Rescheduling for Manufacturing Based on Ontology and Problem Solving Method

Muhammad Firdaus B. Mustapha, Safaai B. Deris
Faculty of Computer Science and Information Systems
Universiti Teknologi Malaysia, 81310 Skudai, Johor, Malaysia.
Email: firdaus19@gmail.com; safaai@utm.my

Abstract

Manufacturing scheduling is a combinatorial problem, particularly in dynamic environment. This paper presents a rescheduling method in solving manufacturing scheduling which uses two main components namely as ontology to model scheduling and PSM. Two main problems are used to solve, firstly is static manufacturing problem and secondly is dynamic manufacturing problem which focus on rescheduling when new job arrive. Computational results show that schedule can be devised efficiently and correct. While when make a rescheduling, mostly of operations is not interfere the current schedule.

Keywords: Ontology, Problem-Solving Method (PSM), Scheduling, Manufacturing.

1. Introduction

Manufacturing scheduling can be defined as the process of allocating and timing resource usage in manufacturing system to complete jobs over time. According to [Pinedo, 2005], the purpose of scheduling in manufacturing is to minimize the production time and costs, by telling a production facility what to make, when, with which staff, and on which equipment in order to maximize the efficiency of operation and reduce costs. Most scheduling problems are combinatorial in the sense that the number of possible courses of actions or the time required to evaluate a given course of action is exponential in the description of the problem or more specifically a non-polynomial (NP) problem [Gupta and Nau, 1992]. The manufacturing scheduling problem becomes even more complex when it takes place in dynamic environment, where changes in the number of jobs or machine occur at any time. In dynamic manufacturing system the changes rarely go as expected such as [Leitao and Restivo, 2007]; (1) new job or rush job arrive, (2) resources breakdown or failure, or (3) delay happen.

Apart from this, scheduling problems arouse many researchers as it becomes an interesting subject to study. Lately, a research in knowledge-based system (KBS) was used to solve scheduling problem. This paper will be concentrating on an application of KBS which is known to have a great potential in reusability. It also has a clear separation of the knowledge about task, method, domain, and application. Substantially the algorithms can be changed by simple modification of the knowledge base.

2. Related Work

Scheduling problems had been solved by using various techniques in Operational Research (OR) and Artificial Intelligence (AI). In OR [Sharma, 1996], [Lin and Kononov, 2006], [Lee and Vairaktarakis, 1998], [Brah and Hunsucker, 1991] techniques such as linear programming, simulation, heuristics, and branch and bound. Research in OR aimed at finding an optimal solution, but optimization normally suffers from combinatorial complexity that can be proven NP-hard. Recently, various techniques in AI such as Constraint Based Reasoning (CBR), heuristics, and Genetic Algorithm (GA) [Burke, 1998], [Bierwirth et. al., 1995], [Bagchi et. al., 1991] have been used to solve scheduling problems. Moreover, several knowledge-based techniques had been reported normally to solve the scheduling problem such as by [Hori and Yoshida, 1998], [Sundin, 1994], and [Dnyanesh, 2006]. Besides that, various intelligent scheduling system like ISIS by [Fox, 1984], OPIS by [Smith, Fox, and Ow, 1987], DAS, YAMS, and SONIA were developed. Although these systems have used various AI techniques successfully, there were hardwired in nature and the domain specific nature of these systems restricted their reusability within a single domain.

Early research in AI was mainly focused on the development of “expert systems” that tried to capture domain knowledge by means of primitives such as production rules which were evaluated by rule interpreters. [Holger, 2002] states that, despite early success with small prototypical systems, research in expert systems failed to scale up for large systems, because the rather simple formalism of production rules did not support an adequate representation of different types of knowledge. Furthermore, knowledge encoded in rules is hard to reuse. In order to overcome these limitations of expert systems, [Holger, 2002] said that the researchers from Knowledge Engineering suggest building KBS form two kinds of reusable components: Ontology and PSM. Ontology [Gruber, 1993] can be seen as an information model that explicitly describes the various entities and abstractions that exist in a universe of discourse, along with their properties. PSM describe the reasoning processes of a KBS in an implementations and domain-independent manner. Ontology and a PSM are two main components in this research which focus on solving the scheduling problems. As discussed by [Van Heijst et al], a PSM can be used as a model-based template to guide the knowledge acquisition process and it can also be used to develop robust and maintainable applications by reuse.
In our approach, we subscribe to the Generic Library of Problem Solving Methods for Scheduling Applications by [Dnyanesh, 2006]. This library has used Task-Method-Domain-Application (TMDA) framework, which can be seen as a four tier hierarchy. At each phase of the development, and ontology is developed that makes the knowledge associated with that particular level explicit. The following types of ontologies for each level:

1. Task ontology: it is a vocabulary for characterizing the nature of scheduling task.
2. Method ontology: the method ontology formalizes the structure or the knowledge roles inherent to the problem solving methods.
3. Domain ontology: this ontology characterizes the body of knowledge associated with the particular domain of a task.
4. Application ontology: this ontology characterizes the body of knowledge that is specific to the particular application of a generic task.

3. Generating Schedules

Firstly the KBS generate schedule in order to determine the assignment of job to resource over time. Two main components in this KBS are task ontology and PSM. The first component is generic task ontology from [Dnyanesh, 2006]. The task ontology formalizes the nature of a scheduling task independent of any particular applications domain or the way problems can be solved. The task ontology is specified by using Operational Conceptual Modeling Language (OCML) [Motta, 1998]. OCML supports knowledge-level modeling specification of (Newell, 1982; Fensel and van Harmelen, 1994) by supporting the classes, relations, instances, functions, rules, etc. In the task ontology, the scheduling task is formalized in terms of the ten-dimensional space \{J, A, R, C, Req, Tr, etc. In initial state a schedule is incomplete because all the jobs and activities are still unassigned while in the solution state it satisfies all solution criteria. For a schedule to be a valid solution various conditions can be imposed, such as should be complete, should not violate any constraints, should maintain all the requirements, and should be cheaper than others states. In a problem space a transition from an initial state to a solution state can be achieved by means of operators where each operator is responsible for assigning jobs and activities to resources and time range.

- **Jobs**, \(J = \{j_1, \ldots, j_M\}\). A set of jobs to be assigned to a set of resources for their execution.
- **Activities**, \(A\). For each job, \(jm\), there are \(N\) uniquely associated activities. The set of all such activities is denoted as, \(A_{jm} = \{ajm_1, \ldots, ajm_N\}\).
- **Resources**, \(R = \{r_1, \ldots, r_I\}\). A set of resources to which the jobs and activities can be assigned for their execution. The constraint and requirement specific knowledge relevant to the resources must be obeyed while assigning the jobs and activities over resources.
- **Constraints**, \(C = \{c_1, \ldots, c_L\}\). A set of constraints that must not be violated by a solution schedule. E.g. resource constraint, job constraint.
- **Requirements**, \(Req = \{req_1, \ldots, req_K\}\). A set of requirements that describe the desired properties of a solution schedule. For instance, in the manufacturing domain, to execute the milling operation 'milling machine-A' must be present along with other tools.
- **Schedule time range**, \(Tr\). The time horizon in which the schedule takes place. It is represented in terms of a start time and an end time.
- **Preferences**, \(P = \{p_1, \ldots, p_T\}\). A set of criteria for choosing among competing solution schedules.
- **Cost function**, \(Cf\). A function that computes the cost of a solution schedule.
- **Schedule**, \(Sc = \{s_1, \ldots, s_W\}\). A schedule, \(Sc\), represents all possible schedules those can be generated as an output by the task ontology. Each schedule, say, \(Sw\) is a set of quadruples of the form: \(\langle \text{job}, \text{activity}, \text{resource}, \text{job time range} \rangle\).
- **Solution criterion**, \(Cr\). A mapping from \(Sw\) to \{True, False\}, which determines whether a candidate schedule is a solution. See below for the definitions of these properties.

3.1 A Generic Method of Scheduling

While the second component, the PSM we use a generic method of scheduling which applies in solving the scheduling problem. The generic method ontology [Dnyanesh, 2006] will take input from the scheduling task ontology and use ‘search’ problem-solving paradigm in constructing a schedule. The search paradigm in generic method ontology of scheduling represents the space of scheduling problem-solving by a state-space and operator.

A space problem has a group of states, where each state is uniquely represented by a schedule associated with it. In initial state a schedule is incomplete because all the jobs and activities are still unassigned while in the solution state it satisfies all solution criteria. For a schedule to be a valid solution various conditions can be imposed, such as should be complete, should not violate any constraints, should maintain all the requirements, and should be cheaper than others states. In a problem space a transition from an initial state to a solution state can be achieved by means of operators where each operator is responsible for assigning jobs and activities to resources and time range.

![Figure 2: The search-based problem space of scheduling.](image-url)
of a job is performed. Finally, a dead-end state is a state from which no solution can be achieved. By the way, the control structure of this method is generic. This generic method ontology of scheduling can be used to construct other specialize PSM such as Propose and Revise, Propose and improve and etc.

3.2 Propose and Revise

The Propose and Revise (Marcus, Stout, and McDermott, 1988) method was originally developed to tackle VT, a system for elevator configuration. Followed below are the procedures of this method:

1. Propose an extension to a schedule by applying procedures (to assign jobs to resources and time).
2. Check the currently extended schedule for constraint violations.
3. Revise a schedule in order to fix constraint violations by applying appropriate fix strategy.

In this research we propose to use extend-model-then-revise (EMR) (Motta and Zdrahal, 1998) to fix constraint violations. EMR fix the constraint violations as soon as they occur while constructing a solution.

4. Repairing schedules

Several method have been studied to solve dynamic rescheduling such (1) right-shift rescheduling, (2) regeneration, and (3) partial rescheduling [Viera et al., 2003]. For this paper use, we Full Rescheduling (FR) or regeneration schedule approach. FR, where all unfinished jobs are rescheduled to satisfy certain objective, can be applied in problems with small or medium size which can achieve optimal solution [Wang and Hong, 2007]. The repairing schedule is done, when new job arrives at time, \( t = k \), all operations behind time, \( t=k \) will reschedule along

5. Computational Results

The proposed KBS has been implemented using lisp and OCML library of (Motta, 1998). During the experimental phase we apply the proposed KBS. Firstly, the generation schedule is performed, and then rescheduling is performed when a new job arrives and should be added to current schedule. The results of rescheduling are presented in following subsections. Table 2 shows the information of new job arrives at time, \( t=8 \). The three of new jobs is namely Job14, Job 15, and Job 16 and the total number of operations is eight where these operations should be inserted into the current schedule. Figure 9 shows the Gantt chart of initial result of schedule and the makespan of schedule is 62 minutes. In the Gantt chart, numbers inside the blocks are the processing time associated with the jobs, and the black rectangles represent the machine idle periods.

Table 2: Information of new job arrives

<table>
<thead>
<tr>
<th>Item</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of new jobs</td>
<td>3 (Job 14, 15, 16)</td>
</tr>
<tr>
<td>No. of operations</td>
<td>8</td>
</tr>
<tr>
<td>No. of machines</td>
<td>6 (group into 3 machine types)</td>
</tr>
<tr>
<td>Earliest starting time</td>
<td>8</td>
</tr>
<tr>
<td>Due date</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 9: Gantt chart of initial schedule and the rescheduling time window is \( t=8 \).
Figure 10: Gantt chart of new job rescheduling.

Figure 10 shows the result of the first rescheduling scenario and the schedule achieves 72 minutes of the makespan. The new operations are scheduled at the behind of Gantt chart. The new operations are colored with grey.

5.1 The evaluation of rescheduling

To evaluate the result, the proposed algorithm is compared to Adaptive Genetic Algorithm (AGA) from (Yang and Wu, 2003). An efficiency measurement is defined as the percentage changes in makespan of repaired schedule compared to the prescheduled. The efficiency of rescheduling can be compute by:

\[ \eta = \left(1 - \frac{M_{\text{new}} - M_0}{M_0}\right) \times 100 \]  

\( k \) = Efficiency  
\( M_{\text{new}} \) = Makespan of the rescheduled schedule  
\( M_0 \) = Makespan of the prescheduled

Table 3: The comparison between AGA and proposed KBS

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Initial schedule</th>
<th>Updated schedule</th>
<th>Size of disruption (%)</th>
<th>Efficiency (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed KBS</td>
<td>62</td>
<td>72</td>
<td>16.13</td>
<td>83.87</td>
</tr>
<tr>
<td>AGA</td>
<td>66</td>
<td>78</td>
<td>18.18</td>
<td>81.82</td>
</tr>
</tbody>
</table>

Table 3 above shows the efficiency comparison between proposed KBS and AGA. The result shows that the efficiency of rescheduling using proposed KBS is better than AGA. According to the makespan, both of initial schedule and updated schedule of the proposed KBS is better than AGA. The second measurement is schedule stability which measures the number of revision or changes in starting time of operations that a schedule undergoes during the execution. The schedule is stable if it deviates minimally from the prescheduled. The schedule stability measurement for rescheduling can be compute like such below:

\[ \xi = \frac{\sum_{i=1}^{k} \sum_{j=1}^{P_j} \left| S_{ji}^* - S_{ji} \right|}{\sum_{i=1}^{k} P_j} \]  

\( \xi \) = Normalized deviation  
\( P_j \) = number of operations of job \( j \).  
\( k \) = number of jobs.  
\( S_{ji}^* \) = Starting time of \( i \)th operation of job \( j \) in repaired schedule.  
\( S_{ji} \) = Starting time of \( i \)th operation of job \( j \) in original schedule.

Table 4: The comparison between AGA and proposed KBS

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Schedule Stability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed KBS</td>
<td>8.91</td>
</tr>
<tr>
<td>AGA</td>
<td>10.21</td>
</tr>
</tbody>
</table>

Table 4 shows the results normalized deviation to measure schedule stability. The less normalized deviation means better for schedule stability. The results show that the stability of Proposed KBS is better than AGA.

6. Conclusion

In this paper, we use a generic library of PSMs from [Dnyanesh, 2006] to solve scheduling task. It is based on the TMDA knowledge framework and follows a top-down
The generic library of PSMs can be reused to construct other PSM by customized which modify or add new function in it. We reused the scheduling task ontology and method ontology from this library to solve job shop scheduling problem.

While doing this we use Propose and Revise paradigm which modify the procedure from the previous. The assignment job to resource and time needs to satisfy all constraint. So, we proposed this paradigm which it check the constraints while proposing the schedule. If it cannot propose then the revising is needed to suggest a solution until find one solution.

The solution of this technique is consistent and feasible because it can fix inconsistencies such as constraint and requirement violation and also get correct result. After getting initial schedule, we use to update it when new time resource or job arrive. The rescheduling is performed using regeneration method. The results show that after rescheduling, the schedule is feasible and consistent which it satisfy all the constraints. The efficiency of schedule is quite big because of rescheduling is done along with new job. And the schedule sustains mostly initial schedule according to stability.

7. References


Holger Knubalauch (2002), Ph.D Dissertation, Universitat ULM.


