Pergamon

Computers ind. Engng Vol. 33, Nos 3–4, pp. 469–472, 1997

0360-8352/97 $17.00 + 0.00

PII: S0360-8352(97)00170-8

Enhanced Exchange Heuristic based Resource Constrained Scheduler using ARTMAP

Inkap R. Song, Taeyong Yang, Jacob Jen-Gwo Chen

Department of Industrial Engineering

University of Houston

Houston, Texas 77204-4812

ABSTRACT

The Exchange Heuristic (EH) has demonstrated superior results compared with other RCS methods in solving Resource Constrained Scheduling (RCS) problems. Selecting the most promising target constitutes the success of EH. The current version of EH highly depends on experts' intuition in selecting a target. Expert systems and Fuzzy rulebase as well as Neural Network (NN) have been considered as alternatives for human experts. Expert systems are brittle in its nature, and Fuzzy rulebase needs membership functions defined for each linguistic variable. However, these membership function can not be justified and can be very subjective. Therefore, Neural Network is employed because of its capability of learning as well as dealing with fuzzy data. Known examples are used to train the NN. Back propagation algorithm is used first, then Adaptive Resonance Theory (ART) network is employed to reduce training time since new rules come up often. Even at the end of the training the NN, we may end up with local optima or the NN which is too general to specific problems. Utilizing Genetic Algorithm (GA) will help to further refine or adapt the weights of the NN which optimizes target selection strategy for a specific problem. © 1997 Elsevier Science Ltd

KEY WORDS: Resource Constrained Scheduling, Exchange Heuristic, Target Selection Methods, Neural Network, Adaptive Resonance Theory, Training Neural Network, Genetic Algorithms

INTRODUCTION

Resource constrained scheduling (RCS) is a scheduling problem which allocates resources over time to perform a collection of tasks. It is known to be NP-complete, i.e., combinatorially explosive; therefore, a RCS problem effectively rules out the possibility of finding an optimal solution by conventional algorithms. RCS is notorious in the sense that it is not only NP-complete, but also solving the problems with general assumptions is extremely difficult. Most solution techniques found in the literature tend to solve problems with often unrealistic assumptions, and ignore multiple criteria. One of the major drawback of many algorithms is that they are mostly based on static situation rather than dynamic, i.e., there is a set of n jobs and an optimal schedule is to be found with respect to these n jobs only while in real flowshop situations, resource constraints as well as job information change frequently as the project progresses. This makes the algorithms difficult to be applied to handle these dynamic situations. For these reasons, there is a lack of papers as well as real world applications for the static flowshop problems. Significant scheduling applications have lagged behind other areas of computer science and engineering.

Yang and Ignizio [YI87] have developed a proven efficient heuristic algorithm for solving RCS problems while developing the scheduling of army battalion training. The EH has been successfully extended and implemented for project scheduling [YIS89], generalized job shop scheduling [YID89], and to specific areas such as the petrochemical industry. The EH has demonstrated superior results compared with other RCS methods including commercially available project management software packages, notably Microsoft Project [T95].

Although the current version of the EH is a powerful tool for RCS, there is a possibility of further improvement by introducing intelligence in selecting a target throughout the procedure. The mechanism which generates the most promising target in intelligent manner will help to reduce the makespan as well as execution time of the scheduler by reducing the number of iterations in Exchange Heuristic (EH) described below. Reduction of execution time will help to handle dynamic situations. The major force that enables to handle dynamic situations comes from the simple but powerful nature of the EH algorithm.

Tiger [T95] introduced multiple objective concept in previous version of EH. The scheduler is termed as Multiple Objective Exchange Heuristic (MOEH). A decision maker may have several conflicting desires. For instance, he may prefer to minimize makespan while meeting the due dates. MOEH attempts to alleviate this problem by separating activity selection from activity scheduling; consequently, many different policies can be incorporated. In MOEH, a decision maker may choose the target selection method according to his preference at the beginning of the program. The scheduler then select the target in every iteration in the way it will optimize the criterion the user chose. However, after each iteration, the configuration of the schedule will be changed, and the best selection method in one iteration might not be the best in the next iteration. Therefore, selecting the most promising target in accordance with the preferred criterion in each iteration will enhance the performance of EH. The question is which is the best policy when a configuration is given. Determining this manually is a cumbersome matter even in small size problems, and when the problem gets large, it is prohibited. The authors have experimented with several different selection methods in each iteration manually, and have discovered some rules for selecting target activity.

In order to automate this process, several options exist. An Expert System was considered first. However, including all the information is not only practically infeasible due to the number of cases, but also impossible since experts would not know all the cases. Furthermore, some rules generated by an expert might conflict by themselves. Therefore, using a classical expert system is not feasible. Besides, the expert system is brittle in its nature, so it could not cover the fuzziness in the rulebase For instance, suppose value 1-5 corresponds to small, 5-7 to medium, and 7-9 to large. Any attributes having value between 1 and 5 is considered to be small in expert system. There is no difference between 1.1 and 4.9. In order to overcome this brittleness, a Fuzzy Rulebase can be considered. In order to have fuzzy rulebase, however, a membership function for each linguistic variable, e.g., small, or medium, or large, has to be defined. But this membership function is not known and only depends on experts' opinion which could be very subjective. Therefore, a system capable of automatic generation of the membership functions as well as learning ability which can teach the system the correct rules automatically is desired. With this in mind, a Neural Network (NN) is a suitable choice for this process.

EXCHANGE HEURISTIC

1. The EH Philosophy

While initially adding activities to a schedule, the number of non-scheduled activities is numerous, and resource availability levels are high; thus, chances for satisfying resource constraints are high. As activities are added to the schedule, resource availability levels fall and satisfying constraints is more difficult. Typical scheduling heuristics assign activities from the beginning of the schedule to the end. This results in high utilization of resources near the beginning of the scheduling horizon, with utilization tailing off towards the end. The basis of the EH strategy is an attempt to alleviate resource utilization at the beginning of the schedule by judiciously selecting activities to assign later in the schedule. This increases the possibility of assigning some promising activity earlier in the schedule that may allow an improvement in the schedule. This activity is defined as the target activity. Earlier scheduling of the target activity may improve the flexibility of shifting its successors earlier in the schedule and reduce the makespan. Realization of this flexibility results in an improved schedule.

2. Exchange Heuristic Algorithm

Given an initial schedule, the EH first finds a promising activity called the TARGET activity, which is an activity that can be moved left in the schedule. That is, the TARGET must have some slack time between itself and its immediate predecessor(s). Selecting the most promising TARGET is one of the most critical decisions in the EH. TARGET selection methods will be discussed in detail throughout this paper. In order to move the TARGET to the left, the resources occupied by other activities in the slack time prior to and needed by the TARGET should be freed first. We will call the time period between the start of the slack time and the end of the TARGET as search region. In order to free the resources in the slack period, the activities scheduled to start in the search region are moved to the right. This operation is called the Right Move Operation (RMO). If there is success in freeing the resources, the TARGET is moved left. This operation is called the Left Move Operation (LMO). Then, the activities previously moved right are rearranged towards the beginning of the schedule. This step may result in the reduction of the makespan.

Target Selection Method

There are 7 single target selection methods, and 136 combination options. The 7 single options are minimum Activity duration (mAd), Maximum time to due date (Mtdd), Maximum length of time window ratio (Mltwr), Maximum slack time (Mst), Maximum number of work remaining (Mnwr), minimum Average length of work remaining (mAlwr), and Maximum resource consumption - availability ratio (Mrcra). We will call these single selection methods as $s_1,.....,s_7$. Combination options use a lexicographic rule to assign higher priorities for the contributing factors (attributes) which have higher preference. Depending on the configuration of the initial schedule, some methods work better than other methods. More importantly, after the LMO or RMO, the configuration of the schedule will be changed, and the method which works best in a previous schedule might not work well in a new schedule. Therefore, choosing the best selection method after each iteration in some intelligent manner will enhance the performance of the algorithm. The degree of priority for a candidate activity to become a target activity can be affected by 7 attributes: activity duration (Ad), time to due date (tdd), length of time window (ltw), slack time from its predecessor (st), number of works remaining after the candidate (nwr), average length of works remaining after the candidate (Alwr), resource consumption - availability ratio (rcra). We will call these attributes as $a_1,....., a_7$.

Before each iteration starts, each of the 8 attributes must be examined not only individually but also in an aggregated manner. In fact, the main reason that combination options are used in determining a target is to aggregate information from all three groups. Expressing it differently, if we have a mechanism that can reflect information about the priorities from all three groups in an aggregated manner, we do not need to consider the 136 combination options.

In previous version of EEH, the configuration of schedule for each candidate job is reported to the target selection routine. This design has two disadvantage: 1. It takes longer execution time than executing target selection routine only once for each iteration. 2. It is hard to compare the measure of effectiveness for each rule developed. This is because by measuring suitability for each of 7 options for each job separately, the difference between one option to another can be ignored. For these two reasons, the way of measuring suitability for a given configuration of schedule has been developed. How each of 7 attributes are measured is described below. Note that all the measure have to be normalized in order to generalize the training rulesets. Calculation of each attribute is shown in Table 1. First 6 rows describe the configuration of the current schedule, and the symbols in far right column represent the value described. These 7 factors can be categorized into 3 groups (except a_1): a_2 and a_3 focus on customer service (group 1), a_4, a_5, and a_6 attempt to minimize the makespan (group 2), and a_7 is concerned about meeting the resource requirements (group 3). At the very beginning of the program, an user

assigns the preference values for each group such as (1, .7, .3). These numbers will be multiplied by the normalized values of the attributes from each of 3 groups. The attribute activity duration is not multiplied since it is associated with all 3 groups, so it does not need to be scaled. The resulting 8 values will be the inputs to the NN described below. This procedure is to reflect the user's preference.

1. Activity duration is obtained as the job duration given and normalized by dividing it by the makespan of the schedule for each candidate job. The minimum of these normalized values is fed to target selection routine.
2. Time to due dates are calculated as follows. First, difference between possible latest finish time and current job finish time, and between time to earliest start and current job start time are calculated. The maximum of these two is calculated for each job (MBdAd). These values are sum up for all the successors and the same value of the job itself. These values are divided by the average of sum of MBdAd for candidate activities, and divides again by the current makespan to normalize values. Maximum of time to due dates is used since the more the job and its successors are behind (or ahead) to the schedule, the better chance of locating the job to the front of the schedule.
3. Length of time window ratio is calculated as the ratio of time window length and duration of the job for the candidate jobs. These values are normalized by maximum time window length. Maximum of time window is used since it has higher potential of moving toward the beginning.
4. Slack time is obtained as the time period between maximum finish time of the predecessor and beginning time of the job for each candidate activity. These values are normalized by being divided by average of the slack time and again divided by the makespan of the schedule. Maximum of these values is reported to the target selection routine.
5. Number of work remaining is obtained by finding all the number of work remaining of the candidate jobs and their successors. These values are normalized by being divided by average number of work remaining and by the total number of jobs. Maximum of these values is taken since having more jobs remaining can expect "chain effect".
6. Average length of work remaining measures the total length of successors. It is obtained by being divided by number of jobs remaining to calculate the average length of successor duration. It is normalized by being divided by the makespan. Minimum value of these is taken since the shorter the average length is the better chance of locating it to the front.
7. Resource consumption - availability ratio calculates the ratio of resource consumption of a candidate job and minimum availability of the resources within the slack region. This value could be negative, and the minimum of these values for each job is compared, and the maximum of these values is taken since it represents the most possible job to locate front.

Table 1. Attribute Calculation

Act	1	2	3	4	5	6	7	8	9	10	11	
Start	0	0	3	9	11	14	5	17	11	16	22	
Finish	3	5	9	11	14	17	9	22	16	17	25	
lst_st	0	9	9	3	7	7	8	16	9	16	20	
lst_fn	9999	9999	16	9	12	12	9999	25	9999	9999	9999	
Dur	3	5	6	2	3	3	4	5	4	2	3	
												mAd
Ad	0.12	0.2	0.24	0.08	0.12	0.12	0.16	0.2	0.16	0.08	0.12	0.06
tdd	10	0	6	13	5	5	0	0	3	3	0	
t_un	3.05	0	1.14	2.48	0.95	0.95	0	0	0.57	0.57	0	Mtdd
t_nm	0.12	0	0.05	0.1	0.04	0.04	0	0	0.02	0.02	0	0.095
LtwR	3333	2000	2.17	2.5	1.67	1.67	2500	1.8	2500	4001	3999	MLtwR
L_nm	0.33	0.2	0	0	0	0	0.25	0	0.25	0.5	0.33	0.333
ST	0	0	0	0	0	3	0	0	2	0	5	
ST_un	0	0	0	1.5	0	0.75	0	0	0.5	0	1.25	MST
ST_nm	0	0	0	0.06	0	0.03	0	0	0.02	0	0.05	0.06
Nwr	8	2	3	5	2	1	1	0	2	1	0	
N_un	4	1	1.5	2.5	1	0.5	0.5	0	1	0.5	0	Mnwr
N_nm	0.36	0.09	0.14	0.23	0.09	0.05	0.05	0	0.09	0.05	0	0.227
k												
Alwr	28	9	9	16	5	5	5	0	5	3	0	
A_un	3.5	4.5	3	3.2	2.5	5	5	9999	2.5	3	9999	mAlwr
A_nm	0.14	0.18	0.12	0.13	0.1	0.2	0.2	400	0.1	0.12	400	0.1

Structure of the Neural Network (NN)

Figure 1 above shows a fully connected NN used in this study. This network consists of three layers of nodes (or neurons): the first layer is the input layer, the second layer is the hidden layer, and the last layer is the output layer. In Figure 1, 7 nodes in the input layer which represent the 7 attributes, $a_1,.....,a_7$, and 7 neurons in the output layer which represent 7 single options in selecting a target, e.g., mAd, Mtdd are shown. The number of neurons in the hidden layer is to be determined after series of experiments. Excessive number of hidden units will cause over-fitting problem which would give undesired results. On the other hand too small number of hidden units will cause under-fitting. So, we have to try different number of hidden units by trial-and-error. In this NN, we assume all the neurons in the input layer are fully connected to the neurons in the hidden layer, and all the neurons in the hidden layer are fully connected to the neurons in the output layer. Each of the neurons except for the neurons in the input layer has a bias which has a value of unity. Each arc in the network has a weight. These weights are called synaptic weights. In this paper, a NN is used first because of its well known training capability. Then, use of Genetic (GA) is suggested to fine tune the problem in global level sense.

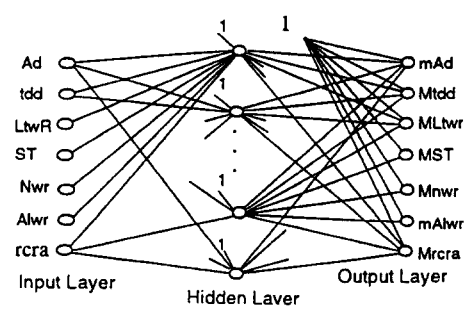

Figure 1. Neural Network for Target Selection Method

Training Fuzzy Neural Network

In the NN shown in Figure 1, initially, we are given some partial knowledge about rules. We try as many known rules as possible. Some rules might have an incomplete antecedent part. As mentioned before, there are 7 neurons in the input layer and 7 neurons in the output layer. Therefore, an example (training set or input-output pair) can be expressed as $[a_1,.....,a_7, s_1,.....,s_7]$ where a_1 through a_7 are the values of the attributes, and s_1 through s_7 are the single selection options described above. These 7 numbers represent the priorities of each of the 7 single target selection methods. Let T denotes the set of training examples. Then, $T= \{t_1, t_2,....., t_n\}$ if there are *n* examples. All the neurons in the input layer take normalized values of contributing factors and all the neurons in output layer generate values between 0 and 1 representing degree of the priorities. As an example, a specific training set t_i can be written as t_i =[1, .2, .01, 0, .21, 0, .2, 0, .15, .1, 1, 0, .15, 0]. After the first training example is done, the weights of the NN will be changed, and after the second training example, the weights will be changed again, and this goes on until all the training sets are considered. After training the NN with all the examples, we train the NN with the same set of examples again, one at a time, until the error between the desired output and the actual output becomes less than some very small number, ε.

After we finish the training, the synaptic weights will represent the importance (energy) of the arc. Some weights can be positive, and some can be negative. The magnitudes of the weights represent the strength of the arc. The more positive value the weight takes, the stronger the positive (synergistic) effect of the arc, and the more negative value the weight takes, the stronger the negative (anti-synergistic) effect of the arc. After the training, we find the trained weight vector for the NN. This weight vector will represent the "best" weight vector which gives the most proper priorities for all the options for each of the activity based on the examples given by the experts. With these output values, the highest value (possibility value) will be recorded and the jobs which gives the highest value, and the output with highest value is most appropriate target selection method. A possibility value represents the possibility that for a given configuration of schedule, the method to be adopted as a target selection method. If more than one output values are high within preset tolerance level, combination options are selected using lexicographic rule. Note that the weights in the NN represent the aggregated information about the priorities of all the single options. Therefore, these priorities of the target selection methods have the same effect as having combination options. The remaining NN contains the information that might have incomplete antecedent parts or conditions. Is there any way to recover these incomplete information? Or, is the resulting NN can be applied to solve any specific problems? In order to fine-tune the weights in problem level, some global search method has to be employed. A Genetic Algorithm (GA) is a suitable choice for this purpose. A diagram describing entire procedure of EEH is shown in figure 2 below.

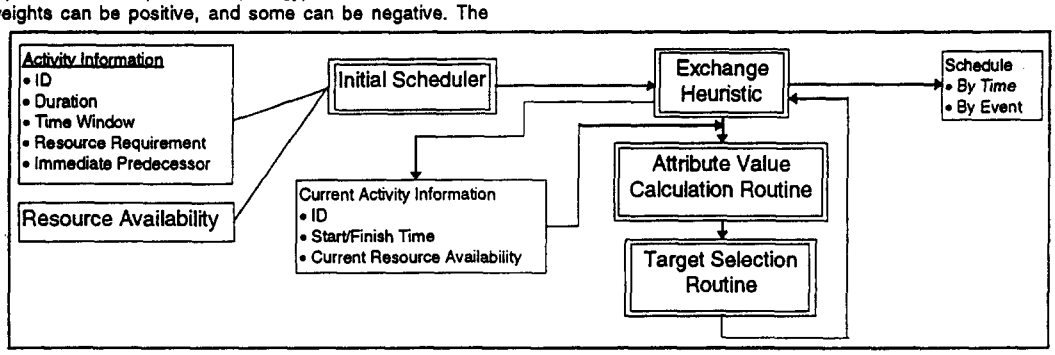

Figure 2 The EEH Architecture

CONCLUSION

The EH already demonstrated its power in solving RCS problems. However, it can be further improved if the target selection method is strengthened. From only the partial information from the experts, the method of generating a complete, and aggregated rulebase is presented in this study in the form of a NN. Through the training mechanism, the weight vector of the NN will be updated to give the most proper priorities for all the options for each of the activity based on the examples given by the experts. Further improvement of weight vectors can be done using a GA. The output of the NN is the possibility value for each output to be selected as a target selection method for the iteration. This method can be applied to many other similar types of problems when only partial information is available to generate a complete rulebase mechanism.

BIBLIOGRAPHY

[T95] Tiger, A. A., 1995, The Multi-Objective Exchange Heuristic (MOEH), A Tool for Solving Resource Constrained Scheduling Problems, *unpublished dissertation.*

[YI87] Yang, T., and Ignizio, J., P., 1987, An algorithm for the scheduling of army battalion training exercises, Computers and Operations Research, vol 14, p479-491

[YID89] Yang, T., Ignizio, J. P., and Deal, D. E., 1989, An exchange heuristic algorithm for generalized job shop scheduling, Engineering Optimization, vol 15, p83-96

[YIS89] Yang, T., Ignizio, J. P., and Song, J., 1989, An exchange heuristic algorithm for project scheduling with limited resources, Engineering Optimization, vol 14, p189-205