Load Information Based Priority Dependant Heuristic for Manpower Scheduling Problem in Remanufacturing

Shantanab Debchoudhury[1], Debabrota Basu[1],
Kai-Zhou Gao[2], and Ponnuthurai Nagaratnam Suganthan[2]

[1] Department of Electronics and Telecommunication Engineering,
Jadavpur University, Kolkata-700032
[2] School of Electrical and Electronic Engineering, Nanyang Technological University,
Singapore- 639798
{sdch10,basudebabrota29}@gmail.com,
{kzgao,epnsugan}@ntu.edu.sg

Abstract. Disassemble scheduling in remanufacturing is an important issue in the current industrial scenario. Allocation of operators for this purpose forms a special class of manpower scheduling problem with an added layer of restrictions. In this paper we propose a set of heuristics that make use of the concept of load information utilization. Several modifications have been incorporated on the basic framework and thorough comparison has been made between the same to devise an efficient way to tackle remanufacturing scheduling problems. The developed method has been put to test on a series of test functions of varying range of difficulty. The results prove the efficiency of these heuristics to solve this scheduling problem.

1 Introduction

One of the basic processes involved in industrial applications, remanufacturing [1] mainly deals with disassembling a complex machine into smaller and simpler subparts. The method helps in identifying root problems which when addressed can help in formation of a new reassembled machine.

From an environmental point of view remanufacturing is considered as the *ultimate form of recycling* and now-a-days it is an interesting topic for researchers [2]. Even remanufacturing is practically used in several countries across the globe for remanufacturing of several products like aerospace, air-conditioning units, bakery equipments, computer and telecommunication equipment, defence equipments, robots, vending machines, motor vehicles and many more. This environmental edge of remanufacturing process [3] has inspired us to deal with this problem.

For any remanufacturing problem, operators are needed to carry out the disassembling. Hence for an efficient allocation of resources an organized scheduling algorithm is of the utmost importance. Manpower scheduling [4-6] is a particular class of such problems when the task at hand is to basically assign a set of human workers with imposed constraints. A broader method would be the classical Job shop scheduling problem [7] when more generalized machines are employed capable of performing at higher efficiencies compared to human labor.

B.K. Panigrahi et al. (Eds.): SEMCCO 2013, Part I, LNCS 8297, pp. 59–67, 2013.

Standard scheduling techniques have been overshadowed by the use of heuristics and meta-heuristics which aim to devise cost effective and efficient means to solve the issue. The latter consist primarily of algorithms like ABC [8], DE [9], GA [10] to employ a population based stochastic process seeking out the best possible combination for an optimal scheduling. Heuristics [11] on the other hand are general dedicated methods that are based on experience and learning.

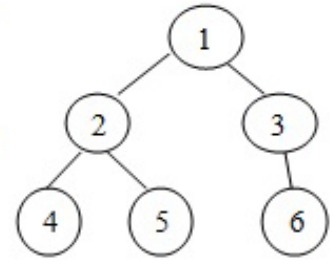
In this paper our primary aim is to utilize the concept of man power scheduling for disassembly in remanufacturing using heuristics. No such works that we are aware of are in the literature. Hence we have made use of a load information retention technique besides priority assignment. The process of disassembly is depicted through tree architecture. Thus, the problem at hand is to assign operators at each node from an available set of operators with specific processing times for working on that node. The algorithm keeps a track on the load of each operator and accordingly completes the assignment process in decreasing levels of priority.

The paper is organized as follows. Section 2 deals with discussion of the problem structure and objective. In section 3, details of the priority dependant heuristic is put forward. As basic skeleton, a primitive simple greedy heuristic with no load retention in adapted. The advantages of adding the load retention scheme is then discussed and five variations of the same are put up. In section 4, all the frameworks discussed have been put to test in some generated test cases. The experimental background is mentioned in the same section as well as the obtained results with discussions and figures. The conclusion is finally drawn in Section 5.

2 Problem Model and Objective

The process of disassembly as mentioned in Section 1 is shown using tree architecture. The root node is the main machine which is to be disassembled into subparts. At each level of disassembly a certain set of operators are employed from an available pool and each operator has a value of processing time to operate on that particular level.

Fig 1 shows a standard tree and the corresponding tree information table. As is evident in the figure node 1 or the root node is demarcated as NA since it has no parent. Thus Parent number 1 beside node number 2 shows that node number 1 is parent to node 2.



Node No	Parent
1	NA
2	1
3	1
4	2
5	2
6	3

Fig. 1. Sample Tree and corresponding Tree Information Table

We now proceed to show a sample benchmark tree information table in Table 1 that has been utilized as a test case. The problem shown is a 4-operator remanufacturing scheduling problem and has been marked as Instance 1 (Simple) in the subsequent discussions. The presence of '-' indicates that the corresponding operator is not available for operation in the particular shift.

Table 1. Sample Test Case- Instance 1(Simple)

Node	Parent	Operator 1	Operator 2	Operator 3	Operator 4
1	N.A.	-	-	-	-
2	1	-	-	15	10
3	1	5	-	10	-
4	2	6	9	-	8
5	2	-	-	7	10
6	2	-	5	6	-
7	3	-	8	-	7
8	3	11	-	8	10

Our objective here is to derive a schedule so that makespan [12] or the maximum time for the completion of the entire process is minimized. A schedule which is gives the least makespan of all the possible schedules is said to be an optimal schedule. Hence we attempt to allocate in such a way that the schedule obtained is as close to being an optimal schedule as possible.

3 Priority Dependant Heuristic

The remanufacturing model bears significant resemblance to a tree. In discussions henceforth we shall be frequently to a disassembled part to be a child of its former state whom we refer to as the parent. The level of disassembling or in other words the stage up to which the original machine has been disassembled is referred to as the depth. With these annotations in mind, we shall proceed to the assumptions and discussions of the priority dependant heuristic for operator assignment.

The assumptions used for the heuristics designed are as follows:

1. Level with lower amount of disassembled parts is considered for allocation prior to a level with higher amount of disassembled parts. In other words priority of assignment decreases as the depth of the tree increases. We refer to this as a LDHP (Lower Depth Higher Priority) assignment.
2. Children or in other words, disassembled parts of Jobs which are assigned the least time in a certain level are considered first for assignment in the next level. The job whose children are being considered remains unaffected by its own parentage. To put in simple words, process flow of all the heuristics is unidirectional along increasing depth.

3.1 Simple Greedy LDHP Heuristic without Load Information

The simple Greedy Lower Depth Higher Priority (LDHP) algorithm is a primitive attempt to solve remanufacturing scheduling problem. This does not involve any prior

load information of the operators and hence this is not an adaptive algorithm. The greedy based selection of operators is utilized when assignment within a same level of priority is under question. In such a scenario, only that job is taken up which permits allocation of operator with the least possible operating time. Hence it is termed as a greedy selection. The entire process flow of the heuristic is outlined in the Fig 2.

Step 1: Go to the first level of disassembling.
Step 2: Prioritize this level and process all the children of this level first
Step 3: All the children are of the same priority since they are all derived from same parent node. Within this priority level, go for greedy selection.

- For each child find minimum processing time possible among all operators capable of operating on that child.
- Order the obtained set or minima.
- Assign the child with least minimum the highest priority for assignment and that with the highest the last priority. If there are multiple similar minima, priority is assigned randomly between them.
- Choose operators in order of decreasing priority.
- In a priority level always choose the operator with minimum processing time.

Step 4: Update the finishing time of operation of each child in the level.
Step 5: On completion, rank the finishing time of operation in increasing order.
Step 6: Go to next level. The children of the job with the least rank (lowest finishing time) as obtained in Step 5 is given highest priority in this level and the priority decreases with increasing rank. Between two similar lower finishing times choice is made randomly.
Step 7: The higher priority set of children are considered first
Step 8: Repeat Steps 3-5 until all the priority levels within this depth are covered
Step 9: Proceed similarly for all the depths

Fig. 2. Process flow for the Simple Heuristic: (Greedy LDHP algorithm)

3.2 Modified Load Information Based LDHP Heuristic with Five Variations

Load Information Retention involves using an adaptive process to determine the existing load on a particular operator. The existing load is added to the offered load in each level of assignment to get a complete picture of the existing scenario, which we refer to here as the Complete Information Schedule (CIS). Real world problems often deal with similar situations when scheduling constraints often demand exclusion of an operator which is already on a certain amount of load. The total time of processing can be taken as a suitable measure of the load and hence this concept is used.

The algorithmic variations from the simple heuristic discussed in Section 3.1 are highlighted in Step 3 of Fig 3. The method of assigning operators in a same priority situation is done using one of five methods discussed subsequently in Fig 4, Fig 5, Fig 6, Fig 7 and Fig 8. The methods or variations deal with choosing between same

priorities and the modes of selecting constitute the basis of the variations. These modes are random, greedy, size, greedy and size, and finally a random pick between the above mentioned four modes.

Step 1: Go to the first level of disassembling. Set Initial load of all the operators to 0.

Step 2: Prioritize this level and process all the children of this level first

Step 3: All the children are of the same priority since they are all derived from same parent node.

- Add the existing load to the corresponding processing times of operators within this set of children and obtain a Complete Information schedule (CIS)
- Consult Methods 1, 2, 3, 4, 5 (Fig 4 – Fig 8) to assign priority based on this CIS.
- Choose operators in order of decreasing priority.

Step 4: Update the finishing time of operation of each child in the level.

Step 5: Update the existing load by adding the finishing time of each operator to their corresponding entries in the list of load.

Step 5: On completion, rank the finishing time of operation in increasing order.

Step 6: Go to next level. The children of the job with the least rank (lowest finishing time) as obtained in Step 5 is given highest priority in this level and the priority decreases with increasing rank. Between two similar lower finishing times choice is made randomly.

Step 7: The higher priority set of children are considered first

Step 8: Repeat Steps 3-5 until all the priority levels within this depth are covered

Step 9: Proceed similarly for all the depth.

Fig. 3. Process flow for the Load Information based LDHP heuristic

Step 1: From the CIS, find a random order of the children in the priority level

Step 2: Assign the first member of this order the highest priority and the last member the least priority.

Fig. 4. Method 1: (R-SP) [Random based Priority assignment in Same Priority level]

Step 1: From the CIS, rank the summed processing times (processing time + existing load) in increasing order of minima of the children. For multiple children with same minimum choose randomly.

Step 2: Assign the first member of this rank the highest priority and the last member the least priority.

Fig. 5. Method 2: (G-SP) [Greedy based Priority assignment in Same Priority level]

Step 1: From the CIS, rank the size list of operators available of operation in increasing order. For multiple children with same size choose randomly.
Step 2: Assign the first member of this rank the highest priority and the last member the least priority.

Fig. 6. Method 3: (S-SP) [Size based Priority assignment in Same Priority level]

Step 1: From the CIS, list the summed processing times (processing time + existing load) of the children and add the size of the corresponding list of operators available. Rank this combined minimum and size parameter in increasing order. For multiple children with same sum choose randomly.
Step 2: Assign the first member of this rank the highest priority and the last member the least priority.

Fig. 7. Method 4: (GS-SP) [Greedy and Size based Priority assignment in Same Priority level]

Choose any of the R-SP, G-SP, S-SP, GS-SP (Methods 1-4) in random manner

Fig. 8. Method 5: (RCS-SP) [Random chosen scheme based Priority assignment in Same Priority level]

4 Experimental Settings, Results and Discussions

The developed heuristics have been put to test through a series of 10 instances with wide range of difficulty. Four are relatively easy scheduling problems where number of operators is limited to a maximum of 6 and depth of disassembling is at most 4. Four cases require assignment of medium level of difficulty where depth of tree is at most 7 and number of operators involved is at most 10. The remaining two cases pose a difficult assignment challenge with involvement of 15 operators and a depth level of 10. The simple instance 1 has been shown in Table 1 in Section 2. In each of the cases, the makespan is noted besides the computational run time. Our objective is to minimize the makespan.

Table 2 and Table 3 show the results of the simple Greedy LDHP heuristic along with the Load information based LDHP algorithm with all the five methods of assignment within a same priority level. In addition two additional hybrid algorithms are devised.

In "Hybrid Pop based complete heuristic" a population pool of 5 is chosen and each is tested with Load information based assignment with one of the five methods, one after the other. The five different methods work together and the best results are reported. However in "Hybrid Pop based R-G-GS algorithm" only 3 membered populations with R-SP, G-SP and GS-SP modes of selection are used in each of the populations. Again the best results are obtained.

A total of 10 runs have been taken for each instance and each algorithm. The best results have been adopted in each case. All the codes have been run on Intel Core™ i3 machine with 2.26 GHz processor and 3GB memory on MATLAB 2012 platform.

Table 2. Results obtained under Instances 1-4

HEURISTIC		Instance 1 (Simple)	Instance 2 (Simple)	Instance 3 (Simple)	Instance 4 (Simple)
Simple Greedy LDHP Heuristic	Makespan	20	45	45	41
	Run timee(sec)	5.96e-02	6.31e-02	7.50e-02	8.01e-02
Load Info based Assignment (R-SP)	Makespan	20	35	36	41
	Run time(sec)	6.20e-02	5.25e-02	5.09e-02	9.37e-02
Load Info based Assignment (G-SP)	Makespan	20	39	29	41
	Run time(sec)	6.01e-02	6.00e-02	6.03e-02	6.87e-02
Load Info based Assignment (S-SP)	Makespan	20	39	29	41
	Run time(sec)	4.99e-02	5.88e-02	5.50e-02	6.34e-02
Load Info based Assignment (GS-SP)	Makespan	20	39	29	41
	Run time(sec)	5.24e-02	6.32e-02	5.41e-02	6.88e-02
Load Info based Assignment (RCS-SP)	Makespan	20	39	33	41
	Run time(sec)	1.18e-01	9.09e-02	5.99e-02	6.97e-02
Hybrid Pop based Complete heuristic	Makespan	20	35	29	41
	Run time(sec)	8.87e-01	7.98e-01	7.86e-01	6.63e-01
Hybrid Pop based R-G-GS heuristic	Makespan	**20**	**35**	**29**	**41**
	Run time(sec)	4.12e-01	3.99e-01	3.32e-01	2.19e-01

Table 3. Results obtained under Instances 5-10

HEURISTIC		Instance 5 (Medium)	Instance 6 (Medium)	Instance 7 (Medium)	Instance 8 (Medium)	Instance 9 (Hard)	Instance 10 (Hard)
Simple Greedy LDHP Heuristic	Makespan	135	110	116	114	198	189
	Run time(sec)	9.33e-02	1.23e-01	1.11e-01	1.11e-01	1.77e-01	1.89e-01
Load Info based Assignment (R-SP)	Makespan	101	83	85	83	128	157
	Run time(sec)	1.17e-01	1.23e-01	8.27e-02	9.66e-02	1.58e-01	1.97e—01
Load Info based Assignment (G-SP)	Makespan	100	86	86	87	125	157
	Run time(sec)	9.93e-02	1.07e-01	9.72e-02	9.28e-02	1.67e-01	1.75e-01
Load Info based Assignment (S-SP)	Makespan	104	77	91	89	126	158
	Run time(sec)	3.09e-01	1.26e-01	9.88e-02	9.11e-02	1.69e-01	1.75e-01
Load Info based Assignment (GS-SP)	Makespan	101	77	82	82	129	161
	Run time(sec)	1.10e-01	1.09e-01	1.03e-01	6.89e-02	2.10e-01	1.50e-01
Load Info based Assignment(RCS-SP)	Makespan	105	81	87	86	126	160
	Run time(sec)	1.12e-01	1.10e-01	1.26e-01	1.09e-01	1.71e-01	1.93e-01
Hybrid Pop based Complete heuristic	Makespan	100	77	82	82	125	157
	Run time(sec)	1.12e+00	9.88e-01	7.97e-01	8.55e-01	1.03e+00	1.63e+00
Hybrid Pop based R-G-GS heuristic	Makespan	**100**	**77**	**82**	**82**	**125**	**157**
	Run time(sec)	5.41e-01	4.13e-01	3.32e-01	3.11e-01	6.98e-01	7.01e-01

The above results show that the simple greedy heuristic is basically a primitive model which fails to address the necessities of a minimal makespan. A marked improvements in the results is obtained when we include the load information retention mechanism. Altough all the five variations of this Load information based heuristic are remarkably superior to their primitive skeleteon – the greedy simple heuristic, it is however seen that on comparison between themselves slight differences arise. In some instances (Instance 2) R-SP mode works the best while on others G-SP (Instance 9) or in some cases (Instance 7) GS-SP outperforms the other heuristcis.

Hence hybrid algorithms were devised. The hybrid pop based complete heuristic manages to achieve minimum makespan on all the cases albeit at the cost of a much increased computational time as compared to the others. The hybrid pop based R-G-GS heuristic is finally seen to be an optimal algorithm since it gives the nimum makespan with a much lesser computational time as compared to the other hybrid algorithm. The Gnatt Chart [13, 14] is depicted below in Fig 9 for the best solution obtained in case of Simple Instance 1. The horizontal axis shows time units elapsed while vertical axis shows node number on which operation takes place.

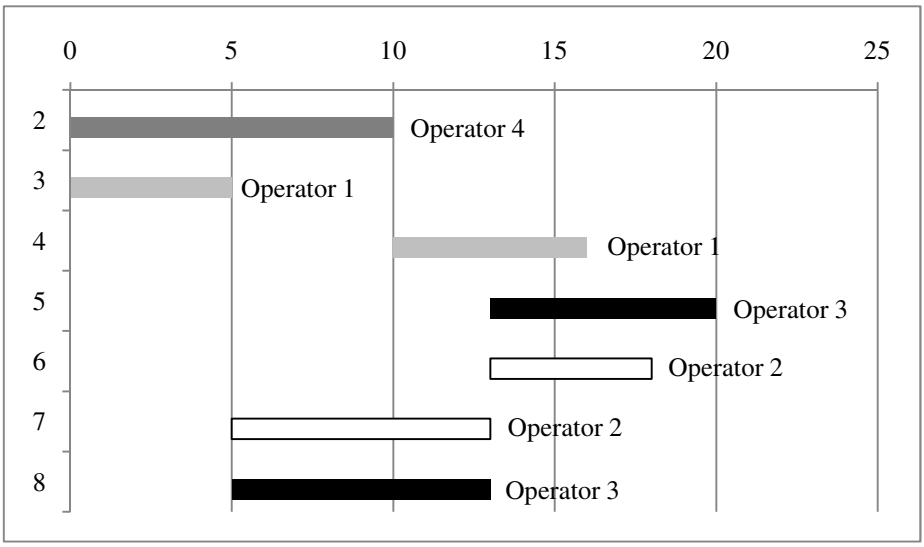


Fig. 9. Gnatt chart corresponding to solution of Table 1

5 Conclusions

Before concluding our discussion we quickly recapitulate the major aspects of the heuristic proposed in our paper

First and foremost, the load information scheme is shown to be a fast performing and highly dependent method of obtaining scheduling solutions. The heuristic is derived from a grass root level and it has been shown that incorporation of load information retention scheme improves its performance to a great extent. The modifications or variations involved in the heuristic are all comparable in terms of performance although certain variations work well in some instances. Hybrid variations thus devised were seen to be highly efficient although compromises were made with the algorithmic run time. Summing it all up, such a proposed heuristic assures a solution that boasts of qualities of reliability robustness and effectiveness.

Hence the load information based priority dependant heuristic is found to be a dependable method to solve manpower scheduling based remanufacturing problems.

References

[1] Remanufacturing, link, http://en.wikipedia.org/wiki/Remanufacturing

[2] The Remanufacturing institute, link, http://reman.org/AboutReman_main.htm

[3] Remanufacturing, http://www.remanufacturing.org.uk/

[4] Lau, H.C.: On the complexity of manufacturing shift scheduling. Computers Ops. Res. 23(1), 93–102 (1996)

[5] Ho, S.C., Leung, J.M.Y.: Solving a manpower scheduling problem for airline catering using metaheuristics. European Journal of Operational Research 202, 903–921 (2010)

[6] Pan, K.-Q., Suganthan, P.N.: Solving manpower scheduling problem in manufacturing using mixed-integer programming with a two-stage heuristic algorithm. Int. J. Adv. Manuf. Technol. 46, 1229–1237 (2010)

[7] Joseph, A., Egon, B., Daniel, Z.: The Shifting Bottleneck Procedure for Job-shop Scheduling. Management Science 34(3) (March 1988) (printed in USA)

[8] Wang, L., Zhou, G., Xu, Y., Wang, S., Liu, M.: An effective artificial bee colony algorithm for the flexible jobshop scheduling problem. The International Journal of Advanced Manufacturing Technology 60(1-4), 303–315 (2012)

[9] Das, S., Suganthan, P.N.: Differential Evolution – A Survey of the state-of-the-art. IEEE Transactions on Evolutionary Computation 15(1), 4–31 (2011)

[10] Pezella, F., Morganti, G., Ciaschetti, G.: A genetic algorithm for the flexible Job-shop Scheduling Problem. Computers & Operations Research 35(10), 3202–3212 (2008)

[11] Heuristics, http://en.wikipedia.org/wiki/Heuristic

[12] Chekuri, C., Bender, M.: An efficient Approximation Algorithm for Minimizing Makespan on Uniformly Related Machines. Journal of Algorithms 41(2), 212–224 (2001)

[13] Geraldi, J., Lechter, T.: Gantt charts revisited: A critical analysis of its roots and implications to the management of projects today. International Journal of Managing Projects in Business 5(4), 578–594 (2012)

[14] Gnatt chart link, http://en.wikipedia.org/wiki/Gantt_chart