A practical decomposition approach for a chemical substance scheduling problem

Patrick Van Bael*, Marcel Rijckaert

Departement Chemische Ingenieurstechnieken, De Croylaan 46, B-3001 Heverlee, Belgium

Received 24 June 1998; accepted 1 November 1999

Abstract

This paper describes a decomposed iterative improvement scheduling algorithm developed to solve a scheduling problem of a chemical substance production machine. The algorithm has to determine the most effective schedule to produce several chemical substances. In order to build optimal feasible schedules a hybridization of AI techniques (heuristics, simulated annealing, constraint propagation) is used in a practical algorithm architecture. The algorithm architecture is compared against human schedule generation and a simple opportunistic algorithm.

Keywords: Scheduling; Decomposition; Local search

1. Introduction

Many practical scheduling problems do not fit into the simplified scheduling models [1]. The shop models like open or closed job shop models, flow shop models, one machine problems or multiple parallel machine problems have been described in literature and solved with general or even specific algorithms [2–5]. However, these simplified model representations cannot always be successfully applied to the practical specific problem detail, which is common for industrial scheduling problems [6]. These problems usually consist of specific constraints which have to be satisfied and cannot always be modeled and solved through the use of one single global method [7]. Hence, one tries to solve large real-world scheduling problems by using modular algorithm structures or architectures. Two elements are important in modular environments:

- Hybridization: the use of different techniques combined in the same architecture to solve the same task;
- Specialization: the combination of different modules using a similar technique for solving different tasks.

Modularity simplifies large complex tasks into a set of small, connected and manageable sub-modules, which can be solved separately whilst reducing complexity and development time.

Within a framework one can combine different (types of) scheduling techniques together with artificial intelligence techniques or even operational
research techniques. The scheduling optimization problem of a chemical substance production machine can also be solved with the use of a more complex architecture.

In Section 2 the problem is described; Section 3 focuses on the algorithm architecture to solve the scheduling problem and Section 4 shows the preliminary results accomplished with this algorithm architecture. Section 5 concludes with a discussion on these results.

2. Problem description

Every day a certain demand of chemical substances is requested. The request is specified with a specific order recipe to produce the chemical substance needed. The scheduler brings these orders together and tries to make a feasible and optimal schedule for the chemical substance machine.

The difficulty to make a feasible schedule comes from the many constraints which are defined by the chemical substance machine and from the chemical substance itself. The types of constraints for the machines are the common capacity constraints (limited resources) and the difference in time resolution between the chemical substance production phase and the following monitoring phase. The capacity constraints are the number of orders that can be scheduled during one day, the amount and number of components that can be added, the limited resources to handle the activities to add a certain component and the single shared monitoring machine. The most important constraint for the chemical substance is the duration of storage. During the production phase the actions representing components to add must be exactly like that specified in the recipe, while the sample procedures are allowed to be scheduled slightly different from the setpoint positions defined in the recipe.

2.1. Production layout

In Fig. 1 the chemical substance production machine is shown. One can see that the number of substances that can be produced together is sixteen and are separated into two production lines A and B. Each production line can use a set of basic components which is limited to eight. Each production line has indeed one single production arm which can get the basic components from the storage places X or Y. Production arm A can only get basic components from storage place X (and

![Diagram of production layout](image-url)
The production recipe consists of two phases (see Fig. 2). The first phase produces the chemical substance while the second describes the monitoring procedure. The production recipe consists of components to be added. Each component has two time specifications. The first is the time period to wait after a chemical substance started production (K) and the second specifies the duration to add the specific component (t₁). The monitoring recipe defines the amount of samples to be taken from the chemical substance together with time indications to start each sample procedure. These time indications are relative towards the end of the production phase (L).

2.2. Production recipe

The simple recipe illustrated in Fig. 2 shows the periods where the chemical substance is altered by the addition of new components. Here, the sample periods only take place after every production step has been performed. All sample procedures within the monitoring phase can be moved all together within the buffer period (b). The possible movements of the individual sample procedures are also shown (t₃).

3. Algorithm description

The algorithm is principally developed to obtain feasible schedules, which is the main difficulty of the problem, and in the second place to obtain an optimal schedule. The problem described is already simplified because the number of orders is already limited to the maximal capacity a priori known. The remaining problem therefore is to find a feasible schedule for one day developed with the given orders.

The algorithm is built with four sub-modules (see Fig. 3). The first sub-module selects a set of orders to be produced on the first production line, the rest of the orders is produced on the other production line. This module selects heuristically the appropriate production line for an order such that the capacity of the number of different components needed on one production line is not exceeded. Therefore, a distance measure is used which measures the number of different components needed. The module can have two possible results, the order set is or is not separable with these capacity constraints.
The second sub-module is created to avoid that the monitoring machine is not able to function without violating the sample constraints for every order. Therefore, every order is assigned an earliest start time (EST) for the monitoring phase. The module uses a simulated annealing algorithm and starts with all orders which are assigned a zero earliest start time monitoring phase. These earliest start times are the begin points from a discrete timeline. The algorithm disturbs the earliest start times one at a time. After every perturbation the simulated annealing algorithm evaluates the cost function which counts the occupation on every discrete period. At the end of a simulated annealing run the orders have an earliest possible start time where no discrete period has an occupation of samples which exceeds 100%.

A second cost function is also created where a predefined occupation is used to set the earliest start times as to achieve this occupation.

The third sub-module schedules the production phase of the orders by using the earliest start times of the monitoring phase as the earliest start times for the production phase. A simulated annealing algorithm is used here, where the cost function is defined as the earliest completion time of the latest production phase of all orders (i.e. the minimal makespan). This sub-module is used once for every production line.

The fourth sub-module schedules the sample procedures of all orders with the complementary information of the already scheduled production phase. The sample procedures are scheduled in two phases. The first phase tries to find an optimal buffer for every order. This buffer defines the distortion from the earliest start time after the production phase. The procedure starts with all buffers set to zero. The algorithm, again a simulated annealing algorithm, alters the buffers between zero and the maximal buffer value for one order each time. The cost function is an evaluation of the distortion of every sample start time from its earliest possible start time. In the second phase a simulated annealing algorithm tries to find the optimal sample start times using the previously defined buffer possibilities. This time the cost function is the earliest completion time of the latest monitoring phase of all orders and the distortion of every sample start time from its earliest possible start time.

Whenever a schedule is built, a partial or even a whole schedule, a generative schedule construction algorithm is used together with constraint propagation techniques [8]. This means that after every assignment of a start time of a production/monitoring action the constraints are updated. With this strategy no single action will overlap another action and no single action will be assigned a start time which will cause a backtrack procedure at the end of a schedule generation.

The simulated annealing algorithm is used in different modules. A general description of this algorithm can be seen in Fig. 4 [9] and has already been used several times with success in different domains [7,10,11].

Simulated annealing is a near-optimal stochastic optimization algorithm and is based on a simulation of a natural optimization process, i.e. the annealing (cooling down) of a liquid metal to a solid metal with a minimal energy level. This physical process can be achieved by gradually lowering the temperature, giving atoms (of the metal) enough time to rearrange themselves into an equilibrium state at each temperature. Hence, an annealing helps the material to avoid local minimal energy states and to find the global minimum energy state.

Simulated annealing as described in Fig. 4 is an analogy of the physical annealing process. It consists of a few general parameters like the initial temperature \((c_0)\), the end temperature \((c_e)\), the

```
INITIALISE (i_{start}, c_0, I_{opt}, k, i)
repeat
  for l = 1 to L_k do
    begin
      GENERATE (j \in S_i)
      if f(j) \geq f(i) then i = j
      else if \left( \frac{f(i) - f(j)}{c} \right) > \text{random}[0,1] \) then i = j
    end
  k = k+1
  CALC(I_k)
  CALC(c_k);
until stop criterion
```

Fig. 4. Simulated annealing algorithm.
temperature decrement function (CALC($c_k$)), the equilibrium definition ($L_k$) and a few special parameters like the perturbation mechanism (GENERATE) and the cost function ($f$) used. The simulated annealing algorithm used in the different sub-modules only differ in the special parameters like the cost function and the perturbation mechanism. Although the general parameters do have other parameter values, the structure of the algorithm is identical. An example of the cost function is given in Fig. 5. Here the cost function of the simulated annealing algorithm in sub-module four is given and shows that only optimization terms are used. The perturbation mechanism used is always SWAP, which is a well known mechanism in perturbations of sequencing structures. The simulated annealing algorithm was already successfully applied in [12].

The algorithm architecture can be seen as a specializing modular architecture. The use of this modular strategy makes it possible to define a subset of possible feasible solutions where the number decreases from sub-module to sub-module. The last sub-module tries to obtain the optimal feasible solution. Within every module the parameters can be optimized for a specific task. The difference between such a strategy and a single algorithm, which tries to optimize the problem with one cost function, is that no search time is wasted with the creation of unfeasible schedules. The last sub-module should thereby have a simpler task to find an optimal schedule since the number of feasible solutions is already limited to a small subset. Also, in the past a lot of single algorithms have failed because it is very difficult to obtain a schedule which is feasible and near-optimal when one tries to optimize both at the same time.

$$f = \sum_{i=1}^{\#\text{actions}} \left( tc_i - t_{ci} \right)^2 + \max_{j \in \text{actions}} (tc_j)$$

$tc_i =$ completion time of an action $i$

$t_{ci} =$ earliest completion time of an action $i$

Fig. 5. Cost function of the simulated annealing algorithm of sub-module four.

4. Test results

The algorithm architecture is applied to a practical set of orders. Various experiments were performed. In Table 1 the results are represented.

The table shows the result from an opportunistic algorithm and the new algorithm architecture where two different cost functions were applied as described in section three. The opportunistic algorithm creates a schedule by setting the orders one after the other without violating constraints. The algorithm improves the schedule by switching the sequence of orders. Table 1 illustrates that the creation of a schedule of near-optimal cost value was remarkably better than a schedule created with an opportunistic algorithm. The schedule quality is shown with two parameters. The first parameter shows the makespan of the global schedule. The second parameter shows the sum of all the disturbances of the sample procedures earliest start times. With these parameters, one can see that the quality of the schedule is increased with the new algorithm construction. Both the makespan and the number of disturbances decreased. The occupation which can be used as a parameter in the algorithm is an important factor and has an impact on the schedule quality. The higher the occupation, the smaller the makespan, but the higher the disturbances for the sample procedures from their earliest start times. At very low occupations the algorithm gives lower quality results according to the second cost factor.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Makespan</th>
<th>Disturbances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opportunistic algorithm</td>
<td>6 h 07 min</td>
<td>68</td>
</tr>
<tr>
<td>New algorithm:</td>
<td>5 h 41 min</td>
<td>38</td>
</tr>
<tr>
<td>New algorithm with occupation in cost function:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>90%</td>
<td>5 h 01 min</td>
<td>15</td>
</tr>
<tr>
<td>90%</td>
<td>5 h 11 min</td>
<td>1</td>
</tr>
<tr>
<td>90%</td>
<td>5 h 11 min</td>
<td>3</td>
</tr>
<tr>
<td>90%</td>
<td>5 h 15 min</td>
<td>1</td>
</tr>
<tr>
<td>60%</td>
<td>5 h 49 min</td>
<td>18</td>
</tr>
<tr>
<td>50%</td>
<td>5 h 13 min</td>
<td>8</td>
</tr>
<tr>
<td>40%</td>
<td>4 h 53 min</td>
<td>38</td>
</tr>
<tr>
<td>40%</td>
<td>4 h 57 min</td>
<td>35</td>
</tr>
<tr>
<td>40%</td>
<td>5 h 03 min</td>
<td>30</td>
</tr>
</tbody>
</table>
This comes from the fact that a very low occupation is impossible to schedule over one day, and therefore the system sets the earliest start times almost at the beginning of the day. This explains the low makespan and the high disturbance factor for the sample procedures. The lowest possible occupation without this phenomenon was 60% in this case.

The schedule generation time over human schedule generation time is drastically decreased. A human scheduler was busy for one day to generate a schedule of equal quality as an opportunistic algorithm, where the new algorithm architecture took only 3 minutes on average to achieve the illustrated results on an HP pentium Pro 180 MHz. Fig. 6 shows a schedule which was created by the above-described algorithm.

5. Conclusion

In this paper, a modular algorithm architecture is developed which was used to solve the scheduling problem of a chemical substance production machine. The algorithm is a decomposed iterative improvement scheduling algorithm consisting of several modules. Each module optimizes a smaller sub-problem focusing on a specific constraint and resulting in a set of feasible solutions. The last module searches the optimal solution from a limited set of feasible schedules. With this algorithm preliminary results have shown that it is possible to solve a scheduling problem by focusing on specific difficulties of the practical scheduling problem. The new algorithm’s performance is an improvement over human schedule generation time and over simple opportunistic algorithm results.

Acknowledgements

This work was financed with a fellowship from the Flemish Institute to Improve the Science-Technological Research in Industry (IWT)

References