Journal of Korean Institute of Intelligent Systems, Vol. 24, No. 2, April 2014, pp. 218-224
http://dx.doi.org/10.5391/JKIIS.2014.24.2.218

Term-Based Approach를 기초로한 자원제한프로젝트스케줄링 문제의 해결

Solution of the Resource Constrained Project Scheduling Problem on the Foundation of a Term-Based Approach

김복선[†]
Pok-Son Kim[†]

국민대학교 자연과학대학 수학과
Department of Mathematics, Kookmin University

Abstract

The logic-based scheduling language RCPSV may be used to model resource-constrained project scheduling problems with variants for minimizing the project completion time. A diagram-based, nonredundant enumeration algorithm for the RCPSV-problem is proposed and the correctness of the algorithm is proved.

Key Words : Resource-Constrained Project Scheduling Problem, Scheduling Algorithm, Correctness of Algorithm

1. Introduction

Scheduling problems are applied in several industrial fields[1, 2]. Especially, resource constrained project scheduling problems are applied for improving effectiveness of project accomplishment[3, 4, 5]. Kim and Schmidt-Schauss suggested a new approach for representing and solving a general class of non-preemptive resource-constrained project scheduling problems in [6]. The new approach is to represent scheduling problems with variant processes as descriptions (activity terms) in a logic-based terminological language called RSV (resource constrained project scheduling with variant processes). The basic ideas of the new approach stem from the terminological methods of KL-ONE based knowledge representation systems[7,8]. Further this approach allows that the alternative processing possibilities can be formulated not only for each ground activity but also for each subproject.

A further logic-based scheduling language RCPSV may be used to model resource-constrained project scheduling problems with variants for minimizing the project completion time. Based on the definition of the language RCPSV, a calculus can be defined. Using this calculus each expression of the language RCPSV can be transformed into an semantically equivalent normalized expression.

We propose a diagram-based, nonredundant enumeration algorithm that can be applied to the normalized expressions for solving the RCPSV-problem. Further we prove the correctness of the algorithm.

접수일자: 2013년 9월 1일
심사(수정)일자: 2013년 10월 12일
게재확정일자 : 2013년 12월 30일
† Corresponding author
이 논문은 2013년도 국민대학교 교내연구비 지원을 받아 수행된 연구임.

2. The Scheduling Language RCPSV

We define a term-based language RCPSV that may be used to model resource-constrained project scheduling problems with variants.

2.1 The syntax of the language

Definition 2.1 The vocabulary of RCPSV consists of two disjoint sets of symbols. These sets are:

- A finite set of ground activities $\{(i,r(i),d(i))|i=1,\cdots,n(n\in N),r(i)\in R,d(i)\in N\}$ where R is a finite set of resources. Each ground activity is atomic and is associated with a resource $r(i)$ and an activity time $d(i)$ needed for completing it. Each resource can be assigned to only one activity at a time (resource constraint). Activity splitting is not allowed (nonpreemptive case).
- A set of two structural symbols (operators) '**xor**' and '**hnet**'.

The activity-terms of RCPSV are given inductively as follows:

1. Each ground activity is an activity-term.
2. If $t_1,t_2,\cdots,t_k$ are activity-terms, then all terms $(\mathbf{xor}\, t_1,t_2,\cdots,t_k)$ and $\mathbf{hnet}[let\, n_1=t_1,\cdots,n_k=t_k;(n_{11},n_{12}),\cdots,(n_{j1},n_{j2})]$ are activity-terms where $n_1,\cdots,n_k$ are distinct constant symbols (names) and it holds $n_{i1}\neq n_{i2}$ and $\{n_{11},n_{12}\}\cup\cdots\cup\{n_{j1},n_{j2}\}=\{n_1,\cdots,n_k\}$ $(=\{t_1,\cdots,t_k\})$. Further $(\{n_1,\cdots,n_k\},\{(n_{11},n_{12}),\cdots,(n_{j1},n_{j2})\})$ specifies a directed acyclic graph.

The operators **xor** and **hnet** are used for constructing activity-terms and have the following meaning:

- **xor**: This operator can be used to select an activity-term among several different alternative activity-terms. *Exactly one* activity-term among alternatives must be selected and executed.
- **hnet**: This operator specifies the arrangement of activity-terms corresponding to the given precedence relations (*precedence constraint*). In term $\mathbf{hnet}[let\, n_1=t_1,\cdots,n_k=t_k;(n_{11},n_{12}),\cdots,(n_{j1},n_{j2})]$, the operator **hnet** forces $[(n_{11},n_{12}),\cdots,(n_{j1},n_{j2})]$ to specify the precedence relation such that n_{i1} is predecessor of n_{i2} for each $i=1,2,\cdots,j$ for the set of vertices $\{n_{11},n_{12}\}\cup\cdots\cup\{n_{j1},n_{j2}\}$.

2.2 Reduced activity terms

An expression s of RCPSV is called a *reduced* activity-term of an activity-term t of RCPSV, if s can be derived from t by replacing each term of the form $(\mathbf{xor}\ l_1,\cdots l_n)$ in t by exactly one $l_i(i=1,\cdots \text{or}\, n)$ so that s is **xor**-free. Associated with any activity-term t of RCPSV, there exist finitely many different reduced activity-terms which can be derived from t. These reduced activity-terms take partially different paths but complete the same project.

2.3 Schedules

If two ground activities require the same resource at the same time, a resource conflict occurs. The occurring resource conflicts have to be resolved. For a reduced activity-term s let $g(s)=\{a_1,\cdots,a_n\}$ be the set consisting of all ground activities occurring in s. The activity-term s defines a strict partial order $<_s$ on $\{a_1,\cdots,a_n\}$, using the precedence relation. It is generated on the set μ of all subterms of s as follows:

- $\mathbf{hnet}[let\, n_1=t_1,\cdots,n_k=t_k;(n_{11},n_{12}),\cdots,(n_{j1},n_{j2})]\in\mu$ $\Rightarrow n_{i1}<_s n_{i2}$ for all $i=1,2,\cdots,j$.
- $n_1,n_2\in\mu\wedge n_1<_s n_2\Rightarrow n_1'<_s n_2'$ for every subterm n_i' of $n_i, i=1,2$.

Definition 2.2 Let s be a reduced activity-term and $g(s)=\{a_1,\cdots,a_n\}$. An active schedule for s is a set of starting times of ground activities $\{t_{a_i}|a_i\in g(s)\}$ such that:

- The precedence constraints are satisfied: $t_{a_h}+d(a_h)\leq t_{a_i}$ for each a_i and each immediate predecessor a_h with $a_h<_s a_i$,
- The resource constraints are satisfied: $t_{a_m}\geq t_{a_l}+d(a_l)$ or $t_{a_l}\geq t_{a_m}+d(a_m)$ for all $a_l,a_m\in g(s)$ with $r(a_l)=r(a_m)(l\neq m)$ and
- No ground activity can be started earlier without changing other start times: There does not exist another set $\{t_{a_i}'|a_i\in g(s)\}$ with a ground activity a_j which satisfies the precedence and resource constraints such that $t_{a_i}=t_{a_i}'$ for $i\neq j$ and $t_{a_j}>t_{a_j}'$.

The makespan of an active schedule is the duration from the first starting time $\min_i(t_{a_i})$ to the stopping time $\max_i(t_{a_i}+d(a_i))$.

Let t be an acyivity-term. Then the set of active schedules for t is the union of the sets of active schedules for all reduced activity subterms of t.

Time is discrete. Accordingly, for any activity term t the set of active schedules derived from t is finite. In the following all schedules are assumed to be active.

2.4 The Semantics of the language RCPSV

Now, the semantics of RCPSV can be defined as follows:

Definition 2.3 The model-theoretic semantics of activity-terms in RCPSV is given by an interpretation I which consists of the set D (the domain of I) and a function $.^I$ (the interpretation function of I). The set D consists of all active schedules derived from activity-terms in RCPSV. The interpretation function $.^I$ assigns to every activity-term t some subset of D that consists of all active schedules derived from t.

2.5 The Scheduling Problem RCPSV

The objective is minimizing the project makespan. So, we define the scheduling problem RCPSV as follows:

For a given activity-term of RCPSV an active schedule which has the minimal project makespan (project completion time) has to be determined.

$$\frac{(\mathbf{xor}\, t_1, t_2, \cdots, t_k, (\mathbf{xor}\, s_1, s_2, \cdots, s_l), t_{k+2}, t_{k+3}, \cdots, t_n)}{(\mathbf{xor}\, t_1, t_2, \cdots, t_k, s_1, s_2, \cdots, s_l, t_{k+2}, t_{k+3}, \cdots, t_n)} \tag{1}$$

$$\frac{(\mathbf{hnet}[let\, n_1 = t_1, \cdots, n_{k+1} = (\mathbf{xor}\, s_1, s_2, \cdots, s_l), \cdots, n_n = t_n; (n_{11}, n_{12}), \cdots, (n_{h1}, n_{k+1}), \cdots, (n_{j1}, n_{j2})])}{\begin{array}{r} (\mathbf{xor}\, \mathbf{hnet}[let\, n_1 = t_1, \cdots, n_{k+1} = s_1, \cdots, n_n = t_n; (n_{11}, n_{12}), \cdots, (n_{h1}, n_{k+1}), \cdots, (n_{j1}, n_{j2})], \\ \mathbf{hnet}[let\, n_1 = t_1, \cdots, n_{k+1} = s_2, \cdots, n_n = t_n; (n_{11}, n_{12}), \cdots, (n_{h1}, n_{k+1}), \cdots, (n_{j1}, n_{j2})], \\ \vdots \qquad\qquad\qquad\qquad \\ \mathbf{hnet}[let\, n_1 = t_1, \cdots, n_{k+1} = s_l, \cdots, n_n = t_n; (n_{11}, n_{12}), \cdots, (n_{h1}, n_{k+1}), \cdots, (n_{j1}, n_{j2})]) \end{array}} \tag{2}$$

$$\frac{(\mathbf{hnet}[let\, n_1 = t_1, \cdots, n_{k+1} = (\mathbf{xor}\, s_1, s_2, \cdots, s_l), \cdots, n_n = t_n; (n_{11}, n_{12}), \cdots, (n_{k+1}, n_{h2}), \cdots, (n_{j1}, n_{j2})])}{\begin{array}{r} (\mathbf{xor}\, \mathbf{hnet}[let\, n_1 = t_1, \cdots, n_{k+1} = s_1, \cdots, n_n = t_n; (n_{11}, n_{12}), \cdots, (n_{k+1}, n_{h2}), \cdots, (n_{j1}, n_{j2})], \\ \mathbf{hnet}[let\, n_1 = t_1, \cdots, n_{k+1} = s_2, \cdots, n_n = t_n; (n_{11}, n_{12}), \cdots, (n_{k+1}, n_{h2}), \cdots, (n_{j1}, n_{j2})], \\ \vdots \qquad\qquad\qquad\qquad \\ \mathbf{hnet}[let\, n_1 = t_1, \cdots, n_{k+1} = s_l, \cdots, n_n = t_n; (n_{11}, n_{12}), \cdots, (n_{k+1}, n_{h2}), \cdots, (n_{j1}, n_{j2})]) \end{array}} \tag{3}$$

그림 1. 계산 법칙 (1), (2), (3)
Fig. 1. Rules (1), (2), (3)

3. A Calculus for the Scheduling Language RCPSV

Similar to RSV [6], there are activity-terms which are syntactically different, but semantically equivalent. Based on the semantics of RCPSV, we can define a calculus called RCPSV-calculus, which transforms a term into another semantically equivalent term of it.

Definition 3.1 If $t_1, t_2, \cdots, t_k, s_1, \cdots, s_l, t_{k+2}, \cdots t_n$ are activity-terms, the calculus has the associative rule (1) and distributive rules (2, 3) (See Figure 1). The associative rule (1) describes a subexpression combined by '**xor**' which is an argument of the operator '**xor**' may be flattened. The distributive rules (2, 3) describe if a subexpression combined by '**xor**' occurs as an argument of the operator '**hnet**', the operator '**xor**' may be moved to the leftmost position.

In rule 2 (3) the right (left) component n_{k+1} of (n_{h1}, n_{k+1}) $((n_{k+1}, n_{h2}))$ corresponds to an expression combined by '**xor**'. In the following we formalize the correctness of the RCPSV-calculus.

Lemma 3.2 The RCPSV-calculus is a correct calculus.
Proof. A rule in the form

$$\frac{A}{B}$$

is "correct" iff the interpretations of the upper expression A and the lower expression B are identical ($A^I = B^I$). In the following we show this for each of the 3 rules.
Rule (1): The sets of all active schedules derived from the upper and lower expression are obviously identical, because both expressions describe that exactly one of the activity-terms $t_1, t_2, \cdots, t_k, s_1, \cdots, s_l, t_{k+2}, \cdots t_{n-1}$ and t_n must be selected and executed. Let $M_{t_i}(M_{s_i})$ be the set of all schedules derived from $t_i(s_i)$. The interpretation of the upper and lower expression corresponds then to the union of the sets $M_{t_1}, \cdots M_{t_k}, M_{s_1}, \cdots, M_{s_l}, M_{t_{k+2}}, M_{t_n}$. Therefore, the following equation holds:

$$(\mathbf{xor}\; t_1, t_2, \cdots, t_k, (\mathbf{xor}\, s_1, \cdots, s_l, t_{k+2}, \cdots t_n)^I$$
$$= (\mathbf{xor}\; t_1, t_2, \cdots, t_k, s_1, \cdots, s_l, t_{k+2}, \cdots t_n)^I$$

Rule (2) : For the $k+1$-st argument of the operator '**hnet**' a choice possibility exists. One of the l activity-terms $s_1, \cdots, s_{l-1}$ and s_l must be selected. For each choice of $s_i (i = 1, \cdots, l)$ a set of all schedules derived from the expression

$$(\mathbf{hnet}[let n_1 = t_1, \cdots, n_{k+1} = s_i, \cdots, n_n = t_n;$$
$$(n_{11}, n_{12}), \cdots, (n_{h1}, n_{k+1}), \cdots, (n_{j1}, n_{j2})])$$

denoted by M_{s_i} is determined. Then the set of all schedules which may be derived from the upper expression of the rule (2) corresponds to the union of the sets $M_{s_1}, \cdots M_{s_{l-1}}$ and M_{s_l}. Further this union corresponds to the set of all schedules which may be derived from the lower expression. So, the interpretations of the upper expression and the lower expression are identical.
Rule (3): This may be proved similar to rule (2). □

The correctness of the RCPSV-calculus permits to formalize the following theorem.

Theorem 3.3. For any activity description t of

RCPSV all operators '**xor**' in the interior of t always may be moved to the leftmost position, so that t is transformed to a semantically equivalent, normalized activity-term s in which the operator '**xor**' can occur uniquely once in the leftmost position combining all reduced activity-terms derived from t.

Proof: It is sufficient to show that for any expression t of RCPSV all derivations terminate in a normalized expression. This is the case when no further RCPSV-rules can be applied (see figure 1). Using an innermost strategy and induction on the number of occurrences of '**xor**'s it is easy to see that in every expression containing the '**xor**'-operator, it can be shifted to the topmost position. Thus all '**xor**'-operators in the interior of the activity-term t may be moved stepwise to the left until at most one topmost '**xor**' remains. □

In theorem 3.3, it follows from semantical equivalence of an activity-term t and its normalized activity-term s that a schedule which is optimal for t is optimal for s too and vice versa. But for a normalized activity-term we can consider the arguments (reduced activity-terms) of the '**xor**'-operator separately in order to compute optimal schedules. So, because of theorem 3.3 the RCPSV-problem can be solved, while first transforming each activity-term t into a semantically equivalent, normalized activity-term s and then computing the schedules with the minimal project makespan for every reduced activity-term of s separately using a solution algorithm for solving the classical RCPS-problem. The schedules among all these computed schedules that have the minimal value correspond then to the optimal schedules of t.

4. Solving the RCPSV-problem using diagram-based algorithm

Many varieties of branch-and-bound-based implicit enumeration methods for solving the RCPS-problem which may be also used for determining the optimal schedules for each reduced activity-term of RCPSV have been reported[3,4,5,9]. Further Kim and Schmidt-Schauss presented a new diagram-based method for representing and solving reduced activity-terms of RSV [6]. In this section we describe an diagram-based algorithm for solving reduced activity-terms of RCPSV which we call the RCPSV-algorithm and it is similar to the method of RSV[6].

A diagram has a time axis and a scan-line. In the diagram each ground activity has a left and right end point (a start and end time). The left and right end point of any ground activity i denoted by $LE(i)$ and $RE(i)$ are referred to as the *stopping times* of the scan-line. (D,t) with $t \geq 0$ denotes the scan-line is found at the stopping time $t_{SL} = t$ in the diagram D. Instead of a continuous moving, the scan-line jumps from a stopping time into the next right stopping time while determining and then resolving resource conflicts. For each given reduced activity-term, a directed acyclic graph (X,P) with $X = \{(i,r(i),d(i)|\ i=1,\cdots n, r(i) \in R, d(i) \in N\}$ and $P = \{(i,j)|i,j \in \{1,\cdots n\}$ can be determined. So, the objective is to determine an optimal active schedule

$\{(i,s(i)|i=1,\cdots,n,s(i) \in N\}$

subject to all constraints where $s(i)$ describes the starting time of i.

In the following we will omit descriptions of the parts equaling to the method of RSV and take the notations and definitions which have been used for the method of RSV if they can be consistently applied to the algorithm.

In the beginning the scan-line is found at the time $t_{SL} = 0$ and the diagram is empty.

Step 1: *Attaching start ground activities to the scan-line*: First, all start ground activities out of X which have no predecessors are attached to the scan-line. "Attaching a ground activity i to the scan-line" means that i is placed in the diagram so that the time at which the scan-line is found is assigned to i as its start time.

Step 2: *Moving the scan-line*: The scan-line jumps to the next stopping time.

Step 3: *Determining and resolving resource conflicts (Multiplying the diagram by the number of the existing conflict combinations); Freezing all definitely placed ground activities; Deleting all definitely placed ground activities from the directed graph*: This step is carried out just as in A_{RSV}.

Step 4: *Attaching further ground activities to the scan-line*: Further ground activities out of X which can be attached to the scan-line are determined in order to place them. For an actual diagram (D,t_{SL}) and an actual (X,P) a ground activity in X can be attached to the scan-line iff

1. i isn't from the diagram (D,t_{SL}),
2. in (D,t_{SL}) there exists no frozen activity j with $r(i)=r(j)$ for which $LE(j) < t_{SL}$ and $RE(j) > t_{SL}$ hold.
3. all immediate predecessors e of i, that is, for e and i $(e,i) \in P$ holds, are already elements of (D,t_{SL}) and for all e $RE(e) \leq t_{SL}$ holds in (D,t_{SL}).

Furthermore the steps 2, 3, 4 are recursively applied until all ground activities have been placed in the diagram, all activities in the diagram have been frozen and the actual graph is empty so that a schedule is completed. Among all computed schedules, those that have the minimal project makespan, are delivered as the optimal schedules for the given reduced activity term.

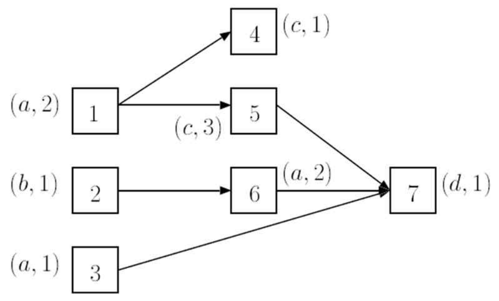

그림 2. 그래프
Fig. 2. A graph

Example 4.1. We consider the following reduced activity term.

hnet$[let n_1 = (1,a,2), n_2 = (2,b,1), n_3 = (3,a,1), \cdots, n_7 = (7,d,1); (n_1,n_4), (n_1,n_5), (n_2,n_6), (n_3,n_7), (n_5,n_7), (n_6,n_7)]$

There are start activities 1,2 and 3 (See Figure 2). The diagram $(D,0)$ of figure 3 shows the resulting diagram after applying the step 1, in which the scan-line time 0 has been assigned to these start activities 1,2 and 3 as their start time. In the next step 2, the scan-line jumps into the next stopping time $t_{SL} = 1$. There is one 1-time conflict free activity 2. In addition, an 1-time resource conflict occurs since there is an activity i such that $RE(i) = t_{SL}(=1)$ (e.g. the activity 3) and the resource $r(i)$ is required by more than one activity in the time interval $[t_{SL}-1, t_{SL}]$. So there exist two 1-time conflict combinations [1] and [3]. The diagram $(D,1)$ is duplicated, let these be D_1, D_2 and [1], [3] are assigned to D_1, D_2 respectively. In each diagram, the 1-time conflict free activity 2 and the assigned 1-time conflict activity are frozen and the other 1-time conflict activity is moved behind the frozen conflict activity. Subsequently we proceed with the next step 4 in each diagram. If we pursue $(D1,1)$ to which the combination [1] is assigned, we have the diagram $(D1,1)$ of figure 3 where 1 and 2 have been frozen and 3 has been moved behind 1. We delete all definitely placed ground activities (all frozen ground activities) from the graph. In the next step 4, further ground activities out of the graph which can be attached to the scan-line of $(D1,1)$ should be determined. The activities 4 and 5 can not be attached since it holds $RE(1) > t_{SL}(=1)$ for their immediate predecessor 1. The activity 6 can not be attached too since in $(D1,1)$, there exists a frozen activity 1 with $r(1) = r(6) = a$ for which $RE(1) > t_{SL}(=1)$ holds. The resource a is being blocked until the time 2. So there is no activity to be attached to the scan-line. We proceed with the next step 2. After moving the scan-line in $(D1,1)$, we have diagram $(D1,2)$. Since in $(D1,2)$ no 2-time resource conflict occurs and no 2-time conflict free activity exists, we proceed with the next step 4 in which the activities 4, 5 and 6 can be attached to the scan-line. After placing these activities, the scan-line jumps to the next stopping time $t_{SL} = 3$. The diagram $(D1,3)$ of figure 3 shows the resulting diagram. Now, there are two 3-time conflict resources a and c. So there exist four 3-time conflict combinations [3,4],[3,5],[6,4] and [6,5] altogether. The diagram $(D1,3)$ is multiplied 4 times, let these be $D11, \cdots, D14$ and [3,4],[3,5],[6,4], [6,5] are assigned to $D11, \cdots, D14$ respectively. If we pursue the diagram $D12$, and apply the further steps recursively, one active schedule is finally generated that the diagram $(D12,6)$ of figure 3 shows. In this way, 8 nonredundant active schedules are computed altogether. Two of these require the minimal project makespan 6.

4. Proving correctness of the RCPSV-algorithm

A correctness proof for the RCPSV-algorithm is given as follows:

Theorem 4.1 For any given reduced activity-term s, the RCPSV-algorithm generates nonredundantly all active schedules which may be derived from s.

Proof. We show the theorem through induction on the number of ground activities (vertices of the corresponding graph).

Induction base: If s is a ground activity, the proof is trivial.

Induction step: In the beginning the scan-line is found at the time $t_{SL} = 0$. After applying the first step all start activities of s are attached to the scan-line. In the following the scan-line jumps to the next stopping time l. Let $a_1, \cdots, a_n$ be all l-time direct scan-line activities, i.e. it holds that $RE(a_1) = RE(a_2) = \cdots RE(a_n) = l$. Now, the following 2 different cases have to be distinguished:

Case 1: There is at least one activity a_i which corresponds to a l-time conflict-free activity. First, all l-time conflict-free activities are frozen and then the occurring resource conflicts are resolved. Here, let the diagram be multiplied to k diagrams $D_1, \cdots D_k$ so that each l-time conflict combination is assigned to a diagram respectively. After resolving the

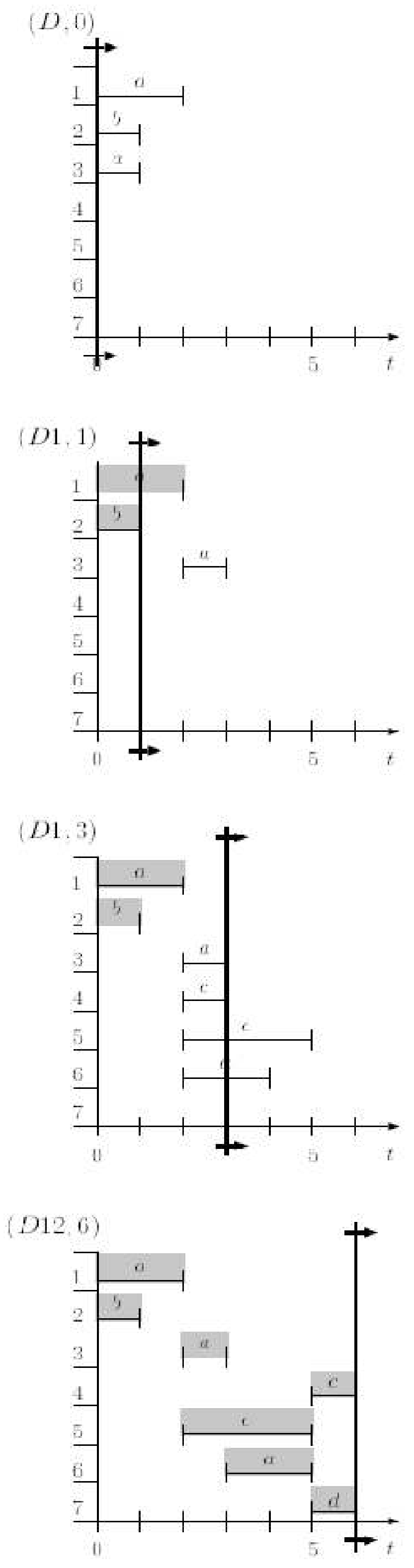

그림 3. RCPSV-다이아그램을 기초로한 계산
Fig. 3. A RCPSV-diagram based calculation

resource conflicts, for every diagram all l-time direct scan-line ground activities are deleted from the graph respectively. So, in every diagram the size (the number of ground activities) of the corresponding actual graph becomes smaller since at least one l-time conflict-free activity a_i is deleted. Furthermore, the RCPSV-algorithm is applied recursively to every diagram accompanied by the corresponding actual graph. By assumption of induction, the RCPSV-algorithm generates nonredundantly all active schedules for every diagram since the size of every corresponding actual graph is smaller than the size of the graph of s. Moreover, A_{RCPSV} generates nonredundantly all active schedules for s since $D_1, \cdots D_k$ are pairwise different. It is obviously true for the case $k=1$, i.e. all $a_1, \cdots, a_n$ are l-time conflict-free too.

Case 2: Each activity a_i is involved in a l-time resource conflict, i.e. there is no l-time conflict-free activity. First the occurring resource conflicts are resolved. Let the diagram be multiplied to k diagrams $D_1, \cdots D_k$ so that each l-time conflict combination is assigned to a diagram respectively. For any D_i the following two subcases have to be distinguished:

Case 2.1: There is at least one l-time direct scan-line activity a_i which is frozen. This case is very similar to the case 1 and can be shown correspondingly that from D_i the RCPSV-algorithm generates nonredundantly all active schedules.

Case 2.2: None of the l-time direct scan-line activities $a_1, \cdots, a_n$ is frozen. Then, there are further l-time conflict activities which are frozen. Let these be $h_1, \cdots h_m$ where $RE(h_j) > l$ for each j. Eventually $h_1, \cdots h_m$ will be deleted from the graph and in the result the size of the graph will become smaller. So, by assumption of induction, the RCPSV-algorithm generates nonredundantly all active schedules.

Consequently, for the case 2, the RCPSV-algorithm generates nonredundantly all active schedule for s since $D_1, \cdots D_k$ are pairwise different.

5. Conclusion

Another terminological scheduling language RCPSV, similar to RSV, may be used to formulate and solve a new general class of RCPSV-problems. The terminological logic RCPSV offers an effective approach for solving the RCPSV-problem. We introduced a solution algorithm based on a scan-line principle, through which all active schedules could be generated for any reduced activity term. The correctness proof of the solution algorithm shows the algorithm generates nonredundantly all active schedules which may be derived from any given reduced activity-term.

References

[1] C.-k. Kwon and K.-s. Oh, “Genetic algorithms based on maintaining a diversity of the population for jobshop scheduling problem,” *Journal of Korean Institute of Intelligent Systems*, vol. 11, no. 3, pp. 191 - 199, 2001.

[2] K. M. Lee and K. M. Lee, “Task allocation and scheduling of multiagent systems with fuzzy task processing times,” *Journal of Korean Institute of Intelligent Systems*, vol. 14, no. 3, pp. 324 - 329, 2004.

[3] E. Demeulemeester and W. Herroelen, “A Branch-and-Bound Procedure for the Multiple Resource-Constrained Project Scheduling Problem,” *Management Science*, Vol. 38, No. 12, pp. 1803-1818, 1992.

[4] E. Demeulemeester and W. Herroelen, “New Benchmark Results for the Resource-constrained Project Scheduling Problem,” *Management Science,*, Vol. 43, No. 11, pp. 1485-1492, 1997.

[5] A. Mingozzi, V. Maniezzo, S. Ricciardelli and L. Bianco, “An exact Algorithm for Project Scheduling with Resource Constraints based on a New Mathematical Formulation,” *Management Science*, Vol. 44, No. 5, pp. 714-729, 1998.

[6] P. S. Kim and M. Schmidt-Schauss, “A Term-Based Approach to Project Scheduling,” *Lecture Notes in Artificial Intelligence Series 2120*, p. 304 ff., Springer-Verlag, 2001.

[7] F. M. Donini, M. Lenzerini, D. Nardi and W. Nutt, “The Complexity of Concept Languages,” *Information and Computation*, Vol. 134, No. 1, pp. 1-58, 1997.

[8] B. Nebel and K. von Luck, “Hybrid Reasoning in BACK,” *Methodologies for Intelligent Systems*, North Holland, Amsterdam, Netherlands, pp. 260-269, 1988.

[9] P. Brucker, S. Knust, A. Schoo and O. Thiele, “A Branch and Bound Algorithm for the Resource-constrained Project Scheduling Problem,” *European Journal of Operational Research*, Vol. 107, pp. 272-288, 1998.

저 자 소 개

김복선 (Pok-Son Kim)
1987년: 국민대학교 수학과 이학사
1995년: 독일 프랑크푸르트 괴태대학교 컴퓨터과학 석사
2001년: 독일 프랑크푸르트 괴태대학교 정보과학전공 박사
2002～현재 국민대학교 수학과 부교수

2009년～현재 : 한국지능시스템학회 이사

관심분야 : Complexity Theory, Merging and Sorting Algorithms, Scheduling Problems
Phone : +82-2-910-4747
E-mail : pskim@kookmin.ac.kr