of the problem such that all the constraints are satisfied simultaneously. The set of all possible assignments of values to the variables is known as the search space. Therefore, solving a CSP is to search for a value assignment in the space.

CST is widely used recently to solve various problems ranging from resource allocation to scheduling. These techniques do not require elaborate mathematical formulae, but instead require a problem to be stated in terms of constraints. Therefore, an immediate advantage of CST is ease of implementation. Another advantage of CST is that, rather than searching blindly the entire space for a solution, it exploits the constraints themselves to reduce its search effort. Constraints are exploited in a constructive way to deduce other constraints and to detect inconsistencies among possible solutions, which are known as constraint propagation and consistency checking. Unsuitable values in the domains of variables due to the inconsistencies will be removed. A backtrack-free search is possible by using these constructive ways. The following simple example illustrates how CST works. Suppose there are two variables "n" and "m", and their domains are {0, 1, 2, …, 10} and {5, 6, 7}, respectively. Two constraints "n < 10" and "n + m = 15" are imposed. Unsuitable values are removed from the domains when the constraints are set. For example, when "n < 10" is set, "10" is removed from the domain of "n" to become {0, 1, 2, …, 9}. When the constraint "n + m = 15" is set, further removal of unsuitable values will update the two domains to {8, 9} and {6, 7}, respectively. Therefore, the search space has been greatly reduced before the actual search. To search for a solution, a choice point (also known as a "node") is set by assigning 6 or 7 to "m". If 6 is assigned to "m", then constraint propagation will remove the value 8 from the domain of "n" and bind "n" at 9, and a solution is found. After that, the system can backtrack to the solution "m = 7" and "n = 8". If the criterion is to minimize "m" then the latter solution with larger value of "m" will be discarded. From this simple example, it can be seen that finding a solution with CST involves two stages: 1) the preprocessing stage, where techniques such as consistency checking are applied to reduce the search space, and 2) the search stage, where the search space is explored using techniques such as backtracking to find different solutions.

CST is a research field that has demonstrated its usefulness and competitiveness in scheduling (Brailsford et al., 1999; Tsang, 1995). Scheduling problems are typically characterized by constraints, such as due dates, maximum resource availability and operation precedence. Therefore, it is appropriate to treat scheduling problems as CSPs and solve them using CST.

Unlike in the MILP approach, variables in CSPs are not limited to taking on numerical values only. They can also be of enumerative type (e.g. John, Mary, Peter, etc.). There is no limitation on the type of constraints that CST can deal with, while in the MILP, constraints have to be linear inequalities or equations. These advantages make CST very flexible and hence have a wide domain of applications (Brailsford et al., 1999; Tsang, 1995).

CST is also different from local search techniques, such as tabu search, simulated annealing and genetic algorithms. CST does not attempt to find an optimal solution, but mainly seeks for a feasible solution that satisfies all imposed constraints simultaneously (Lee, 1997; Tsang, 1995). However, CST can be adapted to find an optimal solution (Brailsford et al., 1999). For example, after a feasible solution is found, a new constraint can be introduced, which specifies that the objective value of the next solution must be better than that of the current solution. This is done repeatedly until no better solution can be found, implying the last solution is optimal. However, finding an optimal solution with CST is often time consuming and it is not the nature of CST. Kelleher and Cavichiollo (2001) indicated that a CSP-based approach might be able to find "better" solutions but it is rarely capable of finding an optimal solution (in reasonable amount of time). This is the weakness of CST.

Compared with local search techniques, which are repairing in nature by iteratively improving a complete solution, CST is constructive and attempts to incrementally extend partial solutions to a complete (feasible) solution. At the beginning of solution search, an attempt is made to satisfy more stringent constraints, leaving less stringent constraints for the final part of the search process. Many techniques have been developed for managing the constraints and speeding up the search (Lee, 1997; Tsang, 1995). Among CST approaches, tree search techniques combined with backtracking and consistency checking are widely used for solving CSPs, while constraint propagation is mainly used to restrict the size of the search tree (Brailsford et al., 1999).

Brailsford et al. (1999) compared CST with some well-known operational research techniques such as branch and bound (B&B) and simulated annealing (SA). It is found that CST compares favourably with these techniques in terms of ease of implementation and the flexibility of adding new constraints. Therefore, CST is very suitable for solving highly constrained problems. On the other hand, performance of CST with respect to solution quality and computation time tends to be problem dependent. Since CST relies mainly on constraint propagation to restrict the size of the search space, while B&B is highly dependent on its bounding scheme, it is likely that CST would be more efficient if the bounds in the B&B approach require a significant amount of computation time and they are not strong enough to allow very much pruning of the search space. CST is unlikely to be competitive with the best local search methods to find the optimal solutions if CST is used in its pure form. However, if ideas from local search methods, such as randomization and restarting, are incorporated, then CST could become a serious competitor. Although CST is still relatively in its infancy, whereas B&B and SA are well developed and sophisticated, there are many problems for which CST is more competitive.

3.3 Scheduling Applications of Batch Processes

Research on scheduling of batch processes has become active since the early 1980s with the development of two approaches: exact algorithms such as MILP and heuristics such as local search and CST.

Egli and Rippin (1986) developed a computer program SRSBP for the short-term scheduling of multi-product batch chemical plants. Given available equipment, delivery date and equipment requirements of each product, batches are sequenced to meet the delivery dates while taking into account of penalties associated with product changeovers and storage costs. Ku and Karimi (1988) used MILP to schedule multi-product batch plants of no or finite intermediate storage with the objective of minimizing the makespan.

Ku and Karimi (1991) investigated the usefulness of simulated annealing (SA) for batch process scheduling problems. They developed an SA methodology for minimizing the total time to produce a set of batches in the serial flow-shop with unlimited storage. Their algorithm is simple and appears to be applicable to different types of scheduling problems. The drawback of their approach is the requirement of large amount of computational effort.

Kondili et al. (1993) presented a general MILP formulation for a wide range of scheduling problems arising from multi-purpose batch chemical plants. They proposed a state-task network (STN) representation of a chemical process. The novel feature of this representation is that both the individual batch operations (“task”) and the feedstock, intermediate and final products (“states”) are included explicitly as network nodes. The proposed MILP formulation is very general. However, there is a main problem of the size of the resulting MILP problem. Solution of even a small example, using a state-of-the-art generic MILP solver, was found to require a substantial amount of computation time. Furthermore, the computational cost increased rapidly with problem size. For realistic scheduling problems, their MILP model may involve several thousands of binary variables, which by far exceed the solution capabilities of general-purpose methods that are currently available.

Goldman and Boddy (1997) developed a constraint-based scheduler, which they called Honeywell Batch Scheduler. This scheduler was based on a scheduling technique called constraint envelope scheduling, in which constraints are added automatically as needed, e.g., to resolve resource conflicts. In several aspects the scheduler can be improved. Constraints such as the finite wait policy were not considered. The current version of the scheduler was created with Common Lisp as a prototype environment. The authors indicated that it was difficult to develop a reasonably sized runtime system because Lisp images were very large. Moreover, conventional software organizations cannot support and maintain Lisp code, which prevents the scheduler from running on a variety of platforms.

Das et al. (1998) investigated a simple but typical batch production scheduling problem and found it was possible to develop constraint-based scheduling solutions within very modest computation time. Huang and Chung (1999) developed a constraint model based on CST principles to represent a common class of batch plant scheduling problems. Based on this model, they produced a simple scheduling system. Das et al. (2000) compared the above approach by Huang and Chung (1999) with established mathematical programming approaches and concluded that it was relatively easier to represent complicated batch plant scheduling problems by using the constraint-based approach.

Shaw et al. (1999) presented a scheduling system based on a multi-objective genetic algorithm (MOGA). The MOGA-based system allowed human interaction with the optimization process, which included the ability to change priorities of the multiple objectives and to change the plant data. Although the authors claimed that their system was promising to provide scheduling solutions to highly complex problems, the example they used was rather small, in which only reaction operations with three reactors were considered.

Brucker and Hurink (2000) used a two-phase tabu search method to solve a chemical batch scheduling problem. In their approach, a “rough” time-lag model was used in the first phase to improve an initial solution and then a general shop scheduling model was used in the second phase to allow tabu search to build up a good solution in a reasonable time. The production constraints imposed on the problem were relatively simple, which only included the precedence relations between the processing tasks, the total number of products to be produced and the deadline. Other constraints such as chemical material allocations, intermediate storage policies and safety constraints etc., which were usually important to a chemical plant, were not taken into account.

Rodrigues et al. (2000) studied a short-term scheduling problem in multi-purpose batch plants, where the main objective was adherence to due dates. Their solution approach with simulated annealing consisted of two phases: planning and scheduling. As explained by the authors, simulated annealing generated a large number of infeasible solutions in heavily constrained situations and hence the planning phase was essential to filter out these infeasible solutions in order that the scheduling phase could focus on generating feasible solutions.

Although many aspects of process plants have now been considered, there is still much room for further work and more complex constraints, such as those on finite wait and finite storage, should be addressed properly.

4. Constraint Model Development

4.1 General Constraint Model

In this subsection we give an overview and categorize the constraint-based scheduling model for batch processes, which includes those constraints presented fragmentally in several previous papers (Huang and Chung, 1999; Huang and Chung, 2000). Our key extension to include shared storage and finite wait policies is described in Section 4.2. First we explain some basic terminology that will be used later.

- Production cycle
  A production cycle refers to a complete set of related operations required to produce a batch of final products. One or more cycles may be needed to produce the required quantities of a product.
- Production activity
  A production activity is an operation that transforms materials from their input state into their output state. A cycle may consist of one or more production activities.
- Run

A production activity may run once or more times, dependent on the availability of feed materials from the previous activity of the same production cycle and on the capacity of the processing unit used for the particular activity.

- Operation (J_i)

  Each run of a production activity in a cycle is called an operation. The total number of operations is the total number of runs of all production activities in all cycles within the whole production process under consideration.

- Batch activity (B_r)

  A batch activity is defined to represent a production cycle. In pipeless plants, a moveable vessel is selected by a batch activity to go through a number of processing stations to produce products. The duration of a batch activity covers the processing time of all the related operations and the time a vessel needs to move from one station to another. If several production cycles are required to meet the quantity demand for a specific product, several batch activities are needed to represent them. The concept of a batch activity is necessary to prevent a vessel from being used by more than one batch activity simultaneously, i.e., once a vessel is selected to produce a batch of products, then it cannot be used by another batch activity until the previous one has finished.

- Moveable vessel (V_x)

  In pipeless batch plants, a suitable vessel is selected by a batch activity to go through a number of processing stations to produce a batch of products.

- Processing station (S_p)

  A processing station is a unit where the material in a vessel is converted from one state into another. Stations are used in pipeless batch plants and their corresponding equipments in other batch plants are called machines.

- Material (M_u)

  Materials are handled in the production processes and will be converted into products eventually.

Let a schedule have n operations, J_1, …, J_n, where J_1 and J_n are respectively the first and last operation of the whole production process. A complete set of notational definitions used in the following model is presented in the appendix.

<u>Constraints on Moveable Vessel Allocation</u>

To produce a batch of products, a batch activity requires a suitable vessel:

$$B_r \leftarrow V_x \tag{1}$$

Here "←" means "requires". This formula implies a suitable vessel will be selected by a batch activity to produce a batch of products.

After a vessel is allocated to a batch activity, the vessel will be kept for that batch activity from the start time of the first operation to the end time of the last operation plus the time for the vessel to move from the last station (e.g., cleaning station) to the start point to produce another batch of products. This period is the duration of a batch activity:

$$ET(B_r) - ST(B_r) = ET(J_{r,\text{last}}) - ST(J_{r,\text{first}}) + T_r \tag{2}$$

Because a moveable vessel is a unary resource, any two batch activities requiring the same moveable vessel cannot overlap, i.e., the end time of a batch activity must precede the start time of another:

$$\text{If } B_r \leftarrow V_x \text{ and } B_s \leftarrow V_x \text{, then either } ET(B_r) \leq ST(B_s) \text{ or } ET(B_s) \leq ST(B_r) \tag{3}$$

Constraints on Processing Station (Machine) Allocation

A suitable processing station or machine is required by an operation to convert materials from the input state into the output state:

$$J_i \leftarrow S_p \tag{4}$$

Since a processing station (machine) is a unary resource, any two operations requiring the same processing station (machine) cannot overlap, i.e., the end time of an operation must precede the start time of another:

$$\text{If } J_i \leftarrow S_p \text{ and } J_j \leftarrow S_p\text{, then either } ET(J_i) \leq ST(J_j) \text{ or } ET(J_\mathrm{j}) \leq ST(J_\mathrm{i}) \tag{5}$$

Under some circumstances such as maintenance, a processing station (machine) may not be available within a time window from T_x to T_y. If it is required by an operation, the operation must end before T_x, or it must start after T_y:

If $J_i \leftarrow S_p$ and S_p is not available during $[T_x, T_y]$, then $ET(J_\mathrm{i}) \leq T_x$ or $ST(J_\mathrm{i}) \geq T_y$ (6)

<u>Constraints on Material Allocation</u>

Besides a processing station (machine), an operation also requires materials and converts them from the input state to the output state. The type and amount of material required by an operation depend on a specific production demand.

$$J_i \leftarrow M_u \tag{7}$$

A material is a discrete resource with certain quantity. If the available quantity is limited, two and more operations requiring the same material at time point T_u may overlap in time as long as the total requirement for the material does not exceed the limited quantity:

$$\sum_i Q_u(J_\mathrm{i}) \leq Q(M_\mathrm{u}) \tag{8}$$

where the summation over i is taken so that $ST(J_i) \leq T_u \leq ET(J_i)$

<u>Precedence Constraints</u>

Under certain requirement, an operation needs to precede another operation:

$$ET(J_i) \leq ST(J_j) \tag{9}$$

To meet the quantity demand, two or more batch activities may be needed for the same product. Since the operations comprising each batch activity for the product are exactly the same, a precedence constraint is set to let an operation within a batch activity start after the start of the corresponding operation within the previous batch activity:

$$ST(J_{B_r}) \leq ST(J_{B_{r+1}}) \quad (10)$$

Although this constraint does not affect the final solutions, it narrows the search space by eliminating needless possibilities (choice points).

The start time of a batch activity must be equal to the start time of the first operation within the batch activity:

$$ST(B_r) = ST(J_{r,\mathrm{first}}) \quad (11)$$

Time-Bound Constraints

The start time of the first operation in the whole production process is the time origin and the end time of any operation cannot exceed the time horizon. Time horizon is a constant determined by the user, which represents the upper time bound for a schedule.

$$ST(J_1) = TO \text{ and } ET(J_i) \leq TH \quad (12)$$

A product may be required to be delivered before a time point T_d. Therefore, any operation within the production cycle that produces this product is constrained to end by time T_d in order to meet this requirement:

$$ET(J_d) \leq T_d \quad (13)$$

Optimal Criterion

The optimality criterion is minimum makespan, by which every batch and production activity is constrained to end:

$$ET(J_i) \leq M_s \quad (14)$$
$$ET(B_r) \leq M_s \quad (15)$$
$$\text{Minimize } M_s \quad (16)$$

4.2 Storage and Finite Wait (FW)

As discussed earlier, intermediate storage can be used to reduce machine idle time by freeing machines for processing other operations. The rules governing the transfer of batches between processing stages are classified into several storage policies: unlimited intermediate storage (UIS), finite intermediate storage (FIS), unlimited wait (UW), finite wait (FW), no intermediate storage (NIS), zero wait or no wait (ZW). In this subsection, we consider constraints of limited capacities of shared storage and finite wait policies. We assume in the storage model that a storage unit is used by a single production schedule at a time and it can only be used to hold one kind of material in a schedule.

Intermediate storage may be assigned to a storing operation if there is still enough space to hold the required amount of material at time point T_c:

If $ST(J_s) \le T_c \le ET(J_s)$, then $C - S_c \ge S(J_s)$ (17)

The duration of the operation requiring intermediate storage to hold the intermediate material may be limited, i.e., the wait time of the intermediate material in the storage may be finite. The following constraint, called the first finite wait policy, is created for the purpose:

$ET(J_s) - ST(J_s) \le T_f$ (18)

If the intermediate material does not have special requirement and can stay in the storage as long as possible until the downstream machine is available, the user can set T_f equal to the whole duration of the schedule.

Intermediate materials may also stay in the processing machine temporarily for some reasons, e.g., no storage is available or storage capacity is insufficient to hold the material at the moment. The wait time in the machine may also be finite. A constraint, called the second finite wait policy, is produced for this purpose:

$ET(J_i) \le ST(J_{i+1}) \le ET(J_i) + T_f$ (19)

The constraint means that the current operation produces the intermediate material, and before the material is taken by the next operation to process, it can wait for a finite time in the machine temporarily. The next operation can start after the end of the current operation but must start within a finite time of the end of the current operation. If the constraint is applied to pipeless plant, it means that the intermediate material can only stay in the moveable vessel for a finite time between two consecutive production activities.

Sometimes it may be necessary for the user to set a constraint to ensure that an operation must end within a time of the end of its previous operation. For example, this is needed if a cleaning operation is required to complete soon after the previous activity (e.g., mixing) and the end time of cleaning is much more important than its start time. The following constraint, called the third finite wait policy, is created for the purpose:

$$ET(J_i) \leq ST(J_{i+1}) \leq ET(J_{i+1}) \leq ET(J_i) + T_f \tag{20}$$

5. Scheduling System Development

We have produced a computer scheduling system, Batch Processing Scheduler (BPS), to apply the model. BPS is a CST-based scheduler developed in C++ using some library classes and functions from ILOG, which is a CST-based commercial software tool.

5.1 Activities and Resources

In the constraint model, variables are declared to represent activities and resources, such as production activities, batch activities, machines (processing stations), moveable vessels, materials and so on. BPS is developed based on object-oriented principles so that the activities and resources are all represented as object classes. Instances of these classes are created to represent the specific resources and activities required for a specific scheduling problem. An OOP (object-oriented programming) class contains the data members (variables) and methods, which manipulate these data. A class may also contain links to other related classes.

All resource classes are children of a base class: *ResourceBase*. The class includes common data members that every resource class will have: name and capacity, and those methods tackling these data. All resource classes, which are children of *ResourceBase*, will inherit these data and methods automatically and will only need to define the specific data and methods related to those resources themselves. The *ResourceBase* class hierarchy is shown in Fig. 1.

The definitions of the Vessel and Material classes are described here as examples. The Vessel class has only one data member and its data type is *IlcUnaryResource*, which is a predefined ILOG class. An instance of this class represents a resource with the value of capacity equal to one in its definition. From the scheduling point of view, the capacity of a vessel is treated as one so that a vessel cannot be shared by more than one activity simultaneously. It will ensure that the time intervals over which two activities require a moveable vessel cannot overlap. The constraint that any two batch activities requiring the same vessel cannot overlap is applied by the definition of the class. A vessel does have a physical capacity that is used to calculate the number of required batches for a product.

The *Material* class has one data member and its data type is *IlcDiscreteResource*, which is also a predefined ILOG class. An instance of the class represents a resource with a discrete capacity. Each activity may require some amount of the material as long as the material used at a particular time point does not exceed the maximum capacity of the discrete resource. According to the definition of the class, when an instance of *Material* is created, its capacity is set. The constraint on material allocation is applied by the definition of the class.

Similar to resource classes, all activity classes are children of the class *ActivityBase*. The class includes the common data members that every activity has: the name of the final product produced by a production cycle that the current activity belongs to, and the sequence number of this cycle. In addition, another data member is used to indicate what kind of activity it is. Its data type is ILOG's predefined class *IlcIntervalActivity*. An instance of *IlcIntervalActivity* represents an activity that executes without interruption from its start time to its end time. By default, it requires a resource from the beginning to the end of its execution. Besides the data member, methods dealing with these data are also defined. The hierarchy diagram for this class is shown in Fig. 2.

The definition of production activity, which is a child of *ActivityBase*, is described here as an example. Besides those data members and member functions it can inherit from *ActivityBase*, it also defines its own data members and methods dealing with these data. When an instance of the class is created, the constructor of the class will be called to initialize the object. The initialization process is shown in Fig. 3, and it creates a production activity, sets constraints to select a suitable machine (station) from those available and sets constraints between the production activity and required materials. It is seen that some constraints proposed in the constraint model are actually set during the initialization of the object.

5.2 Constraint Model Imposed

For a specific scheduling problem, instances of a variety of classes are declared to represent specific resources and activities required, and then constraints among them are set. The creation of resources and activities, and the setting of proposed constraints are mainly carried out by a function, *DefineProblem*, using the input information. A diagram for object classes with attributes shown in Fig. 4 illustrates the relationship among these resources and activities defined in the function.

In this diagram, an object class of activity or resource is represented as a box. The name of the object class is in the first (upper) part of the box and the data members of the class are listed in the second (lower) part of the box. For example, the data member of "Vessel" is "unary resource" and the data member of "Material" is "discrete resource". These data members define the attributes of the corresponding classes. The notation for drawing the diagram can be referred to the book by Rumbaugh et al. (1991), but it is believed that the diagram can be easily understood with the following explanations. "Activity Base" is the parent class of "Production Activity" and "Batch Activity". "Resource Base" is the parent class of "Vessel", "Material" and "Machine". A batch activity consists of one or more production activities. A production activity requires one machine, but it requires one or more materials. In addition, a batch activity requires a moveable vessel.

The most important and difficult task that the function, *DefineProblem*, needs to do is to create every batch and production activity for a specific problem, and to impose corresponding constraints on them. This procedure is achieved by a 4-level iteration routine. The routine is shown in pseudocode in Fig. 5. The conventions used for the pseudocode can be found in Bailey and Lundgaard (1989). The reason to use the 4-level iteration routine is because products are produced in a cyclic mode in batch processing plants. A batch of products is produced by a cycle that consists of a number of related production activities. In order to meet the amount requirement for a product, several cycles may be needed. For a specific scheduling problem, usually several products are required and production processes for them need to be scheduled simultaneously. A production activity may run once or more depending on the capacity of equipment and the amount of processed material. As mentioned, a run of a production activity is called an operation, and any operation is represented by an object of a production activity in BPS. All the operations for a scheduling problem are generated in an ordered sequence – all operations for a product are created first, then all the operations for another product, and then the operations for the third… and so on. For all operations producing a product, those operations in a cycle are created first, and then those operations for another cycle are created … and so on. After each operation is created, the related constraints are imposed on it. For example, an operation requires a suitable machine and requires one or more materials. In addition, each operation is constrained to end by the makespan, which is to be minimized in the model. Once all operations are scheduled, the value of makespan can be obtained. Similarly, batch activities are also created in an ordered sequence and related constraints are imposed on a batch activity after it is created. The whole procedure is shown in Fig. 5. It is necessary to point out that although operations and batch activities are created in an ordered sequence, they are not necessarily performed in the same order. The appropriate processing time of these activities are determined by BPS according to imposed constraints and an activity created later may be performed earlier.

6. Performance Investigation

6.1 A Simple Example and Discussion

BPS can schedule batch plants with intermediate storage. In order to demonstrate the system, the example used in a published paper (Das et al., 2000) is selected and extended to consider storage policies of finite capacity and finite wait time. The whole production process is shown in Fig. 6, in which two ingredients A and B are blended and the resulting product is packed into 1-kg, 2-kg and 3-kg pack sizes as the final products. There are 60 tonne of A and 60 tonne of B available. They are consumed in equal proportion by the blending task to produce the unpacked product (UPP). Two identical blenders are available and each has a capacity of

5 tonne and blending time of 2 hours. The UPP can be stored in a storage unit of 15-tonne capacity while waiting to be packed. Due to the safety concerns, the UPP can only wait in the storage unit for up to 6 hours. The packing tasks are carried out by a single continuous flexible packing line that can produce only one pack-size product at any time and has a capacity of 2500 packs per hour for any pack-size. It is desired to schedule the plant operations over a period of 48 hours, during which 20 tonne 1-kg-pack-size products, 20 tonne 2-kg-pack-size products, and 20 tonne 3-kg-pack-size products should be produced as soon as possible.

In this example, materials are blended and then transferred to the storage unit waiting to be packed into different final products. These batch processes are operated in a cyclic mode in which the same sequence of operations repeats as many times as necessary to fulfil product demand. It is assumed that there is only one intermediate storage unit available and intermediate materials should only be held in the storage unit. The operation of storing a batch of intermediate material in a storage unit is a key operation for the whole process. The storage unit can hold as many batches of intermediate materials as possible unless there is not enough space available to hold another batch. However, the intermediate materials can only wait in the storage unit within a finite time and they have to be transferred to the downstream unit before the time limit.

Although the operations of storing and packing are continuous activities, in the discrete time formulation for the scheduling of batch plants, these continuous tasks are approximated as being batch tasks with their durations equal to a chosen time interval. In this example, a time interval of one hour is used. This means that a packing task is taken to be a batch task of 2500 packs and its duration is one hour. In a similar way, the operation of storing UPP in the storage can be modeled as a task with duration of one hour or a multiple of one hour. This implies that changes to the amount of material in the storage and in the packing line occur on an hourly basis.

In this example, the two blenders, the intermediate storage unit and the flexible packing line together constitute four processing stations of the whole process. There are total 12 batches of products to be produced and the objective is to complete all production processes as soon as possible subject to the time horizon. According to the proposed constraint model, the solution must meet the following constraints:

Time-Bound Constraint

- Time origin is zero and time horizon is 48 hours, i.e. *TO* = 0 and *TH* = 48.
- The delivery time for all products is time 48.

Precedence Constraint

- For producing a batch of products, storing must start immediately after blending, and packing must start immediately after storing.

Machine Allocation Constraint

- Blenders and the packing line are unary resources that can only be used by one activity at any time.

Material Allocation Constraint

- The ingredients A and B are discrete resources, so at any time the total required quantity of these materials cannot exceed the available quantity.

Intermediate Storage Constraint

- The physical capacity of storage is 15 tonne, and one batch of intermediate UPP from a blender is 5 tonne. So, the storage can store at most 3 batches of intermediate UPP at any time.
- Storage wait time is up to 6 hours. The intermediate materials can stay in the storage until a downstream unit is ready to deal with it as long as these materials don't stay in storage for more than 6 hours.
- Intermediate materials cannot stay in other processing units temporally.

The solution statistics of this example are shown in Table 1 and the optimal solution is illustrated in Fig. 7. The results are generated by running BPS on a PC with a P3 500 MHZ processor. The first solution was found in 0.04 seconds and the last solution was found in 0.831 seconds. Four other solutions were obtained between the first and last solutions. CST-based BPS applies a gradual improvement strategy to search for the optimal solution. This means that once a solution is found a new constraint is added to the problem automatically, which constrains the system to search for another solution at least one step better than the previous one according to the optimal criterion unless no further solution can be found. So, in this example after the last solution was found, BPS still used additional time to confirm that there was no better solution. The system was terminated automatically in 23.944 seconds and confirmed that the last solution is the optimal solution. It is found that the first feasible solution was obtained very quickly and other solutions, including the last solution, were also obtained within 1 second. On the contrary, the system spent quite a lot of efforts in confirming that there was no better solution. It indicates that CST-based BPS is very useful for finding feasible solutions that meet all imposed constraints, but it may need a great deal more time to find an optimal solution. The scheduling results show that both time and resources, including the space of storage, are allocated properly. All the imposed constraints, including the storage finite wait constraint as well as the optimization criterion, are met. The makespan of the first solution and that of the optimal solution are 26 hours and 19 hours, respectively. The makespan ratio of the optimal solution to the first solution is 73%, which means that the makespan can be reduced by 27% from the first solution to the optimal solution.

Before we proceed further, let us formally define the following terms:

- The running time: the elapsed CPU time since BPS starts scheduling a problem.
- The running time of the first solution: the running time when the first feasible solution meeting all constraints is found. It is 0.04 seconds in the above example.
- The running time of the last solution: the running time when the last solution is found. It is 0.831 seconds in the above example. But in fact when this solution is found, BPS does not know whether a better solution exists or not and it still continues running to confirm this. This solution is known as the last solution from the final results.
- The running time of the optimal solution: the system termination time when BPS stops running automatically by ensuring that there is no better solution available. It is 23.944 seconds in the above example. The optimal solution is actually the last solution, but their running times are different. The running time of the optimal solution may be much larger than the running time of the last solution.
- The running time of the best solution: in some examples, a running time limit is set to terminate BPS if the running time exceeds a set period. If BPS terminates because the time limit is reached, the last solution obtained is the best solution, but it is not

confirmed whether it is the optimal solution. In this case, the time limit is the running time of the best solution.

For the purpose of research, a default running time limit of three hours (i.e., 10800 seconds) is set to terminate the system in case that an optimal solution still has not been found. The reasons are: (1) Previous study indicates that CST-based BPS is able to find a feasible solution very quickly, but it may need an excessive amount of time, even exponential time, to find an optimal solution. (2) From the industrial point of view, finding a feasible solution quickly is preferred to finding an optimal solution in an excessive amount of time. (3) A number of examples have been devised in this study to investigate the performance of our approach and system, so a uniform running time limit is required to compare the results of these examples to find the trends.

6.2 Performance Investigation and Discussion

In this subsection, we study the system behaviour as the difficulty of the scheduling problems increases. We hope to find some trends in using the CST-based system and to identify its strengths and weaknesses. For the above simple production example of four machines and 12 batches, it takes only 0.04 seconds for the first solution, but needs 23.944 seconds for the optimal solution. While other constraints remain the same, we have also tested on a number of plants with different numbers of batches and machines to investigate the trends. The running times of the resulting first solutions are presented in Table 2. Based on these results, the relationships of the running time vs. the number of batches and vs. the number of machines are derived and shown in Fig. 8 and Fig. 9, respectively.

With the increase of batch size and hence the number of activities, the problem difficulty increases. Two functions are displayed for each relationship in the figures. The first is the regression function representing the relationship based on the data points in the figures. The second is the fitness function representing the fitness of the regression curve and data points. The closer is the fitness (represented as R^2 in the figures) to one, the better is the fitness.

It can be seen that the relationship between the running time of the first solution and the number of batches is polynomial, no matter how many machines are available. The relationship between the running time of the first solution and the number of machines is also polynomial, no matter how many batches are required. Only 99.733 seconds are needed to find the feasible solution when scheduling a batch plant of 20 machines and 99 batches, which is a large-size problem. These results demonstrate that, if the focus is to find a feasible solution, CST-based BPS can solve a scheduling problem quickly even though it may be a large problem. However, if the focus is to find the optimal solution, the results are somehow different because the running time of the optimal solution is usually much longer than the running time of the first solution.

The running time of optimal solutions to problems of relatively small sizes can be obtained within the three-hour running time limit. The running times of optimal solutions are presented in Table 3. Based on these results, relationships of the running time vs. the number of batches and vs. the number of machines are derived and shown in Fig. 10 and Fig. 11, respectively.

It can be seen that the running time of the optimal solution increases exponentially as the number of batches increases. Optimal solutions can only be found within the three-hour running time limit for problems of up to 19 batches. Therefore, it is very expensive to find an optimal solution by using the CST-based system and only small-size problems can be tackled. However, we can also observe that the relationship between the running time of an optimal solution and the number of machines is still polynomial. Therefore, it seems that in general the number of machines will not significantly affect the running time in our approach.

Since the number of batches significantly affects the running time for the optimal solution and it does not for the first solution, we compare the quality of the first solution to that of the optimal solution in terms of the optimality criterion. It can be seen from Table 3 that for batch plants of four machines, the optimal solution can be found with up to 19 batches. For these examples, the makespan ratio of the optimal solution to the first solution is calculated and presented in Table 4. These ratios are all in the range from 70% to 75%, which is a quite small range. Compared to the running time increased exponentially as the batch size increases, these makespan ratios remain relatively unchanged. Other examples with eight, fifteen and twenty machines in Table 3 are also investigated and it is found that there is only one solution eventually obtained for every example, which means the first solution is the optimal solution. Therefore, the examples examined so far demonstrate that, for problems of relatively small sizes, it may be worth waiting for some time to find an optimal solution, however, there is little use to wait for a long time for problems of large sizes, because the makespan is unlikely to improve a lot and its improvement is not worth the exponential running time the system may need. It is, therefore, good practice in using the CST-based BPS that, unless there is no solution at all, the user finds at least one feasible solution and waits for a while until a tolerable time limit is reached. For example, if the time limit is 5 minutes (300 seconds), the last solutions for all the examples in Table 3 have already been found before the time limit.

7. Conclusions

Scheduling production processes by using computer-aided systems is important for improving the productivity of batch plants since it harmonizes the entire plant operation efficiently to achieve production goals. Although a number of papers are reported on the scheduling of batch plants, limitations of current scheduling approaches exist. In this paper we have developed a constraint-based model and system for batch process scheduling, and have investigated their performance. The main contribution of the study is the creation of a constraint-based scheduling model for batch plants. Although some constraints have been considered before, they were not comprehensive and also fragmented. The proposed constraint model analyzes and brings together these scheduling constraints, adds new constraints and categorizes them according to their functions. The comprehensive constraint-based scheduling model has never been presented in the literature.

A number of examples have been devised to investigate the performance of the proposed model and the developed system. These examples have been analyzed and discussed, and the results demonstrate the feasibility of the proposed methodology. It is found that the CST-based approach and system can schedule complex plants and solve large-size problems by finding feasible solutions that satisfies all imposed constraints, including some hard ones such as finite wait time constraints. It is also identified that the first feasible solution can be found very quickly but much more time, even exponential running time, is required to find the optimal solution particularly for complex and large-size problems. One of the contributions of this study is to demonstrate that a CST-based system, such as BPS, is suitable to find feasible solutions for scheduling a complex problem with many constraints, but it is not good to solve a problem optimally.

References:

Applequist, G., O. Samikoglu, J. Pekny, G. Reklaitis, 1997. Issues in the use, design and evolution of process scheduling and planning systems. ISA transactions 36 (2), 81-121.

Bailey, T.E., K. Lundgaard, 1989. Program design with pseudocode, Version 3 (Brooks/Cole Publishing Company, Pacific Grove, California).

Bok, Jin-Kwang, Sunwon Park, 1998. Continuous-time modeling for short-term scheduling of multipurpose pipeless plants. Industrial & Engineering Chemistry Research 37, 3652-3659.

Brailsford, S.C., C.N. Potts, B.M. Smith, 1999. Constraint satisfaction problems: algorithms and applications. European Journal of Operational Research 119, 557-581.

Brucker, P., J. Hurink, 2000. Solving a chemical batch scheduling problem by local search. Annals of Operations Research 96, 17-38.

Chen, B., C.N. Potts, G.J. Woeginger, 1998. A Review of Machine Scheduling: Complexity, Algorithms and Approximability, in: D.-Z. Du and P. Pardalos, eds., Handbook of Combinatorial Optimization, Vol. 3 (Kluwer Academic Publishers), 21–169.

Das, B.P., N. Shah, P.W.H. Chung, 1998. Off-line scheduling a simple chemical batch process production plan using the ILOG scheduler. Computers and Chemical Engineering 22, S947-S950.

Das, B.P., N. Shah, P.W.H. Chung, W. Huang, 2000. Comparative study of time-based and activity-based production scheduling. Hungarian Journal of Industrial Chemistry 28 (1), 7-10.

Datta, B., V.K. Jayaraman, B.D. Kulkarni, 2001. A comparative study of annealing methods for batch scheduling problems. Chemical Engineering Research & Design 79 (A6), Sep., 673-683.

Egli, U.M., D. W. T. Rippin, 1986. Short-term scheduling for multiproduct batch chemical plants. Computers & Chemical Engineering 10 (4), 303-325.

Goldman, R.P., M. S. Boddy, 1997. A constraint-based scheduler for batch manufacturing. IEEE Expert 12 (1), 49-56.

Grau, R., A. Espuna, L. Puigjaner, 1996. Completion times in multipurpose batch plants with set-up, transfer and clean-up times. Computers and Chemical Engineering 20, S1143-S1148.

Huang, W., P.W.H. Chung, 1999. Scheduling of Multistage Multiproduct Chemical Batch Plants Using A Constraint-Based Approach. Computers and Chemical Engineering 23, S511-S514.

Huang, W., P.W.H. Chung, 2000. Scheduling of Pipeless Batch Plants Using Constraint Satisfaction Techniques. Computers and Chemical Engineering 24 (2-7), 377-383.

Kelleher, G., P. Cavichiollo, 2001. Supporting rescheduling using CSP, RMS and POB – an example application. Journal of Intelligent Manufacturing 12, 343-357.

Kim, M., J.H. Jung, I.B. Lee, 1996. Optimal Scheduling of Multiproduct Batch Processes for Various Intermediate Storage Policies. Industrial & Engineering Chemistry Research 35, 4058-4066.

Kondili, E., C.C. Pantelides, R.W.H. Sargent, 1993. A General Algorithm for Short-Term Scheduling of Batch Operations - I. MILP Formulation. Computers and Chemical Engineering 17 (2), 211-277.

Ku, H., D. Rajagopalan, I. Karimi, 1987. Scheduling in Batch Processes. Chemical Engineering Progress (Aug.), pp 35-45.

Ku, H., I.A. Karimi, 1988. Scheduling in serial multiproduct batch processes with finite interstage storage: A Mixed Integer linear program formulation. Industrial & Engineering Chemistry Research 27, 1840-1848.

Ku, H., I.A. Karimi, 1991. An Evaluation of Simulated Annealing for Batch Process Scheduling. Industrial & Engineering Chemistry Research 30, 163-169.

Lee, C.Y., L. Lei, M. Pinedo, 1997. Current Trends in Deterministic Scheduling. Annals of Operations Research 70, 1-41.

Lee, K.H., S. Chung, H. K. Lee, I. B. Lee, 2001. Continuous time formulation of short-term scheduling for pipeless batch plants. Journal of Chemical Engineering of Japan 34 (10), 1267-1278.

Liu, R., 1996. A Framework for Operational Strategies for Pipeless Plants. (Ph.D. thesis, Department of Chemical Engineering, The University of Leeds) 26-28.

Niwa, T., 1993. Pipeless Plants Boost Batch Processing. Chemical Engineering 100 (6), 102-108.

Pantelides, C.C., M.J. Realff, N. Shah, 1995. Short-Term Scheduling of Pipeless Batch Plants. Chemical Engineering Research and Design 73 (A4), 431-444.

Realff, M.J., N. Shah, C.C. Pantelides, 1996. Simultaneous Design, Layout and Scheduling of Pipeless Batch Plants. Computers and Chemical Engineering 20 (6/7), 869-883.

Reklaitis, G.V., 1982. Review of Scheduling of Process Operations. AIChE Symposium 78, 119-133.

Rich, S.H., G.J. Prokopakis, 1986. Scheduling and sequencing of batch operations in a multipurpose plant. Industrial & Engineering Chemistry Process Design and Development 25, 979-987.

Rodrigues, L.A., M.Graells, J.Canton, L.Gimeno, M.T.M.Rodrigues, A.Espuna, L.Puigjaner, 2000. Utilization of processing time windows to enhance planning and scheduling in short term multipurpose batch plants. Computer and Chemical Engineering 24, 353-359.

Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy, W. Lorensen, 1991. Object-Oriented Modeling and Design (Prentice-Hall, Inc. ISBN: 0-13-630054-5).

Shah, N., C.C. Pantelides, R.W.H. Sargent, 1993. Optimal periodic scheduling of multipurpose batch plants. Annals of Operations Research 42, 193-228.

Shaw, K.J., A.L.Nortcliffe, M.Thompson, J.Love, P.J.Fleming, 1999. Interactive Batch Process Schedule Optimization and Decision-Making Using Multiobjective Genetic Algorithms. 1999 IEEE conference on systems, man and cybernetics, Tokyo, VI486-VI491.

Tsang, E.P.K., 1995. Scheduling Techniques ---- A Comparative Study. BT Technology Journal 13 (1), 16-28.

Tsirukis, A., G.V. Reklaitis, 1991. A comprehensive framework for the scheduling of resource constrained multipurpose batch plants. Proceedings of the 4th International Symposium on Process Systems Engineering, Montebello, Canada, Paper III 20, 1-15.

Vin, J. P., M. G. Ierapetritou, 2000. A new approach for efficient rescheduling of multiproduct batch plants, Industrial & Engineering Chemistry Research 39, 4228-4238.

Wang, H., L. Liao, 1997. Framework of constraint-based modelling for cooperative decision systems. Knowledge-Based Systems 10 (2), Aug., 111-120.

Yu, H., S.Lloyd, G. Kelleher, 1998. Closed-loop control and on-line scheduling of the batch process plants. IASTED international conference on intelligent systems and control, Halifax, Canada, June, 1-3.

APPENDIX:

Notation in the proposed constraint model:

Symbol	Definition
B_r, B_s	Batch activities
C	Capacity of an intermediate storage unit
$ST(.)$, $ET(.)$	Start and end time of an activity or operation
J_i, J_j	Operations
J_{B_r}	An operation within batch activity B_r
J_d	A operation producing a product with delivery time T_d
$J_{r,\text{first}}$, $J_{r,\text{last}}$	First and last operation within batch activity B_r
J_s	A storing operation
M_s	Makespan
M_u	Discrete material
S_p, S_q	Processing stations or machines
S_c	The total space already used by all storing operations at time T_c
T_c	Time point when an operation requires storage space to hold material
T_d	Delivery time of a product
T_f	Finite wait time
T_r	The time a vessel needs to move from the last processing station within batch activity B_r to the start point of another batch activity
T_u	A time point when material M_u is required
T_x, T_y	Start and end of a time window
TH	Time horizon of the whole production process
TO	Time origin of the whole production process
$Q_u(J_i)$	Quantity of material M_u required by operation J_i at time T_u
$Q(M_u)$	Total available quantity of material M_u at time T_u
$S(J_s)$	Space of a storage unit required by operation J_s
V_x, V_y	Moveable vessels

Table 1: Solution statistics for scheduling a batch processing plant

Solution Statistics	First Solution	Last Solution	System Termination (Confirm Optimal Solution)
Total Number of Activities	36	36	36
Number of Choice Points	12	132	4095
Elapsed Time since Creation i.e., running time (seconds)	0.04	0.831	23.944
Makespan (hours)	26	19	19

Table 2: The running time (in seconds) of the first solution for scheduling a number of batch plants

Number of Batches	Number of Machines			
	Four	*Eight*	*Fifteen*	*Twenty*
15	0.04	0.1	0.37	0.691
27	0.1	0.37	1.322	2.683
39	0.21	0.891	3.374	6.579
51	0.43	1.722	6.89	13.179
63	0.721	2.994	11.927	23.864
75	1.131	4.867	19.227	39.516
87	1.672	7.29	29.822	62.65
99	2.343	10.575	43.472	99.733

Table 3: The running time (in seconds) of the optimal solution for scheduling a number of batch plants

Number of Batches	Number of Machines			
	Four	*Eight*	*Fifteen*	*Twenty*
12	23.944	96.849	326.399	588.826
13	59.235	239.184	824.465	1481.16
14	140.822	589.297	2050.92	3668.65
15	328.582	1430.61	4929.87	8913.49
16	800.22	3428.9		
17	1851.02	8061.02		
18	4225.34			
19	9864.86			

Table 4: The makespan ratio between the optimal and first solution

Number of Batches	Makespan of Optimal Solution (h)	Makespan of First Solution (h)	Makespan Ratio (Optimal/First)
12	19	26	73.08%
13	21	28	75.00%
14	22	30	73.33%
15	23	32	71.88%
16	25	34	73.53%
17	26	36	72.22%
18	27	38	71.05%
19	29	40	72.50%

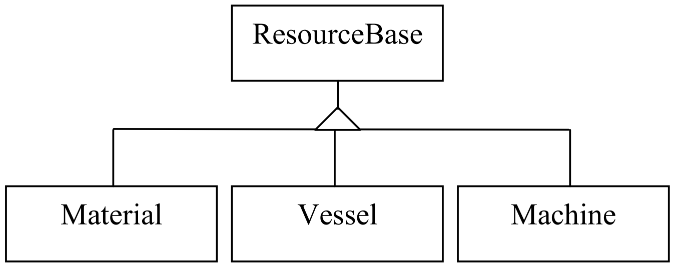


Fig. 1: ResourceBase class hierarchy

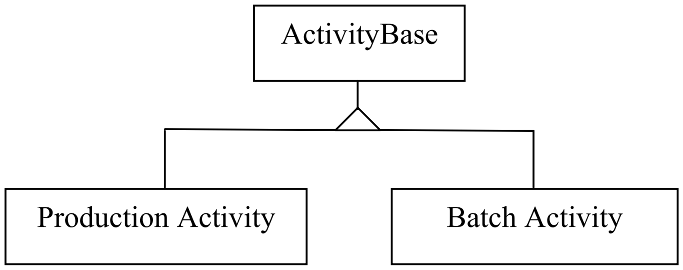


Fig. 2: ActivityBase class hierarchy

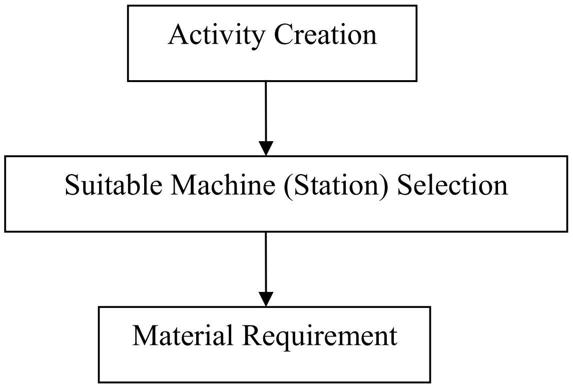


Fig. 3: Initialization of an object of production activity class

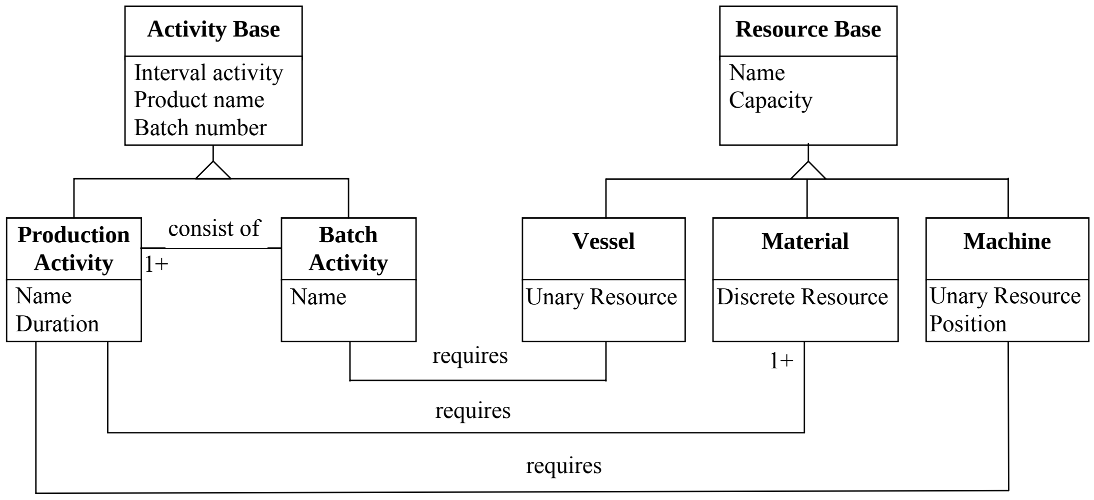


Fig. 4: Object diagram for resources and activities defined in *DefineProblem*

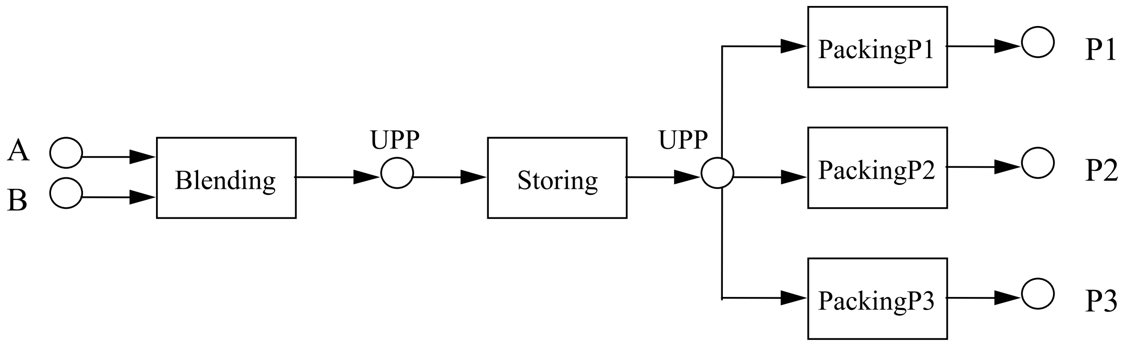


Fig. 6: Production processes of a batch processing plant

```
Read number of products,
number of cycles for a product,
number of production activities in a cycle,
and
number of runs of a production activity.

For each product
    For each product cycle
        If a pipeless batch plant is scheduled, then
            Create a batch activity to require a suitable moveable vessel
            to produce a batch of products.
        End if

        For each production activity in a cycle
        For each run of a production activity

            Create an operation to require suitable stations and
            material needed.

            Constrain the operation and batch activity to end
            before makespan.

            Set relevant constraints such as precedence and
            time-bound constraints.

            End loop
        End loop
    End loop
End loop
```

Fig. 5: The pseudocode for creation of batch and production activities

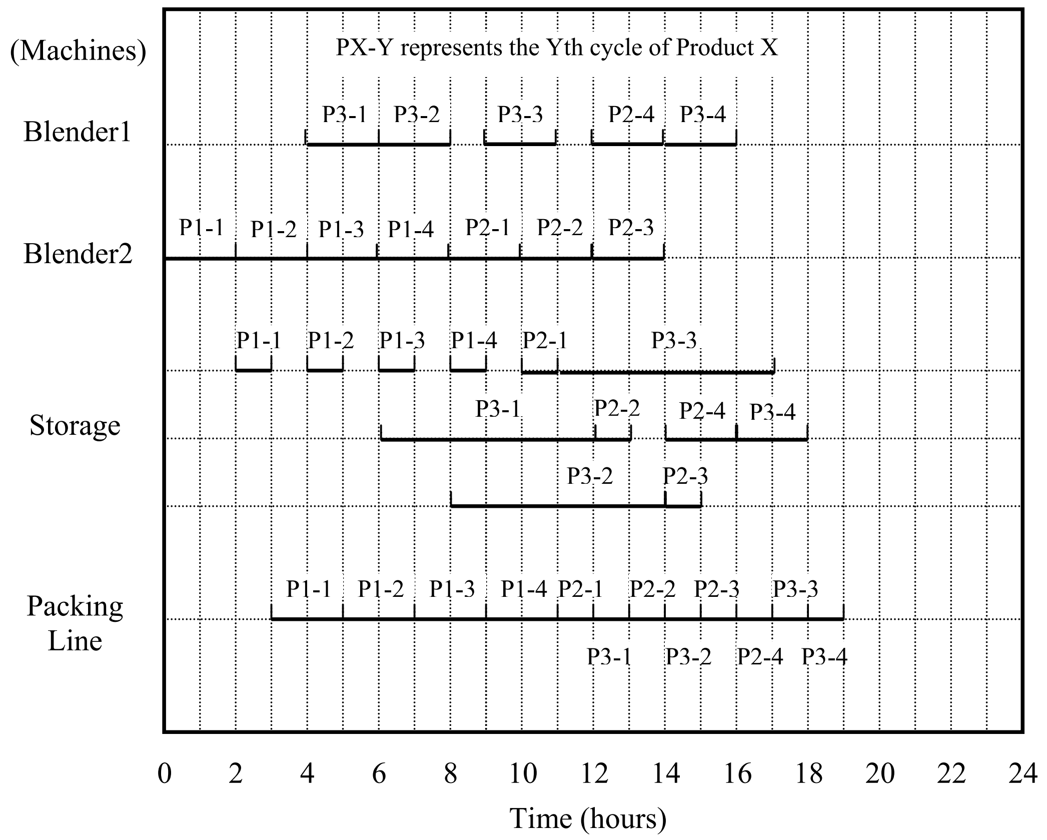
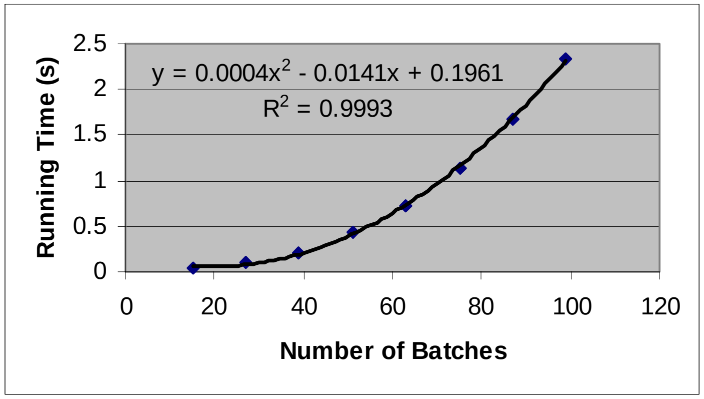
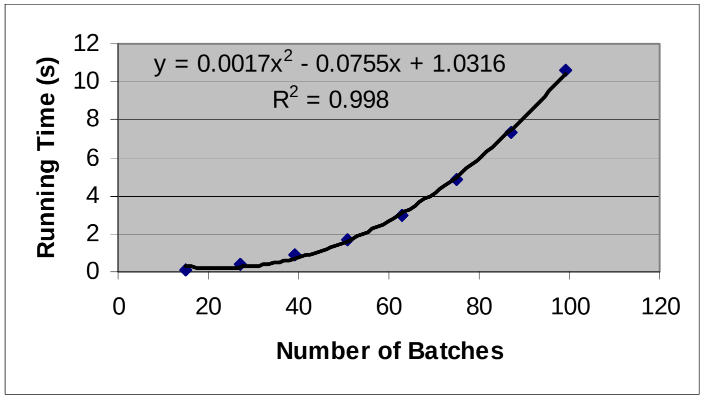


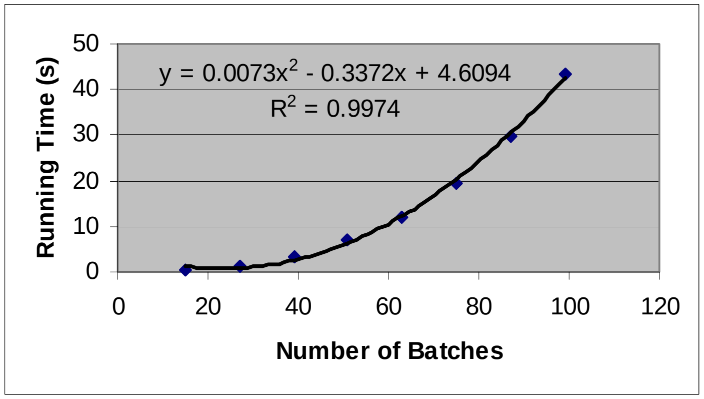
Fig. 7: The optimal solution for scheduling a batch processing plant



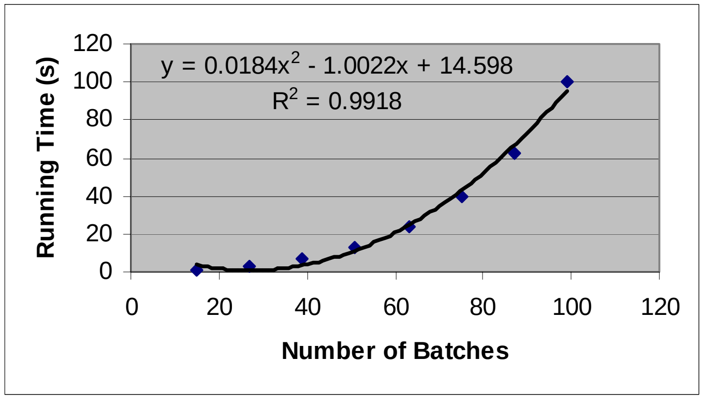
(a) Four machines available



(b) Eight machines available

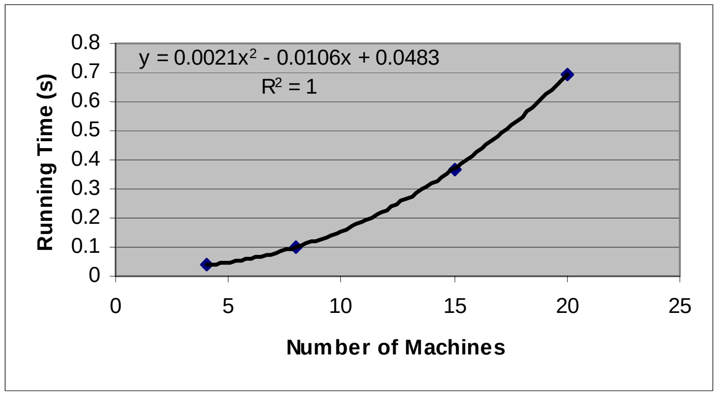


(c) Fifteen machines available

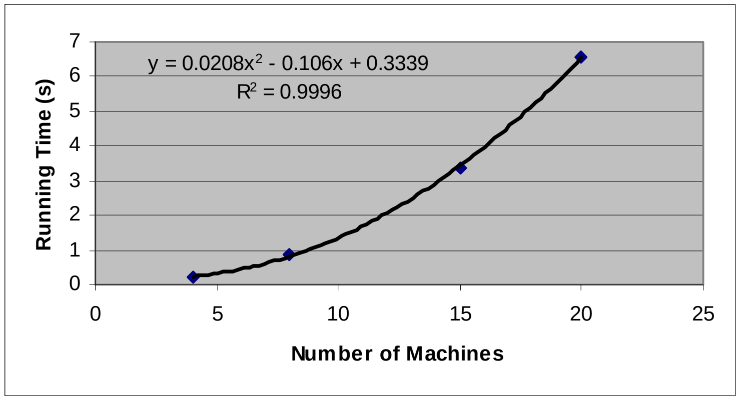


(d) Twenty machines available

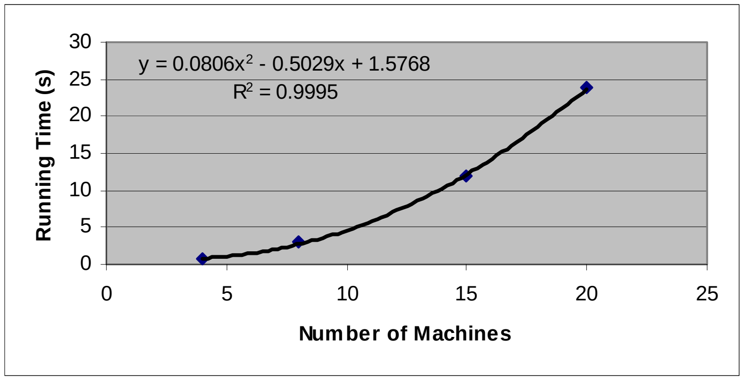
Fig. 8: The relationship between number of batches and running time of the 1st solution for scheduling batch plants



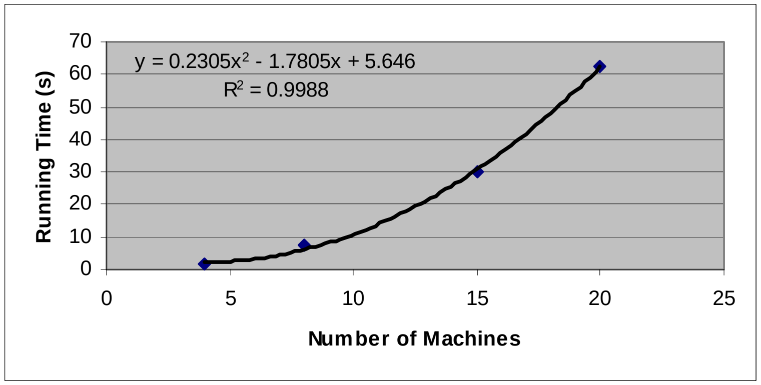
(a) 15-batch production



(b) 39-batch production

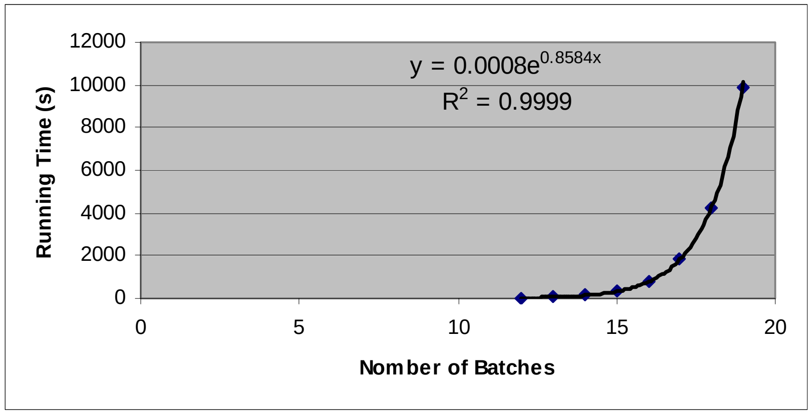


(c) 63-batch production

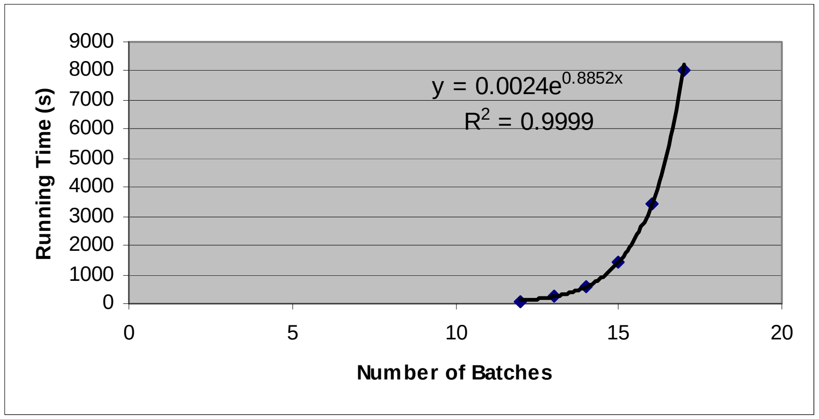


(d) 87-batch production

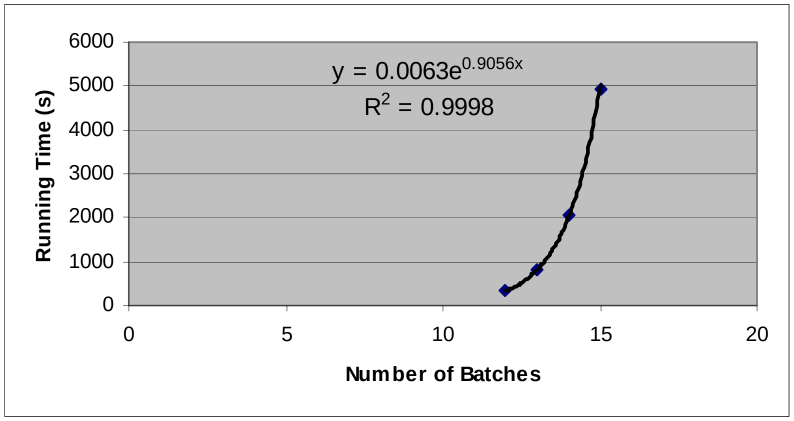
Fig. 9: The relationship between number of machines and running time of the 1st solution for scheduling batch plants



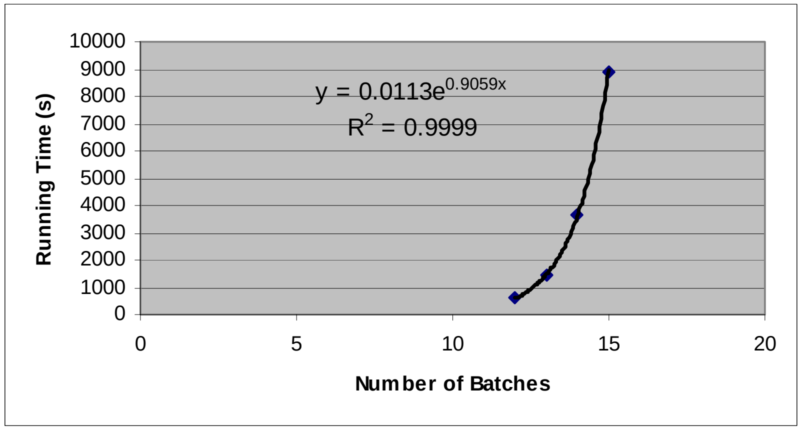
(a) Four machines available



(b) Eight machines available

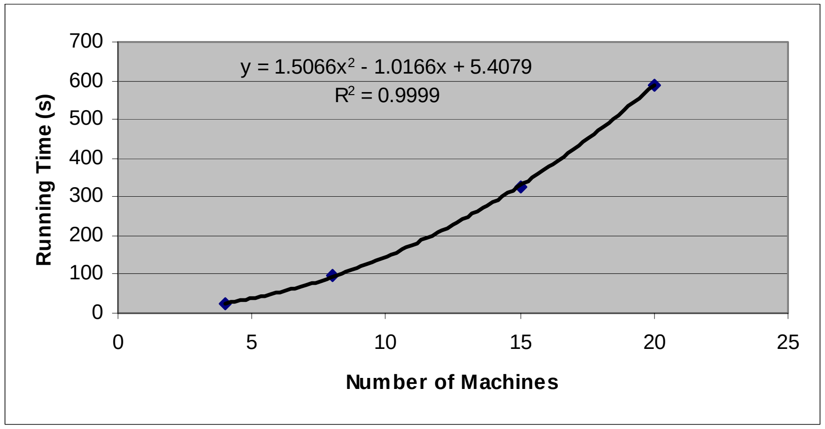


(c) Fifteen machines available

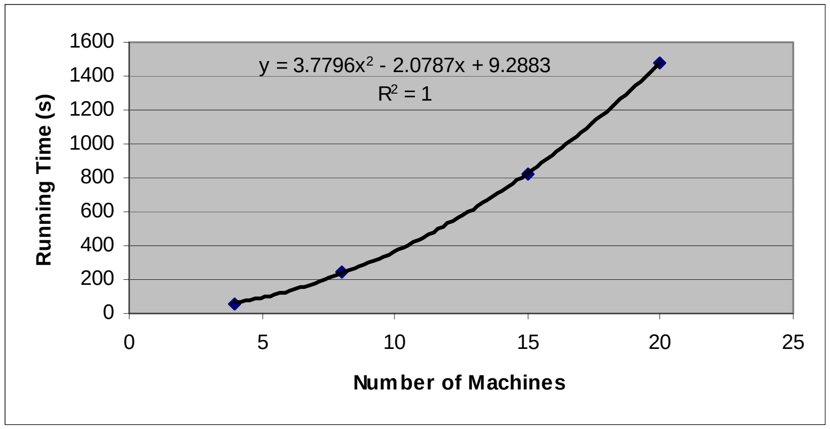


(d) Twenty machines available

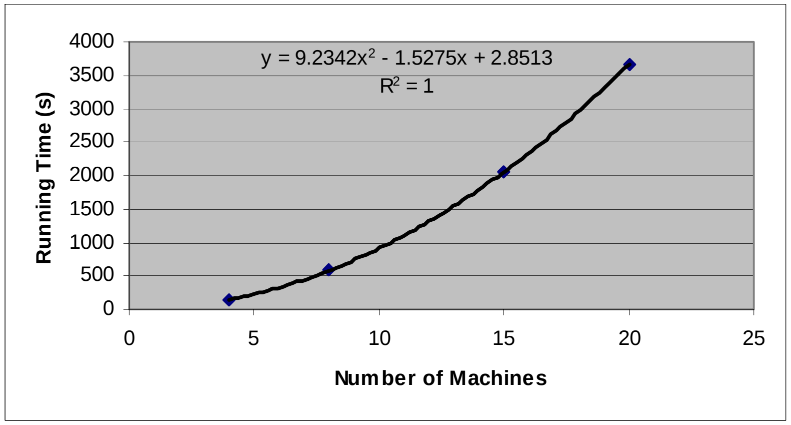
Fig. 10: The relationship between number of batches and running time of the optimal solution for scheduling batch plants



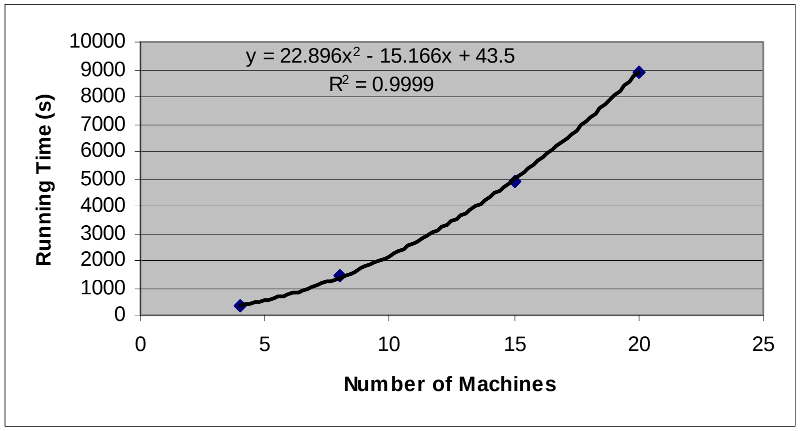
(a) 12-batch production



(b) 13-batch production



(c) 14-batch production



(d) 15-batch production

Fig. 11: The relationship between number of machines and running time of the optimal solution for scheduling batch plants