MILP model for resource disruption in parallel processor system
Syarifah Zyurina Nordin and Louis Caccetta

Citation: AIP Conference Proceedings 1643, 764 (2015); doi: 10.1063/1.4907525
View online: http://dx.doi.org/10.1063/1.4907525
View Table of Contents: http://scitation.aip.org/content/aip/proceeding/aipcp/1643?ver=pdfcov
Published by the AIP Publishing

Articles you may be interested in
Task rescheduling model for resource disruption problem in unrelated parallel processor system
AIP Conf. Proc. 1605, 1086 (2014); 10.1063/1.4887742

ObstacleAware LongestPath Routing with Constraint Programming and Parallel MILP
AIP Conf. Proc. 1373, 18 (2011); 10.1063/1.3627191

A MILPModel for the Optimization of Transports
AIP Conf. Proc. 1281, 570 (2010); 10.1063/1.3498540

Parallel systems using the Weibull model

The TRIMM parallel processor
MILP Model for Resource Disruption in Parallel Processor System

Syarifah Zyurina Nordin\textsuperscript{a} and Louis Caccetta\textsuperscript{b}

\textsuperscript{a}Department of Mathematical Science, Faculty of Science, Universiti Teknologi Malaysia, 81310 UTM Johor Bahru, Johor.
\textsuperscript{b}Western Australian Centre of Excellence in Industrial Optimization (WACEIO), Department of Mathematics and Statistics, Curtin University, Perth 6845, Western Australia.

Abstract. In this paper, we consider the existence of disruption on unrelated parallel processor scheduling system. The disruption occurs due to a resource shortage where one of the parallel processors is facing breakdown problem during the task allocation, which give impact to the initial scheduling plan. Our objective is to reschedule the original unrelated parallel processor scheduling after the resource disruption that minimizes the makespan. A mixed integer linear programming model is presented for the recovery scheduling that considers the post-disruption policy. We conduct a computational experiment with different stopping time limit to see the performance of the model by using CPLEX 12.1 solver in AIMMS 3.10 software.

Keywords: task scheduling, unrelated parallel processors, mixed integer linear programming, rescheduling.

PACS: 02.30.Vv

INTRODUCTION

In classical scheduling, it is assumed that the system is always available for processing from the start time until the end. However, this assumption is invalid nowadays since parallel processing systems arising in many real applications especially in manufacturing applications with high levels of uncertainty, referring to the disruption that occurs in the system within a certain period. In practice, disruption in scheduling is one of the most important unexpected problems that are concern. It is important to revise the schedule and overcome the disruption since it may cause a disaster or a significant loss of income if it involved a major disruption.

There are many internal and external factors that cause the disruptions. For example, the internal factors that contribute to the disruptions are power supply failure, need for machine maintenance, issue in the quality control requirement, raw materials shortage, delay in producing the target and sickness in personnel. There are also external disruptions factors that can occur and affect the performances of the plan. The changes in the weather, changing customer orders, the cancellation/delay from the suppliers, political issues and the new government polices are several reasons that may cause the external disruptions.

The disruption in the system will change the system environment for the scheduling. A disruption problem might impact on the original schedule and in the worse case cause it to become infeasible. Therefore, it is necessary to have a recovery decision. The recovery decision should determine the possible options so that the problem could be solve optimally. A new recovery objective and constraints are important to represent the recovery problem.

In this paper, we focus on the rescheduling on unrelated parallel processor system. In the system, we are concerned with the interruption during the scheduling, a machine becomes unavailable at a certain time. The problem is specifically described as resource disruption. The resource disruption also refers to the machine shortage in the parallel processor system that gives impact to the task scheduling. The potential machine disruption are considered in which the unavailability can be not known in advance and occurred unpredictably, it is considered as post-disruption. The rescheduling for recovery option can only start on or after the disruption period. This post-disruption policy in-charge of the remaining tasks left in the system for the revision process. Following are the three field notations of Graham et al. [1]. We refer to this problem as $R|\text{disruption}|\omega_1 y + \omega_2 z$, i.e. the task scheduling problem for minimizing the weighted of makespan, $y$, and the total deviation costs, $z$, on unrelated parallel processors with the model requires scheduling having disruption.
LITERATURE REVIEW

We now provide further information on the disruption problem in parallel processor system. An extensive review of the literature on the disruption problem can be found in Vieira et al. [2] and Aygut et al. [3]. Qi et al. [4]; Lee and Yu [5]; Goren and Sabuncuoglu [6]; Hall and Potts [7]; and Hall et al. [8] studied the disruption problem on a single machine system. The same problem on parallel processor systems can be found in Alagoz and Azizoglu [9]; Azizoglu and Alagoz [10]; Tang and Zhang [11]; Ozlen and Azizoglu [12]; Ozlen and Azizoglu [13]; and Arnaout and Rabadi [14]. For a multi-stage environment, Rangarajiratnam et al. [15] and Moratori et al. [16] studied the disruption specified on job shop scheduling.

Resource disruption is a disruption that involves a shortage in resources such as machines and personnel. This disruption causes a schedule to become infeasible, as the resource is not available at the specified time. Alagoz and Azizoglu [9]; Azizoglu and Alagoz [10]; and Tang and Zhang [11] are among the researchers that have focussed on machine disruption. They applied the disruption problem in identical parallel processor environment. Alagoz and Azizoglu [9]; and Azizoglu and Alagoz [10] considered the same machine disruption problem and they proposed several heuristic algorithms including a polynomial time algorithm and BB based heuristic to minimize the number of disrupted jobs and the total flow time. Tang and Zhang [11] investigated machine breakdown disruption that can minimize the total weighted completion time and the weighted deviation completion time cost from the original schedule. They used Langragian relaxation to solve their model and obtained near-optimal solutions for all instances.

In this research, rescheduling for resource disruption is considered particularly for unrelated parallel processing systems and it is considered to be an NP hard problem (Azizoglu and Alagoz [10]). Our work is the extension to the research addressed by Nordin and Caccetta [17]. They have developed the model for the post-disruption case with recovery objectives and recovery constraint as a match up strategy to stay close and minimize the deviation with the initial schedule. Then, they examined the stability of the model and the model performed with high value of stability measure on different size problem. We continue the work on disruption problem of R|disruption|\(\omega_1, y + \omega_2 z\) with a larger data set and the value of \(\omega_1 = \omega_2 = 0.5\). Then we evaluate the model performance at different time limit.

RESOURCE DISRUPTION ON UNRELATED PARALLEL PROCESSOR SYSTEM

The scheduling problem that we consider is a system consisting of \(n\) independent tasks, \(J_i (i = 1,2, \ldots, n)\), to be processed by \(m\) unrelated parallel processors, \(M_j (j = 1,2, \ldots, m)\). The following are the conditions that satisfies the system:

1. Both sets of task and processor are all assume available at time zero.
2. Even there are disruptions in the system, we still assume that the task are non-preemptive where the task need to be process without interruption.
3. If the disruption occurs in the middle of processing task, the task should reschedule from the start, which is called as a non-resumable cases.
4. The task can be assigned to any processor.
5. The task migration between processor is allowed for the reschedule process.
6. Each processor is assumed to be continuously available unless if the system having disruption.
7. Once the disruption has been fixed, the processor immediately available until the system satisfies the stopping criteria.
8. Each processor capable to supports one task at a time by an integer processing time except during the disruption activity.

The disruption feature that model and solve in this section is focus on resource disruption. We assume that only this type of disruption may occur at any time. Resource disruption is refer to the availability of resource is decreases for some period of time Zhu et al. [18]. The potential machine disruption is considered in which the unavailability can be predictive. The time of disruption and the duration can be expected due to managerial arrangement such as scheduled machine maintenance or recession of employee. In this problem, we can only know the disruption event until the disruption really enters the system. All remaining tasks in the system that are not complete will be sending for rescheduling. The tasks that are already processed are not involved in the reschedule. We consider the post-disruption model using the rescheduling start time same as the disruption start time to reduce the delay in the rescheduling.
Mixed Integer Linear Programming (MILP) Model for Resource Disruption

Notation

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I$</td>
<td>Index for tasks</td>
</tr>
<tr>
<td>$J$</td>
<td>Index for processors</td>
</tr>
<tr>
<td>$T$</td>
<td>Index for time</td>
</tr>
<tr>
<td>$N$</td>
<td>Number of tasks $i$</td>
</tr>
<tr>
<td>$M$</td>
<td>Number of processors $j$</td>
</tr>
<tr>
<td>$F$</td>
<td>Number of time slot $t$</td>
</tr>
<tr>
<td>$x_{ij}$</td>
<td>The variables for the current schedule for task $i$ that processing on processor $j$ in the rescheduling phase</td>
</tr>
<tr>
<td>$p_{ij}$</td>
<td>The processing time for task $i$ on processor $j$</td>
</tr>
<tr>
<td>$C_i$</td>
<td>The completion time for task $i$</td>
</tr>
<tr>
<td>$x_{ijt}$</td>
<td>The 0-1 time indexed decision variable of task $i$ starts processing on processor $j$ at time $t$</td>
</tr>
<tr>
<td>$A_{jt}$</td>
<td>The processor availability</td>
</tr>
<tr>
<td>$\phi_{ij}$</td>
<td>The assignment for task $i$ on processor $j$ for the initial schedule</td>
</tr>
<tr>
<td>$D_{ij}$</td>
<td>The migration tasks $i$ from processor $j$ to another processor after the disruption occurred and can be written as $D_{ij} = \phi_{ij} - x_{ij}$</td>
</tr>
<tr>
<td>$\omega$</td>
<td>The weighted</td>
</tr>
<tr>
<td>$Y$</td>
<td>The makespan where $y = \max{C_i</td>
</tr>
<tr>
<td>$Z$</td>
<td>The total deviation costs, $z = \sum_i \sum_j D_{ij}$</td>
</tr>
</tbody>
</table>

MILP Formulation

The initial schedule before the resource disruption happen to the schedule are obtain using an optimum $R||C_{max}$ model that has been formulated by Potts [19]. The variables from the initial schedules are transferred to the disruption model as parameter. Then, we implement the $R|\text{disruption}|\omega_1 y + \omega_2 z$ model. The MILP model for the problem $R|\text{disruption}|\omega_1 y + \omega_2 z$ can be written as follows:

Let the 0-1 time indexed decision variable is denoted as follows:

$$x_{ijt} = \begin{cases} 1, & \text{if processing of task } i \text{ starts processing on processor } j \text{ at time } t \\ 0, & \text{otherwise} \end{cases}$$

(1)

where $i = 1, 2, ..., n$, $j = 1, 2, ..., m$ and $t = 1, 2, ..., f$.

In the model, we consider the disruption allocation problem and remain the scheduling efficiency measure in minimizing the makespan. To describe the resource disruption that have been allocated, we let $A_{jt}$ is the processor availability when,

$$A_{jt} = \begin{cases} 1, & \text{if processor } j \text{ is available at time } t \text{ without disruption} \\ 0, & \text{otherwise} \end{cases}$$

(2)

The following MILP model contains the recovery constraints set for the rescheduling time horizon where one processor is disrupted at time $0$. The assignment variables (1) are used in the model and the rescheduling model can be formulated as follows with all information have been updated to reflect the disruption allocation:

Minimize $Q = \omega_1 y + \omega_2 z$,

subject to

$$\sum_{j=1}^{m} \sum_{t=1}^{f} A_{jt} x_{ijt} = 1 \quad \text{for } i = 1, 2, ..., n$$

(3)

$$C_i = \sum_{j=1}^{m} \sum_{t=1}^{f} (t + p_{ij}) x_{ijt} \quad \text{for } i = 1, 2, ..., n$$

(4)

$$\sum_{j=1}^{m} \sum_{t=1}^{f} x_{ijt} \leq 1 \quad \text{for } j = 1, 2, ..., m \quad t = 1, 2, ..., f$$

(5)

$$\sum_{i=1}^{n} \sum_{t=1}^{f} p_{ij} x_{ijt} + \sum_{t=1}^{f} (1 - A_{jt}) \leq y \quad \text{for } j = 1, 2, ..., m$$

(6)

$$D_{ij} + \sum_{t=1}^{f} x_{ijt} \geq \phi_{ij} \quad \text{for } i = 1, 2, ..., n \quad j = 1, 2, ..., m$$

(7)

$$y, C_i, D_{ij} \geq 0 \quad \text{for } i = 1, 2, ..., n \quad j = 1, 2, ..., m$$

(8)

$$x_{ijt} \in \{0,1\} \quad \text{for } i = 1, 2, ..., n \quad j = 1, 2, ..., m \quad t = 1, 2, ..., f$$

(9)

Constraints (3) give restriction to the task assignment at the disruption time slot. Constraints (4) define the completion time of a task. Constraints (5) ensure that only a task is progressing during the execution time slot.
Constraints (6) represent the completion time on a processor is less than makespan. Constraints (7) represent the deviation cost for the task migration and constraints (8) require the non-negativity for the deviation cost and makespan. Constraints (9) define the 0-1 variables used in the model.

**COMPUTATIONAL EXPERIMENTS**

In this computational experiment, we carry out a comprehensive computational testing of the proposed post-disruption model to see how well the model performs. In Nordin and Caccetta [17], we have look at the results on the stability measure of the model. Hence, we extend the computational testing on the model performance at different time limit.

**Computational Design**

We generate the initial schedule and implement the disruption model using AIMMS 3.10 software on a PC with Intel Core 2 2.66 GHz 1.95 GB RAM. We setup a deterministic disruption environment with different length of disruption slots at the beginning of the schedule. The data set are generated using the following data:

1. The total number of tasks enters the system, \( n = \{100, 200\} \).
2. For every set of tasks, there are three number of processors used where \( m = \{6, 8, 10\} \).
3. For every combination, we generate 20 instances. We compute the processing time as the following uniform distribution: \( p_{ij} = U[\min p_{ij}, \max p_{ij}] = U[1, 100] \).
4. The disruption execution time is generated as follows: \( d_{ad} = L_\ell \max\{p_{ij}\} \) where \( L_\ell = 1, 0.5, 0.25 \) for \( \ell = 1, 2, 3 \).

**Gap on Different Time Limit of Computational Time**

We now present the performance of the disruption model with \( \omega_1 = \omega_2 = 0.5 \) and 10 simulation problems for each test case. In this experiment, the gap is recorded at the specified solving time limit that obtained from the following equation:

\[
\text{Gap} \% = \frac{\text{Best solution}-\text{Best LP bound}}{\text{Best solution}} \times 100
\]  

where the best LP bound is refers to the lower bound obtain from the model.

The motivation of the experiment is to observe the quality of the solving time of the model for a large data set. The MILP model is tested with 9 different stopping limits (in seconds): 30, 60, 150, 300, 600, 900, 1200, 1500 and 1800.

Table 1 shows the average of the gap (%) recorded at different time limit. From this experiment, we can see the performance of the model from the beginning of 30 seconds until the last stopping limit at 30 minutes. At time \( t = 30 \), all the instances have the gap less than 30% from the optimum and at \( t = 300 \) all cases already less than 10% gap. In other words, the model has a good quality of solutions and can be obtained within a short amount of computational time.

**TABLE 1. Average gap at the specified stopping criteria.**

<table>
<thead>
<tr>
<th>Number of processors ( m )</th>
<th>Number of Tasks ( n )</th>
<th>Level ( L )</th>
<th>30</th>
<th>60</th>
<th>150</th>
<th>300</th>
<th>600</th>
<th>900</th>
<th>1200</th>
<th>1500</th>
<th>1800</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>100</td>
<td>( L_1 )</td>
<td>6.07</td>
<td>5.10</td>
<td>4.09</td>
<td>3.76</td>
<td>2.82</td>
<td>2.74</td>
<td>2.46</td>
<td>2.42</td>
<td>2.35</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( L_2 )</td>
<td>6.23</td>
<td>4.62</td>
<td>3.88</td>
<td>3.51</td>
<td>2.57</td>
<td>2.42</td>
<td>2.29</td>
<td>2.18</td>
<td>2.07</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( L_3 )</td>
<td>5.76</td>
<td>4.04</td>
<td>3.02</td>
<td>2.69</td>
<td>2.22</td>
<td>2.11</td>
<td>2.07</td>
<td>2.01</td>
<td>2.00</td>
</tr>
<tr>
<td>200</td>
<td>( L_1 )</td>
<td>4.37</td>
<td>3.82</td>
<td>2.99</td>
<td>2.62</td>
<td>2.31</td>
<td>2.13</td>
<td>1.82</td>
<td>1.78</td>
<td>1.56</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( L_2 )</td>
<td>4.68</td>
<td>3.65</td>
<td>2.46</td>
<td>2.31</td>
<td>2.21</td>
<td>1.94</td>
<td>1.87</td>
<td>1.65</td>
<td>1.63</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( L_3 )</td>
<td>3.72</td>
<td>2.70</td>
<td>1.83</td>
<td>1.52</td>
<td>1.43</td>
<td>1.25</td>
<td>1.20</td>
<td>1.13</td>
<td>1.11</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>( L_1 )</td>
<td>9.15</td>
<td>8.00</td>
<td>6.60</td>
<td>6.12</td>
<td>4.98</td>
<td>4.61</td>
<td>4.56</td>
<td>4.47</td>
<td>4.14</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( L_3 )</td>
<td>8.36</td>
<td>5.62</td>
<td>5.06</td>
<td>4.15</td>
<td>3.37</td>
<td>3.32</td>
<td>3.26</td>
<td>3.13</td>
<td>3.08</td>
</tr>
</tbody>
</table>

This article is copyrighted as indicated in the article. Reuse of AIP content is subject to the terms at: http://scitation.aip.org/termsconditions. Downloaded to IP: 161.139.220.12 On: Thu, 05 Feb 2015 05:41:40
<table>
<thead>
<tr>
<th></th>
<th>L1</th>
<th>L2</th>
<th>L3</th>
<th>L4</th>
<th>L5</th>
<th>L6</th>
<th>L7</th>
<th>L8</th>
<th>L9</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>6.73</td>
<td>4.92</td>
<td>4.16</td>
<td>3.63</td>
<td>3.60</td>
<td>3.48</td>
<td>3.47</td>
<td>3.30</td>
<td>3.18</td>
</tr>
<tr>
<td></td>
<td>8.46</td>
<td>5.50</td>
<td>4.19</td>
<td>3.90</td>
<td>3.7</td>
<td>3.54</td>
<td>3.38</td>
<td>3.33</td>
<td>3.18</td>
</tr>
<tr>
<td></td>
<td>6.82</td>
<td>4.66</td>
<td>3.68</td>
<td>3.41</td>
<td>3.23</td>
<td>2.69</td>
<td>2.42</td>
<td>2.39</td>
<td>2.36</td>
</tr>
<tr>
<td>10</td>
<td>11.66</td>
<td>9.92</td>
<td>8.08</td>
<td>6.41</td>
<td>5.70</td>
<td>5.55</td>
<td>5.55</td>
<td>5.55</td>
<td>5.55</td>
</tr>
<tr>
<td></td>
<td>15.97</td>
<td>12.02</td>
<td>8.84</td>
<td>7.99</td>
<td>7.24</td>
<td>7.17</td>
<td>7.16</td>
<td>7.08</td>
<td>6.99</td>
</tr>
<tr>
<td></td>
<td>12.05</td>
<td>7.29</td>
<td>6.41</td>
<td>6.08</td>
<td>5.85</td>
<td>5.74</td>
<td>5.64</td>
<td>5.43</td>
<td>5.23</td>
</tr>
<tr>
<td>200</td>
<td>25.76</td>
<td>15.61</td>
<td>9.41</td>
<td>5.57</td>
<td>5.33</td>
<td>5.23</td>
<td>5.22</td>
<td>5.14</td>
<td>5.12</td>
</tr>
<tr>
<td></td>
<td>22.55</td>
<td>15.29</td>
<td>11.31</td>
<td>7.16</td>
<td>5.88</td>
<td>5.85</td>
<td>5.71</td>
<td>5.34</td>
<td>4.73</td>
</tr>
</tbody>
</table>

Figure 1 shows the average gap for all three disruption levels at respective combination of tasks and processors that we described as \( n \times m \). In the figure, the \( x \)– axis indicates the percentage of the average gap while the \( y \)– axis is the time limit (in seconds) that we consider. The graph is obtained from the previous table and illustrates the performance of the model for every combination of tasks and processors at the beginning of the computational time. In the diagram, we can see that the performances of the system with 200 tasks are better than 100 tasks for every processor at the first 30 minutes. The most interesting observation is when the largest data set (i.e 200 \( \times \) 100) improved very fast within 300 seconds of the CPU times. It indicated the model can obtained small gap for a large data set before reach the optimum value in short computational time.

**SUMMARY**

In this paper, a case of \( R|\text{disruption}|\omega_1 y + \omega_2 z \) is described which is refer to the task scheduling on unrelated parallel processor system having a disruption problem. The objective function of the problem is to minimize the weighted of makespan and the total deviation costs. Our strategy in solving the disruption problem is rescheduling the initial schedule. The rescheduling approach is a recovery option to manage the disruption by address new constraint and new objectives. MILP model are formulated for the recovery model with match-up schedule technique. A computational study is conducted for a large data set and found that the results are good since the first 30 seconds of the computational time. Then the gaps are reducing until obtain less than 7.99% starting at 300 second of CPU time.

**ACKNOWLEDGMENTS**

The authors would like to thank especially to MOE, Universiti Teknologi Malaysia (UTM) and GUP (the vote number 08H49) for funding the project.
REFERENCES