Scheduling of Shipyard Block Assembly Process Using Constraint Satisfaction Problem

Heemoon Kim*, Jangha Kang** and Sungsoo Park***
(received November 2001; accepted December 2001)

We consider a scheduling problem on a shipyard block assembly process. Given a set of blocks and a set of bays, we need to determine the bay and the starting time to assemble each block while satisfying the precedence constraints, the time window constraints and the resource constraints. We mainly consider two types of resources for each bay: space resource and man-power resource. Our primary objectives are to minimize the number of blocks failed to be scheduled and load balancing for man-power. To solve this problem, an algorithm based on CSP (Constraints Satisfaction Problem) technique is proposed. Computational experiences are reported for real world problems.

Keyword : Scheduling, Constraint Satisfaction Problem, Shipyard Block Assembly Process

1. Introduction

In this paper, we consider a scheduling problem on a shipyard block assembly process. Shipbuilding process is illustrated in Figure 1. After contract signing and design process, we cut steel plates into small parts in accordance with the design and construct blocks with them. Block is a basic component to construct a ship. Blocks are assembled in workshops which are called bays. Outfitting process is to outfit items like pipes inside the block. After that, painting of the block is performed. After outfitting and painting process, the blocks are assembled in the dock to form the hull of the ship, which is called the erection process. The ship is then launched and finishing process is done at the harbor.
In most shipbuilding companies, docks are considered as the most important resources. So the schedule for the erection process in the docks is made first so that the docks are fully utilized. Schedules for the other processes are made considering the dock schedule, other restrictions and resources pertinent to each process.

In this paper, we only consider the scheduling problem of block assembly process. The scheduling problem for the block assembly process is to determine, for each block, the bay to assemble the block and the starting time of the block assembly. The most important constraints that we need to consider in the block assembly are as follows: the precedence constraints, the time window constraints and the resource constraints. The precedence constraints restrict the order in which the assembly jobs for some blocks are performed. The time window constraints limit the starting times of the jobs. The resource constraints refer to the available resources at the bays. They include the space and man-power limitations in the bays.

We briefly summarize the recent researches about shipbuilding scheduling. Sang-Gyu Min et al. [1] developed a genetic algorithm for the erection scheduling in shipbuilding. Tae Hyun Baek et al. [2] developed resource balancing heuristics for the scheduling of the erection process. They derived earliest start (ES) schedule by PERT to get an initial resource limit. And then, the heuristics lower the resource limit by a given increment and derive a new schedule. K. Park et al. [3] developed a scheduling algorithm on a shipyard block assembly process satisfying time window constraints and space constraints. Shie-Gheun Koh et al. [4] introduced a production scheduling system for the block assembly workshops in a shipyard.
objectives of their system are balancing of the work load of the subassembly production workshops as well as the assembly workshops.

These recent studies mainly have used heuristic algorithms to solve the problems. In this paper, we propose an algorithm based on the constraint satisfaction problem (CSP) approach.

This paper is organized as follows. In section 2, we introduce details of the block assembly process. We introduce the CSP approach and model the scheduling problem using CSP in section 3. In section 4, we present an algorithm using CSP for the problem. Next, computational results are shown in section 5. Finally, concluding remarks are given in section 6.

2. The block assembly process

In this section, we introduce the details of the block assembly process. In the block assembly process, we make blocks in the bays by assembling the small parts made in the cutting process. There are two types of bays in the particular shipyard we considered. One is called fixed-bay and the other is called moving-bay. Moving-bay is more efficient in its operations but it can only assemble small and flat blocks. On the other hand, fixed-bay is mainly used to assemble curved blocks and large blocks. We only consider the scheduling problem for blocks to be assembled in the fixed-bays and will refer fixed-bay as bay hereafter.

Resources in a bay are used in the assembly process. Space resource and man-power resource are the two most important resources. The man-power resource is measured by available man-hours for a day at each bay, which can be obtained by multiplying the number of workers by normal working hours a day. The number of workers is considered fixed for each bay during the scheduling horizon. In reality, however, the amount of man-power resource may not be the same as the standard man-hours since overtime working is permitted. Overtime working is permitted for a short time period but prolonged overtime is not allowed. For this reason, we obtain the amount of man-power resource by multiplying the standard man-hours by man-power utilization ratio, which may take on a value of over 100%.

Bays have rectangular shapes. Each block assigned to a bay occupies the space of the bay while it is being built. Hence total area occupied by the blocks assigned to a bay should not exceed the area of the bay at any day in the planning horizon. Therefore we consider the space of each bay as a resource of the bay. We call it space resource.

However, the condition that the area occupied by the blocks should not
exceed the area of the bay does not necessarily guarantee the schedule is workable since the blocks have irregular shapes. We actually need to determine the locations of the blocks to obtain a workable schedule, which is a very difficult problem to solve. But we know by experience that if the areas of the blocks are less than a certain portion of the area of the bay, then a workable positioning of the blocks can be obtained in most cases. Therefore, we multiply the area of the bay by a space utilization ratio, which is given as a parameter in the model, to obtain the amount of space resource of a bay.

There are three types of blocks we consider: grand assembly blocks, unit assembly blocks, and sub-assembly blocks. The relationship between blocks is depicted in Figure 2.

![Figure 2 Relationship of blocks](image)

Sub-assembly blocks are made by assembling the small parts made in the cutting process. A unit assembly block consists of several sub-assembly blocks one of which is called the main sub-assembly block. A grand assembly block consists of several unit assembly blocks one of which is called the main unit assembly block. The assembling job of a main unit assembly block must end before the assembling job of the corresponding grand assembly block starts. Similar restriction also exists between the main sub-assembly block and the corresponding unit assembly block. Grand assembly blocks are used to make the hull of the ship in the erection process.

There are occasions that several grand assembly blocks are assembled
together to form a very large block before erection. This process is called pre-erection (PE) and the resulting block is called the PE block. Meanwhile, the grand assembly blocks which constitute a PE block is referred to as the PE block set. A PE block is considered as a block and it is assembled in the erection process with other grand assembly blocks. It is desirable that the assembly jobs for the grand assembly blocks in a PE block set end almost at the same time to reduce the work-in-process and to expedite the assembly of the PE block. This condition is imposed in the model so that the ending times of assembly jobs for blocks in a PE block set must be within a time interval. The length of this time interval is called the maximum PE interval and the value is the same for all PE blocks.

There are some grand assembly blocks which need special skills and much effort to assemble. These are Engine Room/Double Bottom blocks (ER/DB blocks). Since it requires much of the resources of a bay to assemble an ER/DB block, it is difficult to assemble more than one ER/DB blocks at the same time in a bay. Hence, if we need to assemble two ER/DB blocks in a bay, the starting time of the second ER/DB block must be apart at least a certain period of time after the starting time of the first ER/DB block.

An example of the schedule of ER/DB blocks assigned to a bay is shown in Figure 3. Block 1 and block 3 are ER/DB blocks. The starting time of block 1 is 7 and that of block 3 is 15. The difference of the starting time of these two blocks must be at least 8 in this example. However, Block 1 and block 3 can be assembled together from 15 to 22 in the same bay.

![Figure 3 Example of ER/DB block schedule](image)

It may happen that we cannot schedule all of the blocks while satisfying the constraints. Those blocks which failed to be scheduled are called failure blocks and they are given out as outside orders. One of our major objectives is to obtain a schedule which minimizes the number of failure blocks.
The latest starting time of a grand assembly block is computed backward from the schedule of the erection process. We refer to this time as schedule start day (SSD) for each grand assembly block. Also each grand assembly block has the earliest starting day (ESD). The difference between SSD and ESD is constant for all grand assembly blocks. Actually ESD for each block is computed from SSD and the constant. SSD and ESD are not defined for unit assembly blocks and sub-assembly blocks.

3. Modeling using CSP

An instance of a constraint satisfaction problem (CSP) involves a set of variables \( X = \{x_1, \ldots, x_n\} \), a domain \( D_i \) for each variable \( x_i \), \( 1 \leq i \leq n \) and a set of constraints \( C = \{C_1, \ldots, C_q\} \) such that \( C_j \subseteq D_1 \times \cdots \times D_n \), \( 1 \leq j \leq q \), which define feasible combinations of domain values. A solution is an assignment of domain values to all variables that is consistent with all imposed constraints. In this section, we propose a model of the problem using CSP.

We used two kinds of variables in the model. One is the variable \( b_i \), \( i \in B \) which are used to denote the bay in which block \( i \) is assembled. Here \( B \) denotes the set of bays. The other is the variable \( t_i \) which denotes the starting time of work on block \( i \).

The domain of each variable is the set of values that each variable can have. The domain of variable \( b_i \), \( i \in B \) is the set of the available bays located in the shipyard. We call it bay domain for simplicity. The bay domain of each block can be different depending on the block because of the height and weight limitations on the block that the crane in the bay can handle. We define the fail bay as the bay to assemble the failure blocks. Each block has the fail bay in the bay domain. Note that fail bay actually does not exist in the shipyard. It is introduced here for modeling purposes only.

The domain of variable \( t_i \) is determined by the value of SSD and ESD if the block is a grand assembly block. We call it starting time domain for simplicity. The time unit is not an hour but a day. If the SSD of a grand assembly block is 25 and the difference between ESD and SSD is 5, ESD of the grand assembly Block is 20 and the starting time domain of this block is \( \{20, 21, 22, 23, 24, 25\} \).

The constraints we considered are categorized in the following. First, we introduce the notation and parameters used in the description of
constraints. They are summarized in Table 1 for ease of reference.

### Table 1 Notation and model parameters

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SSD_i$</td>
<td>SSD of grand assembly block $i$</td>
<td>$pl_v$</td>
<td>low bound on ending time for blocks in a PE block set $v$</td>
</tr>
<tr>
<td>$ESD_i$</td>
<td>ESD of grand assembly block $i$</td>
<td>$pu_v$</td>
<td>upper bound on ending time for blocks in a PE block set $v$</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>SSD - ESD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d_i$</td>
<td>processing time of block $i$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$te_i$</td>
<td>ending time of block $i$</td>
<td>$\nu$</td>
<td>low bound on ending time for blocks in a PE block set $v$</td>
</tr>
<tr>
<td>$tb_{ij}$</td>
<td>buffer between block $i$ and $j$</td>
<td>$\nu$</td>
<td>upper bound on ending time for blocks in a PE block set $v$</td>
</tr>
<tr>
<td>$\beta$</td>
<td>maximum PE interval</td>
<td>$P(j)$</td>
<td>set of sub-assembly blocks constituting unit assembly block $j$</td>
</tr>
<tr>
<td>$\delta$</td>
<td>ER/DB starting time interval</td>
<td>$\nu$</td>
<td>low bound on ending time for blocks in a PE block set $v$</td>
</tr>
<tr>
<td>$M$</td>
<td>set of unit assembly blocks</td>
<td>$s_b$</td>
<td>area of bay $b$</td>
</tr>
<tr>
<td>$P$</td>
<td>set of sub-assembly blocks</td>
<td>$sb_i$</td>
<td>area required by block $i$</td>
</tr>
<tr>
<td>$N$</td>
<td>total number of blocks</td>
<td>$m_b$</td>
<td>standard man-hours available for a day in bay $b$</td>
</tr>
<tr>
<td>$B$</td>
<td>set of bays</td>
<td>$R_S$</td>
<td>space utilization ratio</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>decrement in man-power utilization ratio</td>
<td>$mb_i$</td>
<td>amount of man-hours required by block $i$ for a day</td>
</tr>
<tr>
<td>$R_M$</td>
<td>man-power utilization ratio</td>
<td>$x_{i,b,t}$</td>
<td>1 : if block $i$ is assigned to bay $b$ at time $t$</td>
</tr>
<tr>
<td>$\max_R$</td>
<td>maximum work load ratio</td>
<td></td>
<td>0 : otherwise</td>
</tr>
<tr>
<td>$TIME$</td>
<td>set of times for scheduling horizon</td>
<td>$U_f$</td>
<td>upper bound on the number of failure blocks</td>
</tr>
<tr>
<td>$N_f$</td>
<td>the number of failure blocks</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### a. Time window constraints

All grand assembly blocks must start the job between the ESD and SSD of each block.

$$ESD_i \leq t_i \leq SSD_i \quad \text{where} \quad ESD_i = SSD_i - \alpha, \quad i \in L$$

The grand assembly blocks in a PE block set have constraints on the ending times. The maximum gap of the ending times of the grand assembly blocks in a PE block set must be less than or equal to the maximum PE interval $\beta$. The maximum PE interval $\beta$ is constant for all PE block sets. The earliest ending time $pl_v$ and the latest ending time $pu_v$ of grand
assembly blocks in a PE block set $\nu$ are not fixed. These values can vary during the search procedure but maximum PE interval $\beta$ must be kept.

$$pl_\nu \leq te_i \leq pu_\nu, \quad pl_\nu + \beta = pu_\nu$$
for all grand assembly block $i$ in PE block set $\nu$

b. Precedence constraints

There is a buffer between the starting time of a grand assembly block and the ending time of unit assembly blocks constituting the grand assembly block. Suppose that we have a grand assembly block $i$ and the set of unit assembly blocks $M(i)$ constituting block $i$. The latest ending time $te_j$ of a block $j \in M(i)$ is determined by the starting time of block $i$ and the buffer of the block $j$. One of the unit assembly blocks in $M(i)$ is the main unit assembly block. Main unit assembly block's buffer is 0 and other unit assembly blocks' buffers are positive values. For example, suppose there are one grand assembly block and three unit assembly blocks that constitute the grand assembly block. We suppose the starting time of the grand assembly block is 10 and the buffers of unit assembly blocks are 2. Then main unit assembly block must be completed by 10. Other unit assembly blocks must be completed by 12.

In the same way, main sub-assembly block's buffer is 0 and other sub-assembly blocks' buffers are positive values.

$$te_j \leq t_i + tb_{ij}, \quad \forall i \in L, \forall j \in M(i)$$

$$te_l \leq t_k + tb_{kl}, \quad \forall k \in M, \forall l \in P(k)$$

c. Resource constraints

The total amount of space and man-power resources needed for blocks assigned to a bay should not exceed the available resources in each day in the planning horizon. As is remarked earlier, we used the man-power utilization ratio and the space utilization ratio to reflect overtime working and to guarantee actual positioning of blocks in the bay.

$$\sum_{i} x_{i,b,t} \times s_{b} \leq R_s \times s_b, \quad \forall b \in B, \forall t \in TIME$$

$$\sum_{i} x_{i,b,t} \times m_{b} \leq R_m \times m_b, \quad \forall b \in B, \forall t \in TIME$$

d. Other constraints
The starting times of any two ER/DB blocks should be apart at least a certain period of time to avoid overloading. We call this period as ER/DB starting time interval and denote it as $\delta$.

Suppose that two ER/DB blocks $E_1$ and $E_2$ are assigned to the same bay. The starting time of $E_1$ is $t_{E_1}$ and the starting time of $E_2$ is $t_{E_2}$. Then either $t_{E_1} + \delta \leq t_{E_2}$ or $t_{E_2} + \delta \leq t_{E_1}$ should hold for a feasible schedule. Hence we need the constraints.

$~(b_i = b_j) \lor (t_i + \delta \leq t_j) \lor (t_j + \delta \leq t_i)$ for all possible pair of ER/DB blocks $i_1, i_2$

Note that if the time windows of two ER/DB blocks do not overlap, we don't need to specify the ER/DB constraint for the pair of ER/DB blocks.

Finally, we explain the objectives of the scheduling problem. There may exist many objectives for a scheduling problem, some of which may be conflicting. Here, we consider two objectives which are regarded as the most important in the shipyard.

The first objective is minimizing the number of failure blocks and the second is load balancing of works for man-power. Load balancing is achieved by minimizing the maximum work load ratio over all bays and over the time in the planning horizon.

Work load ratio for each bay and for each day in the planning horizon is computed as the percentage ratio of assigned work load to the standard man-hours of the bay. Since CSP only finds solutions which satisfy the constraints, those objectives are included in the model as constraints with possible upper bounds. We repeatedly try to find a feasible solution of the model after decreasing the upper bounds on the objectives as long as we can find an improved solution.

There still remains a subtle point which needs to be mentioned. In figure 4, two schedules are shown for the same bay. The two schedules have the same work load and the same maximum work load ratio. However, schedule 2 is preferred to schedule 1 in the shipyard since daily work load in schedule 2 fluctuates less. It is difficult to reflect this point in the CSP model. Hence, rather than trying to model this situation, we try to find a schedule with less fluctuations in the work load over time when we search for a schedule in the algorithm.
4. Algorithm

In this section, we give an overall explanation of our algorithm. First, we illustrate the overall structure of our algorithm to find a feasible solution in subsection 4.1. In subsection 4.2, we will illustrate the steps to construct a search tree throughout the algorithm.

4.1. Overview of the algorithm

In this subsection, we present the overall structure of the algorithm, which is illustrated in Figure 5. We try to find an improved solution repeatedly after a feasible solution is found.

Initially, we set the man-power utilization ratio as the maximum possible value and the upper bound on the number of failure block as the total number of blocks. After finding a feasible solution of the model, we may obtain a solution with a smaller number of failure blocks and the maximum work load ratio is less than or equal to the current man-power utilization ratio. We then set the upper bound on the number of failure blocks as the newly obtained value and decrease the man-power utilization ratio by a certain percentage points and we try to find a feasible solution of the new model. This procedure is repeated until we fail to find a feasible solution of the model.
4.2. Constructing a search tree

In this subsection, we describe the part "A" in Figure 5. A feasible schedule is found in the part "A". First of all, we illustrate briefly how CSP searches for a feasible solution. CSP considers all possible cases for all variables to find a feasible solution. We can represent the search procedure as a tree which we call a search tree. A node in the search tree represents a status of decision variables and their domains.

If there exist a variable such that the domain of it has two or more values in a node of the search tree, we make two child nodes by restricting the domain of the variable, which we call branch operation. This procedure is repeated until the values of all variables are determined, in which case we obtain a completed search tree. The values of variables that are no longer feasible because of constraints are removed from variable domains in CSP. If

---

Figure 5 The flow chart of overall algorithm
the domain of a variable is modified, then the domains of other variables are checked through constraints and the values that cannot be part of a solution are removed from the domains of the variables. We refer to this modification as propagation. We will use the depth first search technique during the search procedure to find a feasible solution.

We use a sub-function in part "A". It is used repeatedly until we obtain a feasible completed search tree for all blocks. The sub-function consists of three parts: selection of a block, branching on the bays to assemble the selected block, and branching on the starting times of the selected block. We apply the sub-function at each node of the search tree to obtain a completed search tree. The steps taken in the sub-function are given below. The flow chart of the sub-function is illustrated in Figure 6.

**CSP algorithm: sub-function**

*Step 0.* Select a node in the search tree.

*Step 1.* Block selection.

If there are unscheduled blocks, select an unscheduled block and go to step 2. Otherwise, a feasible solution is obtained. Stop the search.

*Step 2.* Branching on the bays.

If no bay is available for the selected block, the node is infeasible. Go to step 4.

If only one bay is available for the selected block, go to step 3.

If two or more bays are available for the selected block, we branch out two child nodes. One child node represents that the selected block is assigned to a bay $b$ and the other child node represents that the block should be assigned to one of the available bays except the bay $b$. Go to step 4.

*Step 3.* Branching on the starting times.

If the current starting time domain of the selected block is empty, the node is infeasible. Go to step 4.

Otherwise, We branch out two child nodes. One child node represents that the selected block starts its assembly on a specified time $t$ selected from the current starting time domain of the block and the other child node represents that the block starts its assembly on one of the times in the starting time domain except time $t$. 
Step 4. End of sub-function.

Figure 6 The flow chart of sub-function
In step 1, we select a block having the latest LST (latest start time) among the blocks for which the bay or the starting time are not determined yet. Note that the LST of a block is not necessarily the same as the SSD of the block since the starting time domain of the block may have been changed by propagation or branching operation in the search procedure.

When we branch on the bays, the bay $b$ to which the selected block is assigned is chosen randomly if there are two or more bays available. However, selecting the starting time $t$ in step 3 is performed more judiciously. As we remarked earlier, the schedule with less fluctuations in the work load in a bay is preferred in the shipyard. But examining all feasible schedules and selecting the schedule with the smallest fluctuations in the work load is impractical since the search tree is enormous in size. Hence a plausible approach is trying to construct a schedule with small fluctuations in the work load while we search for a feasible solution. This objective can be achieved in part by selecting the starting time $t$ in the branching procedure with more care.

Suppose we select block $i$ in step 1 and the bay is selected in step 2. Let $L_t$ be the work load in the bay at time $t$, measured in man-hours, and $C_t^\text{max}$ be the maximum work load ratio of the bay in time $[t, t + d_i]$ in the partial schedule constructed so far. We define

$$T_t = \{(L_{t-1} - L_t) + (L_{t+d_i} - L_{t+d_i-1})\} \times \frac{R_M}{C_t^\text{max}}, \text{ for all } t \in [ESD_i, SSD_i].$$

Then, among the times in the starting time domain, the time $t$ which maximizes $T_t$, $t \in [ESD_i, SSD_i]$ is selected as the starting time in step 3.

For example, consider the case shown in Figure 7. Figure 7 shows the work load of a bay over time for a partial schedule.
Suppose that \( d_i = 3 \) and the starting time domain of block \( i \) is \([12, 15]\). Also suppose that \( R_M = 110\% \) and standard man-hours of the bay is 12.

<table>
<thead>
<tr>
<th>( t ) (day)</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L_t )</td>
<td>6</td>
<td>10</td>
<td>8</td>
<td>7</td>
<td>5</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>((L_{t-1} - L_t) + (L_{t+d_i} - L_{t+d_{i-1}}))</td>
<td>-1</td>
<td>-6</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( C_t^{\max} )</td>
<td>83%</td>
<td>83%</td>
<td>67%</td>
<td>58%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( T_t )</td>
<td>-1.33</td>
<td>-7.95</td>
<td>1.64</td>
<td>5.69</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

From Table 2, maximum of \( T_t \), \( t \in [12, 15] \) is 5.69 on day 15. Hence day 15 is chosen as the starting time.

5. Computational results

We tested the algorithm on two data sets. The characteristics of data used in the tests are briefly described in Table 3. The data set 1 and data set 2 are from real data.
Table 3 Characteristics of data sets

<table>
<thead>
<tr>
<th>Data set</th>
<th>Number of grand assembly Blocks</th>
<th>Number of unit assembly Blocks</th>
<th>Number of sub-assembly Blocks</th>
<th>Number of PE blocks</th>
<th>Time horizon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data set 1</td>
<td>459</td>
<td>308</td>
<td>19</td>
<td>144</td>
<td>174 days</td>
</tr>
<tr>
<td>Data set 2</td>
<td>506</td>
<td>255</td>
<td>17</td>
<td>138</td>
<td>174 days</td>
</tr>
</tbody>
</table>

There exist seven bays in the shipyard. Grand assembly blocks can be assembled in all bays except bay 7 and unit assembly blocks can be assembled in bay 6 and bay 7. Sub-assembly blocks are only assembled in bay 7.

The values of parameters used in our tests are as follows.

- space utilization ratio $\sigma = 80\%$
- initial maximum man-power utilization ratio $R_M = 120\%$
- maximum PE interval $\beta = 10$ days
- difference between SSD and ESD of grand assembly blocks $\alpha = 12$ days
- ER/DB starting time interval $\delta = 4$ days
- decrement in man-power utilization ratio $\lambda = 3\%$

The tests were run on a Pentium PC (800MHz) and we used ILOG Solver 5.0 and ILOG Scheduler 5.0 as CSP engines[8]. We set the time limit as 20 minutes for each search to find a feasible solution. Test results are shown in Table 4 and Table 5.

Table 4 Results for data set 1

<table>
<thead>
<tr>
<th>Search</th>
<th>$\max_{} R_M$ (%)</th>
<th>Number of failure blocks</th>
<th>Running time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search 1</td>
<td>119</td>
<td>52</td>
<td>9.72</td>
</tr>
<tr>
<td>Search 2</td>
<td>114</td>
<td>44</td>
<td>9.68</td>
</tr>
<tr>
<td>Search 3</td>
<td>109</td>
<td>43</td>
<td>9.71</td>
</tr>
<tr>
<td>Search 4</td>
<td>failed to find a feasible solution within the time limit</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 5 Results for data set 2

<table>
<thead>
<tr>
<th></th>
<th>max $R_m$ (%)</th>
<th>Number of failure blocks</th>
<th>Running time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search 1</td>
<td>115</td>
<td>57</td>
<td>9.55</td>
</tr>
<tr>
<td>Search 2</td>
<td>111</td>
<td>57</td>
<td>11.20</td>
</tr>
<tr>
<td>Search 3</td>
<td>107</td>
<td>48</td>
<td>9.49</td>
</tr>
<tr>
<td>Search 4</td>
<td>failed to find a feasible solution within the time limit</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In those tables, we notice that the algorithm finds a feasible solution quickly if it ever can find one. But, if the algorithm fails to find a feasible solution, it could not verify that a feasible solution does not exist even for a long time. We think that this might happen because of the enormous size of the search space inherent to this scheduling problem. The schedules obtained by the algorithm were examined by a scheduler of the shipbuilding company and he regarded the results satisfactory.

We also compared our results with a schedule actually implemented earlier in the company. The work load of the schedule is comparable to our test problems but the planning horizon is shorter. Since we could only obtain information of the work load in each bay over the planning horizon, we could not apply our algorithm to obtain a new schedule. Instead we examined whether our algorithm is effective in reducing the daily fluctuations in work load in each bay. For this purpose, we computed the absolute value of the difference in work load for day $t$ and day $t-1$ for all days in the planning horizon. We then computed the mean and standard deviation of those values for the schedules obtained from data set 1, data set 2, and the actual schedule. The results are shown in Table 6.
### Table 6 Comparison of fluctuations in work loads

<table>
<thead>
<tr>
<th></th>
<th>mean of fluctuations for actual schedule</th>
<th>mean of fluctuations for data set 1</th>
<th>mean of fluctuations for data set 1</th>
<th>standard deviation for actual schedule</th>
<th>standard deviation for data set 1</th>
<th>standard deviation for data set 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>bay 1</td>
<td>15</td>
<td>8.62</td>
<td>11.56</td>
<td>12.23</td>
<td>10.65</td>
<td>12.19</td>
</tr>
<tr>
<td>bay 2</td>
<td>15.11</td>
<td>8.13</td>
<td>10.19</td>
<td>16.55</td>
<td>12.78</td>
<td>12.51</td>
</tr>
<tr>
<td>bay 3</td>
<td>18.28</td>
<td>9.23</td>
<td>13.65</td>
<td>13.08</td>
<td>10.15</td>
<td>17.29</td>
</tr>
<tr>
<td>bay 4</td>
<td>22.89</td>
<td>7.32</td>
<td>6.19</td>
<td>42.82</td>
<td>11.84</td>
<td>9.6</td>
</tr>
<tr>
<td>bay 5</td>
<td>27.11</td>
<td>9.08</td>
<td>7.95</td>
<td>50.72</td>
<td>10.75</td>
<td>8.85</td>
</tr>
<tr>
<td>bay 6</td>
<td>16</td>
<td>6.01</td>
<td>5.93</td>
<td>29.93</td>
<td>6.16</td>
<td>8.37</td>
</tr>
<tr>
<td>bay 7</td>
<td>17.11</td>
<td>4.59</td>
<td>5.68</td>
<td>32.01</td>
<td>5.93</td>
<td>7.82</td>
</tr>
<tr>
<td>Total work load</td>
<td>114.61</td>
<td>19.78</td>
<td>19.09</td>
<td>143.89</td>
<td>20.01</td>
<td>15.95</td>
</tr>
</tbody>
</table>

In table 6, the last row shows the results when we compute the values considering the work loads over all bays in each day as the total work load of the shipyard. We can observe that the daily fluctuations in work load are significantly reduced for the schedules obtained by our algorithm, although the comparisons are not made for the same data sets.

### 6. Concluding remarks

This paper focused on the scheduling of block assembly process using CSP. We proposed an algorithm to generate a schedule that satisfies many complicated constraints. We considered the objectives of minimizing the number of failure blocks and balancing the work loads for man-power. However, other objectives pertinent to a particular shipyard can be incorporated in the model similarly. We used a sub-function recursively in the proposed algorithm until the values of all variables are determined. Our algorithm also includes the feature to find a schedule with small fluctuations in work loads. Computational results on real problems show that our algorithm can find good schedules in a short time.

Our results show the potential of using CSP for complicated scheduling problems. In real scheduling problems, there are many complicated constraints which need to be observed in a schedule. CSP provides more freedom to model those constraints than mathematical models or heuristic approaches. Another advantage of CSP approach is the relative easiness in
modifying the model when situations are changed in the shipyard. Usually, a scheduling system needs to be modified frequently to reflect the varying conditions in the workplace, which is not an easy task. For CSP approach, it is relatively easy to change the model since the search procedure and the representation of constraints are considered separately.

References
