Journal of the Korean Institute
of Industrial Engineers
Vol.23, No.2, June, 1997

유연자원할당 및 자원제약하의 일정계획을 위한 발견적 알고리즘

A heuristic algorithm for resource constrained scheduling with flexible resource allocation

유재진*

Yoo, Jaegun*

Abstract

In this study, a heuristic algorithm is developed to solve a resource-constrained scheduling problem. The problem involves multiple projects and multiple resource categories, and allows flexible resource allocation to each activity. The objective is to minimize the maximum completion time. The algorithm takes advantage of the basic structure of a heuristic algorithm, called the exchange heuristic, but employs different strategies on some critical steps of the original algorithm which have significant effects on the algorithm performance. The original algorithm and the modified algorithm were compared through an experimental investigation. The modified algorithm produces significantly shorter schedules than the original algorithm, though it requires up to three times more computation time.

1. Introduction

Resource-constrained scheduling (RCS) is constrained by the competition for available resources (resource constraints) and by the time interdependencies of the numerous activities (precedence constraints) [5]. The RCS paradigm is fundamentally more pragmatic than other paradigms such as the flow-shop model, simple job-shop model, or basic network model of PERT/CPM [2, 8]. It takes into account many more of the complexities of the scheduling environment. As a result,

* 한성대 산업공학과

the difficulties in simpler problems are superimposed in RCS problems. Most models proposed for the RCS assume that an activity must be performed in a prespecified way. That is, each activity consumes certain types of resources at fixed rates and is completed in a fixed duration. However, in reality, many activities are capable of consuming resources at a certain range of degrees. That is, they have flexibility in consuming their resources. Naturally, their durations are also variable. The more resources are applied to them, the more quickly they are processed and completed. These activities that have flexible resource requirements and variable durations are termed "variable-intensity" activities [6]. The term "Intensity" indicates the rates of consumed resources.

The assumptions of the RCS problems which are dealt with in this study are as follows:

(1) a set of projects is to be scheduled,
(2) each project must perform a set of predetermined activities,
(3) all activity durations are taken to be integers,
(4) an activity may have flexible resource requirements,
(5) activities cannot be started until specific, preceding activities are completed,
(6) each activity may have multiple predecessors and multiple successors, but looping and dangling of activities are not allowed,
(7) there is no mutual exclusiveness among activities, that is, there is no restriction such that a group of activities cannot be simultaneously processed,
(8) activities once started cannot be interrupted (splitting of activities is not permitted),
(9) the amounts of resources are integer-valued,
(10) the minimum amounts of different resources that an activity requires during each time period, remain constant throughout the processing of the activity, and
(11) the amounts of resources available during each time period are constant and known.

Much attention has been given to RCS problems since 1970's. This is because basic PERT/CPM procedures have been successfully and popularly employed for a wide variety of scheduling problems, but it has also been recognized for a long time that the assumption of unlimited resource availability is too naive and too restrictive for certain real-life situations [5]. Existing scheduling procedures may be grouped into two major categories. First, there are heuristic procedures which aim at producing good, feasible schedules. In contrast, the second major group consists of procedures which aim at producing the best possible, or optimal, schedule. Most exact procedures found in the literature tend to solve very limited problems. The RCS problem is NP-hard [1]. That is, as the problem size increases, the time to find an exact solution grows exponentially. Since exact algorithms are not available, heuristic methods have to be used, as long as they show the ability of obtaining consistently good solutions in reasonable computing time. Most heuristic procedures are single pass heuristics based on priority rules. In single pass heuristics,

once a schedule is obtained, it is never rescheduled. Since other heuristics were developed only for very special circumstances, they can hardly be applied to other problems. The exchange heuristic (EH), developed by Yang and Ignizio [12, 13, 14], is a multiple pass heuristic procedure. The EH consists of two phases. First, an initial feasible schedule is obtained. In the second phase, the algorithm is applied to the initial schedule. During the implementation of the algorithm the feasibility is never lost and the objective value never deteriorates. Therefore the EH is always superior to any single pass heuristic procedure. In this study, a heuristic algorithm is proposed, which takes advantage of the basic structure of the exchange heuristic, but employs different strategies on some critical steps of the original algorithm which have significant effects on the algorithm performance. Therefore, the exchange heuristic needs to be described before the modified algorithm is explained.

2. The Exchange Heuristic

The EH was originally developed by Yang and Ignizio to solve a scheduling problem of army battalion training exercises [12]. It was applied later to project scheduling with limited resources [14] and generalized job-shop scheduling [13]. The EH makes improvements over an initial feasible schedule obtained by other solution methods. It is also designed such that each pass of the heuristic cannot construct a schedule which is inferior to a schedule previously constructed. It therefore provides a schedule which is most likely improved, or, in a worst-case scenario, is the same as the schedule provided by other scheduling procedures. The heuristic is essentially the second phase in a two-phase approach.

2.1 Phase I - preparing an initial schedule

An initial feasible schedule is obtained during Phase I. Feasibility ensures that all the constraints including resource limitations, precedence relationships, and nonpreemptiveness are not violated. Any method can be used to find an initial schedule as long as it is feasible. It is generally the case in a numerical search method that the better a starting solution is, the faster will be the convergence. For this reason a better initial schedule may be preferred. However, it is not always the case for the EH. On the contrary, even though a better initial schedule is usually obtained by a heuristic zero-one sub-optimization method of Thesen [10], it has been observed that for such schedule there tends to be less opportunity for improvement [13, 14]. Therefore, a random schedule generator is developed and used to generate initial feasible schedules in this study.

2.2. Phase II - the exchange heuristic

A survey of heuristic approaches to RCS problems indicates that most researchers have hitherto adopted the forward-loading technique in which activities are scheduled as early as possible [7]. For the forward-loading technique, there

is a strong tendency that resource consumption levels will become lower along the time horizon, as shown in Figure 1-(a). The EH takes advantage of this tendency. That is, the EH reduces schedule length through the leveling of such uneven resource consumption rates. Figure 1 shows simplified representations of how the EH achieves the reduction of schedule length. First, the activities in a specific time region, which is called "search region", are rescheduled backward, therefore consuming the unused resources at the later part of the schedule. The resources in the search region may be freed by this backward-scheduling and resource leveling is achieved after the search region, as shown in Figure 1-(b). Then, by rescheduling the activities forward that are previously scheduled after the search region, the schedule length may be reduced, as shown in Figure 1-(c). The original algorithm for the EH is described below in detail so that explaining and understanding the modified algorithm might be easier. Terminology and notation used to present the algorithm are listed and defined in Table 1.

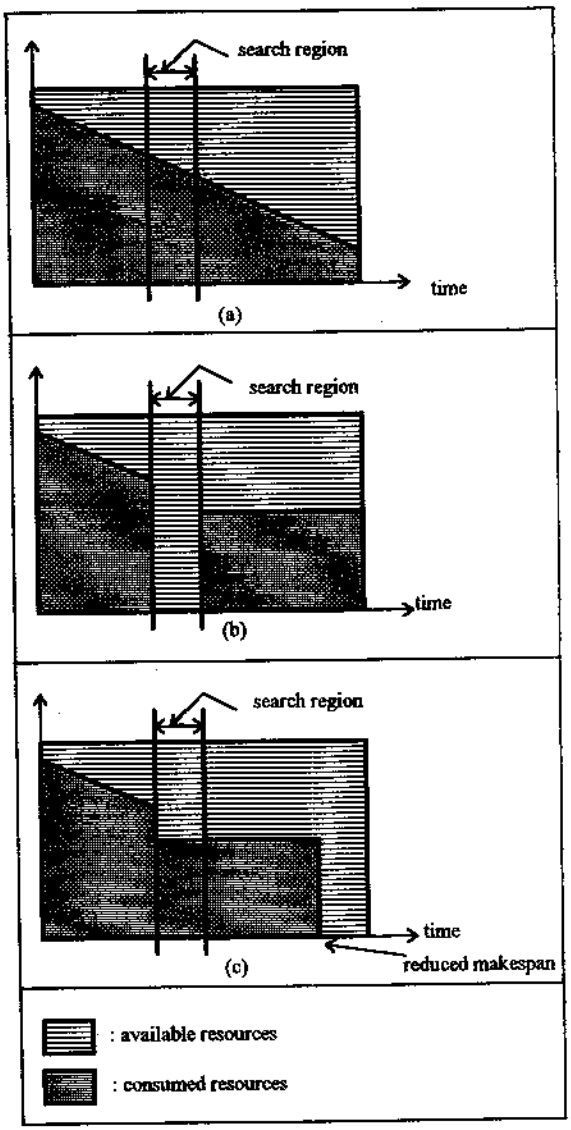

Figure 1. A simplified illustration of how the EH reduces schedule length

The Algorithm of the EH

Step 0 (Initialization)

Step 0-1: Let the current schedule (S^{curr}) be a given initial schedule.

Step 0-2: Let ITER = 0.

Step 1 (TARGET selection and removal)

Step 1-1: Let the temporary schedule (S^{tem}) be S^{curr} (The S^{tem} is used until it is accepted as S^{curr} at the end of an iteration or discarded during an iteration).

Step 1-2: Find a TARGET activity.

· The TARGET has the latest start time among the activities having slack time

Table 1. Definitions of terminology and notation used in the algorithm.

Notation	Definitions
B	set of activities which start in the search region
D	set of activities which start at the right boundary of the search region
E	set of activities which start after the search region
ITER	iteration number
LRO	(Leftward Rescheduling Operation) an operation of rescheduling activities to finish as early as possible
PLB	(Precedence Left Boundary) the latest finish time of immediate predecessors of an activity
PRB	(Precedence Right Boundary) the earliest start time of immediate successors of an activity
RRO	(Rightward Rescheduling Operation) an operation of rescheduling activities to start as late as possible
S^{curr}	initial or new improved schedule
S^{tem}	temporary schedule
search region	time region bounded by SLB and SRB
SLB	(Search region Left Boundary) PLB of a TARGET
SRB	(Search region Right Boundary) finish time of a TARGET
TARGET	an activity which has slack time between itself and its immediate predecessors, and plays a key role in reducing schedule length

between itself and its immediate predecessors.

- If an activity has been chosen as a TARGET in a previous iteration that has failed to shorten schedule length, it is not eligible.

Step 1-3: If no activity is eligible as a TARGET, let the current schedule be the final schedule and STOP.

Step 1-4: Let ITER = ITER + 1.

Step 1-5: Remove the TARGET from the schedule.

- The resources used by the TARGET are made available.

Step 2 (Determining the search region, and sets B, D, and E)

Step 2-1: Determine the boundaries of the search region.

- The left boundary of the search region (SLB) is the latest finish time of the immediate predecessors to the TARGET and the right boundary of the search region (SRB) is the finish time of the TARGET.

Step 2-2: Collect a set of activities (set B) which start between SLB and SRB.

Step 2-3: Collect a set of activities (set D) which start at SRB.

Step 2-4: Collect a set of activities (set E)

which start at or after SRB+1.

Step 3 (RRO of set B - Freeing the resources in the search region)

Step 3-1: If B = ϕ, go to step 1-2.

Step 3-2: Choose an activity that has the latest finish time among the activities in set B and has not been chosen for RRO in step 3.

Step 3-3: Perform RRO for the activity.

- Find the right boundary for RRO (PRB) which is determined only by precedence constraints.
- Check resource availability between the PRB and the old finish time of the activity.

Step 3-4: If all the activities have been chosen, go to step 4. Otherwise, go to step 3-2.

Step 4 (LRO of the TARGET - Triggering the overall leftward rescheduling)

Step 4-1: Perform LRO for the TARGET.

- Check resource availability between the SLB and the old start time of the TARGET.

Step 4-2: If the TARGET is not rescheduled earlier, go to step 1-1. Otherwise, go to step 5.

Step 5 (LRO of set B)

Step 5-1: Choose an activity which has the earliest start time among the activities in set B and has not been chosen for LRO in step 5.

Step 5-2: Perform LRO for the activity.

- Find the left boundary for LRO (PLB) which is determined only by precedence constraints.
- Check resource availability between the PLB and the old start time of the activity.

Step 5-3: If all the activities have been chosen, go to step 6. Otherwise, go to step 5-1.

Step 6 (LRO of set D and predicting the possibility of reducing the schedule length)

Step 6-1: If D = ϕ, go to step 7.

Step 6-2: Choose an activity which has the earliest start time among the activities in set D and has not been chosen for LRO in step 6.

Step 6-3: Perform LRO for the activity.

- Find the left boundary for LRO which is determined only by precedence constraints (PLB).
- Check the resource availability between the PLB and the old start time of the activity.

Step 6-4: If all the activities have been chosen, go to step 6-5. Otherwise, go to step 6-2.

Step 6-5: If no activity in set D is rescheduled earlier, go to step 1-1. Otherwise, go to step 7.

Step 7 (LRO of set E)

Step 7-1: If E = ϕ, go to step 7-5.

Step 7-2: Choose an activity which has the earliest start time among the activities in set E and has not been chosen for LRO in step 7.

Step 7-3: Perform LRO for the activity.
- Find the left boundary for LRO which is determined only by precedence constraints (PLB).
- Check the resource availability between the PLB and the old start time of the activity.

Step 7-4: If all the activities have been chosen, go to step 7-5. Otherwise, go to step 7-2.

Step 7-5: If the schedule length is not reduced, go to step 1-1. Otherwise, update the current schedule, i.e., $S^{curr} = S^{tem}$ and go to step 1-2.

3. The Modified Algorithm

3.1 Improving the performance

The procedures of the original algorithm are examined to find ways to improve its performance. Each iteration of the algorithm tries to achieve some reduction in schedule length, but it may or may not be successful. Its success strongly depends on the success of the following steps:

Step 1: (RRO of set B) The resources in the search region should be freed so that the TARGET can be rescheduled earlier.

Step 2: (LRO of the TARGET) The TARGET should be rescheduled earlier to trigger the LRO of the subsequent activities.

Step 3: (LRO of sets B, D, and E) The activities in the sets should be rescheduled earlier to reduce the schedule length.

It is step 3 that the actual reduction of the schedule length is achieved. However, if step 2 is not successful, step 3 is not even attempted. In other words, the failure of step 2 certainly excludes any possibility of reducing the schedule length at the iteration. In turn, the success of step 2 totally depends on step 1. A simplified flow chart of the original algorithm, which is shown in Figure 2, will be helpful to understand

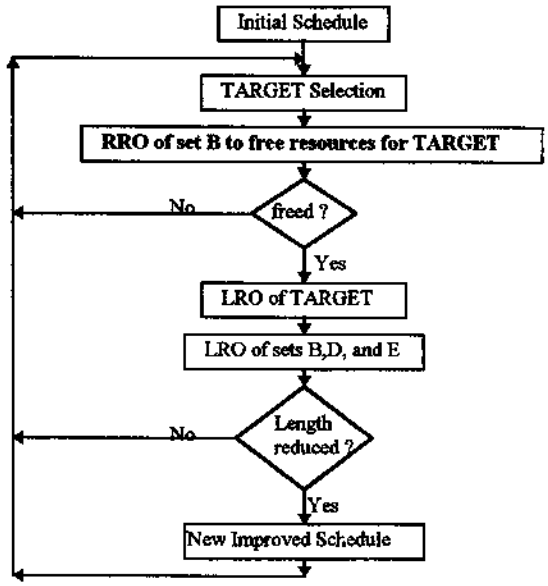

Figure 2. A structural flow chart of the EH.

such logical relations. The most important step can be recognized from this flow chart. It is that of RRO of set B. If the step fails, there is no chance to reduce schedule length. Only when it succeeds, there is a chance of improving the current schedule. Therefore, the success of an iteration totally depends on the success of the step which tries to free resources for TARGET. It is then believed that the performance of the

original algorithm can be improved by modifying the step of RRO. Therefore, efforts to improve the performance of the heuristic are focused on improving the step of freeing the search region resources for the TARGET.

The purpose of rescheduling the activities in the search region is to free resources for the TARGET. Therefore, the activities which do not consume the same types of resources as the TARGET need not be removed from the search region. In addition, the unnecessary rescheduling of these activities and their successors may result in blocking the movement of other activities towards the right because the former will reduce the availability of resources in the later part of the schedule. Therefore the activities in the search region which do not consume the same types of resources as the TARGET should not be included in set B. When the activities in the search region are rescheduled to the right for freeing the resources for the TARGET, they are frequently blocked by the activities after the search region due to both of precedence constraints and resource conflicts. Therefore the activities after the search region as well as those in the search region should be rescheduled to the right for freeing the resources for the TARGET. Hence, in the modified algorithm, set B is redefined to include the activities which start in the search region and compete for resources with the TARGET, and their successors. Set B is defined as equation (1).

$$B=A\cup(\text{big}\bigcup_{j\in A} S_j), \qquad (1)$$

where

S_j is a set of successors to activity j,

$A = \{j \mid SLB \le t_j^s \le SRB,\ j \neq TARGET,$ and $RT_j \cap RT_{TARGET} \neq \phi\}$,

t_j^s is the start time of activity j,

$RT_j=\{k \mid mr_{jk} \neq 0\}$, and

mr_{jk} is the minimum amount of resources of type k required by activity j,

When the activities of set B are rescheduled towards the right, the slack times between an activity and its immediate successors are reduced, when there are enough resources. When the slack times are reduced enough, the activities in the search region are moved beyond the SRB to free resources in the search region for the TARGET. The slack times are scattered between the SLB and the end. Resources in the period are consumed partly by the activities which do not belong to set B. The set of these acitivites is termed "Set$\overline{B}$". Therefore, the activities starting in and after the search region are divided into set B and set$\overline{B}$. There are resource conflicts between the activities of set B and set$\overline{B}$. Such resource conflicts as well as precedence constraints block the rightward rescheduling of the activities of set B. Such resource conflicts are partially resolved by extra rescheduling of the activities of set$\overline{B}$. Pushing the activities of set$\overline{B}$ towards the right or left will increase resource availability around the SLB or the end of the schedule, respectively. The increased availability of resources on one side increases the chance to reduce the slack

times there. Thus, it is expected that by alternately performing LRO and RRO of set$\overline{B}$, the slack times will be more efficiently reduced and thus resources in the search area are more likely to be available for the TARGET. However, it is also most likely that computing time will increase. There are two kinds of extra rescheduling of set $\overline{B}$: RRO and LRO. Whenever RRO of set B is not successful, either RRO or LRO of set$\overline{B}$ is performed and then RRO of set B is performed again. However, there should be a limit to the number of the extra rescheduling. The maximum number of the extra rescheduling of set$\overline{B}$ is denoted as "*imax*". The value of *imax* which gives the best results is experimentally determined in this study. Set$\overline{B}$ is defined as equation (2).

$$\overline{B} = \{j \mid t_j^s \geq SLB,\ j \notin B,\ j \neq TARGET,\ j \notin P_i,\ i \in B\}, \quad (2)$$

where $P_i = \{ j \mid j$ is a predecessor to $i\}$.

Step 3 and 4 of the original algorithm are modified as shown below(the modified or added steps are bold-faced).

Modified or Added Steps of the Modified Algorithm

Step 3 (RRO of set B - Freeing the resources in the search region)

Step 3-1: If B = ϕ, go to step 1-2.

Step 3-1.1: Let i = 1. Choose the maximum number of extra rescheduling of set$\overline{B}$, i.e., the value of *imax*.

Step 3-2: Choose an activity which has the latest finish time among the activities in set B and has not been chosen for RRO in step 3.

Step 3-3: Perform RRO for the activity.

- Find the right boundary for RRO which is determined by precedence constraints (PRB).
- Check the resource availability between the PRB and the old finish time of the activity.

Step 3-4: If all the activities have been chosen, go to step 4. Otherwise, go to step 3-2.

Step 4 (LRO of the TARGET - Triggering the overall leftward rescheduling)

Step 4-1: Perform LRO for the TARGET

- Find the left boundary for LRO which is determined by only precedence constraints (PLB).
- Check the resource availability between the PLB and the old start time of the TARGET.

Step 4-2: If the TARGET is not rescheduled earlier, go to step 4-3. Otherwise, go to step 5.

Step 4-3: If $i > imax$ or $\overline{B} = \phi$, go to step 1-1.

Step 4-4: If i is odd, i.e. the last rescheduling operation of set$\overline{B}$ was performed towards the left, or this is the first res-

cheduling operation of set$\overline{B}$, go to step 4.1 for RRO of set $\overline{B}$. Otherwise, go to step 4.2 for LRO of set$\overline{B}$.

Step 4.1 (RRO of set$\overline{B}$ - Helping the RRO of set B)

Step 4.1-1: Choose an activity which has the latest finish time among the activities in set $\overline{B}$ and has not been chosen for RRO in step 4.1.

Step 4.1-2: Perform RRO for the activity.

- Find the right boundary for RRO which is determined by precedence constraints (PRB).
- Check the resource availability between the PRB and the old finish time of the activity.

Step 4.1-3: If all the activities have been chosen, go to step 4.1-4. Otherwise, go to step 4.1-1.

Step 4.1-4: If no activity in set$\overline{B}$ is rescheduled later, go to step 1-1. Otherwise, let $i = i+1$ and go to step 3-1.1.

Step 4.2 (LRO of set$\overline{B}$ - Helping the RRO of set B)

Step 4.2-1: Choose an activity which has the earliest start time among the activities in set $\overline{B}$ and has not been chosen for LRO in step 4.2.

Step 4.2-2: Perform LRO for the activity.

- Find the right boundary for LRO which is determined by precedence constraints (PLB).
- Check the resource availability between the PLB and the old start time of the activity.

Step 4.2-3: If all the activities have been chosen, go to step 4.2-4. Otherwise, go to step 4.2-1.

Step 4.2-4: If no activity in set$\overline{B}$ is rescheduled earlier, go to step 1-1. Otherwise, let $i = i+1$ and go to step 3-1.1.

3.2 Extending the applicability

In this study, the modified algorithm is also applied to resource constrained scheduling with flexible resource allocation. The amount of resources applied to an activity can be varied according to the amount of unused resources. The processing time of an activity is reduced when more resources are allocated to it. That is, the processing time of variable-intensity activities is a function of the rates of resource applications. The following defines the relations between resource applications and processing time:

1. Each variable-intensity activity has minimum resource requirements per period to make its processing possible. The amount of minimum resource requirements is fixed.
2. The mix of all the different types of resources which are minimally required by a

variable-intensity activity is termed "basic-mix".

3. All the different types of resources which are required by variable-intensity activities are applied proportionally throughout the activity execution. That is, each variable-intensity activity utilizes a multiple of the basic-mixes. Thus, the intensity of applications of resources to an activity can be indexed in terms of the number of basic-mixes.
4. The activity intensity is assumed to be an integer value from 1 to the upper limit. Each variable-intensity activity has a fixed upper limit.
5. The rate of progress of an activity is assumed to be proportional to its intensity. That is, by increasing the intensity of an activity by one basic-mix, its processing time tends to be reduced. This concept of the relationship between the rate of progress and the intensity of the resources is used by Leachman et al. [6] and Weglarz [11].
6. The total amount of resources required to complete a variable-intensity activity is fixed and indexed in terms of the number of basic-mixes.

Hence, the amount of resources consumed by a variable-intensity activity j and the duration of the activity j are defined as equation (3) and (4), repectively.

$$CBM_j(t) = \text{Min}\left\{ MBM_j, \left[\frac{AR_k(t)}{mr_{jk}} \right] \text{ for } k \text{ such that } mr_{jk} \neq 0 \right\}, \text{ for } j \in J, \quad (3)$$

where

$CBM_j(t)$ is the number of basic-mixes consumed by activity j at time t,

MBM_j is the number of basic-mixes which can be maximally applied to acitivity j in a time period,

$AR_k(t)$ is the amount of unused resources of type k at time t,

mr_{jk} is the minimum amount of resources of type k required by acitivity j,

J is the set of variable-intensity activities.

$$d_j = \text{Min}\{T \mid \sum_{t=t_0+1}^{t_0+T} CBM_j(t) \geq TBM_j, T=1,2,\ldots\}, \text{ for } j \in J, \quad (4)$$

where

t_0 is a candidate starting time for acitivity j,

TBM_j is the total number of basic-mixes required to complete activity j,

$[x]$ is the greatest integer which is less than or equal to x.

4. Experimental Study

4.1 Aims

The aims of the experimental investigation were to answer the following questions:

1. How much does the modified algorithm improves the schedule length more than the original?

2. How much is the schedule length shortened by modeling the flexibility of resource requirements in scheduling procedures?

4.2 Measures of performance

Two measures of effectiveness can be used for each schedule: the schedule length, SL, and the utilization factor, UF, as given by equation (5).

$$UF = \left(\frac{\sum_{j=1}^{N} d_j \, mr_{jk}}{\sum_{k=1}^{K} R_k \cdot SL} \right) \times \frac{100}{K} \qquad (5)$$

where

K is the total number of resource types,

N is the total number of activities,

R_k is the amount of available resources of type k.

These are in a sense equivalent measures in that minimizing schedule length is equivalent to maximizing the utilization factor for fixed R_k. However, UF is more useful as a measure of effectiveness across different problems, as it is normalized, dimensionless, independent of work content, and has better statistical properties [4].

4.3 Description of problems

A scheduling problem is characterized by its size, network structure, and resource structure. The size parameters are the number of projects, activities, variable-intensity activities, and resource types. The parameters used to describe the other characteristics of the problems are as follows [3].

Structure parameter

The structure of the network is characterized by the parameter "order strength" which is the ratio of the total number of actual precedence relations to the total number of possible precedence relations. Order strength is defined by equation (6).

$$os = \frac{\text{the total number of actual precedence relations}}{\text{the total number of possible precedence relations}} = \frac{\sum_{j=1}^{N} n(S_j)}{\frac{N(N-1)}{2}} \qquad (6)$$

where

$n(S_j)$ is the number of successors of activity j.

Resource parameters

The number of different kinds of resources used by activities in a problem is measured by the parameter "resource factor," rf, defined as the ratio of the average number of the different kinds of resources used per activity to the total number of the different kinds of resources. Resource factor is defined by equation (7).

$$rf = \frac{\sum_{j=1}^{N} n(RT)_j}{N \cdot K} \qquad (7)$$

The relative quantities of resource type k required by the project, in relation to the amount available, R_k, is measured by the parameter resource strength, rs(k), defined by

$$rs(k) = \frac{R_k}{\text{average requirement for resource } k \text{ per activity}}$$

$$= \frac{R_k \cdot N}{\sum_{j=1}^{N} mr_{jk}} \quad (8)$$

Sixteen problems were generated as shown in Table 2. Note that problems 1~9 have no variable-intensity activity while problems 10~16 have some.

generator and rescheduled by both algorithms to compare their performances. The modified algorithm can be varied by using different values of *imax*. The value of *imax* is the maximum number of extra rescheduling of set$\overline{B}$, as defined in Section 3.1. Three variations of the algorithm were

Table 2. Characteristics of the example problems.

Problem Serial No.	1, 10	2, 11	3, 12	4, 13	5, 14	6, 15	7, 16	8	9
No. of Projects	1	4	5	6	6	10	20	10	20
No. of Activities	11	12	12	36	36	100	100	200	400
No. of Var-Int Act.'s	4	4	4	12	12	32	32	0	0
No. of Res. Types	3	3	5	6	9	10	5	10	20
Order Strength (os)	0.45	0.18	0.14	0.13	0.13	0.08	0.04	0.05	0.05
Resource Factor (rf)	0.94	0.33	0.35	0.26	0.23	0.14	0.39	0.50	0.45
Avg.Res.Strength(rs)	3.13	4.89	4	7.23	6.82	26.77	10.21	13.39	88.25

4.4 Computational results

For each example problem, 30 random active schedules were generated by a random schedule

tested by using *imax* values of 1, 2, and 3. Those are labeled as EH^1_n, EH^2_n, and EH^3_n. The original algorithm is also labeled as EH_o. The aver-

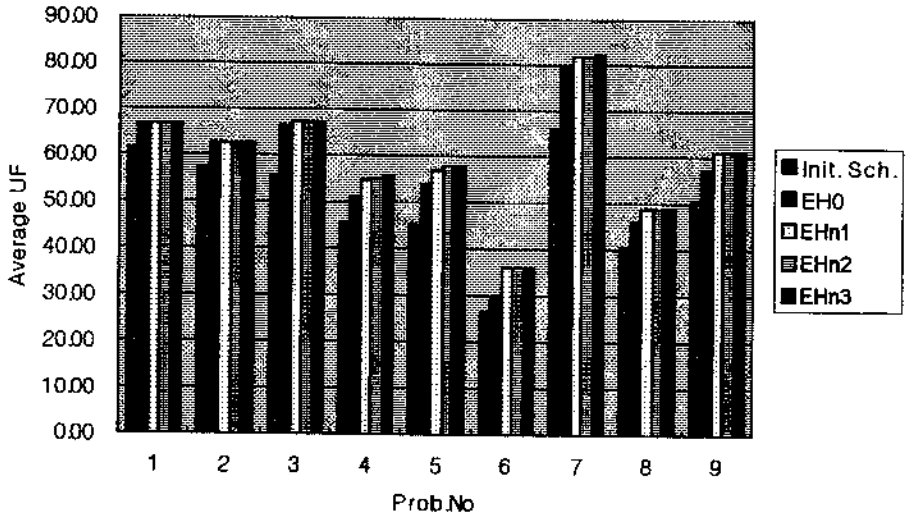

Figure 3. Average utilization factors of 30 final schedules for each problem

age utilization factors of the final schedules obtained by the algorithms for each problem are shown in Figure 3. It is quite obvious that the modified algorithm(EH_n) is superior to the original(EH_o). That is, we can expect that EH_n produces better schedules. When the number of resource types and activities are very small as problem 1 and 2, both algorithms produce the same quality of schedules as shown in Figure 3. However, for the other problems with more resource types and activities, EH_n obviously outperforms EH_o. However, the figure shows that more than one extra rescheduling of set$\overline{B}$(i.e., *imax*= 2) does not make extra reducon significantly. EH^{1}_{n} increases about 6% of average utilization factors and decreases about 20% of average schedule lengths over EH_o. EH_n also shows more stable performance as shown in Figure 4. Especially, for problem 6, EH_n always produces the same schedule length (SL=540), while EH_o produces the worst, 770 and the best, 540 with standard deviation of 56.29. As shown in Figure 5, there is not much difference between the best schedule among 30 final schedules produced by EH_o and the worst schedule by EH_n. Figure 6 shows the schedule lengths of the best schedule among 30 final schedules obtained by EH_o and EH^{1}_{n}. Though both algorithms produce almost the same length of best schedules for small problems, EH_n produces better best schedule for large problems.

To address the second question which was

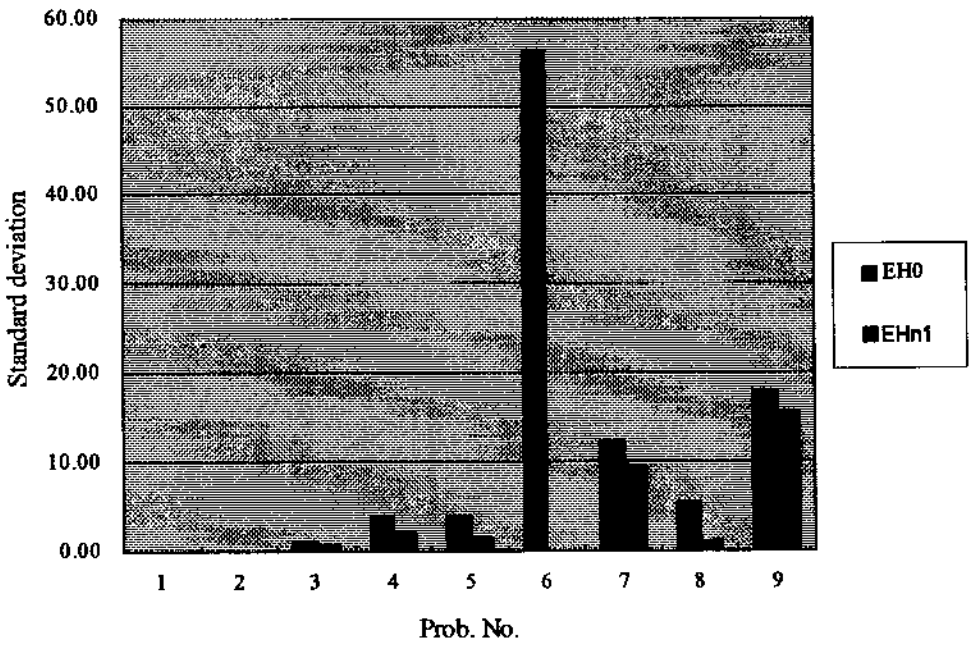

Figure 4. Standard deviations of utilization factors for EH_o and EH^{1}_{n}

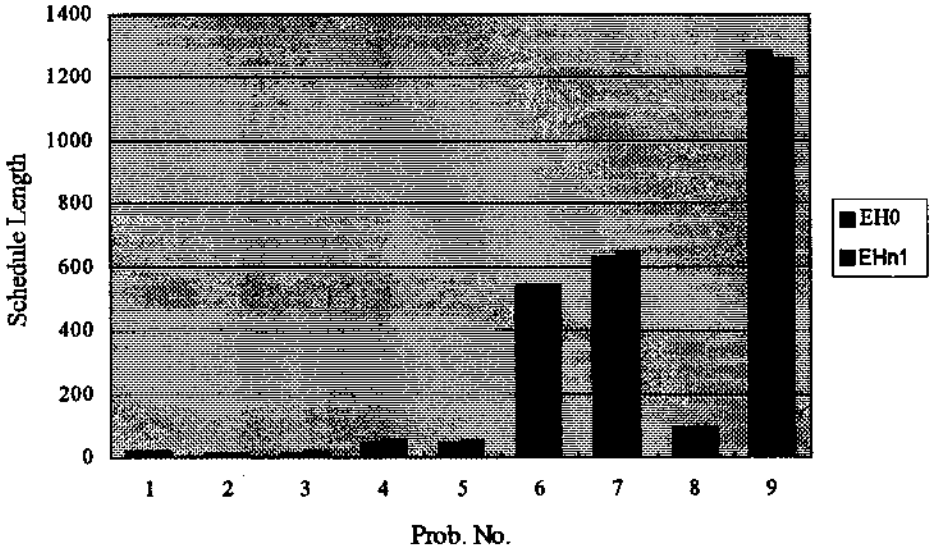

Figure 5. Comparison of SL between best schedules by EH_o and worst schedules by EH_n^1

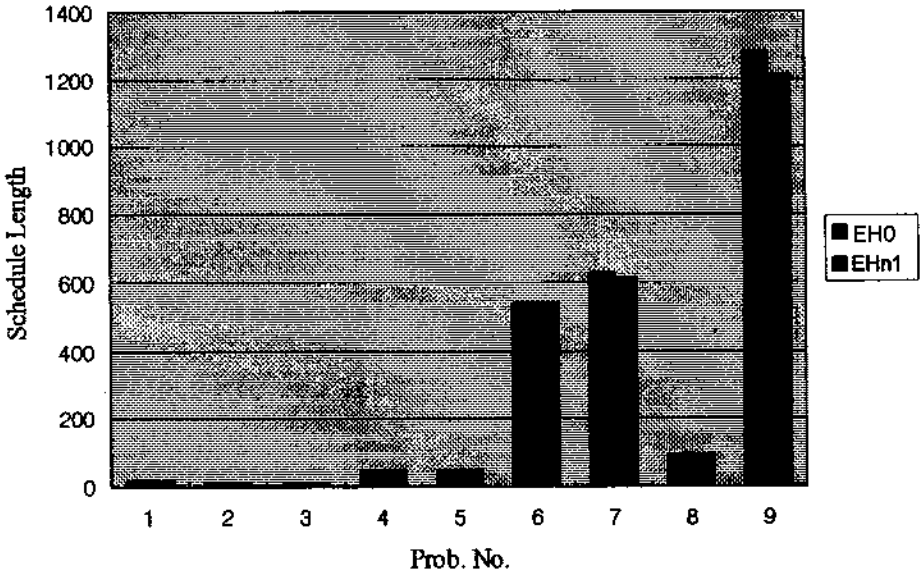

Figure 6. Schedule lengths of best schedules among 30 final schedules

mentioned earlier, the schedule lengths of the best schedule among 30 final schedules obtained by EH_o for the fixed cases(problems 1~7) and the flexible cases(problems 10~16) were compared as shown in Figure 7. It shows that we can get some benefit from flexible resource allotment

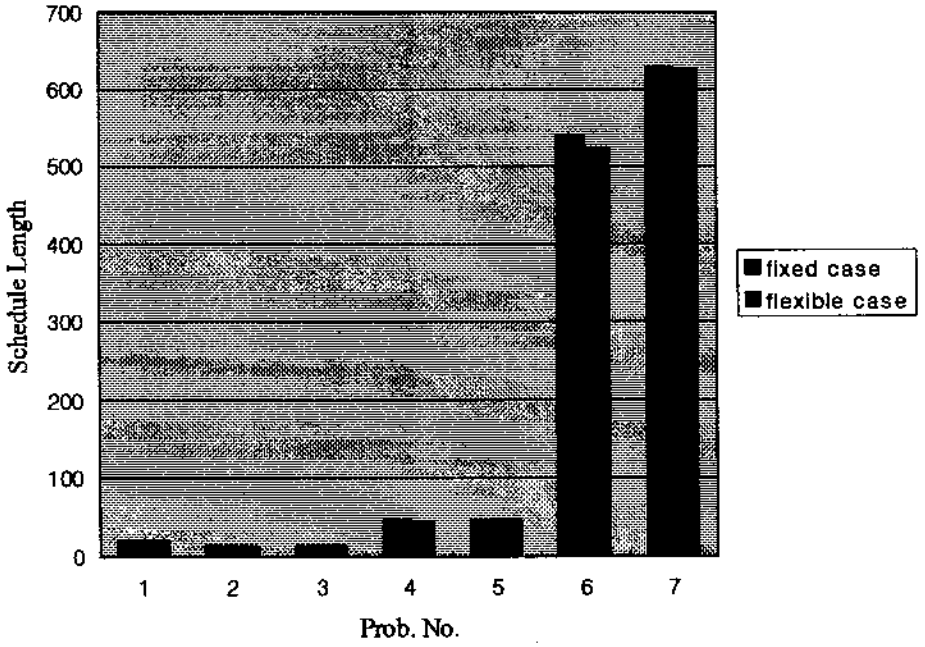

Figure 7. Schedule lengths of best schedules on fixed and flexible cases

when there are enough resources available. Especially, for problem 6 and 15 with 26.77 of resource strength, the extra reduction of schedule length is much larger than that for other problems.

5. Conclusions and Further Research

In this study, a heuristic algorithm has been proposed by modifying the original EH algorithm to solve a RCS problem with multiple projects and resource categories, and flexible resource allocation. The algorithm steps which are critical for improving the performance of the original algorithm are found to be those of freeing resources for the TARGET. The modified algorithm was obtained by applying different strategies on those steps of the original algorithm. It was experimentally proved that the modified algorithm is better than the original algorithm with respect to the schedule length, which is the objective function, but it requires more computation time. The computational results also show that extra reduction in schedule length can be achieved by introducing variable-intensity activities. Though the test problem sizes in this study were up to 400 activities and 20 resource types, the computational results will be similar for larger problems. As stated by Pascoe [9], most effective heuristics for smaller problems were also most effective for larger problems.

The algorithm may be used in two ways. One way is to generate a single random schedule and

obtain an improved final schedule. The other way is to generate multiple random schedules, obtain a set of improved final schedules, and select the best. It is obvious that the latter way will find a better schedule. Thus, it is recommended that the latter method be used, when the computing time is not a significant cost factor.

The assumptions in this study are relatively generous and are close to the real-world RCS problems. The problem is indeed combinatorial in nature and the presence of polynomial time exact algorithms, except in some special cases, is highly unlikely. A heuristic procedure is the only answer to the problem. The proposed algorithm can be a practical solution in a scheduling system especially where the information on resources and activities(or operations in FMS) are electronically available. However, it still has some nonrealistic restrictions such as renewable resource types, fixed resource availability, discrete time, discrete resource amounts, and no mutual exclusiveness. It is strongly felt that by further research these limitations can be removed to expand the applicability of the algorithm.

References

[1] Blazewicz, J., Lenstra, J.K., and Rinnooy Kan, A.H.G. "Scheduling subject to Resource Constraints: Classification and Complexity", *Discrete Applied Mathematics*, Vol.5, pp.11-24, 1983.

[2] Busch, D.H. *The New Critical Path Method*, Probus, Chicago, 1991.

[3] Cooper, D.F. "A Computer Analysis of Some Project Scheduling Heuristics", *Proceedings of the Sixth Australian Computer Conference*, pp.11-24, 1974.

[4] Cooper, D.F. "Heuristics for Scheduling Resource-Constrained Projects: An Experimental Investigation", *Management Science*, Vol. 22, No.11, pp.1186-1194, 1976.

[5] Davis, E.W. "Project Scheduling under Resource Constraints - Historical Review and Categorization of Procedures", *AIIE Transactions*, Vol.5, No.4, pp.297-313, 1973.

[6] Leachman, R.C., Dincerler, A., and Kim, S. "Resource-Constrained Scheduling of Projects with Variable-Intensity Activities", *IIE Transactions*, Vol.22, No.1, pp.31-40, 1990.

[7] Li, K.Y. and Willis, R.J. "An Iterative Scheduling Technique for Resource-Constrained Scheduling", *European Journal of Operational Research*, Vol.56, pp.370-379, 1992.

[8] Moder, J.J., Phillips, C.R. and Davis, E.W. *Project Management with CPM and PERT*, 3rd Edition, Van Nostrand Reinhold, New York, 1985.

[9] Pascoe, T.L. *An Experimental Comparison of Heuristic Methods for Allocating Resources*, Ph.D. Dissertation, Cambridge University, England, 1965.

[10] Thesen, A. "Heuristic Scheduling of Operations under Resource and Precedence Restrictions", *Management Science*, Vol.23, No.4, pp.412-422, 1976.

[11] Weglarz, J. "Project Scheduling with Continuously Divisible, Doubly Constrained Re-

sources", *Management Science*, Vol.27, No. 9, pp.1040-1053, 1981.

[12] Yang, T. and Ignizio, J.P. "An Algorithm for the Scheduling of Army Battalion Training Exercises", *Computers and Operations Research*, Vol.14, No.6, pp.479-491, 1987.

[13] Yang, T. and Ignizio, J.P. and Deal, D.E. "An Exchange Heuristic Algorithm for Generalized Job Shop Scheduling", *Engineering Optimization*, Vol.15, pp.83-96, 1989.

[14] Yang, T. and Ignizio, J.P., and Song, J. "An Exchange Heuristic Algorithm for Project Scheduling with Limited Resources", *Engineering Optimization*, Vol.14, pp.189-205, 1989.

97년 3월 최초 접수, 97년 4월 최종수정