A JIT Algorithm for Offshore Rig Assembly Sequencing

Ernest Abbott[1] and David Chua[2]

Abstract

Lean principles in production are synonymous with the Toyota motor company, probably the most productive and innovative car manufacturer in the world. However, in a construction environment Lean principles are not so easy or obvious. The major difference between a production and a construction environment is the synchronicity of the work and the longer time horizons. In the production environment the product stages are often short, whereas in the construction environment they are long.

Constructing offshore rigs is a long process involving many stages where the correct assembly sequence is paramount. Offshore rigs are composed of units, referred to a blocks. The building of the block involves sub-blocks; each sub-block differs in size, shape and construction time. At the block level, with more than one team of workers constructing the sub-blocks, it is possible to apply Lean principles to the sub-block work so that they arrive at the block assembly point with the minimum of waiting time.

A number of Greedy based sequencing algorithms have been developed by way of a compare and contrast. The Greedy sequencing algorithm augmented with back-tracking coupled with post schedule allocation reordering has proved to be a good one within the bounds of the random nature of the data.

Keywords: Assembly sequence, lean construction, offshore rig, sequencing algorithms.

Introduction

Lean principles in production have been successfully applied in a number of industries, most notably the car industry and consumer electronics industry. Toyota has been credited with developing the modern Lean production approach. Toyota identified seven areas of waste in manufacturing, overproduction, defective product, high inventories, non-value action, unnecessary or superfluous processes, transportation of materials and idle time for material and people (Lang et al., 2001). These concepts of Lean production have permeated other sections of industry. This present work is to apply, as far as possible, lean principles to the construction of off-shore rigs, and in the process derive a just-in-time algorithm.

Lean production has an emphasis on efficiency with the aim of eliminating waste at all stages of the production process. The aim of Lean production is to add value to the produce at every stage of its process. Waste, in Lean manufacturing, increases the time of production or increases the cost of production, but does not add value to of the product (Liker & Lamb, 2002). Lean production is not an end in itself (Womack, James. P. & Jones, 1994), but part of a process that stretches the full length of the value chain.

One characteristic of Lean production is the continuous supply of parts; this eliminates time wasted in waiting for parts to arrive. This continuous supply is better known as Just-in-Time. This approach differs from the traditional method of supplying parts in a

[1] Research Engineer, Dept. of Civil & Environmental Engineering, National University of Singapore, Block E1A #07-03, No.1 Engineering Drive 2, Singapore, 117576, (65) 65164643, FAX (65) 67791635, Email: ceeaels@nus.edu.sg

[2] Associate Professor, Dept. of Civil & Environmental Engineering, National University of Singapore, Block E1A #07-03, No.1 Engineering Drive 2, Singapore, 117576, (65) 65162295, FAX (65) 67791635, Email: ceedavid@nus.edu.sg

continuous production process, where parts are produced at one stage of the process for the next, irrespective of the requirement of the next stage. This is called a *push* system and leads to high levels of inventory (Womack, J. P. et al., 1991).

Lean principles, in part, have been applied in the Japanese shipbuilding industry leading to an improvement in productivity of 150% in the 30 years from 1965 to 1995 (Liker & Lamb, 2002). Liker and Lamb do advocate organizing a shipyard by "product line". A similar approach has been adopted by Lang et al. (2001) in a theoretical simulation of a Lean shipyard.

In the construction of offshore rigs, the Lean principle of continuous supply, along the lines of those identified by Liker and Lamb (2002), are not easily implemented. In shipbuilding blocks are standard, whereas for offshore rigs, block repeatability is at the best rare and often unique in construction. However, in the final assembly of the block, some progress can be made towards reducing waste in terms of waiting time and double handling of sub-assemblies.

Block Assembly

Liu et al. (2011) recognizes the inherent challenges of having long production lead times for a product, such as blocks for Offshore rigs. They use aggregate production planning in the area of workforce levelling as well as inventory usage. They achieve their objective using a multi-objective genetic algorithm. Their work deals with smoothing the work flow of block construction, and as with Liker and Lamb (2002), their objective is to construct a grand block just-in-time. The work of the present research is to deal, not at the macro level as is being done by Liu et al. (2011) and Liker and Lamb (2002) but with the micro level of the sub-assembly construction so that it arrives at the final assembly point just-in-time, or close to just-in-time.

To have a better grasp of the issues involved a brief description of the block parts and construction is given. The Offshore rigs are constructed of units called *blocks*. The blocks are composed of a number of different part types: *plates*, *brackets*, *stiffeners*, *flanges* and *bulkheads*. The plate is the foundation for the other parts. A plate with one or more other parts welded to it is called a sub-assembly. Two or more sub-assemblies welded together is called a sub-blocks. Sub-blocks (& sub-assemblies) are welded together to form a block.

The first stage of building a block is to create a base panel from the various plates. These panels are cut out from a much larger sheet of steel using a laser cutter. The laser cutting process is fast compared with welding of the seams, so does not present a problem in ensuring that the plates arrive just-in-time at the location where they are welded together in a processes known as union melt.(see Figure 1).

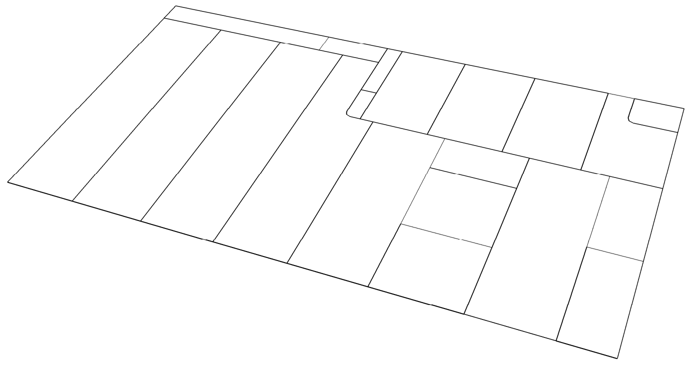

Figure 1. Creating a Base Panel

The next step in the construction process is the welding of stiffeners to the base panel. Stiffeners are special steel profiles sourced from an outside supplier which are cut to the required length in the shipyard. The shipyard has amply stock of these standard parts. As the only process required here is the cutting to length, there is no problem in ensuring that these arrive at the point of requirement just-in-time.

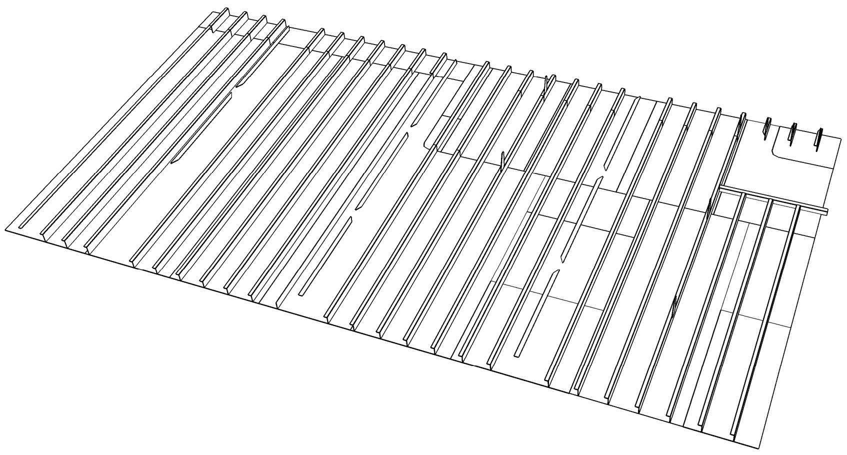
Figure 2. Stiffeners Add to Base Panel

The next stage of the construction process is the assembling of the various sub-assemblies. The order of assembly has already been determined, for example Abbott and Chua (2013) for an outline on how this is achieved. The welding of the sub-blocks to the base panel is in two parts. The sub-assemblies are fitted and tack welded into place. The final welding only takes place after all sub-blocks have been fitted and tack-welded into place. The fitting sequence is not the same as the welding sequence. The fitting sequence of a sub-block and/or sub-assembly is determined by a number of factors, size, weight, position within the block, being some of them. (The actual details are beyond the scope of this paper.) The final welding is from the centre of the block outwards. This is done so as to minimize the distortion caused by the welding process.

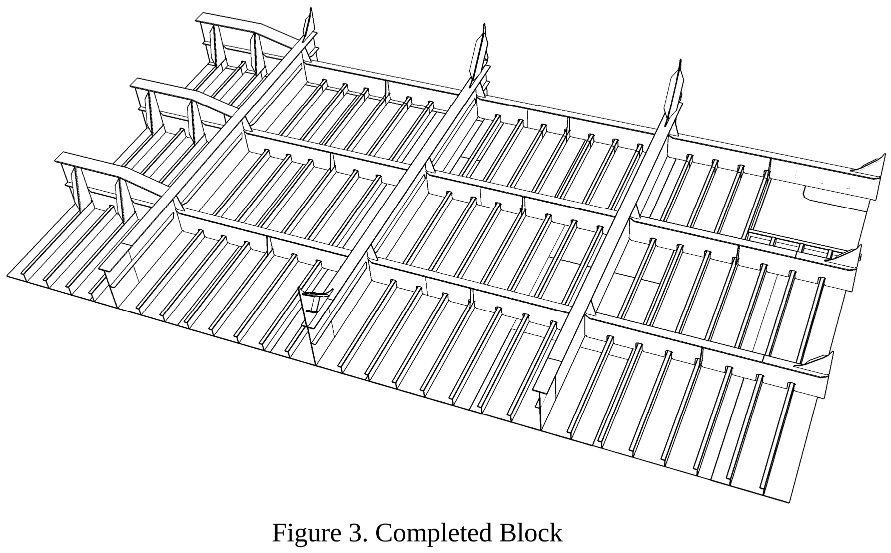
Figure 3. Completed Block

The work content of each sub-block/sub-assembly differs, yet they have to arrive in a particular predetermined assembly sequence. This research is concerned in developing some algorithms with a number of objectives, to minimize the makespan, to minimize the waiting time at the final assembly point and to minimize double handing of parts.

Block Assembly Sequence and Just-in-Time

The basic requirement for a just-in-time block assembly is that the sub-block/sub-assembly arrives at the assembly point the moment they are required. The construction time of the sub-block/sub-assembly is considerably longer than the assembly time. The fitting and tack welding of a sub-block in its final location takes from 30 to 60 minutes. Even if this is extended, it is multiple times faster than the construction of the sub-block/sub-assembly. A sub-block/sub-assembly can take several hours to several days to complete.

The construction of a sub-block/sub-assembly is the task of a *team*, which usually consists of a couple of workers. There are a number of reasons for this: the size of the component being constructed and safety during the welding process. With a limited number of teams and a large number of sub-blocks/sub-assemblies all of which have different construction times, it is impossible to achieve just-in-time assembly.

A computer simulation was performed to demonstrate this point. The simulation consist a number of sub-blocks ranging from 10 to 300. The construction time for each sub-block was randomly generated in the range 2 to 30 time units. The longest construction time was taken as the limiting value. All other sub-blocks were grouped so that no group of sub-blocks had a total construction time greater than the longest sub-block construction time. This grouping does not take into account the required assembly sequence, since the purpose of the simulation is to get the minimum number of teams for JIT assembly. For each set of sub-block the process was repeated 60 times, this should remove any ‘freak’ values that could occur from using the randomly generated values. The average of the 60 repetitions was recorded; the results are in **Table 1**.

Table 1. Simulation Results for Minimum Number of Teams for JIT Construction

No. of Sub-Blocks	10	20	30	40	50	60	70	80	90	100
Avg. No. Teams	6.93	12.73	18.43	24.30	29.22	34.63	40.80	46.23	51.23	57.92
Ratio	0.69	0.64	0.61	0.61	0.58	0.58	0.57	0.58	0.57	0.58
No. of Sub-Blocks	110	120	130	140	150	160	170	180	190	200
Avg. No. Teams	62.88	67.92	73.40	78.57	83.93	89.27	94.62	100.15	106.25	111.52
Ratio	0.57	0.57	0.56	0.56	0.56	0.56	0.56	0.56	0.56	0.56
No. of Sub-Blocks	210	220	230	240	250	260	270	280	290	300
Avg. No. Teams	117.62	122.00	127.92	133.38	139.20	143.28	150.07	155.18	160.87	166.08
Ratio	0.56	0.55	0.56	0.56	0.56	0.55	0.56	0.55	0.55	0.55

The simulation results demonstrate that with a small number of teams JIT production is not achievable. However, a measure of JIT assembly is possible; it is that it will not be 100% JIT.

Developing the Algorithm[3]

As has been mentioned, the task of constructing a sub-block/sub-assembly is assigned to a *team*. This naturally leads to scheduling the work among the teams treating each team as parallel machine. For this research each team is considered to possess an identical skill set.

[3] The notation used here is that of Graham et al. (1979) which is now the standard notation

The work is carried out without pre-emption. Each sub-assembly/sub-block is completed before work begins on the next.

It has been pointed out (Blazewicz et al., 1983; Chen et al., 1988; Karger et al., 2010) that scheduling with parallel identical machines without pre-emption is an *NP*-hard problem. It has been suggested (Mosheiov, 2001), however, that flow-time minimization on parallel identical machines, albeit only two parallel identical machines, is polynomial solvable. The solution is far too problem-specific for this research, since it posits a learning effect, which may be possible in construction environments that deal with repetitious units, which is not the case in the construction of the offshore rig. Further, the computation resource required to resolve this for only two parallel machines is $O(n^4)$. The focus here will be on an algorithmic approach. The problem under investigation in this research is quite constrained in that there is a predefined output sequence; hence the choice of work allocation is more limited which makes it possible to use an algorithm to resolve the problem. Further, this is not really a shop-flow issue, as the various sub-blocks are confined to one process, as it were, on any ‘machine’, and then is ready for assembly.

The most obvious method is the use a Greedy algorithm to allocate work among the teams. Other algorithms used for identical parallel machines in a machine shop environment, such as *Longest Processing Time First* (LPT), are not suitable here as this algorithm is more focussed on the minimization of the *makespan*. For LPT the completion order of the jobs is of no relevance, however, in the assembly of sub-assemblies for blocks for Off-shore rigs the order the jobs are completed in is of great importance.

Each algorithm that is developed depends on the objective, although one objective is to adopt Lean principles, minimizing the makespan for the schedule is clearly an important objective.

Makespan is defined as $C^S_{\max} = \max_j C^S_j$ of a schedule *S* to be the maximum completion time of any job in *S*, where C^S_j is the completion time of job *j* in schedule *S*. For *m* machines where the processing time for job j is p_j, then $C^*_{\max} \geq \sum_{j=1}^{n} \frac{p_j}{m}$ where *n* is the number of jobs, $C^*_{\max}$ is the makespan of the optimal schedule and $C^*_{\max} \geq p_j$ for all jobs *j*. That is makespan for the whole assembly process cannot be less than the sum of the average processing time for all the individual jobs.

A greedy algorithm for identical parallel machines assigns the next available job to the next free machine. The sub-blocks are ordered in their required assembly sequence. **Figure 4** and **Figure 5** illustrate the process. Job 10 is successively assigned to each machine and the completion time, C^S_{10}, of the schedule is calculated. The machine with $\min C^S_{10}$ is the one to which the job is assigned. **Figure 5** illustrates the machine that job 10 is finally assigned to.

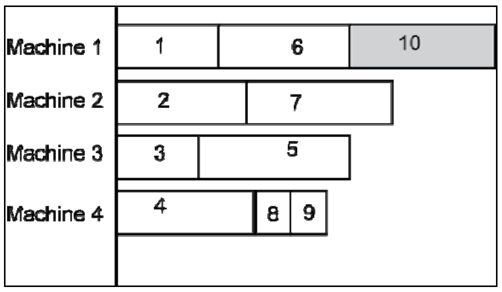

Figure 4. Test Assignment of Job

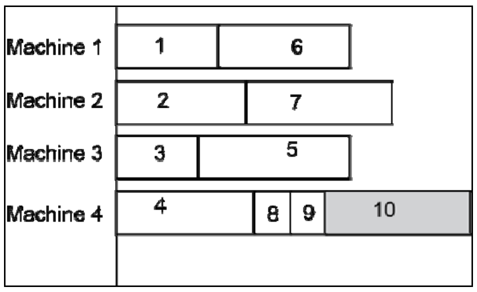

Figure 5. Final Assignment of Job

More formally, the ith job is added to the machine with the minimum completion time, i.e. the ith job is added to $\min C^m$, where C^m is the completion time for machine m. Since the ith job is added in this wall, the overall all schedule is always going to be a minimum.

There is a possibility of improving the algorithm by applying a back tracking technique. With back tracking the job is added to the minimum machine schedule, provided that when it is added to the minimum machine schedule the maximum schedule is not exceeded. If the maximum schedule is exceeded the job is swapped with the previously added job, provided the overall schedule is now earlier than when the new job was added to the minimum machine schedule.

If i jobs have been added to the schedule, using the greedy algorithm, the ith job was added to the machine with a completion time earlier than $C_i - p_i$. Similarly for the $i+1$th job it is added to the machine with a completion time earlier than $C_{i+1} - p_{i+1}$.

When the ith job was added to the schedule, it was added to the machine with the shortest completion time, $\min C^m$. This machine is chosen irrespective of the work content of job i.

The $i+1$th job is added to the schedule. If $C_{i+1} \leq C^*_{\max}$ the addition of the job to the assigned machine is accepted. There is no change in the expected waiting time between the completion of job i and job $i+1$, i.e. $C_{i+1} - C_i$. If, however, $C_{i+1} > C^*_{\max}$ then back tracking is tested with ith job being replaced by $i+1$th job in the schedule. Following the replacement of the ith job by the $i+1$th job, the ith is added to the schedule, which will be a different machine, provided that $C_i^{'} \leq C_{i+1}$ where $C_i^{'}$ is the revised completion time for job i, otherwise job $i+1$ is added as per the greedy algorithm.

The waiting time, if the ith and $i+1$th jobs are not swapped is given by

$$\begin{aligned} W_i &= C_{i+1} - C_i \text{ if } C_{i+1} > C_i \\ &= 0 \text{ if } C_{i+1} \leq C_i \end{aligned}$$

The waiting time, if the ith and $i+1$th jobs are swapped is given by

$$\begin{aligned} W_i^{'} &= C_{i+1} - C_i^{'} \text{ if } C_{i+1} > C_i^{'} \\ &= 0 \text{ if } C_{i+1} \leq C_i^{'} \end{aligned}$$

When swapping ith and $i+1$th jobs, the condition for the swap was that $C_i^{'} \leq C_{i+1}$. The waiting time is zero time units. Without the swap the waiting time is $C_{i+1} - C_i \geq 0$.

This is illustrated in Figure 6 and Figure 7. There are 10 sub-assemblies P1 to P10, with varying construction time, shown in parenthesis. Sub-assembly P1 has a construction time of 10 days. In the greedy algorithm the completion is at day 49, the total work is 91 days. In Figure 7, sub-assembly P9 is constructed by team 1. This leads to a completion on day 46, thus a saving of 3 days over the normal greedy schedule.

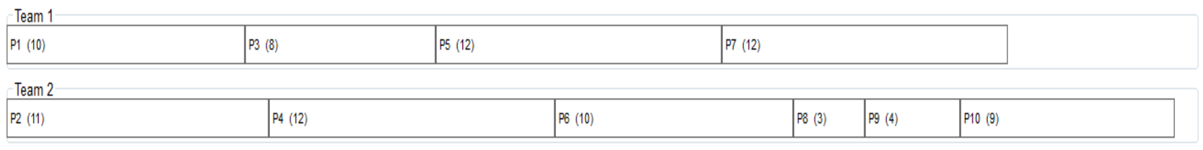

Figure 6. Results of Greedy Algorithm

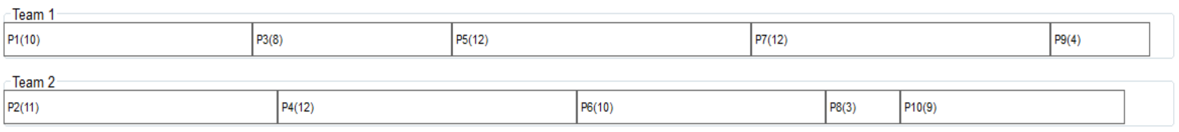

Figure 7. Results of Greedy Algorithm with Backtracking

Backtracking with post sequence reordering is when all the jobs have been allocated to the machines. The completion time of each job is compared to its required time. Job *i+1* is required after the job *i* has been assembled, but its construction completion may be earlier. **Figure 8** and **Figure 9** serve as an illustration. Examining team 2, it can be seen that jobs P2, P3 and P4 will be completed before job P1 is completed. This leads to a problem of double handling. Jobs that are completed ahead of their required assembly time are placed in a stack. In this example the stack would consist of jobs P2, P3 and P4. After P1 is completed, jobs P3 and P4 would have to be removed from the stack in order to retrieve job P2. This is inefficient work where the jobs are double handled. If job P3 is placed in a new stack and job P4 is placed on top of it, the same situation arises when job P3 is required.

To remove this double handing of jobs in line with lean principles, the sequence of construction for jobs P2, P3 and P4, is reordered, as may be seen in **Figure 9**. With this reordering, the stack now has job P2 on the top, so there is no double handling of previously completed jobs when P1 is completed and P2 is required.

It needs to be noted that this reordering of the queue does not change the makespan for the jobs; rather it improves the work flow and reduces the overhead of double handling of the jobs.

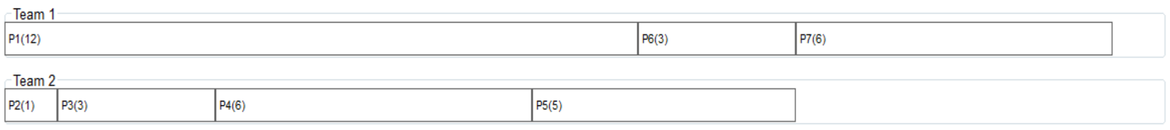

Figure 8. Greedy Algorithm with Backtracking

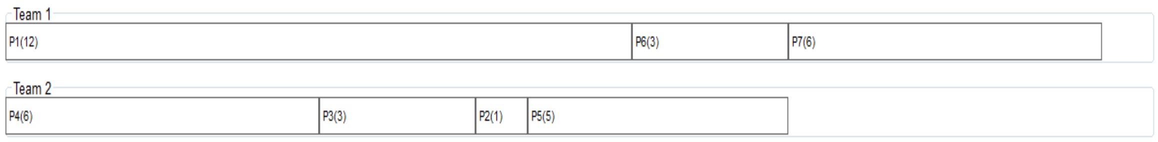

Figure 9. Greedy Algorithm with Backtracking and Reordering

Each job has a start time, construction duration and required time. The start time and construction duration need no explanation. The time a particular job is required is defined as when the job is required at the assembly point. As here lean principles are being applied, the job is required at the same time the previous job is required with the addition of the fitting time. Since the construction time for a sub-assembly is comparatively long compared with the fitting time at the assembly point, for scheduling purposes the fitting time is ignored, hence the required for $i+1^{th}$ job is deemed to be at the finish time of job i^{th}.

Figure 8 shows P1 finishing at time 12. P2 finishes at time 1, but is not required until time 12; the same is true for Jobs P3 and P4. The jobs in team are ordered so that those with the same requirement time are order in reverse required assembly sequence. **Figure 9** shows the sequence of P2, P3, and P4 reordered as P4, P3, and P2.

An assembly has 2 time values: its completion time (CT) and its required time (RT). The completion time is the point the assembly is completed, whereas the required time is the point at which the assembly is required to be just-in-time.

For a part *i*, if $RT_i > CT_i$ then the part is early. If $RT_i=CT_i$, the part is on time. $RT_i< CT_i$ the part is late.

The sub-blocks are order into their required assembly sequence. Then the algorithm is processed:-

```
Begin:
   For $i = 1$ to $n$
      If $RT_{i+1} \leq CT_i$
         set $RT_{i+1} = CT_i$
   Endfor
End
```

The sub-assembly is now ordered as follows:

Team Number - Ascending
Required Time – Ascending
Original required sequence – Descending

Testing the Algorithm

To test the algorithm four computer models were developed in C#, the sub-block construction times were randomised using a seed of 42, with range of 1 to 8 days. The models were (a) greedy algorithm, (b) greedy algorithm with post reordering, (c) greedy algorithm with backtracking only and (d) greedy algorithm with backtracking and post reordering.

The following tables all have 30 items with a total work time of 99 days. The results are shown for 2, 3, and 4 teams working on the construction.

Table 2. Greedy Algorithm Only

# Teams	# in Stack	Max in Stack	Total Stack Time	Avg. Stack Time	JIT Items	% JIT Items	Total Waiting Time	Avg. Waiting Time	Sequence Changes	Optimum Makespan	Makespan
2	7	2	12	1.71	11	36.67	45	1.5	0	49.5	50
3	13	4	28	2.15	16	53.53	30	1.0	0	33	35
4	15	6	37	2.47	21	70.00	22	0.73	0	24.75	27

Table 3. Greedy with Post Reordering

# Teams	# in Stack	Max in Stack	Total Stack Time	Avg. Stack Time	JIT Items	% JIT Items	Total Waiting Time	Avg. Waiting Time	Sequence Changes	Optimum Makespan	Makespan
2	7	2	15	2.14	17	56.67	45	1.5	4	49.5	50
3	13	2	32	2.46	21	70.00	30	1.0	4	33	35
4	15	1	35	2.33	24	80.80	22	0.73	7	24.75	27

Table 4. Greedy with Backtracking Only

# Teams	# in Stack	Max in Stack	Total Stack Time	Avg. Stack Time	JIT Items	% JIT Items	Total Waiting Time	Avg. Waiting Time	Sequence Changes	Optimum Makespan	Makespan
2	8	2	17	2.13	13	43.33	45	1.5	0	49.5	50
3	13	5	34	2.62	19	63.33	30	1.0	0	33	35
4	11	5	22	2	18	60.00	21	0.7	0	24.75	26

Table 5. Greedy with Backtracking and Post Reordering

# Teams	# in Stack	Max in Stack	Total Stack Time	Avg. Stack Time	JIT Items	% JIT Items	Total Waiting Time	Avg. Waiting Time	Sequence Changes	Optimum Makespan	Makespan
2	8	2	21	2.63	18	60.00	45	1.5	18	49.5	50
3	13	2	41	3.15	23	76.67	30	1.0	27	33	35
4	11	1	24	2.18	23	76.67	18	0.6	32	24.75	26

The column headings in the table have the following meaning:
#Teams = number of teams.
#in Stack = the total number of items that were ready early and place in a waiting stack
Max in stack= the maximum number of items in the stack at any one time
Total Stack Time= sum of the times all items spent in the stack.
JIT Items = items that were ready when required.
% JIT Items = the percentage of items that were JIT
Total Waiting Time = amount of time waiting for an item.
Avg. Waiting Time = average waiting time for all items .
Sequence Changes = number of items whose sequences were change
Optimum Makespan= total amount of work divided by number of teams
Makespan = number of days taken to complete all work for the team.

Analysis of Results

Table 6. Summary of % JIT Items

	Table 2		Table 3		Table 4		Table 5	
Teams	JIT Items	% JIT Items	JIT Items	% JIT Items	JIT Items	% JIT Items	JIT Items	% JIT Items
2	11	36.67	17	56.67	13	43.33	18	60.00
3	16	53.53	21	70.00	19	63.33	23	76.67
4	21	70.00	24	80.80	18	60.00	23	76.67

With this particular set of random data, it may be see that backtracking &/or post reordering has an advantage in terms of increasing the number of JIT items. Comparing **Table 2** with **Table 3** (Greedy only and Greedy with Post Reordering) there is an increase in the number of JIT items. Further, with Post Reordering, there is a reduction in wasted time since there is will not be double handling of items. Similarly, the same can be said in comparing **Table 2** and **Table 4** and **Table 2** and **Table 5** data. There is a clear increase in the JIT Items. The highest number of JIT items is when the Greed algorithm is augmented with backtracking and Post Reordering.

Conclusions

Identical parallel machine theory from scheduling has been adopted to resolve the sequencing of the construction of sub-assemblies with a limited number of work teams.

As good as the greedy algorithm is in this constrained problem, it may be improved in many circumstances with the addition of backtracking. While it has been shown that JIT may not be fully realised with a limited number of teams for work, some lean principles may be applied to the construction process by using reordering following the completion of the assignment of the construction sequences of the sub-assemblies. Although this does not improve the makespan for the schedule, it does remove the need for double handling of the sub-blocks and thus makes a step towards a better implementation of lean principles in the construction of offshore rigs.

As with all algorithms and random input, there is no guarantee that it will give an acceptable result on all occasions. The algorithms developed here may be seen as an arsenal, in that the best one for the set of data may be chosen.

References

Abbott, E., & Chua, D., K. H.;. (2013, October 23-25 2013). *The Application of Polychromatic Set Theory in the Assembly Sequencing of Blocks for Offshore Rigs.* Paper presented at the Proceedings of the 4th International Conference on Engineering, Project, and Production Management (EPPM 2013), Bangkok, Thailand.

Blazewicz, J., Lenstra, J. K., & Rinnooy Kan, A. H. G. (1983). *Scheduling subject to resource constraints: classification and complexity.* Paper presented at the Combinatorial Optimization Conference, 24-28 Aug. 1981, Netherlands.

Chen, B., Potts, C. N., & Woeginger, G. J. (1988). *A Review of Machine Schedling: Complexity, Algorithms and Approximations*: Kluwer Academic Press.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Rinnooy Kan, A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Discrete Optimisation, Aug. 1977, II*, 287-326.

Karger, D., Stein, C., & Wein, J. (2010). *Algorithms and theory of computation handbook* (M. J. Atallah & M. Blanton Eds.): Chapman & Hall.

Lang, S., Dutta, N., Hellesoy, A., Daniels, T., Liess, D., Chew, S., & Canhetti, A. (2001). *Shipbuilding and Lean Manufacturing - A Case Study*. Paper presented at the THE SOCIETY OF NAVAL ARCHITECTS AND MARINE ENGINEERS. Conference Paper retrieved from

Liker, J. K., & Lamb, T. (2002). What is lean ship construction and repair? *Journal of Ship Production, 18*(3), 121-142.

Liu, Z., Chua, D. K. H., & Yeoh, K. W. (2011). Aggregate production planning for shipbuilding with variation-inventory trade-offs. *International Journal of Production Research, 49*(20), 6249-6272. doi: 10.1080/00207543.2010.527388

Mosheiov, G. (2001). Parallel machine scheduling with a learning effect. *52*(10), 1165-1169.

Womack, J. P., Jones, D., & Roos, D. (1991). *The Machine That Changed the World: The Story of Lean Produciton*.

Womack, J. P., & Jones, D. T. (1994). From lean production to lean enterprise *Harvard Business Review 74*(Mar-Apr), 21.