A Model To Optimize Projekt Resource Allocation By Construction Of A Balanced Histogram

by

Dr. Univ. Petros Konstantinidis

Ph.D Thesis

University of Pécs
Faculty of Business and Economics

2002

A MODEL TO OPTIMIZE PROJECT RESOURCE ALLOCATION BY CONSTRUCTION OF A BALANCED HISTOGRAM

by

Dr. Univ. Petros Konstantinidis

Dipl. Civil-Engineer

TEI of Athens, Greece

Ph.D Thesis

Supervisor

Dr. Habil György Csébfalvi CSc

Department of Business Informatics

University of Pécs

Faculty of Business and Economics

2002

Abstract

Resource scheduling is a challenging but potentially very frustrating NP-hard problem of project management. There are two basic categories of resource scheduling *resource allocation* and *resource leveling*.

In the *resource allocation* problem, the resource-availabilities are limited. The scheduling objective is to keep the project completion time as close as possible to the critical path length such that the resource constraints are met.

In the *resource leveling* (balancing) problem the resource availabilities are unlimited but the maximal project makespan is fixed. The scheduling objective is to minimize some measure of the *variation* of the resource usage.

The main objective of this thesis is to demonstrate how modeling ideas and solution procedures, borrowed from the field of production scheduling, can be used to formulate and solve several types of project scheduling (resource balancing) problems.

In *production scheduling*, the idle time plays a central role. We hope to prove that the idle time may play a similar role in project scheduling and it may be the starting base of several new *"global"* performance measures.

In the first part of the thesis we introduce a new *interruption* oriented global resource leveling measure. In the proposed new approach, the desired resource profile is defined as *quasi-concave* one. The proposed new measure based on the *total number of interruptions*.

In the second part of the thesis, we present new *exact* interruption oriented resource balancing (leveling) models for *single-mode*, *multi-mode*, and *double-constrained multi-mode* projects.

Theoretically, every new model is formulated as a *mixed integer linear programming problem* (MILP), which may be solved directly in the case of small-scale projects with the help of any commercial MILP package. In our study, the LINDO system was used to solve the MILP problems.

According to the NP-hard nature of the resource-balancing problem, for medium-size projects we present tree-search *implicit enumeration* algorithms with effective pruning rules for every case. In the case of double-constrained multiple-mode projects, the proposed algorithm is formulated as a *multi-level "tree of trees"* like searching problem.

For large-scale problems, simple *"beam-search"* heuristics may be developed by constraining the size of the searching-trees.

To develop and run the *tree-search* algorithms, generate the LINDO input files, draw projects and searching trees the *ProMan* system developed by A. Ghobadian and Csébfalvi (1995), Csébfalvi (2002) was used.

In order to illustrate the essence of the proposed models computational results for several sample problems are presented.

In the last part of the thesis, we summarize our most important new results.

TABLE OF CONTENTS

1. INTRODUCTION

Resource scheduling is a challenging but potentially very frustrating NP-hard problem of the project management. There are two basic categories of resource scheduling resource allocation and resource leveling (for example Brucker et al. (1999) and Weglarz (1999)).

In the *resource-allocation problem* (RAP), the resource-availabilities are limited. The scheduling objective is to keep the project completion time as close as possible to the critical path length such that the resource constraints are met.

In the *resource-balancing problem* (RLP) the resource availabilities are unlimited but the maximal project completion time is fixed. The scheduling objective is to minimize some measure of the variation of the resource usage. The best known procedure for the RLP is the Burgess-Killebrew (1962) heuristics.

Very little work has been done to solve RLP exactly. An explicit enumeration approach has been devised by Ahuja (1976). A non-serial dynamic programming approach has been proposed by Bandelloni et al. (1994) and an integer programming approach has beep developed by Younis and Saad (1996). An implicit enumeration approach has been discussed by Zimmermann and Engelhart (1998) and Neumann and Zimmermann (1999).

All of the works mentioned above, connected to the traditional resource balancing measures and single-mode projects.

The traditional measures used in resource leveling are the following: maximal resource usage (MU), resource usage fluctuation around average (FA), and resource usage fluctuation between consecutive periods (FB).

The concave resource profile as a desired shape was mentioned by several authors without formal description or models. For example: Tavares (1987), Hajdu (1997), and Lova et al. (2000).

To the best of our knowledge, Konstantinidis (1998) was the first who presented a multi-mode heuristic with a new interruption oriented resource balancing measure based on quasi-concavity.

The first exact resource balancing procedure based on quasi-concavity was presented by Csébfalvi and Konstantinidis (1998, 1999) for single-mode projects.

The first exact resource balancing procedure based on quasi-concavity with hard and soft resource constraints was presented by Kruzslicz (2002) for single-mode projects.

2. DEFINITIONS AND NOTATIONS

A project is a set of interrelated activities. Without loss of generality, we assume that a project can be depicted by an acyclic activity-on-node (AoN) graph where activities are numerically labeled such that successor activities always have higher numbers (labels) than all their predecessors.

Associated with each activity is a set of possible durations and the corresponding resource requirements, which would permit the activity to be completed in the stated durations.

Each duration-resource combination is called *"activity-operating mode"* or simply a *"mode"*. For example, it may be possible to complete an activity in 5 days using a skilled person (mode 1), or in 7 days using an unskilled person (mode 2). In the single-mode case, every activity has exactly one operating mode.

In keeping with the general nature of the model, three resource categories are included. Using the terminology introduced by Talbot (1982) these resources are defined as renewable, nonrenewable or doubly constrained.

If resources are available in each period, the resources are considered *renewable*. An example of such a resource would be skilled labor: the number of skilled laborers available to work on the project each day is limited, although no constraint is placed on the number of days skilled labor may be used. Thus, the resource labor is renewed each period to a predetermined level.

Note that, in the resource-leveling problem we usually assume that the renewable resource availabilities are unlimited.

If the total consumption of the resource over the life of the project is constrained, it is called *nonrenewable*. Money is perhaps the best example of

a nonrenewable resource: total project costs are frequently limited to a fixed predetermined contract price.

Finally, resources are defined as *double constrained* if both their daily (per-period) and total availability is limited. Money is again probably the best example of a double constrained resource: cash flows per day as well as total cash expended on a project are often restricted.

A formal definition of the project is the following:

We consider a project consisting of N *real* activities $1, 2, \ldots, N$ and introduce *dummy* activities 0 and $N+1$ which represent the beginning and the completion of the project respectively.

The project modeled by an activity-on-node (AoN) network $G = (V, E)$, where each node $i \in V$ represents an activity $i, (i = 0, 1, \ldots, N+1)$. An arc $i \rightarrow j \in E$ represents a *finish-start precedence relation* between activity i and activity j.

We assume that the periods are labeled $t \in \{0,1,\ldots,T,T+1\}$ where T is a prescribed *maximal project duration*. In our notation period $T+1$ is the place of the fixed *finishing milestone* activity $N+1$ ($T+1$ is the first free period after finishing the project).

A vector of integer start times $S = (S_0, S_1, \ldots, S_{N+1})$ which assigns a start time $S_i \in \{0, 1, \ldots, T+1\}$ to each activity is called a *schedule*. In this study, we deal with the single-project case, therefore $S_0 = 0$.

The maximal project duration is fixed so $S_{N+1} = T + 1$ and the resource balancing procedure does not modify it.

We will say that a schedule is *network-feasible* if it satisfies the network-relations. Let $\Re = \Re(E)$ be the set of network-feasible schedules.

Let R denote the number of renewable resources required for carrying out the project.

2.1 Single-mode case

In the *single-mode case*, the duration of activity i $i \in \{0, 1, \ldots, N, N+1\}$ is denoted by $D_i \in \{1, 2, \ldots\}$.

Let $AS = \{AS_{it} \mid i = 1,2,\ldots, N; t = ES_i, \ldots, LS_i\}$ denote the set of *zero-one* starting time variables, where ES_i (LS_i) is the earliest (latest) starting times for activity i. As usual, $AS_{it} = 1$ if activity i starts in period j, and 0 otherwise.

In a schedule, every activity has exactly one starting time, therefore $\sum_{ES_i}^{LS_i} AS_{it} = 1$ and $S_i = \sum_{ES_i}^{LS_i} t * AS_{it}$ for $i = 1,2,\ldots, N$.

Assume that, each resource r where $r \in \{1,2,\ldots, R\}$ has unlimited resource availability and each real activity $i \in V' = V \setminus \{0, N+1\}$ uses $r_{ir} \geq 0$ units of resource r over its duration.

2.2 Multiple-mode case

In the *multiple-mode* case, let $M_i \quad i \in V' = V \setminus \{0, N+1\}$ denote the number of possible operating-modes for activity i.

Let $AMS = \{AMS_{imt} \mid i = 1,2,...,N; m = 1,...,M_i; t = ES_{im},...,LS_{im}\}$ denote the set of *zero-one* starting time variables, where $ES_{im}(LS_{im})$ is the earliest (latest) starting time in operation-mode m, where $m \in \{1,2,...,M_i\}$. As in the single-mode case, $AMS_{imt} = 1$ if activity i starts in mode m in period t, and 0 otherwise.

In a schedule, every activity has exactly *one* operating-mode and exactly *one* starting time, therefore $\sum_{m=1}^{M_i} \sum_{ES_{im}}^{LS_{im}} AMS_{imt} = 1$ and $S_i = \sum_{m=1}^{M_i} \sum_{ES_{im}}^{LS_{im}} t * AMS_{imt}$ for $i = 1,2,...,N$.

In the *multiple-mode* case, the duration of activity i, where $i \in V'$ is denoted by $D_{im} \in \{1, 2, ...\}$ where $m \in \{1,2,...,M_i\}$

Assume again that, each resource r where $r \in \{1,2,...,R\}$ has unlimited resource availability and each real activity $i \in V'$ uses $r_{imr} \geq 0$ units of resource r over its duration in mode m, where $m \in \{1,2,...,M_i\}$.

2.3 Double constrained multiple-mode case

In the *double constrained* case, let $M_i \; i \in V' = V \setminus \{0, N+1\}$ denote the number of possible operating-modes for activity i.

Let $AMS = \{AMS_{imt} \mid i = 1,2,...,N; m = 1,...,M_i; t = ES_{im},...,LS_{im}\}$ denote the set of *zero-one* starting time variables, where $ES_{im}(LS_{im})$ is the earliest (latest) starting time in operation-mode m, where $m \in \{1,2,...,M_i\}$. As in the single-mode case, $AMS_{imt} = 1$ if activity i starts in mode m in period t, and 0 otherwise.

In the *double constrained* case, the duration of activity i where $i \in V'$ is denoted by $D_{im} \in \{1, 2, ...\}$, where $m \in \{1,2,...,M_i\}$

Assume again that, each resource r where $r \in \{1,2,...,R\}$ has unlimited resource availability and each real activity $i \in V'$ uses $r_{imr} \geq 0$ units of resource r over its duration in mode m, where $m \in \{1,2,...,M_i\}$.

In the *double constrained* case, the per-period activity cost for activity i is denoted by C_{im} where $i \in V'$ and $m \in \{1,2,...,M_i\}$. In this study, without loss of generality we assumed that only the activity *cost* is *nonrenewable* resource. The *maximum daily* (per-period) *cash flow* and the *maximum project cost* are denoted by $\overline{DC}$ and $\overline{PC}$ respectively. We note that, the essence of the proposed models is not effected by this simplification and more or less agree to the everyday project- managing practice.

3. A NEW INTERRUPTION MEASURE

The objective of all resource leveling models is the determination of start time $S_i \in \{ES_i,...,LS_i\}$ for each activity $i \in V'$ on the set of network-feasible schedules $S \in \mathfrak{R}$, which minimizes the given measure of performance.

Let U_{tr} be the amount of a resource used in period t in a schedule $S \in \mathfrak{R}$ where $t \in \{1,2,...,T\}$, $r \in \{1,2,...,R\}$.

Let $U_r = \{U_{1,r}, U_{2r},..., U_{Tr}\}$ be the so-called *resource profile* for resource $r \in \{1,2,...,R\}$.

The traditional *fluctuation-oriented* leveling (balancing) measures for a schedule $S \in \mathfrak{R}(E)$ and a resource r where $r \in \{1,2,...,R\}$ are as follows:

Resource usage fluctuation around average usage:

$$FA_r(S) = \sum_{t=1}^{T} \left(U_{tr} - \overline{U}_r \right)^2, \text{ where } \overline{U}_r = \frac{1}{T} \sum_{t=1}^{T} U_{tr} \qquad \text{(FA)}$$

Resource usage fluctuation between consecutive periods:

$$FB_r(S) = \sum_{t=2}^{T} \left(U_{tr} - U_{t-1r} \right)^2 \qquad \text{(FB)}$$

It is important to note, that the *traditional* fluctuation oriented resource leveling measures, namely the *resource usage fluctuation around average* (FA) and the *resource usage fluctuation between consecutive periods* (FB) measures are essentially *"local"* measures. They concentrate on *local* fluctuations, so they are unable to characterize the resource profile from a global point of view. The *"dream shape"* of the traditional fluctuation oriented resource leveling measures is defined as a rectangle. Unfortunately, in practice the dream usually remains a dream, so the *"optimal shape"* may *be "all hills and dales"*, which from managerial point of view always means undesirable idle times and *interruptions*.

In this chapter, we introduce a new *"global"* resource leveling measure, which is able to evaluate the resource profile from a global point of view.

In the proposed approach, we define the desired resource profile as a *"quasi-concave"* one and introduce a penalty function as a new resource leveling measure, which penalizes the deviations from the desired quasi-concave profile. The penalty is the *number of interruptions*.

The objective is to demonstrate how modeling ideas and solution procedures, borrowed from the field of production scheduling, can be used to formulate and solve several types of project scheduling problems. In production, scheduling the idle time plays a central role.

We hope to prove that the idle time may play a similar role in project scheduling and it may be the starting base of several new "global" performance measures.

In general, a resource unit is idle in a time unit when it is available for the project but it is not assigned to any activity. The *"active life"* of a resource unit may be much diversified in the life of a project. It is not necessarily continuous, and it may be interrupted several times, which always mean undesirable intermediate idle periods.

The first (last) active period of a resource unit not necessarily coincides with the first (last) active period of the whole project (for example: waiting, working, waiting, waiting, working, and working…).

According to a *"lifelike"* definition, a *quasi-concave* resource profile has no *interruptions*; it is ascending, descending or ascending and descending, like a hill. (Note that, a rectangle is a quasi-concave shape.). Figure 3-1 illustrates a typical quasi-concave resource profile.

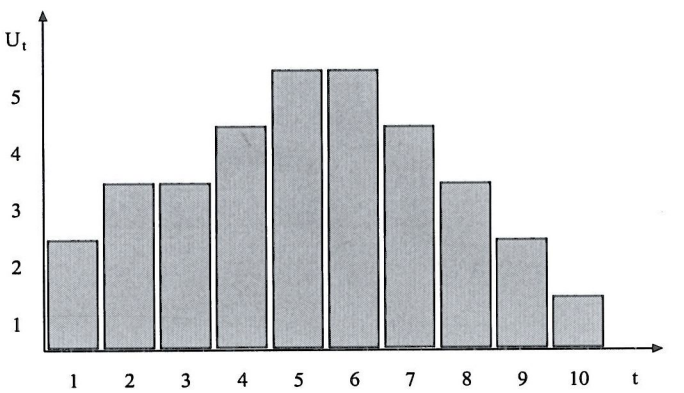


Figure 3-1 A typical quasi-concave resource profile

A formal definition of *quasi-concavity* would be the following:

Let $U_r = \{U_{1,r}, U_{2r}, ..., U_{Tr}\}$ be the so-called resource profile for resource $r \in \{1, 2, ..., R\}$.

Let $MAXL_{tr} = Max(U_{1r}, U_{2r}, ..., U_{tr})$, where $t \in \{1, 2, ..., T-2\}$.

Let $MAXR_{tr} = Max(U_{3r}, U_{4r}, ..., U_{tr})$, where $t \in \{3, 4, ..., T\}$.

The resource profile $U_r = \{U_{1r}, U_{2r}, ..., U_{tr}, ..., U_{T-1r}, U_{Tr}\}$ is *quasi-concave* if

$$U_{tr} \geq Min(MAXL_{t-1r}, MAXR_{t+1r}) \quad (3\text{-}1)$$

for $\forall t \in \{2, 3, ..., T-1\}$

The *quasi-concave hull* $H_r = \{H_{1r}, H_{2r}, ..., H_{tr}, ..., H_{T-1r}, H_{Tr}\}$ for the resource profile U_r is the following:

$$H_{1r} = U_{1r} \quad (3\text{-}2)$$

$$H_{tr} = Max(Min(MAXL_{t-1r}, MAXR_{t+1r}), U_{tr})$$

for $t \in \{2, 3, ..., T-1\}$

$$H_{Tr} = U_{Tr}$$

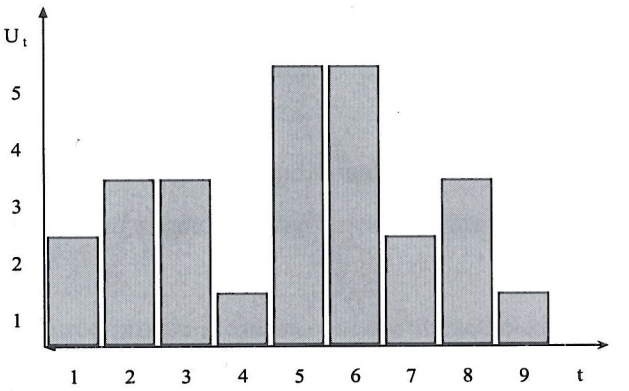


Figure 3-2 A typical non-quasi-concave resource profile

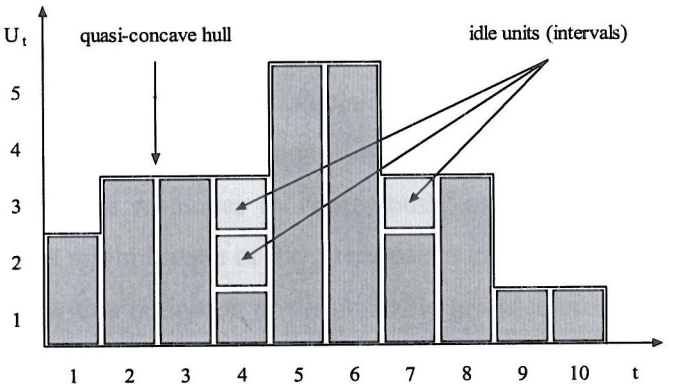


Figure 3-3 The quasi-concave hull and the intermediate idle units (intervals)

Figure 3-2 illustrates a typical *non-quasi-concave* resource profile. Its *quasi-concave hull* and the intermediate idle resource units (intervals) are shown in Figure 3-3. In this example, the first resource unit (a worker or a machine) works continuously (it is always busy), but the second resource

unit is idle in the fourth period, and finish the work a bit earlier. The total number of the idle resource units is three.

The reasons of replacing the traditional *"local fluctuation oriented"* measures by new *"global shape oriented"* measures are the following:

1. When a resource profile is *quasi-concave* then the idle resource units are clustered at the beginning and at the end of the project, so they can be used more efficiently. For instance, in a multi-project environment this clustered idle resource units (intervals) may be transferred to other projects.

2. The idle time (the interruption) minimization and the traditional local fluctuation minimization are not the same, and the management is much more interested in the *intermediate idle time* (*interruption*) minimization than in the *"local"* fluctuation minimization. It is clear that the project cost can be reduced by using the resources as continuously as possible. In fact, it is equivalent with a minimization of the intermediate idle periods (intervals) or, in other words, with a definition of the desirable resource profile as a *quasi-concave* one.

Now we present a new global *interruption oriented resource leveling (balancing) measure* (IN) based on the *number of intermediate idle intervals*.

The objective function of the approach is the total number of such intervals (the sum of the *interruptions*, or in other words, the sum of the restarting events).

Note that, this model belongs to a larger model family. This family is closely related to the *setup time (setup cost) oriented production-scheduling models* because an idle interval before the first active period (an intermediate idle interval) always means a starting (restarting) event with time and cost consequences.

The project of Figure 3-4 can help to understand the essence of the proposed new *interruption oriented* resource leveling approach based on quasi-concavity. The efficient (non-dominated, Pareto optimal) solutions obtained by IN, FA, FB, and ES (early start) models are given in Table 3-1. The ES and the *"best"* schedules for measures IN, FA, and FB are shown in Figure 3-3 and 3-4. Note that, in this context *"best"* always means the following schedule: $\sum_{r=1}^{R} M_r \rightarrow \text{Min}, S \in \Re(E)$, where $M \in \{IN, FA, FB\}$.

The results were obtained by *"brute force search"* among the $|\Re(E)| = 504$ combinations. The solutions are characterized by the set of activity starting times and the set of IN, FA, and FB measure values. According to the *ProMan* notation, an activity shift is denoted by $i \Rightarrow t$, where $i \in V'$ and $t \in [ES_i, LS_i]$. In every case the *"first"* optimal solution instance was considered as a final solution. In this example, the *average vector* $\overline{U}$ was the following: $\overline{U} = \{5,4\}$.

M	IN	FA	FB	Efficient schedules
IT	**{0,1}**	{34,43}	{14,21}	{2⇒5,8⇒9,9⇒9,10⇒7}
FA	{4,3}	**{24,19}**	{27,22}	{2⇒3,8⇒9,9⇒14,10⇒5}
FB	{3,3}	{24,27}	**{28,17}**	{2⇒5,8⇒9,9⇒14,10⇒1}
ES	{3,3}	{38,33}	{32,30}	{}

Table 3-1 Computational results (IN, FA, FB) for a sample project

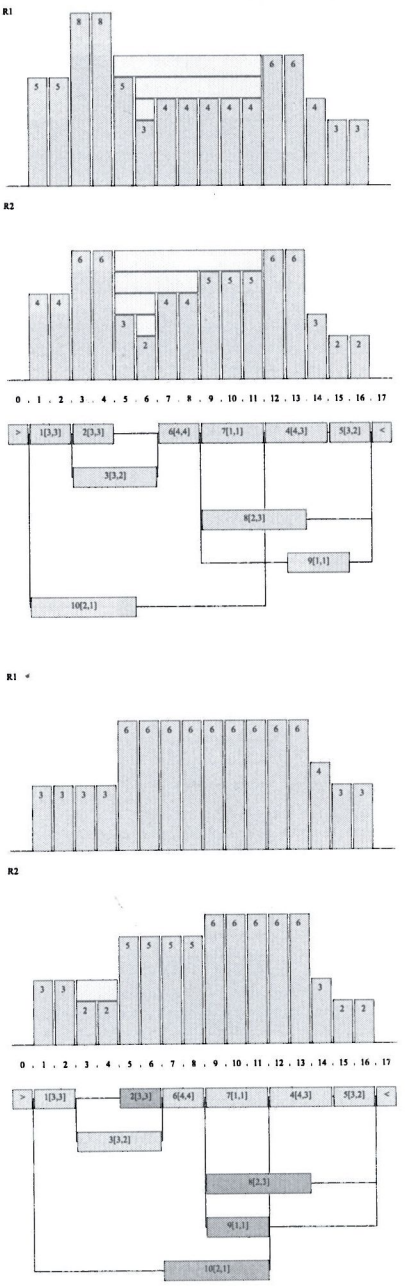


Figure 3-3 The ES schedule and the best IN schedule

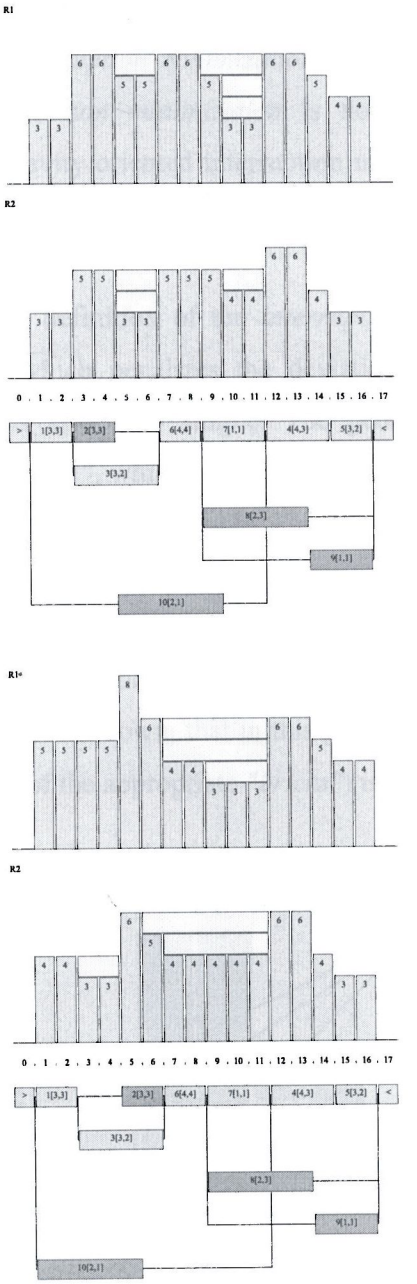


Figure 3-4 The best schedules for measure FA and FB

Computational results can be summarized as follows: (1) the results well illustrate the fact, that *interruption minimization* based on quasi-concavity and the *traditional fluctuation minimization* is not the same. (2) The proposed new quasi-concavity oriented interruption measure (IN) is able to smooth the resource profiles out according to a global point of view.

The another *"lifelike"* definition of the *interruption* oriented resource-leveling measure (IN), which penalizes the deviations from the *desired quasi-concave profile,* is also very simple. The penalty is the minimal amount of *"unit height bricks"*, which is needed to fill all "dales" up, where the width of the *"bricks"* as large as possible. This modified *"definition"* will be very useful in the subsequent discussion.

The Figure 3-5 illustrates this definition. In this resource-profile, there are two *"dales"*. It is easy to count down, that in this profile the total amount of penalty (the total amount of the appropriate *"bricks"*) is $IN = 2 + 1 = 3$.

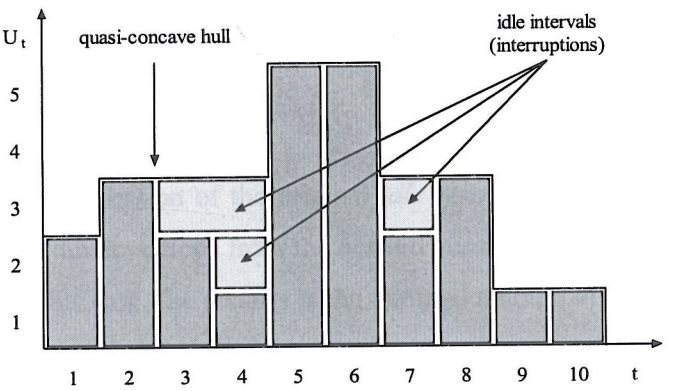


Figure 3-5 A resource profile with three interruptions (idle intervals)

According to previous verbal definition, the *interruption* oriented new global leveling measure mathematically can be expressed in the following form for a resource r, where $r \in \{1,2,...,R\}$:

Let $Act(t) = \{i \in V' \mid S_i \le t < S_i + D_i\}$ denote the set of activities in progress in time t, where $t \in \{1, 2, ..., T\}$ for a schedule $S \in \Re$.

Moreover, let $U_t = \sum_{i \in Act(t)} r_{ir}$, where $t \in \{1, 2, ..., T\}$, be the total amount of resource used at time t for a schedule $S \in \Re$.

Let $MAXL_t = Max(U_1, U_2, ..., U_t)$, where $t \in \{1, 2, ..., T-2\}$.

$$IT_r(S) = \sum_{t=2}^{T-1} Max(Min(MAXL_{t-1r}, U_{t+1r}) - U_{tr}, 0) \qquad (IN)$$

The presented *new interruption measure* was given by an appropriate modification of the *idle resource unit measure* (IT) developed by Csébfalvi (2000).

The *"lifelike"* definition of the original *idle resource unit measure* (IT), which penalizes the deviations from the desired quasi-concave profile is also very simple. In this case, the penalty is the minimal amount of *"unit bricks"*, which is needed to fill all *"dales"* up.

For a given resource r where $r \in \{1,2,...,R\}$:

Let $MAXL_{tr} = Max(U_{1r}, U_{2r}, ..., U_{tr})$, where $t \in \{1, 2, ..., T-2\}$.

Let $MAXR_{tr} = Max(U_{3r}, U_{4r}, ..., U_{tr})$, where $t \in \{3, 4, ..., T\}$.

According to the previous verbal definition, the *idle resource unit* oriented global resource-leveling measure mathematically can be expressed in the following form for a resource r, where $r \in \{1,2,...,R\}$:

$$IT_r(S) = \sum_{t=2}^{T-1} \text{Max}(\text{Min}(MAXL_{t-1r}, MAXR_{t+1r}) - U_{tr}, 0) \qquad \text{(IT)}$$

Proposition 3-1

When we replace $MAXR_{t+1r}$ with U_{t+1r} in the *original measure* then we get the *total number of idle intervals* (*the total number of interruptions*) measure.

Proof

The result immediately follows from the modification. ♦

4. SINGLE-MODE CASE

In this chapter, we present a MILP formulation and an implicit enumeration algorithm for single-mode projects. The MILP formulation and the algorithm for the proposed new *interruption measure* are based on the models developed by Csébfalvi (2000) for the *idle resource unit measure.*

4.1 MILP FORMULATION

It is known that the *interruption* measure mathematically can be expressed in the following form for a resource r where $r \in \{1,2,...,R\}$:

Let $Act(t) = \{i \in V' \mid S_i \le t < S_i + D_i\}$ denote the set of activities in progress in time t, where $t \in \{1, 2, ..., T\}$ for a schedule $S \in \Re$.

Moreover, let $U_t = \sum_{i \in Act(t)} r_{ir}$, where $t \in \{1, 2, ..., T\}$, be the total amount of resource used at time t for a schedule $S \in \Re$.

Let $MAXL_t = Max(U_1, U_2, ..., U_t)$ where $t \in \{1, 2, ..., T-2\}$.

$$IT_r(S) = \sum_{t=2}^{T-1} Max(\, Min(MAXL_{t-1r}, U_{t+1r}) - U_{tr}, 0) \qquad (IN)$$

This mathematically beautiful and *"easy to understand"* global measure has at least two very disadvantageous properties: (1) The Min and Max functions are *non-smooth* functions. Therefore, to maintain linearity, we must replace them with linear constraints or some appropriate combination of linear constraints and integer variables. (2) The problem of replacements is effected by the direction of optimization (we minimize the penalty function), therefore, the Min and Max replacements will be totally different.

In the following we will show, that the *interruption* oriented resource leveling model (IN) for a single resource can be formulated as a *mixed integer linear programming problem*, which may be solved directly in the case of small-size projects. In the case of multi-resource-projects, the problem can be solved by introducing a weighting function.

We consider a project consisting of N real activities. The duration of an activity i, where $i \in \{0,1,\ldots,N,N+1\}$, is denoted by $D_i \in \{1,2,\ldots\}$ and its start time by $S_i \in \{0,1,\ldots\}$.

The project modeled by an activity-on-node (AoN) network $G=(V,E)$, where each node $i \in V$ represents an activity $i, (i=0,1,\ldots,N+1)$.

An arc $i \rightarrow j \in E$ represents a finish-start precedence relation $S_i + D_i \leq S_j$ between activity i and activity j.

There is a *renewable* resource required for carrying out the project. This resource has unlimited resource availability and each real activity $i \in V' = V \setminus \{0, N+1\}$ uses $r_i \geq 0$ units of resource over its duration.

Let $AS = \{AS_{it} \mid i = 1,2,...,N; t = ES_i,...,LS_i\}$ denote the set of the zero-one starting time. As we know, $AS_{it} = 1$ if activity i starts in period j and 0 otherwise.

We describe the network-relations with the well-known formulation

The network-relations are the following:

$$\sum_{t=ES_i}^{LS_i} AS_{it} = 1 \text{ for } i = 1,2,...,N \tag{4-1}$$

$$\sum_{s=ES_i}^{LS_i} s * AS_{is} + D_i \leq \sum_{s=ES_j}^{LS_j} s * AS_{js} \text{ for } i \rightarrow j \in E$$

Let U_t, where $t \in \{1,2,...,T\}$ be the total amount of a resource, be used in period t for a schedule $S \in \Re$. As we know, U_t, where $t \in \{1,2,...,T\}$, is a linear function of the starting time variables.

Let $Act(t) = \{i \in V' \mid S_i \leq t < S_i + D_i\}$ denote the set of activities in progress in period t, where $t \in \{1,2,...,T\}$ for a schedule $S \in \Re$.

Moreover, let $U_t = \sum_{i \in Act(t)} r_i$, where $t \in \{1,2,...,T\}$, be the total amount of resource used at time t for a schedule $S \in \Re$.

Let $MAXL_t = Max(U_1, U_2, \ldots, U_t)$, where $t \in \{1, 2, \ldots, T-2\}$ and let $S \in \Re$ a schedule from the set of network-feasible schedules.

The formal description of the optimization problem may be the following:

$$IN(S) \rightarrow Min! \qquad (4\text{-}2)$$
$$S \in \Re$$

In the subsequent discussion, we show that problem (4-2) can be linearised by *introducing new variables and constraints*.

Step One

First, to avoid the multiple computations (to increase the computational efficiency), we replace all Max functions with more than two arguments with a series of Max functions with two arguments:

$$MAXL_1 = U_1 \qquad (4\text{-}3)$$
$$MAXL_t = Max(MAXL_{t-1}, U_t) \text{ for } t \in \{2, 3, \ldots, T-2\}$$

Step Two

Let I_t, where $t \in \{2, 3, \ldots, T-1\}$, be the total amount of the idle units in period t for a schedule $S \in \Re$. Let $MIN_t = Min(MAXL_{t-1}, U_{t+1})$, where $t \in \{2, 3, \ldots, T-1\}$.

Proposition 4-1

Problem (4-2) can be replaced by the following *non-smooth* problem:

$$\sum_{t=2}^{T-1} I_t \rightarrow \text{Min!} \qquad (4\text{-}4)$$

$$U_t + I_t \geq MIN_t \text{ for } t \in \{2, 3, \ldots, T-1\}$$

$$MIN_t = \text{Min}(MAXL_{t-1}, U_{t+1})$$

$$S \in \Re$$

Proof. It is straightforward. ♦

Step Three

Proposition 4-2

$MAXL_t = \text{Max}(MAXL_{t-1}, U_t)$, $t \in \{3, 4, \ldots, T-1\}$, can be replaced by the following simple constraint set:

$$MAXL_t \geq U_t \qquad (4\text{-}5)$$

$$MAXL_t \geq MAXL_{t-1}.$$

Proof

These results immediately follow from the structure of $IT(S)$ and the direction of optimization ($IT(S) \rightarrow \text{Min!}$). By *"squeezing"* $MAXL_t$ to the *correct value*, we avoid using Max function. ♦

At the end of Step 3, the *non-smooth* optimization problem will be the following:

$$\sum_{t=2}^{T-1} I_t \rightarrow \text{Min!} \qquad (4\text{-}6)$$

$$U_t + I_t \geq MIN_t \text{ for } t \in \{2,3,\ldots,T-1\}$$

$$MIN_t = \text{Min}(MAXL_{t-1}, U_{t+1}) \text{ for } t \in \{2,3,\ldots,T-1\}$$

$$MAXL_1 = U_1$$

$$MAXL_t \geq U_t \text{ for } t \in \{2,3,\ldots,T-2\}$$

$$MAXL_t \geq MAXL_{t-1} \text{ for } t \in \{2,3,\ldots,T-2\}$$

$$S \in \Re$$

Step Four

Proposition 4-3

$MIN_t = \text{Min}(MAXL_{t-1}, U_{t+1})$, $t \in \{2,3,\ldots,T-1\}$, can be replaced by the following nonlinear constraint set, where $LLTR_{t-1}$ is a zero-one variable:

$$MIN_t = LLTR_{t-1} * MAXL_{t-1} + (1 - LLTR_{t-1}) * U_{t+1} \qquad (4\text{-}7)$$

$$MIN_t \leq MAXL_{t-1}$$

$$MIN_t \leq U_{t+1}$$

$$LGTR_{t-1} \in \{0,1\}$$

According to the formulation, $LLTR_{t-1} = 1$ if $MAXL_{t-1} > U_{t+1}$, and 0 otherwise (Notation: Left-Less-Than-Right→ LLTR!).

Proof. It is straightforward. ♦

Proposition 4-4

According to the well-known *"big-M"* formulation, the nonlinear (bilinear) constraint set (2.2.1.1-9) can be replaced by the following linear constraints:

$$MIN_t \le MAXL_{t-1} \tag{4-8}$$

$$MIN_t \le U_{t+1}$$

$$MIN_t = PROL_{t-1} + PROR_{t+1}$$

$$PROL_{t-1} \ge \underline{MAXL}_{t-1} * LLTR_{t-1}$$

$$PROL_{t-1} \ge MAXL_{t-1} + \overline{MAXL}_{t-1} * LLTR_{t-1} - \overline{MAXL}_{t-1}$$

$$PROR_{t+1} \ge -\underline{U}_{t+1} * LLTR_{t-1} + \underline{U}_{t+1}$$

$$PROR_{t+1} \ge U_{t+1} - \overline{U}_{t+1} * LLTR_{t-1}$$

$$LLTR_{t-1} \in \{0,1\}$$

In the formulation $\underline{MAXL}_{t-1}$, $\overline{MAXL}_{t-1}$, $\underline{U}_{t+1}$, and $\overline{U}_{t+1}$ denote the lower and upper bounds for $MAXL_{t-1}$ and U_{t+1}, respectively.

Proof

It is straightforward. … ♦

After Step 4, we get the following mixed integer linear programming problem:

$$\sum_{t=2}^{T-1} I_t \rightarrow \text{Min!} \tag{4-9}$$

$$U_t + I_t \geq MIN_t \quad t \in \{2,3,\ldots,T-1\}$$

$$MIN_t \leq MAXL_{t-1} \quad t \in \{2,3,\ldots,T-1\}$$

$$MIN_t \leq U_{t+1} \quad t \in \{2,3,\ldots,T-1\}$$

$$MIN_t = PROL_{t-1} + PROR_{t+1} \quad t \in \{2,3,\ldots,T-1\}$$

$$PROL_{t-1} \geq \underline{MAXL}_{t-1} * LLTR_{t-1} \quad t \in \{2,3,\ldots,T-1\}$$

$$PROL_{t-1} \geq MAXL_{t-1} + \overline{MAXL}_{t-1} * LLTR_{t-1} - \overline{MAXL}_{t-1} \quad t \in \{2,\ldots,T-1\}$$

$$PROR_{t+1} \geq -\underline{MAXR}_{t+1} * LLTR_{t-1} + \underline{MAXR}_{t+1} \quad t \in \{2,3,\ldots,T-1\}$$

$$PROR_{t+1} \geq U_{t+1} - \overline{U}_{t+1} * LLTR_{t-1} \quad t \in \{2,3,\ldots,T-1\}$$

$$MAXL_1 = U_1$$

$$MAXL_t \geq U_t \quad t \in \{2,3,\ldots,T-2\}$$

$$MAXL_t \geq MAXL_{t-1}. \; t \in \{2,3,\ldots,T-2\}$$

$$LGTR_{t-1} \in \{0,1\} \text{ for } t \in \{1,3,\ldots,T-2\}$$

$$\sum_{t=ES_i}^{LS_i} AS_{it} = 1 \quad i = 1,2,\ldots,N$$

$$\sum_{s=ES_i}^{LS_i} s * AS_{is} + D_i \leq \sum_{s=ES_j}^{LS_j} s * AS_{js} \quad i \rightarrow j \in E$$

$$AS_{it} \in \{0,1\} \text{ for } i = 1,2,\ldots,N$$

To illustrate the essence of the MILP formulation a simple project of Figure 4-1 will be presented. The structure of the LINDO input file generated automatically by the *ProMan* program will be shown in Table 4-1.

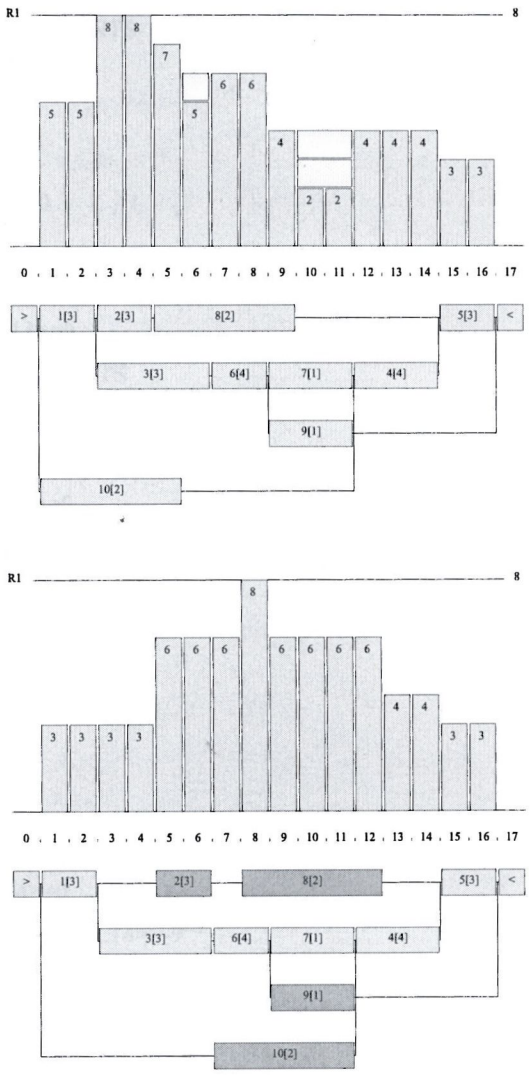


Figure 4-1 The Earliest and the Optimal Schedules

```
MIN +I2+I3+I4+I5+I6+I7+I8+I9+I10+I11+I12+I13+I14+I15

SUBJECT TO
! MAXIMUM CONSTRAINTS
MAXL1-U1=0

MAXL2-U2>=0
-MAXL1+MAXL2>=0
...
MAXL14-U14>=0
-MAXL13+MAXL14>=0
! MINIMUM CONSTRAINTS
MIN2-MAXL1<=0
MIN2-U3<=0
MIN2-PROL1-PROR3=0
-3LLER1+PROL1>=0
-MAXL1-5LLER1+PROL1>=-5
+3LLER1+PROR3>=3
-U3+8LLER1+PROR3>=0
...
MIN15-MAXL14<=0
MIN15-U16<=0
MIN15-PROL14-PROR16=0
-4LLER14+PROL14>=0
-MAXL14-9LLER14+PROL14>=-9
+3LLER14+PROR16>=3
-U16+4LLER14+PROR16>=0
! STARTING TIME CONSTRAINTS
A1S1=1
...
A10S1+A10S2+A10S3+A10S4+A10S5+A10S6+A10S7=1
```

! NETWORK CONSTRAINT

5 A8S5+ … +10 A8S10-3 A2S3- … -8 A2S8>=2

! INTERRUPTION CONSTRAINTS

I2-MIN2+U2>=0

…

I15-MIN15+U15>=0

! RESOURCE USAGE CONSTRAINTS

+3A1S1+2A10S1-U1=0

…

+3A5S15+A9S14-U16=0

END

INT A1S1 … A10S7

INT LLER1 … LLER14

PROBLEM STATISTICS

ROWS = 171

VARIABLES = 131

INTEGER (ZERO-ONE) VARIABLES = 45

NONZEROS = 579

CONSTRAINT NONZEROS = 522

DENSITY = 0.026

OBJECTIVE = MININIZATION

LESS THAN OR EQUAL TO CONSTRAINTS = 33

EQUALITY CONSTRAINTS = 41

GREATER THAN OR EQUAL TO CONSTRAINTS = 96

SOLUTION TIME (166 MHz Pentium PC with 64 MB) = 2 seconds

Figure 4-1 The Structure of the MILP formulation

4.2 AN IMPLICIT ENUMERATION ALGORITHM

In the following, we present an *implicit enumeration* algorithm for the new resource-balancing model (IN) based on the *interruption measure*. The algorithm of the proposed model is formulated as a tree search problem with an effective pruning rule.

We will say, that a schedule $S^j \in \Re$ is an *efficient* schedule if no other schedule $S^k \in \Re$ exists such that $IT_r(S^k) \le IT_r(S^j)$ for each $r = 1, 2, \ldots, R$ with strict inequality for at least one case.

According to the above definition, the *resource-balancing problem* (IN) can be conceptually formulated as follows:

given the prescribed *maximal project duration* T, (IN)

find all efficient solutions of

$$IT(S) = (IT_1(S), IT_2(S), \ldots, IT_R(S)),$$

Subject to $S \in \Re(E)$.

Let $\Re$ denote the set of network-feasible schedules. In the searching process, a schedule set is characterized by the set of original network relations and a particular set of explicitly prescribed activity starting times.

An explicitly prescribed activity starting time is represented by $i \Rightarrow t$, where $i \in V'$ and $t \in \{1,2,\ldots,T\}$. Let $\Re = \Re(F)$ denote a schedule set with additional explicit activity starting time prescriptions, where F denotes a particular set of explicit starting time prescriptions. For a schedule set $\Re = \Re(F)$, $ES(\Re)$ will denote the earliest CPM solution.

The root node of the search tree corresponds to the set $\Re(F)$, where $F = \{\}$. At this node, each movable activity is movable; therefore, the set of explicitly prescribed activity starting times is empty.

Our node evaluation function is very simple: it assigns to any node the following quasi-cost value:

$$IN = \sum_{r=1}^{R} IN_r \tag{4-10}$$

Naturally, this quasi-cost may be replaced by a real cost function, which is able to express a tradeoff of the decision-maker. Thus, at each step of the tree-building process, we select the most promising state, which has been generated but not expanded. In the tree-building process, a parent node which has one or more movable activities is transformed into a set of child nodes by fixing the starting time of the first such activity all the possible ways.

Note that, in this context "first" always means the first activity in topological order. The leaves of the search tree correspond to schedules which have no movable activity anymore or have been pruned according to the "at least as good" pruning rule will be introduced below. The tree-building process maintains the dynamically changing set of efficient schedules.

Let $Act(t) = \{i \in V' \mid S_i \le t < S_i + D_i\}$ denote the set of activities in progress in time t, where $t \in \{1, 2, \ldots, T\}$ for a schedule $S \in \Re(F)$.

Moreover, let $U_{tr} = \sum_{i \in Act(t)} r_{ir}$ be the total amount of resource r used at time t in a schedule $S \in \Re(F)$, where $t \in \{1, 2, ..., T\}$ and $r \in \{1,2, ..., R\}$.

Let $C \subset \{1,2, ..., T\}$ be the subset of such periods in which the resource profiles cannot be modified by activity movements. In other words, if $t \subset C$ then U_{tr} is constant for every $r \in \{1,2, ..., R\}$.

Let C_{tr} , where $t \in \{1, 2, ..., T\}, r \in \{1,2, ..., R\}$, and $0 \leq C_{tr} \leq U_{tr}$, be the constant part of U_{tr}.

The project of Figure 4-1 illustrates the definitions. In the schedule, the starting position of activity 10 is fixed by the $10 \Rightarrow 7$ description, so $F = \{10 \Rightarrow 7\}$. In the figure U_{tr} and C_{tr} values are *light gray* and *gray* bars, respectively. The constant profile values are *dark gray* bars ($C = \{1,2,7,8\}$).

We will show that the quasi-concavity has a special monotone property, which can be exploited in the pruning rule construction.

Let $\Re = \Re(F)$ an arbitrary schedule set.

In this set let

$$MAXCL_{tr} = Max(C_{1r}, C_{2r}, ..., C_{tr}), \qquad (4\text{-}11)$$

$t \in \{1, 2, ..., T-2\}$, $r \in \{1,2, ..., R\}$.

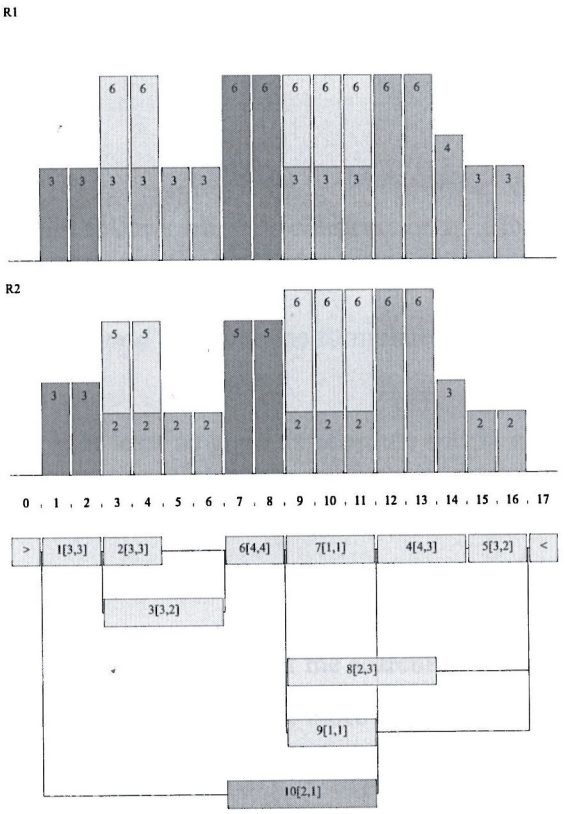


Figure 4-1 Resource profiles with constant and partly constant values

Proposition 4-1

Given a schedule set $\Re = \Re(F)$ and a resource $r \in \{1,2,...,R\}$, let $LIT_r(\Re)$ defined as follows:

$$LIN_r(\Re) = \sum_{t \in C} Max(\ Min(MAXCL_{t-1r}, C_{t+1r}) - C_{tr}, 0). \qquad (4\text{-}12)$$

$LIN_r(\Re)$ is a lower bound of *interruptions* on schedule set $\Re$: $LIT_r(\Re) \le IT_r(S)$ for $\forall S \in \Re$.

Proof

This property easily follows from the construction of $LIN_r(\Re)$. ♦

Now the *"at least as good"* rule may be formulated as follows:

Schedule set $\Re'$ is at least as good as schedule set $\Re$ and thus the node corresponds to schedule set $\Re$ may be pruned if $IN_r(ES(\Re')) \le LIN_r(\Re)$ for $\forall r \in \{1,2,...,R\}$.

Note that according to the progress of the searching process the schedules become more-and-more constrained. The more constrained the schedules, the more effective the pruning rule.

To illustrate the essence of the *"at least as concave rule"* an example will be presented in Figure 4-2. It is easy to verify that the *first* schedule is dominated by the *second* one. In Figure 4-2 the *actual* and the *partly constant* values are represented by *light gray* and *gray* bars respectively. The *constant* histogram values are represented by *dark gray* bars.

The example of Figure 4-3 will be used to illustrate the *"efficient concave schedule searching phase"* of the proposed model. The searching tree is shown in Figure 4-4.

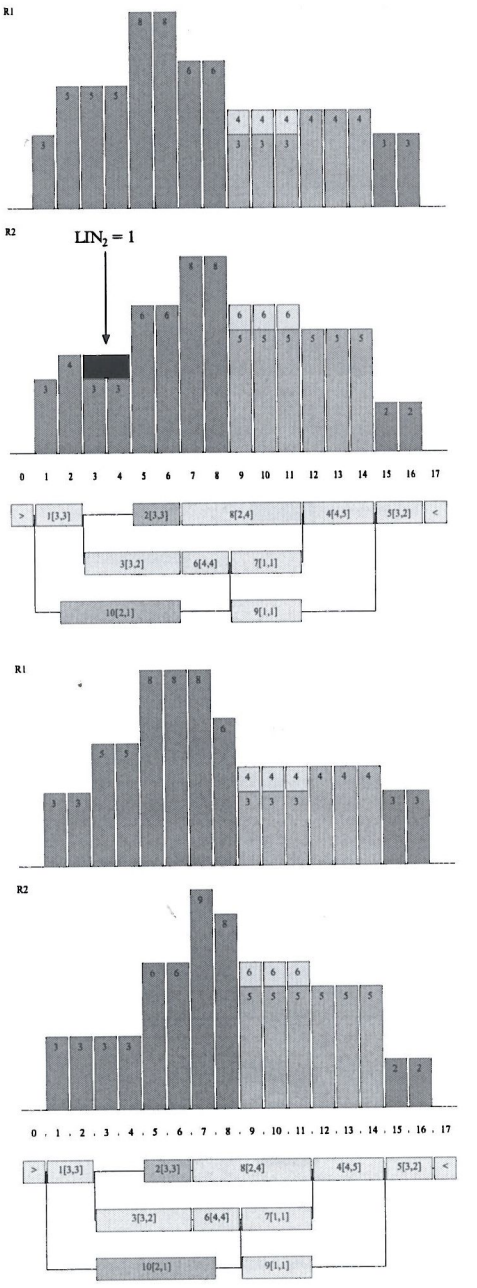


Figure 4-2 The "at least as good" rule (the first schedule is dominated by the second one).

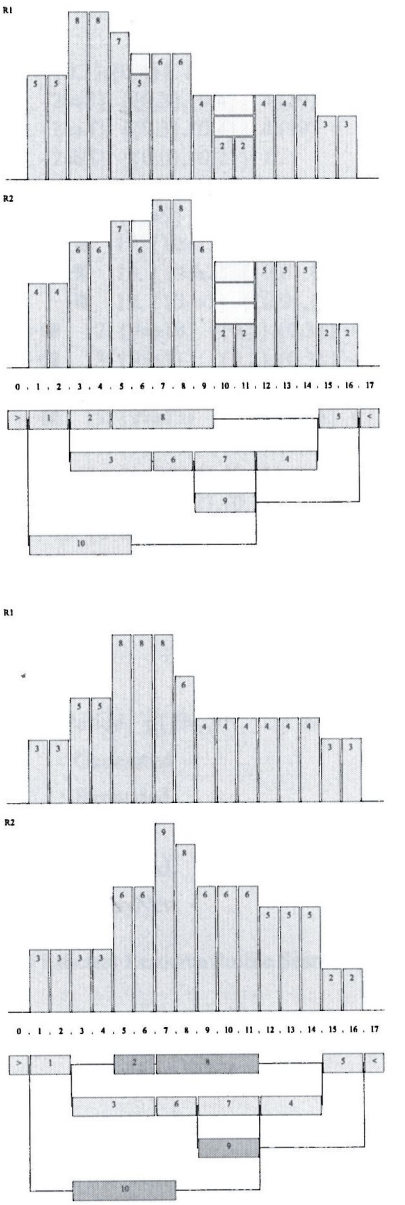
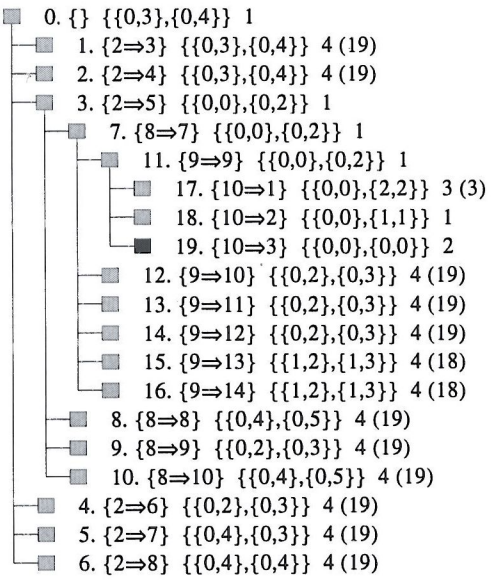


Figure 4-3 Early Start and Optimal Schedules



Legend:

Node. $\{\text{Activity} \Rightarrow \text{Period}\}\ \{\{\text{LIN}_1, \text{IN}_1\},...\}$ Indicator (PrunedBy)

$$\text{Indicator} = \begin{cases} 0 & \text{Generated} \\ 1 & \text{Expanded} \\ 2 \quad \text{if} & \text{Efficient} \\ 3 & \text{PruByOld} \\ 4 & \text{PruByNew} \end{cases}$$

Figure 4-4 Efficient Resource Profile Searching Tree

The second problem is a *"easy"* instance from the well-known Patterson's set. The *problem 20* (P20) well illustrates the fact that problems that are difficult for RLP are not necessarily difficult for RAP and vice versa.

The P20 was investigated by Bell and Park (1990) using the A^* algorithm for RAP. The number of resources is three and the resource availability

vector: $A = \{10,10,10\}$. The *optimal* resource-feasible makespan and the network-feasible makespan are the same. To obtain the *first* resource-feasible solution, the applied A^* algorithm evaluated only 31 *network-feasible* schedules, so P20 is an *"easy"* instance for RAP. The structure of P20 is shown in Figure 12. The number of *network-feasible* schedules $|\Re(E)| = 182\ 772\ 781\ 668$, so P20 is a *"really hard"* problem for RLP.

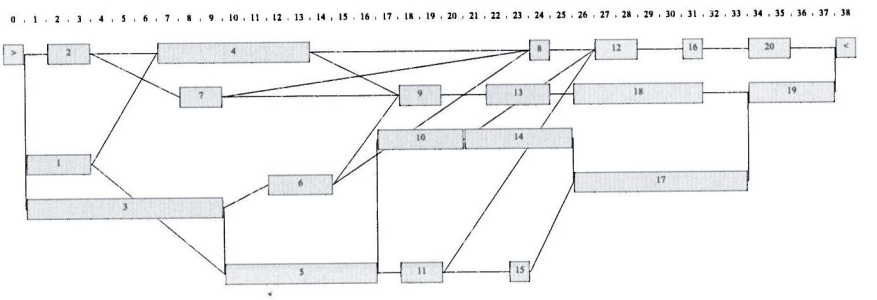


Figure 4-5 The Structure of P20

In the early start CPM schedule: $IN = \{17,20,19\}$, which is shown in Figure 4-6. When we constrain the maximal size of the searching tree (the maximal number of the node expanding steps), and stop the tree building process, for example, after five expanding steps, the *"best heuristic solution"*, which is shown in Figure 4-7, will be: $IN = \{11,14,17\}$. The "truncated tree" is shown in Figure 4-8. After five steps the number of efficient schedules is three. The result well illustrates the fact that the proposed method can be used as a heuristic without essential modifications.

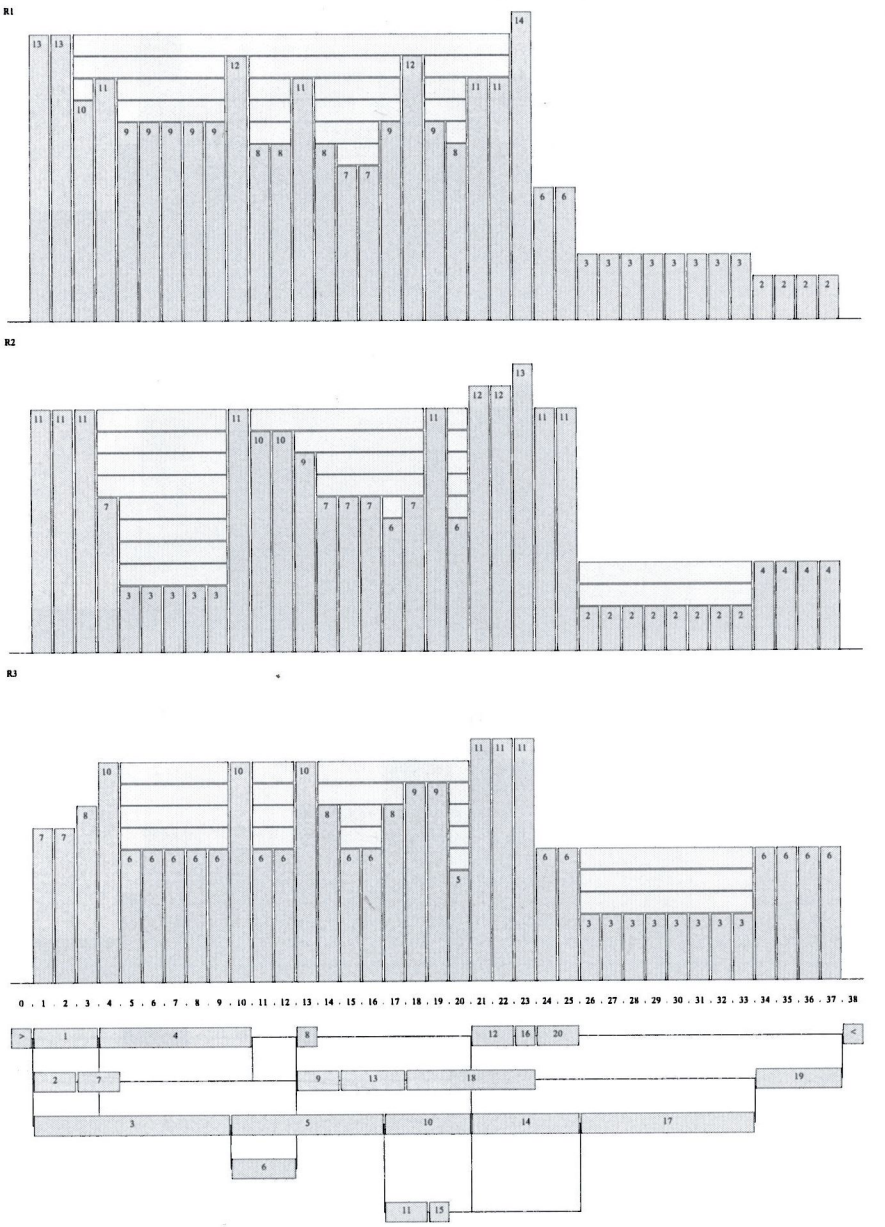


Figure 4-6 The Early Start CPM Schedule

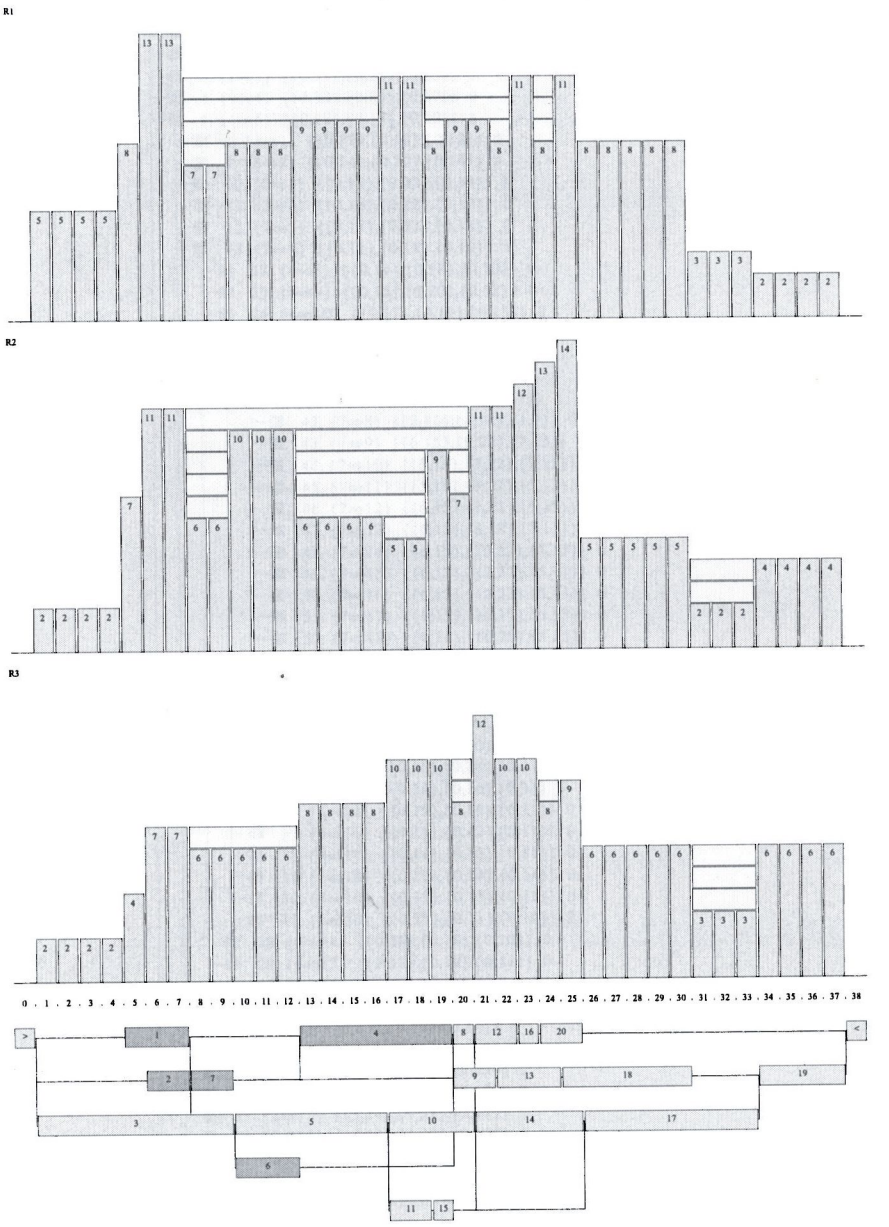


Figure 4-7 The *"best"* Schedule

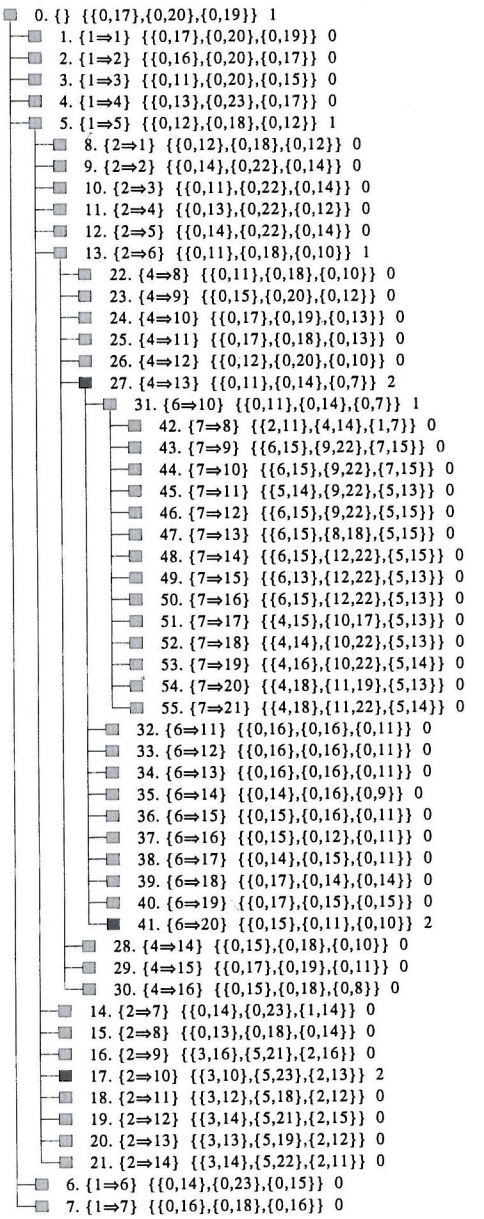


Figure 4-8 The *"truncated"* searching tree

5. MULTI-MODE CASE

The following example, adopted from Talbot (1982) illustrates the foregoing discussion.

In Table 5-1 and Figure 5-1 the project is shown to consist of six activities, each of which may be accomplished in one of two modes.

i	j	D_{ij}	r_{ij1}	r_{ij2}	r_{ij3}	r_{ij4}
1	1	2	1	0	2	1
	2	3	0	1	2	1
2	1	1	1	0	3	3
	2	3	0	1	3	2
3	1	3	1	0	1	4
	2	4	0	1	1	4
4	1	5	1	0	1	3
	2	7	0	1	1	3
5	1	4	1	0	2	2
	2	6	0	1	2	2
6	1	1	1	0	3	2
	2	4	0	1	3	4

Figure 5-1 A Six-Activity Project with four Resources (Talbot (1982))

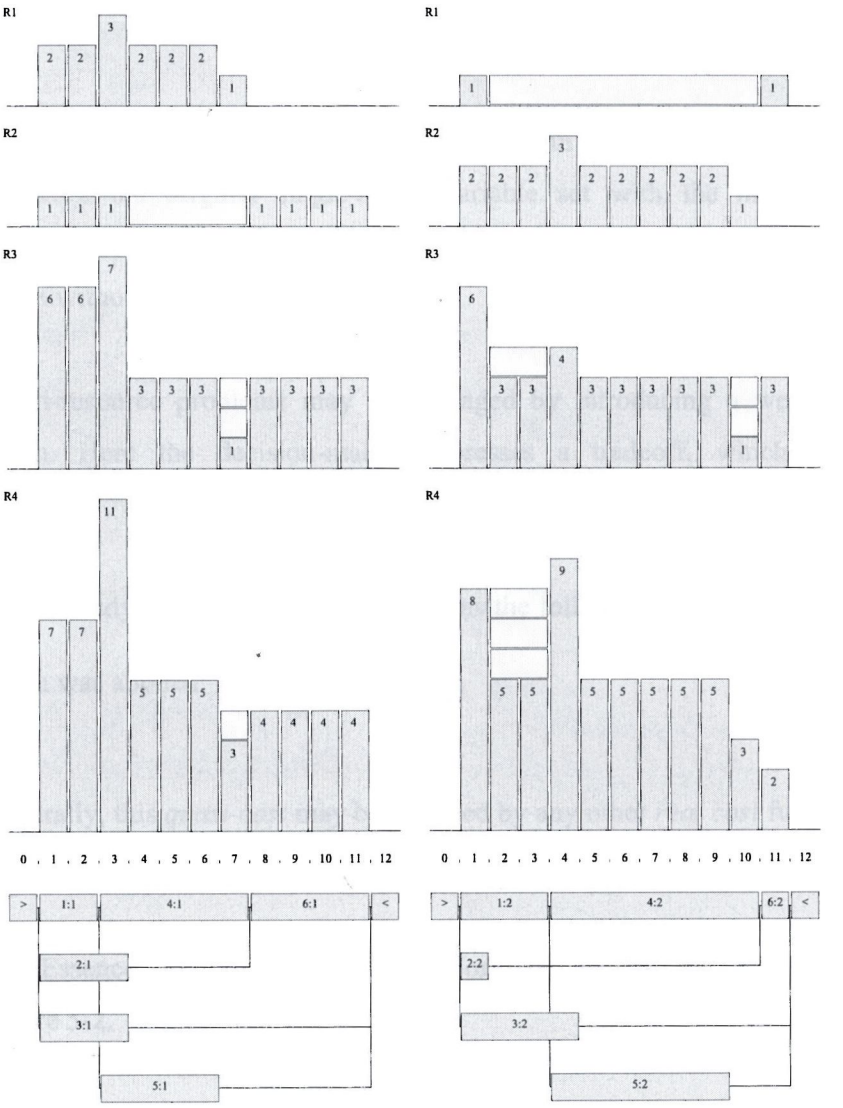


Figure 6-1 The Mode 1 and Mode 2 CPM Schedules of a Six-Activity Project (Talbot (1982))

5.1 MILP FORMULATION

It is easy to realize that when we replace in the *single-mode* MILP formulation the original single-mode variable set with the *multi-mode variable set* then we get the *multi-mode* MILP formulation of the balancing-problem without any other modification.

Multi-resource problems may be managed by introducing a weighting function. Here the decision-maker expresses a tradeoff, which, once specified, allows the problem to be solved with a single criterion.

In this study, for multi-resource problems the following simple *quasi-cost* function was applied: $IN = \sum_{r=1}^{R} IN_r$.

Naturally, this *quasi-cost* may be replaced by any other *real cost* function, which is able to express a tradeoff of the decision-maker.

The Essence of the generated LINDO Formulation (*ProMan*) is presented in Figure 5-2.

The optimal solution using the interruption measure (IN) is shown Figure 5-3.

! OBJECTIVE FUNCTION

MIN

+R1I2+R1I3 … +R1I13
+R2I2+R2I3 … +R2I13
+R3I2+R3I3 … +R3I13
+R4I2+R4I3 … +R4I13

SUBJECT TO

! STARTING TIME CONSTRAINTS

+A1M1S1+A1M1S2 … +A1M2S3=1
…
+A6M1S8+A6M1S92S11 … +A6M2S14=1

! NETWORK CONSTRAINTS

+3 A4M1S3+4 A4M1S4 … +4 A4M2S4-3 A1M1S1-4 A1M1S2 … -6 A1M2S3>=0
…
+8 A6M1S8+9 A6M1S9 … +14 A6M2S14-8 A4M1S3-9 A4M1S4 … -11 A4M2S4>=0

! RESOURCE-USAGE CONSTRAINTS

+1 A1M1S1+0 A1M2S1 … +0 A3M2S1-R1U1=0
…
+4 A3M1S12+4 A3M2S11 … +2 A6M2S14-R4U14=0

```
! MAXIMUM CONSTRAINTS
...
+R1MAX2-R1U2>=0
+R1MAX2-R1MAX1>=0
...
+R4MAX12-R4U12>=0
+R4MAX12-R4MAX11>=0

+R1I2-R1MIN2+R1U2>=0
...
+R4MAX12-R4U12>=0
+R4MAX12-R4MAX11>=0

! INTERRUPTION CONSTRAINTS

+R1I2-R1MIN2+R1U2>=0
...
+R4I13-R4MIN13+R4U13>=0

! MINIMUM CONSTRAINTS

+R1MIN2-R1MAX1<=0
+R1MIN2-R1U3<=0
+R1MIN2-R1PROL1-R1PROR3=0
+R1PROL1-R1MAX1-99 R1LLER1>=-99
+R1PROR3-R1U3+99 R1LLER1>=0
...
+R4MIN13-R4MAX12<=0
+R4MIN13-R4U14<=0
+R4MIN13-R4PROL12-R4PROR14=0
+R4PROL12-R4MAX12-99 R4LLER12>=-99
+R4PROR14-R4U14+99 R4LLER12>=0
END
```

PROBLEM STATISTICS

ROWS = 447
VARIABLES = 425
INTEGER (ZERO-ONE) VARIABLES = 177

NONZEROS =2071
CONSTRAINT NONZEROS = 1969
DENSITY = 0.011

OBJECTIVE = MINIMIZATION

LESS THEN OR EQUAL TO CONSTRAINTS = 96
EQUALITY CONSTRAINTS = 114
GREATER THEN OR EQUAL TO CONSTRAINTS = 236

"BIG-M" CONSTANT = 99

SOLUTION TIME (166 MHz Pentium PC with 64 MB) = 128 seconds

Figure 5-2 The Essence of the Generated LINDO Formulation to Solve a Problem (Talbot (1982))

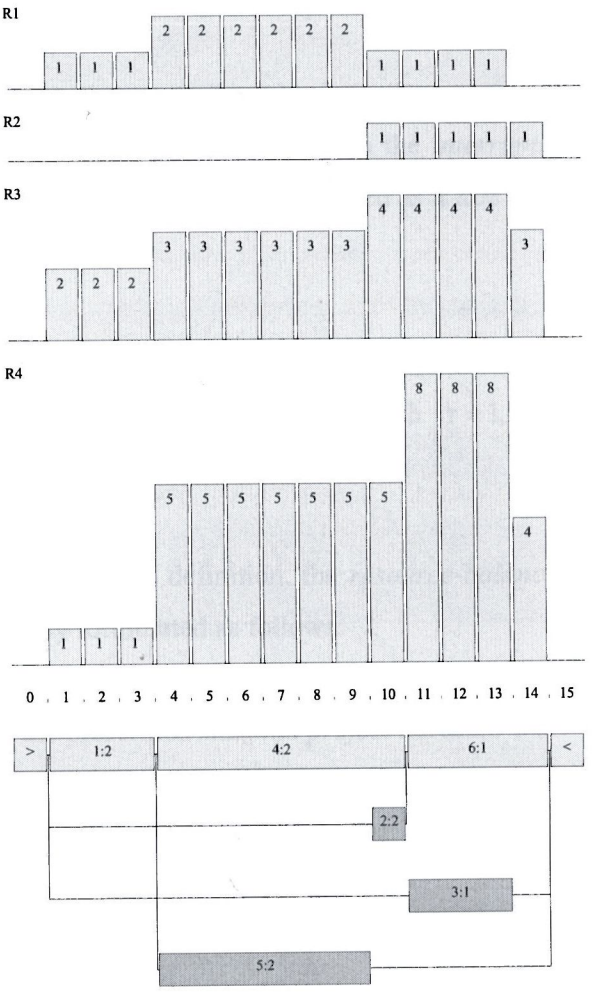


Figure 5-3 Optimal Schedule of a Six-Activity Project with four Resources (Talbot (1982))

5.2 AN IMPLICIT ENUMERATION ALGORITHM

In the following, we present an *implicit enumeration* algorithm for the *multiple-mode-leveling* model (IN) based on the *interruption measure*. The algorithm of the proposed model is formulated as a tree search problem with an effective pruning rule.

A schedule $S^j \in \Re$ is an *efficient* schedule if no other schedule $S^k \in \Re$ exists such that $IN_r(S^k) \le IN_r(S^j)$ for each $r = 1, 2, \ldots, R$ with strict inequality for at least one case.

According to the above definition, the *resource-balancing problem* (IN) can be conceptually formulated as follows:

$$\begin{gathered} \text{given the prescribed } \textit{maximal project duration } T, \qquad \text{(IN)} \\ \textit{find all efficient solutions of} \\ IN(S) = (IN_1(S), IN_2(S), \ldots, IN_R(S)), \\ \text{Subject to } S \in \Re(E). \end{gathered}$$

The *efficient* schedule searching process can be formulated as a tree search problem, where the nodes are characterized by the *set of explicitly prescribed activity operating-modes and starting times*.

According to the *ProMan* notation, the {activity, operation-mode, starting time} triplet (a multiple-mode activity shift) is denoted by $i : m \Rightarrow t$ where $i \in V'$ $m \in \{1,2,\ldots,M_i\}$ and $t \in [ES_i, LS_i]$.

An activity is defined *"movable"* if it has no explicitly prescribed start time and it is *non-critical*. By definition, the starting position of such activities will be the *earliest* CPM starting time.

The *root* node of the search tree corresponds to set $\Re(E,\{\})$, and represented by its early start schedule. At this node, each movable activity is movable so the set of explicitly prescribed activity starting times is *empty*.

Our node evaluation function is very simple: It assigns to any node the following *quasi-cost value*: $IN = \sum_{r=1}^{R} IN_r$.

Naturally, this *quasi-cost* may be replaced by a real cost function, which is able to express a tradeoff of the decision-maker.

Thus, at each step of the searching process, we select the most promising node ($IT \rightarrow Min!$), which has been generated but not expanded.

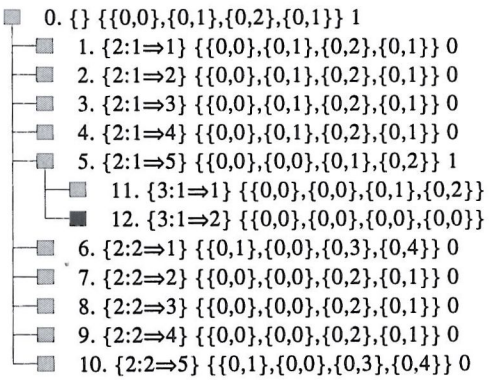
In the tree-building process, a *parent* node which has one or more movable activities is transformed into a *set of child nodes* by fixing the starting time of the *first* such activity all the possible ways.

Note that, in this context *"first"* always means the *first* activity in *topological order*.

The *leaves* of the search tree correspond to schedules which have no movable activity anymore or have been pruned according to the *"at least as good"* pruning rule will be introduced below.

The tree-building process maintains the *dynamically* changing set of *efficient schedules*.

The searching tree (*ProMan*) is presented in Figure 5-4. The optimal solution using the interruption measure (IN) is shown Figure 5-5.



Legend :

Node. {Activity : Mode ⇒ StartingTime}{{LIN_1, IN_1},...}NodeIndicator (PrunedBy)

$$\text{NodeIndicator} = \begin{cases} 0 & \text{Generated} \\ 1 & \text{Expanded} \\ 2 & \text{if Efficient} \\ 3 & \text{PrunedByOld} \\ 4 & \text{PrunedByNew} \end{cases}$$

Figure 5-4 Optimal Resource Profile Searching Tree {2:1⇒5,3:1⇒2}

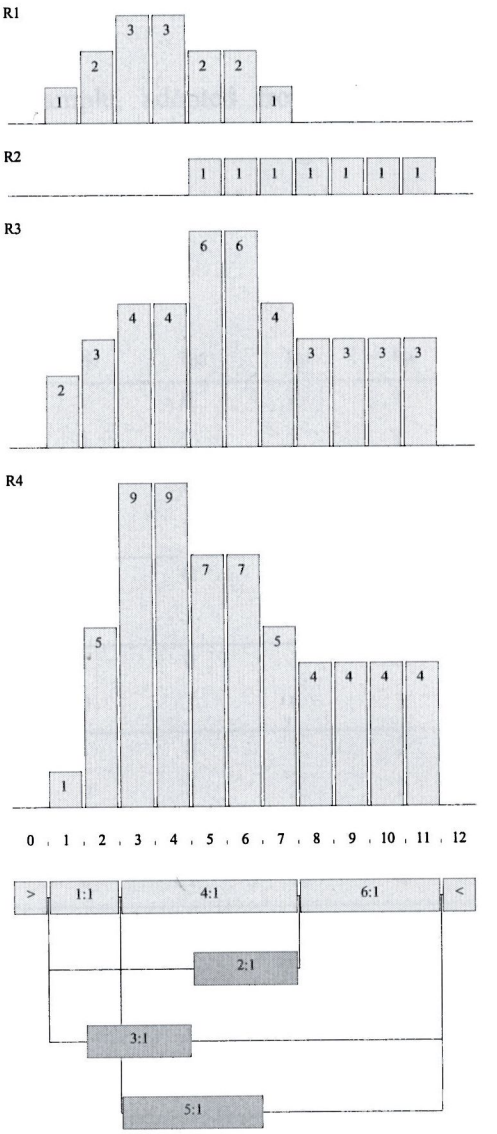


Figure 5-5 Optimal Resource Profile: {2:1⇒5,3:1⇒2}

6. DOUBLE-CONSTRAINED CASE

The following example, adopted from Talbot (1982) illustrates the foregoing discussion. In Table 6-1 and Figure 6-1 the project is shown to consist of six activities, each of which may be accomplished in one of two modes.

i	j	D_{ij}	r_{ij1}	r_{ij2}	r_{ij3}	r_{ij4}	DC_{ij}	TC_{ij}
1	1	2	1	0	2	1	135	270
	2	3	0	1	2	1	65	195
2	1	1	1	0	3	3	160	160
	2	3	0	1	3	2	90	270
3	1	3	1	0	1	4	170	510
	2	4	0	1	1	4	100	400
4	1	5	1	0	1	3	155	775
	2	7	0	1	1	3	85	595
5	1	4	1	0	2	2	150	600
	2	6	0	1	2	2	80	480
6	1	1	1	0	3	2	190	190
	2	4	0	1	3	4	120	480
Unit Costs			100	30	10	15		
Cost Constraints							300	2020

Table 6-1 A Six-Activity Double Constrained Project with four Resources (Talbot (1982))

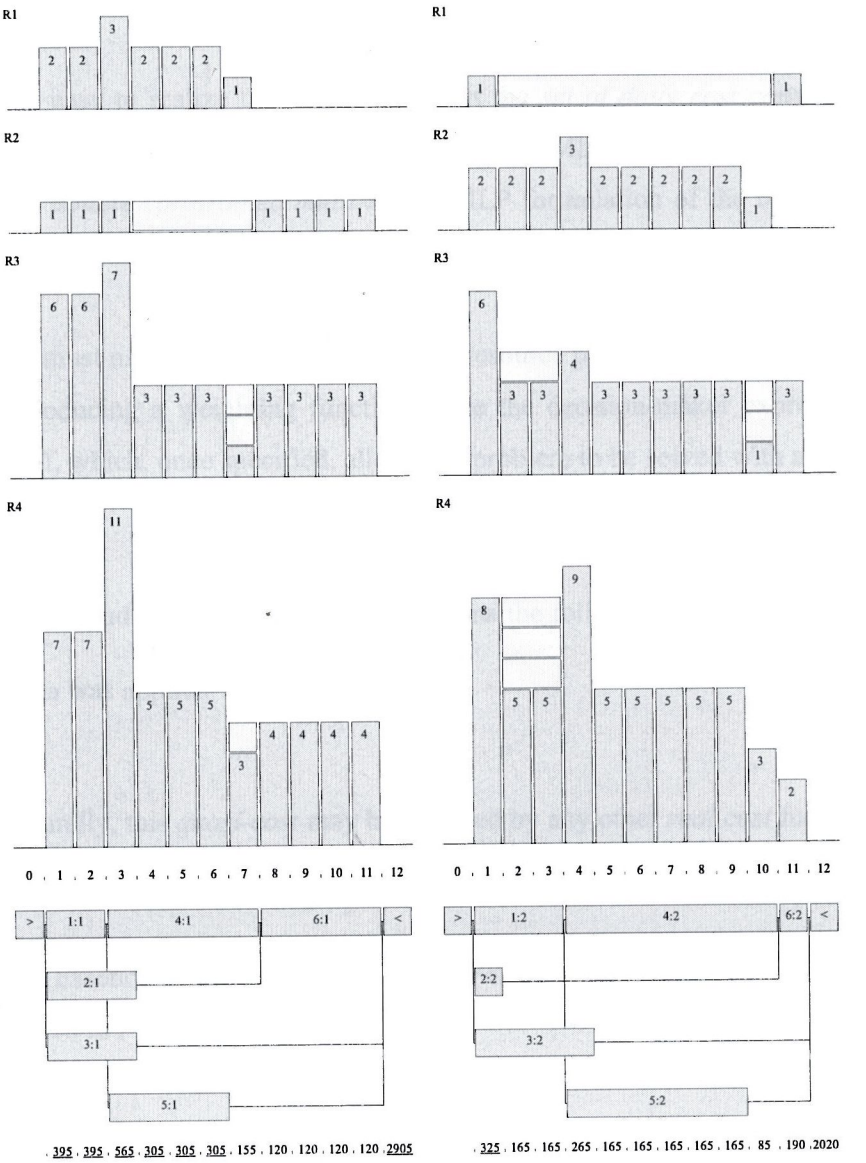


Figure 6-1 The Mode 1 and Mode 2 CPM Schedules of a Six-Activity Project (Talbot (1982))

6.1 MILP FORMULATION

It is easy to realize that when we insert the *set of daily cost constraints* and *project cost constraint* into the *multi-mode* MILP formulation then we get the *double constrained multi-mode* MILP formulation of the balancing-problem without any other modification.

We must mention it again, that a multi-resource problem may be managed by introducing a weighting function. Here the decision-maker expresses a tradeoff, which, once specified, allows the problem to be solved with a single criterion.

In this study, for multi-resource problems the following simple quasi-cost function was applied: $IN = \sum_{r=1}^{R} IN_r$.

Naturally, this *quasi-cost* may be replaced by any other *real cost* function, which is able to express a tradeoff of the decision-maker.

The Essence of the generated LINDO Formulation (given by *ProMan*) is presented in Figure 6-2.

The optimal solution using the interruption measure (IN) is shown Figure 6-3.

! OBJECTIVE FUNCTION

MIN

+R1I2+R1I3 … +R1I13
+R2I2+R2I3 … +R2I13
+R3I2+R3I3 … +R3I13
+R4I2+R4I3 … +R4I13

SUBJECT TO

! STARTING TIME CONSTRAINTS

+A1M1S1+A1M1S2 … +A1M2S3=1
…
+A6M1S8+A6M1S92S11 … +A6M2S14=1

! NETWORK CONSTRAINTS

+3 A4M1S3+4 A4M1S4 … +4 A4M2S4-3 A1M1S1-4 A1M1S2 … -6 A1M2S3>=0
…
+8 A6M1S8+9 A6M1S9 … +14 A6M2S14-8 A4M1S3-9 A4M1S4 … -11 A4M2S4>=0

! RESOURCE-USAGE CONSTRAINTS

+1 A1M1S1+0 A1M2S1 … +0 A3M2S1-R1U1=0
…
+4 A3M1S12+4 A3M2S11 … +2 A6M2S14-R4U14=0

! MAXIMUM CONSTRAINTS

+R1MAX1-R1U1=0
+R1MAX2-R1U2>=0
+R1MAX2-R1MAX1>=0
...
+R4MAX12-R4U12>=0
+R4MAX12-R4MAX11>=0

+R1I2-R1MIN2+R1U2>=0
...
+R4MAX12-R4U12>=0
+R4MAX12-R4MAX11>=0

! INTERRUPTION CONSTRAINTS

+R1I2-R1MIN2+R1U2>=0
...
+R4I13-R4MIN13+R4U13>=0

! MINIMUM CONSTRAINTS

+R1MIN2-R1MAX1<=0
+R1MIN2-R1U3<=0
+R1MIN2-R1PROL1-R1PROR3=0
+R1PROL1-R1MAX1-99 R1LLER1>=-99
+R1PROR3-R1U3+99 R1LLER1>=0
...
+R4MIN13-R4MAX12<=0
+R4MIN13-R4U14<=0
+R4MIN13-R4PROL12-R4PROR14=0
+R4PROL12-R4MAX12-99 R4LLER12>=-99
+R4PROR14-R4U14+99 R4LLER12>=0

! DAILY COST CONSTRAITS

+135 A1M1S1+65 A1M2S1 ... +100 A3M2S1<=300

...

+170 A3M1S12+100 A3M2S11 ... +190 A6M2S14<=300

! PROJECT COST CONSTRAIT

+270 A1M1S1+270 A1M1S2 ... +190 A6M2S14<=2020

END

PROBLEM STATISTICS

ROWS = 462

VARIABLES = 425

INTEGER (ZERO-ONE) VARIABLES = 177

NONZEROS =2433

CONSTRAINT NONZEROS = 2316

DENSITY = 0.012

OBJECTIVE = MINIMIZATION

LESS THEN OR EQUAL TO CONSTRAINTS = 111

EQUALITY CONSTRAINTS = 114

GREATER THEN OR EQUAL TO CONSTRAINTS = 236

"BIG-M" CONSTANT = 99

SOLUTION TIME (166 MHz Pentium PC with 64 MB) = 162 seconds

Figure 6-2 The Structure of the Generated LINDO Formulation to Solve a Problem (Talbot (1982))

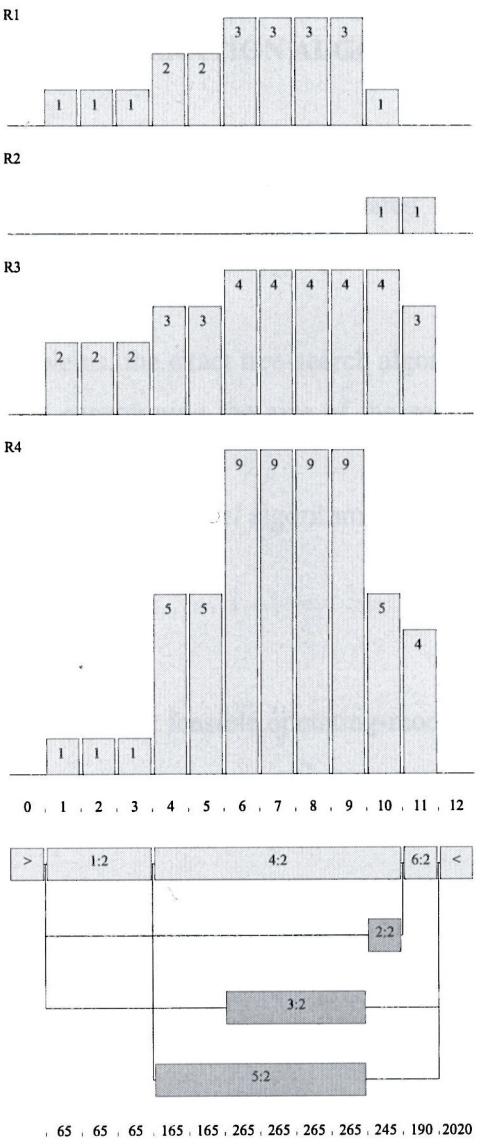


Figure 6-3 Optimal Schedule of a Six-Activity Project with four Resources (Talbot (1982))

6.2 AN IMPLICIT ENUMERATION ALGORITHM

In the case of double-constrained multiple-mode projects, the *exact* implicit enumeration algorithm may be formulated as a *trilevel "tree of trees"* like searching problem.

For large-scale problems, the exact tree-search algorithm may be replaced a simple heuristics by constraining the size of the searching-trees at every level.

The essence of the proposed *trilevel* algorithm is very simple:

First Level:

We generate all the total cost feasible operating-mode combinations.

Second Level:

On the set of total cost feasible operating-mode combinations, we solve the make-span minimization problem subject to the daily cost constraints.

Third Level:

On the set of *make-span minimal cost feasible schedules*, we solve the resource-balancing problem.

6.2.1 The First Level Searching Tree

On the *first* level, we must generate all the total cost feasible operating-mode combinations.

Let $A = \{A_i \mid i = 1,2,...,N; A_i \in \{1,2,...,M_i\}\}$ denote a feasible operating-mode combination. Suppose that for every $i \in V'$ activity the possible operation-modes $m \in \{1,2,...,M_i\}$ are in *descending order* according to their *total cost values* ($D_{im} * COST_{im}$).

Suppose that the nodes of the searching-tree are numbered $n \in \{0,1,2,...\}$.

In the searching process node $n \in \{0,1,2,...\}$ of the tree is characterized by set $\{A^n, PC^n\}$ where $PC^n = \sum_{i-1}^{N} D_{i A_i^n} * COST_{i A_i^n}$ denotes the *total project cost* according to the given A^n operation-mode combination.

The *root* node of the tree will be the *"cheapest"* mode combination $A^0 = \{A_i^0 = M_i \mid i = 1,2,...,N\}$. If $PC^0 \leq \overline{PC}$ then A^0 will be a *total project cost feasible solution* else the searching process is immediately terminated *without feasible solution.*

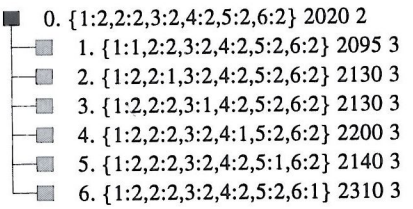
We consider a node *expandable* if in the corresponding *activity-mode combination* at least one *mode index* is greater than one. Thus, in each step of the tree-building process, we select the *first expandable* node, which has

been generated but not expanded. In the tree searching process, a *parent node* is transformed into a *set of child nodes.* A *child node* will be created from its parent node by changing *exactly one* mode index to the next smaller one. Because a child node will be *"at least as expensive"* as its parent node, a *cost infeasible child* can be immediately discarded (pruned).

A *project cost feasible mode combination* generally defines a *project cost feasible mode combination set.*

The reason is very simple. In a *cost feasible mode combination* every mode A_i, which satisfies the relation $A_i < M_i$, may be replaced by a set of cheaper nodes $\{A_i, A_i + 1, ..., M_i\}$.

For the sample problem, the first level searching tree is presented in Figure 6-4.



Legend:

Node {Activity :Mode} ProCost Indicator if
0 Generated
1 Expanded
2 Feasible
3 Infeasible

Figure 6-4 First Level Searching Tree of a Six-Activity Project (Talbot (1982))

6.2.2 The Second Level Searching Tree

On the *second* level, on the set of project cost feasible operating-mode combination sets, we solve the make-span minimization problem subject to the daily cost constraints.

It means that we have to solve the following decision problem:

Given a project cost feasible operating-mode combination set and a maximal project length, does there exist a solution with a project length, that does not exceed the maximal project length and none of the daily cost constraints is violated.

The essence of our algorithm is very simple:

Starting from the network-feasible project length and increasing it step-by-step, in each step we try to solve the *daily inequality system on the set of project cost feasible operating-mode combination sets.*

If the solution set is *not empty*, the process is terminated and the actual project length will be the *cost feasible minimal makespan.*

From methodological point of view, there are at least two new elements in this algorithm Csébfalvi (2000): (1) the objective function (IN) will be used on the third level is *irregular*. Therefore, we must modify the usual makespan minimization procedure according to this fact. (2) We must know *all* the cost feasible solutions, which may be very a complicated problem.

Objective function IN_r is *irregular* since for some $r \in \{1,2,...,R\}$ there may be two feasible schedules $S^1 \in \Re$ and $S^2 \in \Re$ with $IN_r(S^1) > IN_r(S^2)$ and $S_i^1 \leq S_i^2 \ \forall i \in V$.

In the daily cost constrained makespan minimization phase, the project is characterized by the set of its *daily cost violating sets.*

A *daily cost violating set* of activities means the following: (1) All activities in the set *may be* executed concurrently. (2) The total daily cost requirement of these activities exceeds the maximal daily cost $\overline{DC}$. (3) The set does not contain another *daily cost violating set* as a proper subset. See for example Bell and Park (1990).

We note that a cost conflict with *cost violating set* can be explicitly repaired by inserting a *feasible* precedence arc between some pair of activities in the set. This will guarantee that not all members of the chosen resource-violating set can be executed concurrently.

According to our approach, we must redefine the traditional makespan minimization process. The reasons are the following: (1) we must find *all* cost feasible solutions. (2) We must replace the traditional *"visible conflict"* oriented approach by a new *"feasible conflict"* oriented one. In other words, we have to repair every feasible cost conflict regardless of whether it is *"visible"* or *"hidden"*. In our modified approach, in a cost-constrained schedule each *non-critical* activity can be shifted within their available float without affecting the cost feasibility (Csébfalvi (2000)).

In the cost-constrained solution finding process, the nodes of the tree correspond to *"partial"* schedules. In our model, any partial schedule satisfies *all original* precedence constraints and assigns a start time to *all* activities. Nevertheless, it is *"partial"* because it may violate one or more *"visible"* or *"hidden"* cost constraints.

The nodes of the tree are characterized by the set of *original* network relations and a set of *additional* cost conflict repairing relations. The *root* node contains only the network relations.

Leaf nodes of search tree are network and cost feasible schedules or pruned schedules. Our node evaluation function is very simple: It assigns to any node the *number of feasible cost conflicts.* Thus, at each step of the tree-building process, we select the most promising state, which has been generated but not expanded.

A *parent node* is transformed into a *set of child nodes* by repairing its *first* resource conflict all the possible ways. Note that, in this context *"first"* always means the *earliest* conflict in time interval $[1,T]$.

According to our *"best-first"* searching strategy, a node without feasible cost conflict will be a solution of the cost-constrained scheduling problem.

In this simple form, the searching process is very inefficient as too many states are generated. Pruning rules must be employed to cut down the effective branching factor of the search tree.

According to our modified solution strategy, we must revise the traditional pruning rules of resource-constrained project scheduling very carefully since our resource-leveling criterion is not a *regular* measure of performance.

In this study, we applied a pruning rule which are suitable for our *irregular interruption* measure and which are able to reduce the number of generated nodes (Csébfalvi (2000)).

The *"at least as shiftable rule"* is a straightforward modification of the well-known *"left-shiftable rule"* which is a well-known and very efficient *"regular rule"*.

Let $\Re = \Re(E)$ denote the set of *network-feasible* schedules.

In the *cost feasible schedule searching phase* a partially or completely *cost feasible* schedule set is characterized by the set of *original* network relations and a particular set of *additional* cost conflict repairing relations. Let $\Re = \Re(E \cup E^*)$ denote a partially or completely *cost feasible* schedule set, where E^* denotes an additional conflict repairing relation set.

In the *"at least as shiftable rule"* each node of the *searching tree* is characterized by the corresponding schedule set:

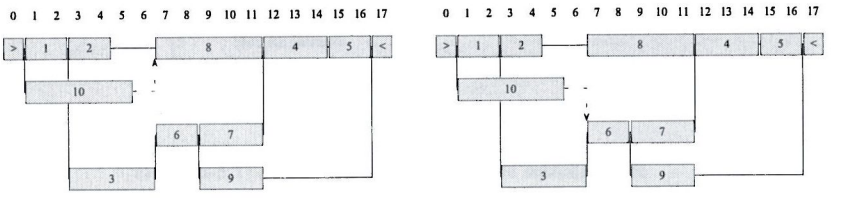
$$\Re(\mathrm{E} \cup \mathrm{E}^{\mathrm{n}}), \text{ where } \mathrm{n} = 0, 1, 2, \ldots \qquad (6\text{-}1)$$

The *"at least as shiftable rule"* involves the comparison of two schedule sets $\Re(E \cup E^i)$ and $\Re(E \cup E^j)$ to conclude that the best feasible schedule

derivable from $\Re(E \cup E^i)$ must be at least as good as the best feasible schedule derivable from $\Re(E \cup E^j)$. This will be the case when $\Re(E \cup E^j)$ no more constrained than $\Re(E \cup E^i)$ (in some well-defined sense).

The *"at least as shiftable rule"* is implemented as follows: Schedule $\Re(E \cup E^j)$ is *"at least as shiftable"* as $\Re(E \cup E^i)$ (and thus node i may be pruned) if: (1) Schedule sets $\Re(E \cup E^i)$ and $\Re(E \cup E^j)$ have *the same cost violating sets*. (2) Each *earliest starting time* in schedule set $\Re(E \cup E^j)$ is less then or equal to the corresponding earliest starting time in schedule $\Re(E \cup E^i)$. (3) Each *latest finishing time* in schedule $\Re(E \cup E^i)$ is less then or equal to the corresponding latest finishing time in schedule $\Re(E \cup E^j)$.

The example project of Figure 6-5 well illustrates the essence of the proposed *"at least as shiftable rule"*. In this example, the left schedule is dominated by the right schedule. Therefore, the left schedule may be immediately discarded because in the schedule set connected to the right schedule activity 8 is *more shiftable* than in the schedule set connected to the left schedule.

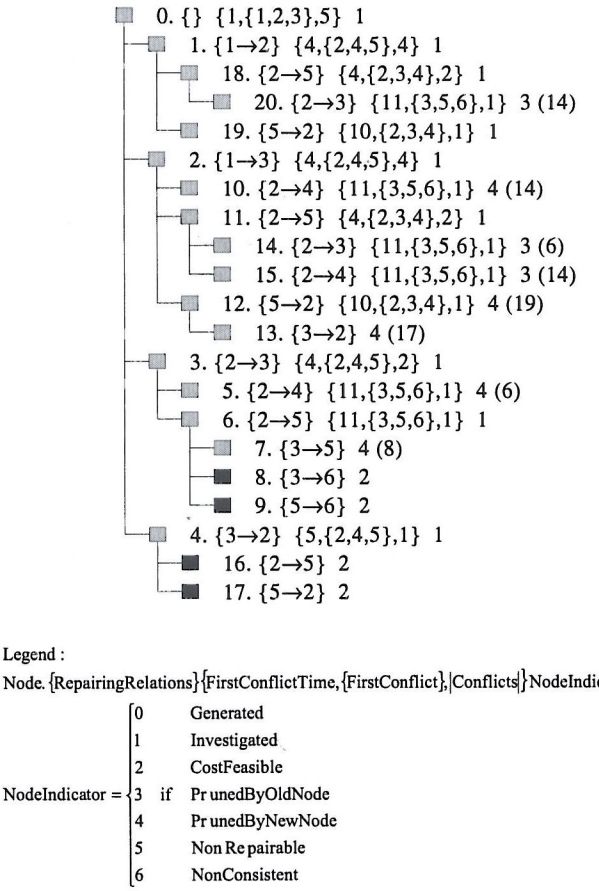


Conflict Repairing Relation: {10→8}

Conflict Repairing Relation: {10→6}

Figure 6-5 The essence of the "at least as shiftable" rule

The daily cost feasible schedule-searching tree is presented in Figure 6-6. The problem has four daily cost feasible solutions.



Legend :

Node. {RepairingRelations}{FirstConflictTime,{FirstConflict},|Conflicts|}NodeIndicator (PrunedBy)

$$\text{NodeIndicator} = \begin{cases} 0 & \text{Generated} \\ 1 & \text{Investigated} \\ 2 & \text{CostFeasible} \\ 3 & \text{if PrunedByOldNode} \\ 4 & \text{PrunedByNewNode} \\ 5 & \text{NonRepairable} \\ 6 & \text{NonConsistent} \end{cases}$$

Figure 6-6 The daily cost feasible schedule searching tree

6.2.3 The Third Level Searching Tree

In this level, on the set of *make-span minimal cost feasible schedules*, we solve the resource-balancing problem.

Given the set $\Re^i$ of *cost feasible* schedules $i \in \{1,2,...,c\}$, a schedule $S^j \in \Re^i$ is an *efficient concave* schedule if no other schedule $S^k \in \Re^i$ exists such that $IN_r(S^k) \leq IN_r(S^j)$ for each $r = 1, 2, ..., R$ with strict inequality for at least one r.

According to the above definition, the *shape optimization phase* (IN) can be conceptually formulated as follows:

$$\begin{gathered} \textit{for each } i \in \{1,2,...,c\} \\ \textit{find all efficient solutions of} \\ IN(S) = (IN_1(S), IN_2(S), ..., IN_R(S)) \\ \text{subject to } S \in \Re^i \end{gathered} \qquad \text{(IN)}$$

In our approach, an *efficient balanced* schedule S^{ij} is characterized by the set $\{E \cup E^i, F^j\}$, where E is the set of *original* precedence relations, E^i is the set of additional cost conflict repairing relations, and $F^j, (j = 1, 2, ...)$ is the set of *explicitly* prescribed activity starting times. As mentioned above, an *explicitly* prescribed activity starting time is represented by $i \Rightarrow t$ where $i \in V'$ and $t \in [ES_i, LS_i]$.

For a given cost constrained schedule, the *efficient balanced* schedule searching process can be formulated as a tree search problem, where the

nodes are characterized by the set of explicitly prescribed activity starting times.

An activity is considered *"movable"* in a schedule set if it has no explicitly prescribed starting time and it is *non-critical*. By definition, the starting position of such activities is the *earliest* CPM starting time.

The *root* node of the search tree is a *dummy* node. The *first level nodes* are characterized by the particular cost conflict repairing relation sets. In this level, each movable activity is movable so the set of explicitly prescribed activity starting times is *empty*.

From the second tree level, every child node inherits its resource conflict repairing relation set from its parent node.

Our node evaluation function is very simple: $\sum_{r=1}^{R} IN_r$. Thus, at each step of the tree-building process, we select the most promising state, which has been generated but not expanded.

A low $\sum_{r=1}^{R} IN_r$ value indicates that a node may be *"near"* to optimal.

In the tree-building process, a *parent* node which has one or more movable activities is transformed into a *set of child nodes* by fixing the starting time of the *first* such activity all the possible ways.

Note that, in this context *"first"* always means the *first* activity in *topological order*.

The *leaves* of the search tree correspond to schedules, which have no movable activity anymore, or have been pruned according to the *"at least as good rule"* introduced previously. The tree-building process maintains the dynamically changing set of *efficient* schedules.

A sub-tree is presented in Figure 6-7. The *cost feasible* early CPM schedule and the *"best" cost feasible balanced solution* are shown in Figure6-8.

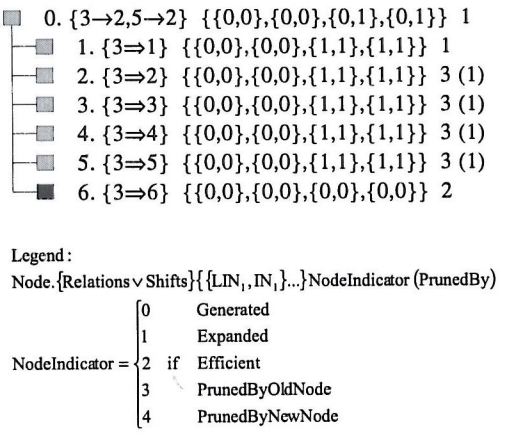
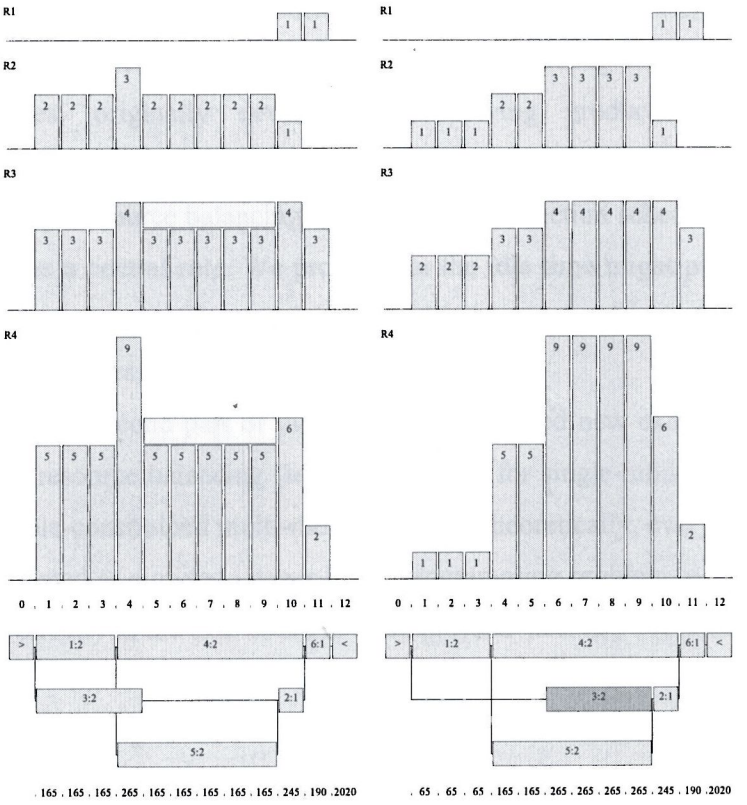


Figure 6-7 A sub-tree $\{3 \rightarrow 2,5 \rightarrow 2\}$ from the searching tree



Cost Feasible Early Start Schedule

Optimally Shifted Schedule

Figure 6-8 The Early Start and the *"best"* Schedules

7. NEW RESULTS

This thesis contributes the following new results:

1. In the first part of this thesis, we presented a new global interruption-oriented resource-balancing measure based on quasi-concavity. It was the main objective of this thesis is to show how modeling ideas and solution procedures, originally developed for solving production scheduling problems, can be used to formulate and solve several types of project scheduling (resource balancing) problems. In production scheduling, the idle time plays a central role. We proved that the idle time might play a similar role in project scheduling and it might be the starting base of new global performance measures.

2. In the second part of the thesis, we presented new exact interruption oriented resource balancing (leveling) models for single-mode, multi-mode, and double-constrained multi-mode projects. Theoretically, every new model is formulated as a mixed integer linear programming problem, which may be solved directly in the case of small-scale projects with any commercial MILP package.

3. According to the NP-hard nature of resource balancing, for medium-size projects we introduced implicit enumeration algorithms with effective pruning rules for every problem type. In the case of double-constrained multiple-mode projects, the proposed algorithm was formulated as a multilevel tree-searching problem. For large-scale problems, simple *"beam-search like"* heuristics can be applied by constraining the size of the searching-trees.

REFERENCES

Brucker, P., Drexl, A., Möhrig, R., Neumann, K. and Pesch, E. Resource-constrained scheduling: Notations, classifications, models, and methods. European Journal of Operational Research, 112(1): 3-41, 1999.

A. Ghobadian and G. Csébfalvi, Workshop on developing interactive learning material for project management and statistical quality control, Proc. 1995 Annual Meeting, Decision Sciences Institute (US), Boston, Massachusetts, US, 20-22 November, 1995.

Bandelloni, M., Tucci, M. and Rinaldi, R. Optimal resource leveling using non-serial dynamic programming. European Journal of Operational Research, 78:162-167, 1994.

Bell, C. E. and Park, K. Solving resource-constrained project scheduling problems by A* search. Naval Research Logistics, 37: 61-84. 1990.

Burgess, A. R. and Killebrew, J. B. Variation in activity level on a cyclical arrow diagram. Journal of Industrial Engineering, 13: 76-83, 1962.

Csébfalvi, G. A fast exact solution procedure for the multiple resource-constrained project scheduling problem. Proc. APMOD '98 Extended Abstracts, Limasol, Cyprus, 1998.

Csébfalvi, G. A new exact resource leveling procedure for the multiple resource-constrained project scheduling problem, Naval Research Logistics, 1998 (second revised version submitted).

Csébfalvi, G. A new global resource leveling procedure for the multiple resource-constrained project scheduling problem, Proc. NAPW, NY US, 2000.

Csébfalvi, G. A new multi-criteria resource leveling procedure for the multiple resource-constrained project scheduling problem, Proc. EURO XVII, Budapest, Hungary, 2000.

Csébfalvi, G. Egy optimális erőforrás kiegyenlítő eljárás tevékenységi hálókra, "Új utakon a magyar operációkutatás", tanulmánykötet, szerkesztők: Komlósi S. és Szántai T. Dialóg Campus, 1999.

Csébfalvi, G. és Konstantinidis P. Egy implicit leszámláláson alapuló új erőforrás kiegyenlítő eljárás, Szigma, 29 (43-52), 1998.

Csébfalvi, G. Habilitation Dissertation, PTE-KTK, Pécs, 2000.

Csébfalvi, G. ProMan Manual, PTE-KTK, Pécs, 2002.

Csébfalvi, G. and P. Konstantinidis, A new exact resource balancing procedure for the multiple resource-constrained project scheduling problem, Proc. APMOD '98 Extended Abstracts, Limasol, Cyprus, 11-13 March, 1998

Csébfalvi, G. and P. Konstantinidis, A new exact resource leveling procedure for the multiple resource-constrained project scheduling problem, Proc. 5th International Conference of the Decision Sciences Institute (US), Athens, Greece, 1999.

Easa, S. M. Resource leveling in construction by optimization. Journal of Construction Engineering and Management, 115: 302-316, 1989.

Hajdu, M. Network Scheduling Techniques for Construction Project Management, Kluwer Academic Publisher, Boston, US, 1997.

Konstantinidis, P. A model to optimize project resource allocation by construction of a balanced histogram. European Journal of Operational Research, 29: 559-571, 1998.

Konstantinidis, P. Arising the productivity of resources with the construction of balanced histograms Journal of Technical Chamber of Greece (Greek),, 28: 80-89, 1996.

Konstantinidis, P. Optimal resource allocation with network analysis in project management, Journal of Civil Engineers of Greece (Greek) 29: 23-29, 1996.

Kruzslicz, F. A new exact resource allocation model with hard and soft resource constraints, Proc. OR2002, Klagenfurt, Austria, 2002.
LINDO Release 6.1, LINDO Systems, Inc., Chicago. 1997.
Lova et al. A multicriteria heuristic method to improve resource allocation in multi-project scheduling. European Journal of Operational Research,127: 408-424, 2000.
Neumann, K. and Zimmermann, J. Resource leveling for projects with schedule-dependent time windows. European Journal of Operational Research, 117(3): 591-605,1999.
Patterson, J. A comparison of exact procedures for solving the multiple resource constrained project scheduling problem. Management Science, 30(7): 854-867. 1984.
Talbot, F. B. Resource-constrained project scheduling with time-resource tradeoff: the nonpreeemty case. Management Science, 28(10): 1197-1210, 1982.
Tavares, V. L.. Optimal resource profiles for program scheduling. European Journal of Operational Research, 29: 83-90,1987.
Weglarz, J. Project Scheduling Recent Models, Algorithms and Applications. Kluwer Academic Publishers, Boston/New York/Dordrecht, 1999.
Youness, E. A. A direct Approach for finding all efficient solutions for multiobjective programming problems. European Journal of Operational Research, 81(2): 440-443, 1994.
Zimmermann, J. and Engelhart, H. Lower Bounds and Exact Methods for Resource Leveling Problems. Report WIOR-517, University of Karlsruhe, 1998.