## Work continuity constraints in project scheduling

the Autonomous Management School of
Ghent University and Katholieke Universiteit Leuven

**Vlerick Leuven Gent Working Paper Series 2005/10**

**WORK CONTINUITY CONSTRAINTS IN PROJECT SCHEDULING**

MARIO VANHOUCKE

Mario.Vanhoucke@vlerick.be

D/2005/6482/10

# WORK CONTINUITY CONSTRAINTS IN PROJECT SCHEDULING

MARIO VANHOUCKE

Vlerick Leuven Gent Management School

**Contact:**

Mario Vanhoucke

Vlerick Leuven Gent Management School

Tel: +32 09 210 97 81

Fax: +32 09 210 97 00

Email: Mario.Vanhoucke@vlerick.be

**ABSTRACT**

Repetitive projects involve the repetition of activities along the stages of the project. Since the resources required to perform these activities move from one stage to the other, a main objective of scheduling these projects is to maintain the continuity of work of these resources so as to minimize the idle time of resources. This requirement, often referred to as work continuity constraints, involves a trade-off between total project duration and the resource idle time.

The contribution of this paper is threefold. Firstly, we provide an extensive literature summary of the topic under study. Although most research papers deal with the scheduling of construction projects, we show that this can be extended to many other environments. Secondly, we propose an exact search procedure for scheduling repetitive projects with work continuity constraints. This algorithm iteratively shifts repeating activities further in time in order to decrease the resource idle time. We have embedded this recursive search procedure in a horizon-varying algorithm in order to detect the complete trade-off profile between resource idle time and project duration. The procedure has been coded in Visual C++ and has been validated on a randomly generated problem set. Finally, we illustrate the concepts on three examples. First, the use our new algorithm is illustrated on a small fictive problem example from literature. In a second example, we show that work continuity constraints involve a tradeoff between total project duration and the resource idle time. A last example describes the scheduling of a well-known real-life project that aims at the construction of a tunnel at the Westerschelde in the Netherlands.

Keywords: Project Management; CPM; work continuity; repetitive project scheduling.

# 1 INTRODUCTION

Construction projects are often characterised by *repeating activities* that have to be performed from unit to unit. Highway projects, pipeline constructions and high-rise buildings, for example, commonly require resources to perform the work on similar activities that shift in stages. Indeed, construction crews perform the work in a sequence and move from one unit of the project to the next. This is mainly the result of the subdivision of a general activity (e.g. carpentry) into specific activities associated with particular units (e.g. carpentry at each floor of a high-rise building).

The repetitive processes of these construction projects can be classified according to the direction of successive work along the units. In *horizontal repetitive projects* the different processes are performed horizontally, as seen in pipeline construction or paving works. These construction projects are often referred to as *continuous* repetitive projects or *linear* projects due to the linear nature of the geometrical layout and work accomplishment. When progress is performed vertically, we refer to v*ertical repetitive projects*, among which high-rise building construction is the classical example. Rather than a number of activities following each other linearly, these construction projects involve the repetition of a unit network throughout the project in discrete steps. It is therefore often referred to as *discrete* repetitive projects. Kang, Park and Lee (2001) argue that construction projects can consist of both horizontal and vertical repetitive processes among several multi-storey structures and refer to this type as *multiple repetitive projects*.

El-Rayes and Moselhi (1998) distinguish between typical and atypical repetitive activities. *Typical repetitive activities* are characterized by identical durations over all units, while *atypical repetitive activities* assume variation of duration from one unit to another. This variation can be attributed to variations in the quantities of work encountered or crew productivity attained in performing the work of these units (Moselhi and El-Rayes, 1993).

A crucial point in scheduling these projects is to ensure the uninterrupted usage of resources of similar activities between different units. Idleness that does not find its roots in forced causes, such as bad weather or equipment breakdowns, is classified as unforced idleness or waste. This waste in repetitive projects stems from resources (crew, equipment,…) waiting for preceding resources to finish their work and has to be eliminated to maintain continuity of work (Harris and Ioannou, 1998). Consequently, in order to maintain work continuity, repetitive units must be scheduled in such a way as to enable timely movement of resources from one unit to the next, avoiding resource idle time. This is knows as the *work continuity constraints* (El-Rayes and Moselhi, 1998).

In this paper, we focus on the vertical (or discrete) repetitive project scheduling problem with work continuity constraints. The organization of the paper is as follows. In section 2 we present an overview of the scheduling literature of projects with repeating activities. Section 3 describes the features of the project scheduling problem under study. In section 4 we present our algorithm for scheduling a repetitive project with work continuity constraints. In section 5 we illustrate our new algorithm on three project examples. Section 6 reports detailed computational results on two randomly generated problem sets. In section 7 we give our overall conclusions and suggestions for future research.

## 2 LITERATURE OVERVIEW

The effective planning, scheduling and control of repetitive construction projects are of crucial importance and lead to a reduced construction time, reduced cost overruns and minimization of disputes (Callahan et al., 1992). Different scheduling methods have been proposed in literature in order to obtain these benefits. A common remark, however, is that most techniques dealing with this scheduling problem are overshadowed by the critical path method (CPM) which lacks a number of features needed in the repetitive scheduling industry. Therefore, a need exists to expand the features of the CPM to include some extra repetitive scheduling features. El-Rayes (2001) states that scheduling of repetitive construction projects can be significantly improved by three main practical requirements, i.e. (*i*) crew work continuity, (*ii*) optimize scheduling and resource utilization so as to minimize project duration and (*iii*) the integration of repetitive and nonrepetitive scheduling techniques. These details of these three requirement are explained below.

*(i)* The application of crew work continuity constraints provides room for an effective resource utilization strategy by minimizing crew idle time. Consequently, this leads to the maximization of the benefits from the learning curve effect for each crew (Shtub et al., 1996) and the minimization of the off-on movement of crews on a project once work has begun (Ashley, 1980, Birrell, 1980). However, Russell and Wong (1993) take a critical look and state that this requirement should not be strictly enforced in scheduling repetitive activities. Selinger (1980) recognizes a trade-off in scheduling repetitive units: work interruption indeed results in an increased direct cost because of the idle crew time and

therefore needs to be avoided. But violation of these work continuity constraints by allowing work interruption may possibly lead to an overall project duration reduction and the corresponding indirect costs, and consequently, a careful trade-off should be made between these two extremes.

*(ii)* We note that minimizing the project duration is a more complex process for repetitive projects than for nonrepetitive ones. Indeed, the simple logic of crashing critical activities in order to shorten the total project duration does no longer hold with the presence of work continuity constraints. Therefore, Rowings and Harmeling (1993, 1994) proposed the linear scheduling method (LSM) to determine the critical path in a linear schedule. Similar to the forward and backward calculations of CPM, the algorithm identifies to so-called *controlling activity path* (CAP). Activities or segments of activities not on the CAP must have float (Harmelink, 2001).

*(iii)* Most construction projects contain both repetitive and nonrepetitive activities. Since the nonrepetitive activities can be scheduled using traditional network techniques while the repetitive activities require more specialized tools with work continuity features, the two scheduling techniques need to be combined in an efficient scheduling model (O'Brein et al, 1985; Russell and Wong, 1993).

In this paper we focus on the work continuity constraints of repetitive project scheduling by taking both the resource idle time and the total project duration into account. In doing so, we take into account the practical requirements mentioned above and recognize the trade-off between idle time minimization and project duration.

As mentioned previously, traditional network techniques such as the critical path method have been criticized in literature for their major drawbacks when applied to scheduling of repetitive projects. According to Reda (1990), the use of the CPM for scheduling repetitive projects has three major disadvantages. First, CPM needs to rely on a large number of activities needed to represent repetitive projects. Indeed, each unit in a repetitive network contains the same activities which can be represented by a project network. Due to the repeating character of the activities between the units of the project, the complete CPM network will have a ladder-like appearance. Each stair denotes the work at one unit consisting of the several activities and precedence relations for that unit. Since the CPM network shows all the links between similar activities of successive units, the number of nodes and arcs of the complete network will be very large. A second drawback is the inability of CPM to guarantee continuity of work. Although it has been reported by several authors that the uninterrupted utilization of resources is an extremely important issue, neither CPM nor its resource-oriented extensions take these work continuity constraints into account. Finally, CPM incorporates the notion of activity crashing by assigning extra resources to the activities, resulting in the well-known time/cost trade-off activity profile. When applying this principle to repetitive projects, activity crashing at one unit leads to a modification of production rates between similar activities at different units. Mattila and Abraham (1998) come to similar conclusions and focus on linear construction as a segment of construction scheduling in which CPM is inadequate. This includes the inability of CPM to (*i*) model work continuity constraints, (*ii*) to handle the large number of activities needed to represent repetitive projects and (*iii*) to accurately reflect actual conditions.

Recognition of the drawbacks of traditional CPM network models in scheduling repetitive projects has led to the development of several scheduling methodologies under different names. Harris and Ioannou (1998) give an overview and distinguish between methodologies for vertical projects with discrete units and horizontal projects where progress is measured linearly. For projects with discrete units, they mention line of balance (Carr and Meyer, 1974; Harris and Evans, 1977), construction planning technique (Peer, 1974; Selinger, 1980), vertical production method (O'Brein, 1975), time-location matrix model (Birrell, 1980), time space scheduling method (Stradal and Cacha, 1982), disturbance scheduling (Whiteman and Irwig, 1988), horizontal and vertical logic scheduling for multistorey projects (Thabet and Beliveau, 1994). For horizontal projects, developed techniques are time versus distance diagrams (Gorman, 1972), linear balance charts (Barrie and Paulson, 1978), velocity diagrams (Dressler, 1980), linear scheduling method (Johnston, 1981; Chrzanowski and Johnston, 1986; Russell and Casselton, 1988). In their paper, Harris and Ioannou (1998) integrate these methods into the *repetitive scheduling method* (RSM) which is a practical scheduling methodology that ensures continuous resource utilization applicable to both vertical and horizontal construction scheduling. In section 5.1 of this paper, we will refer to the project example of Harris and Ioannou (1998) in order to compare our algorithm with the repetitive scheduling method.

Due to the overwhelming use of the CPM technique in repetitive project scheduling, a number of attempts have been made to compare this traditional technique with the more specialized tools. Schoderbek and Digman (1967) have introduced PERT/LOB as an attempt to combine the merits presented by the Program Evaluation and Review Technique with the Line-Of-Balance principles. Al Sarraj (1990) presented a mathematical model for LOB in order to find the start and finish times for repetitive activities and the corresponding project duration. Suhail and Neale (1994) develop a new methology to integrate CPM and LOB. To that purpose, they incorporate the resource leveling principles and float times calculations of the CPM into the LOB technique. Yamin and Harmelink (2001) compare the linear scheduling method (LSM) and the critical path method (CPM) in detail and conclude that specialization can be beneficial to the project. However, although LSM can be superior to CPM for very specific projects, further research is needed to elevate the LSM to the CPM level. In this paper, we try to narrow the gap between the traditional scheduling methods and the repetitive project scheduling requirements by adding work continuity constraints into the CPM scheduling philosophy. Despite the fact that the number of activities of a CPM network for a repetitive project can increase dramatically, we will show in the computational results section that this does not harm the efficiency of our new algorithm in a severe way.

## 3 WORK CONTINUITY CONSTRAINTS FOR CPM NETWORKS

Throughout this paper, we assume that a project is represented by an activity-on-the-node (AoN) network where the set of nodes, $N$, represents activities and the set of arcs, $A$, represents the precedence constraints. Since progress is performed in discrete steps (as in vertical repetitive projects), we assume that this network is repeated in $K$ units. The duration of each activity $i$ at unit $k$ is denoted by $d_{ik}$ ($1 \leq i \leq n$ and $1 \leq k \leq K$). In a similar way, we denote the starting and finishing time of activity $i$ at unit $k$ by $s_{ik}$ and $f_{ik}$, respectively. Consequently, we extend the original unit network to a large network consisting of repeating activities between units. Moreover, we add a dummy start activity 0 at the first unit to denote the start of the project. This dummy activity is a predecessor for all activities of the first unit that have zero predecessors. In a similar way, we add a dummy end activity $n + 1$ at the last unit $K$ to denote the finishing of the project. Consequently, this activity is a successor for all activities belonging to unit $K$ with no successors.

Resources are needed for each activity $i$ that shifts along the units, from unit 1 to unit $K$. The problem under study involves the minimization of the idle time of resources between different units for a project with a given deadline.

In literature, work continuity constraints are often linked with the minimization of crew idle time. In the sequel of this paper, we use the more general term, *resource idle time*, since the minimization of idle time of resources may not be restricted to crews only. Two examples are given below:

- De Boer (1998) introduced *spatial resources* as a resource type that is not required by a single activity but rather by a group of activities. Examples are dry docks in a ship yard, shop floor space or pallets. Since the spatial resource unit is occupied from the first moment an activity from the group starts until the last activity of the group finishes, work continuity constraints can be of crucial importance.
- Gong (1997) has introduced the concept *time dependent cost (TDC)* as a part of the project costs that changes with the variation of activity times. The TDC is defined as the product of unit time cost and service time. Goto et al. (2000) elaborate on that concept and argue that the service time of a *time dependent cost resource* is the time duration starting form the first use and ending at the last. They refer to the use of a tower crane in the construction industry and argue that the reduction of waiting times of TDC resources naturally reduces the time dependent cost.

These research papers motivated us to use the general term ‘resource idle time’ rather than the more specific ‘crew idle time’. The project scheduling problem with work continuity constraints can be formulated as follows:

$$\text{Minimize} \sum_{i=1}^{n} (s_{iK} - s_{i1}) \qquad [1]$$

Subject to

$$s_{ik} + l_{ijkk} \leq s_{jk} \qquad k = 1, \ldots, K \text{ and } \forall (i, j) \in A \qquad [2]$$

$$s_{ik} + l_{iik,k+1} \leq s_{i,k+1} \qquad i = 1, \ldots, n \text{ and } k = 1, \ldots, K-1 \qquad [3]$$

$$s_{01} = 0 \qquad [4]$$

$$s_{n+1,K} \leq \delta_{n+1} \qquad [5]$$

$$s_{ik} \in \text{int}^{+} \qquad i = 1, \ldots, n \text{ and } k = 1, \ldots, K \qquad [6]$$

$$s_{01}, s_{n+1,K} \in \text{int}^{+} \qquad [7]$$

where $l_{ijkl}$ denotes the time-lag for the precedence relation between activity $i$ on unit level $k$ and activity $j$ on unit level $l$. These time-lags representing the different types of generalized precedence relations can be represented in a standardized form by reducing them to minimal start-start precedence relations as shown by the transformation rules of Bartusch et al. (1998). As an example, $l_{ijkk}$ has to be replaced by $d_{ik}$ in Eqs. [2] to model the simple CPM case where only minimal precedence relations with zero time-lags are involved.

The objective in Eq. 1 denotes the work continuity constraints and minimizes the resource idle time between similar activities at different units. We note that the word ‘constraint’ is somewhat confusing since the work continuity of the schedule is guaranteed in the objective function of the model. Since the resource idle time is measured for resources that shift between units, it is sufficient to minimize the timespan of activities between the first and last unit. Indeed, these resource are needed at the start of the activity at the first unit and will only be released at the completion of this activity at the last unit K. Consequently, the starting times of all intermediate activities have no influence on the idle time of this resource and are therefore not included in the objective function. The constraint set given in Eq. 2 maintains the (generalized) precedence relations among the activities of the project network at each unit. The constraint set in Eq. 3 maintains the (generalized) precedence relations among similar activities between consecutive units. Eq. 4 forces the dummy start activity 0 to start at time zero and Eq. 5 forces the dummy end activity n + 1 (and consequently the project) to end on or before a negotiated deadline $\delta_{n+1}$. Eq. 6 ensures that the activity starting times assume nonnegative integer values. Eq. 7 ensures that the single dummy start and single dummy end activity takes a nonnegative integer value.

Note that the formulation of Eqs.[1]-[7] is general and allows the use of both typical and atypical activities. Therefore, we use $l_{ijkk}$ rather than a unit-independent $l_{ij}$, to express that the duration of each activity is not assumed to have a fixed value over all units. In doing so, we allow to incorporate crew productivity, differences in amounts of work between units or learning effects of crews. It has been noted by several authors that it may be necessary to incorporate the learning effects into activity time estimates in order to improve the accuracy of schedules, especially for programs consisting of repetitive projects (Amor (2002), Amor and Teplitz (1993, 1998), Badiru (1995), Shtub (1991) and Shtub et al. (1996)).

As described in section 2, most research efforts in literature focus on the construction industry where the project consists of repeating activities along the units of a project. Even in the formulation of Eqs.[1]-[7] we use $l_{ijkk}$ and consequently, we again assume that the project consists of repetitive subparts. However, numerous examples outside the construction industry can be described where the minimization of the resource idle time is of crucial importance, without being confronted with repeating activities along the units. The following four examples illustrate the possible generalization of the work continuity constraints to other project environments:

**Outsourcing activities / hired material.** Project scheduling problems where a set of activities has been outsourced, or that rely on external resources (subcontracting, consultants, etc…) need to be scheduled with work continuity constraints. This means that the set of activities can be divided into activity groups that have to be executed within the smallest possible time-span in order to minimize the total cost of outsourcing.

**Programme scheduling with different stakeholders.** In programmes that consist of different projects for different stakeholders, each subproject can be seen as an individual ‘activity group’ where work continuity can be of importance. Consequently, it is beneficial to schedule the activities within an activity group within the smallest possible time-span (within the precedence and resource constraints of the complete programme) rather than simply resolving resource conflicts without taking the different subprojects into account. In doing so, we minimize the project duration towards each stakeholder and increase the satisfaction of the different stakeholders. As an example, Vanhoucke and Demeulemeester (2003) discussed a capacity expansion project at a Flemish water production company where different stakeholders are involved. The project consists of different subprojects that are all crucial for the realization of the complete project. Some subprojects, however, can be considered as an individual project since they have immediate repercussions to some stakeholders. The schedule is constructed in a way that subprojects are scheduled within the smallest possible time-span in order to satisfy the individual stakeholders. This can be done by minimizing work continuity constraints within each subproject.

**Learning effects in projects.** Learning effects between activity groups in projects result in reduced activity durations and costs, and consequently reduced overall project duration. In order to fully exploit these advantages of learning effects between activity groups, it is beneficial to minimize the total duration of each individual activity group (work continuity). Indeed, learning effects occur whenever completing an activity group before another one. By minimizing the work continuity per group, the different activity groups can be scheduled one after the other (rather than 'mixing' the activities between groups within the precedence and resource constraints of project), leading to a maximal effect of learning.

**Project with time-critical subprojects.** In projects, where only a subpart is time-critical, work continuity constraints can be important for scheduling the time-critical sub-network. As an example, we refer to a maintenance project in a luggage handling system at an airport. These projects typically have a small project duration (about 3 weeks), and time is the main objective in the project scheduling phase. However, only a subpart of the project is time critical, i.e. the part which involves a shutdown of a part of the luggage handling line, which involves a penalty cost when exceeding the negotiated shut-downtime. Consequently, the activities which involve a shutdown of the luggage line can be seen as an activity group where work continuity is the main issue.

The problem formulation represented by Eqs. [1]-[7] is to schedule all project activities with minimal resource idle time, without violating a given project deadline. However, Hegazy and Wassef (2001) and Selinger (1980) argue that minimizing work continuity as such is not the target since adding work interruption in the schedule can be beneficial. Hegazy and Wassef (2001) present a cost optimization model in order to minimize total construction costs comprising direct cost, indirect cost, interruption cost as well as incentives and liquidated damages. They use a genetic algorithm approach to consider project deadline, crew synchronization and resource constraints simultaneously. They argue that adding work interruption can be beneficial to the total project duration by allowing earlier units of activities to start earlier than in the schedule with minimal resource idle time. Inspired by these observations, we developed an algorithm which search for the *complete trade-off profile between project duration and resource idle time*. To that purpose, we use a horizon-varying approach which involves the iterative optimal solution for Eqs. [1]-[7] over the feasible project durations in the interval bounded from below (the minimal project duration) and from above (the maximal project duration). The minimal project duration corresponds to the critical path length while the maximal project duration corresponds to a schedule with minimal resource idle time. This means that a further increase in this project duration will not lead to an improvement of the resource idle time. In the next section, we discuss a recursive search algorithm to solve the problem given by Eqs.[1]-[7] (i.e. with a given fixed deadline). In section 4.4, we embed this procedure into our horizon-varying approach in order to find the complete trade-off profile between resource idle time and total project duration.

## 4 THE ALGORITHM

The proposed algorithm to minimize the resource idle time of problem [1]-[7] consists of three steps: an activity labeling step, the construction of a search tree and the recursive search in this tree. This algorithm is a modified version of the recursive search algorithm proposed by Vanhoucke et al. (2001) in order to solve a totally different problem (the so-called max-*npv* problem).

### 4.1 Step one. Activity labeling to simulate attraction

The procedure starts with assigning a label to each activity at each unit in the following way: each activity of the project network receives a label with no value (0) except for non-dummy activities at the first and the last unit. These activities get a negative label (-1) at the first unit and a positive label (+1) at the last unit, as follows:

Label $(0, 1)$ = Label $(n + 1, K) = 0$
**Do** for $i = 1, \ldots, n$
    Label $(i, 1) = -1$
    Label $(i, K) = 1$
    Label $(i, k) = 0$ for $k = 2, \ldots, K - 1$

The purpose of this simple labeling mechanism is to simulate attraction between activities in order to minimize resource idle time. Indeed, if we simulate attraction between similar activities $i$ at the first (1) and last unit ($K$), we force the schedule to minimize the time distance $s_{iK} - s_{i1}$ between these activities. As mentioned before, the starting times of all intermediate activities (i.e. $s_{ik}$ with $k = 2, \ldots, K - 1$) are irrelevant for the work continuity calculations since they have no influence on the total resource idle time of this activity over all units. Similarly, Suhail and Neale (1994) illustrate in their LOB calculations that it is only necessary to show only the activities at the first and the last unit.

## 4.2 Step two. Build a search tree

It has been mentioned before that the complete CPM network has a ladder-like appearance, due to the repeating character of the activities between the units. For the sake of simplicity, we rename all activities in the sequel of this paper. In doing so, we can transform the repetitive network into a CPM network where precedence relations exist between activities at the same unit level or between successive levels. As an example, dummy activity 0 at level 1 will be activity 0 in the new network, activity 1 at unit 1 will be activity 1, activity 1 at level 2 will be activity $n + 1$ and so on. In a similar way, we modify the subscripts of the symbols $s_{ik}$ and $l_{ijkl}$ to $s'_{i'}$ and $l'_{i'j'}$. Consequently, $l'_{i'j'}$ can be used to denote a time-lag between activities at the same unit level (i.e. $l_{ijkk}$) or between units (i.e. $l_{iikk+1}$).

Note that we have used $n$ to denote the number of non-dummy activities at each unit (i.e. $n = |N|$), $K$ to denote the number of repetitive units of the project and $A$ to denote the set of precedence relations between the activities at each unit level. The new network containing all repetitive units has $n'$ activities and $|A'|$ precedence relations, for which $n' = |N'| = K * n + 2$ and $|A'| = K * |A| + n * (K - 1) + 2$.

In this step, we create an earliest start schedule by simply scheduling all the activities as soon as possible within the precedence constraints. On top of that, we construct a spanning tree that forms the basis of our recursive search of step 3. The latter has been investigated by Grinold (1972), who has shown that the search for an optimal schedule for the payment scheduling problem can be restricted to feasible trees in the project graph.

Using *bool* as a boolean variable used in the while test, *ST* to denote the spanning tree and *CA* to denote the set of considered activities, the pseudocode to build the spanning tree *ST* is as follows:

---

**BUILD SPANNING TREE**

Initialize $CA = \{0\}$, $ST = \emptyset$ and $bool$ = true
Set $s'_0 = 0$ and $s'_i = -\infty \mid i = 1, \ldots, n'$
**While** $bool$ = true
  $bool$ = false
  **Do** $\forall(i, j) \in A'$
    **If** $s'_i + l'_{ij} > s'_j$ **then** $bool$ = true and $s'_j = s'_i + l'_{ij}$
**While** $CA \neq N$
  **Do** $\forall(i, j) \in A'$
    **If** $I \in CA$ and $j \notin CA$ and $s'_i + l'_{ij} = s'_j$ **then**
      $CA = CA \cup \{j\}$
      $ST = ST \cup (i,j)$
**Return**

---

After this step, all activities are scheduled as soon as possible. Moreover, in the project network, only a subset of precedence relations *ST* is highlighted such that they form a tree. For more details, we refer the reader to Vanhoucke et al. (2000) in which four recursive search procedures for the so-called max-*npv* problem are compared to test their efficiency.

### 4.3 Step three. Recursive search

In a third step the spanning tree is the subject of a *recursive search* (using the dummy start activity 0 as the search base) in order to identify sets of activities (*SA*) that might be shifted forward (away from time zero) to improve the work continuity of the repetitive project. When a set of activities *SA* is found for which a forward shift leads to an improvement of the work continuity, the algorithm computes an allowable displacement interval and updates the spanning tree *ST*. The starting times of the activities of *SA* are increased by the allowable displacement interval and the algorithm repeats the recursive search. The allowable displacement interval $v_{r^*s^*}$ simply calculates the minimal distance over which an activity $r \in SA$ can be shifted until it connects with an activity $s \notin SA$. If no further shift can be accomplished, the algorithm stops and the starting times of the activities of the project are reported.

This recursive search is similar to the recursive procedure proposed by Vanhoucke et al. (2001) for the maximization of the net present value of an unconstrained project scheduling problem. In this procedure, activities with a negative cash flow are scheduled as late as possible while activities with a positive cash flows are scheduled as soon as possible. By labeling our activities we simulate attraction between these activities since a negative label (-1) is similar to a negative cash flows and consequently has to be scheduled as late as possible. A positive label (+1) is linked with a positive cash flows which is forced to be scheduled as soon as possible.

The pseudocode of the third step, in which the recursion step is repeated several times, can be written as given below. The set *CA* denotes the set of already considered activities, *ST* denotes the spanning tree, *CL* denotes the cumulative label values and $v_{r^*s^*}$ the allowable displacement interval.

---

**procedure** Step 3: the recursive search method
$CA = \varnothing$
**Do** *RECURSION(*1*)* $\rightarrow$ *SA'*, *CL'* (parameters returned by the recursive function)
Report the optimal starting times of the activities. STOP.

---

**RECURSION(NEWNODE)**
Initialize $SA = \{newnode\}$, $CL = Label_{newnode}$ and $CA = CA \cup \{newnode\}$
**Do** $\forall$node$_i$ | node$_i \notin CA$ and node$_i$ succeeds *newnode* in the current tree *CT*:
(*remark that* node$_i$ *can be on the same level k or on the following level than newnode*)
*RECURSION(*node$_i$*)* $\rightarrow$ *SA'*, *CL'*
**If** $CL' \geq 0$ **then**
Set $SA = SA \cup SA'$ and $CL = CL + CL'$
**Else**
$CT = CT \setminus (newnode, \text{node}_i)$
Compute $v_{r^*s^*} = \min_{\substack{(r,s)\in A' \\ r \in SA' \\ s \notin SA'}} \{s'_s - s'_r - l'_{rs}\}$ and set $ST = ST \cup (r^*, s^*)$
**Do** $\forall j \in SA'$: set $s'_j = s'_j + v_{r^*s^*}$
Go to Step 3: the recursive search method
**Do** $\forall$node$_i$ | node$_i \notin CA$ and node$_i$ precedes *newnode* in the current tree *CT*:
(*remark that* node$_i$ *can be on the same level k or on the previous level than newnode*)
*RECURSION(*node$_i$*)* $\rightarrow$ *SA'*, *CL'*
Set $SA = SA \cup SA'$ and $CL = CL + CL'$
**Return**

---

Note that the renumbering of all the activities of the repetitive project was just for the sake of simplicity. As an example, the allowable shift interval $v_{r^*s^*} = \min_{\substack{(r,s)\in A' \\ r\in SA' \\ s\notin SA'}} \{s'_s - s'_r - l'_{rs}\}$ equals

$v_{k^*l^*} = \min\{\min_{\substack{k=1,\ldots,K \\ (i_1,i_2)\in A \\ i_1\in SA' \\ i_2\notin SA'}} \{s_{i_2k} - l_{i_1i_2kk} - s_{i_1k}\}, \min_{\substack{k=1,\ldots,K-1 \\ \forall i\in N}} \{s_{ik+1} - l_{iikk+1} - s_{ik}\}\}$ in our previous notation used for Eqs. [1]-[7]. In the next section, we embed the recursive search procedure in the horizon-varying approach in order to look for the complete optimal profile between work continuity and project duration.

### 4.4 Horizon-varying approach for the complete time/work continuity profile

The horizon-varying approach boils down to the initialization step (section 4.1) and an iterative call to the recursive search procedure of sections 4.2 and 4.3. The heart of the algorithm lies in the while loop of the pseudocode below. We start our search with a project deadline $\delta_{n+1}$ equal to the critical path length (denoted by *cpl*). In doing so, the algorithm reports a schedule with minimal resource idle time (*idle*) and a corresponding project duration $C_{\max}$ which is smaller than or equal to the critical path length (i.e. the solution of Eqs. [1]-[7] with $\delta_{n+1}$ = *cpl*). The algorithm repeats its search with a project deadline $\delta_{n+1} = \delta_{n+1} + 1$ and, again, reports on results for the resource idle time (*idle*) and $C_{\max} \leq \delta_{n+1}$. The algorithm continues this way until it has found a schedule with a minimal value for the crew idle time (denoted by *min*). At this point, it stops the horizon-varying approach since a further increase in the project deadline will not lead to a decrease in resource idle time.

---

**HORIZON-VARYING APPROACH**

Step 1: label activities
Determine *min*
Set $\delta_{n+1}$ = *cpl* and *idle* = ∞
**While** (*idle* > *min*)
  Step 2: Build the spanning tree
  Step 3: Recursive search with deadline $\delta_{n+1} \rightarrow (C_{\max}, idle)$
  Store $C_{\max}$ and *idle*
  $\delta_{n+1} = \delta_{n+1} + 1$
**Return**

---

Remark that the *min*-value depends on the type of generalized precedence relations of the repetitive project. On the one hand, for a project with only minimal precedence relations with time-lag of zero (*cpm* case) it is always possible to find a project deadline that results in a schedule with an *idle*-value equal to zero. This means that, by allowing enough slack to the activities due to a project deadline increase, activities can be scheduled such that crews can work without idle time. Therefore, the horizon-varying approach starts from the critical path length and continues until it has found a schedule without idle time (i.e. *min* = 0). For projects with generalized precedence relations (i.e. both minimal and maximal precedence relation or *gpr*), on the other hand, there is no such guarantee. Due to the nature of the precedence relations, a schedule with zero resource idle time does often not exist. In order to determine what the *min*-value is, we initially solve the recursive search with a project deadline $\delta_{n+1} = \infty$ to report the minimal idle time value (*min*). Similar to the *cpm* case, we then solve the problem starting from the critical path length until we have found a schedule with that minimal value for the resource idle time.

## 5 THREE ILLUSTRATIVE EXAMPLES

In this section, we illustrate the use of our algorithm on three problem examples. The first example is taken from Harris and Ioannou (1998) to schedule a repetitive project to minimize crew idle time. In this example, we solely rely on the recursive search to solve the problem, without starting the horizon-varying approach. In a second example, we make use of the horizon-varying approach to report the complete trade-off between project duration and work continuity. A last example describes the scheduling of a real-life project that aims at the construction of a tunnel at the Westerschelde in the Netherlands.

### 5.1 A six-unit repetitive project

In figure 1 we illustrate an activity-on-the-node project network for the activities in the first unit, as published in Harris and Ioannou (1998). Each of these 6 activities have a duration, denoted above the node and need to use a certain resource $R_i$, denoted below the node. The solid lines are technological precedence relations between the activities. The default value for each precedence relation is a finish-start relationship with a minimal time-lag of zero (i.e. $FS^{min} = 0$), unless indicated otherwise. In this example, we imply a minimal time-lag of 2 time units between activity 1 and 3 (indicated as a ‘lead time’ in Harris and Ioannou (1998)). Note that our algorithm can also deal with maximal time-lags between activities.

Insert Figure 1 About Here

If figure 2, we display the complete network for a project with six repeating units, each having the six discrete activities of figure 1. The dashed lines link similar activities from unit to unit and are used to represent resource availability constraints. Note that $n' = K * n + 2 = 6 * 6 + 2 = 38$ and $|A'| = K * |A| + n * (K - 1) + 2 = 6 * 7 + 6 * 5 + 2 = 74$. Since we assume that the work to be done in units 3 and 4 for activity 1 is twice the work to be done in unit 1, we have doubled the duration of activities 13 and 19. Moreover, we have added a minimal time-lag of five between unit 3 and unit 4 (i.e. $FS_{14,20}^{\min} = 5$). This planned interruption in resource continuity is to meet some known or predicted circumstance. Harris and Ioannou (1998) mention that the delivery of materials by a subcontractor's truck is sufficient to completing only three units, and consequently, a work break period is needed after unit 3. Remark that this repetitive project does not have an activity 3 in unit 5, which is a characteristic of an atypical project. To that purpose, we set the duration of activity 27 to zero. Of course, we could also have deleted this activity from the network.

The algorithm starts with calculating the minimal value for the resource idle time (i.e. *min*) by solving the recursive search procedure with a project deadline $\delta_{n+1} = \infty$. The reported value *idle* = 5 with a total project duration of 30. Since this project duration equals the critical path length, there is no need to start the while-loop of the horizon-varying approach. Indeed, by increasing the project deadline $\delta_{n+1}$ we will not be able to further decrease the idle time to a value smaller than 5 (because of the precedence relation $FS_{14,20}^{\min} = 5$). The starting times reported by the algorithm are $s'_0 = s_{01} = 0$, $s'_1 = s_{11} = 0$, $s'_2 = s_{21} = 6$, $s'_3 = s_{31} = 4$, $s'_4 = s_{41} = 11$, $s'_5 = s_{51} = 19$, $s'_6 = s_{61} = 24$, $s'_7 = s_{12} = 2$, $s'_8 = s_{22} = 7$, $s'_9 = s_{32} = 8$, $s'_{10} = s_{42} = 14$, $s'_{11} = s_{52} = 20$, $s'_{12} = s_{62} = 25$, $s'_{13} = s_{13} = 4$, $s'_{14} = s_{23} = 8$, $s'_{15} = s_{33} = 12$, $s'_{16} = s_{43} = 17$, $s'_{17} = s_{53} = 21$, $s'_{18} = s_{63} = 26$, $s'_{19} = s_{14} = 8$, $s'_{20} = s_{24} = 14$, $s'_{21} = s_{34} = 16$, $s'_{22} = s_{44} = 20$, $s'_{23} = s_{54} = 22$, $s'_{24} = s_{64} = 27$, $s'_{25} = s_{15} = 12$, $s'_{26} = s_{25} = 15$, $s'_{27} = s_{35} = 20$, $s'_{28} = s_{45} = 23$, $s'_{29} = s_{55} = 23$, $s'_{30} = s_{65} = 28$, $s'_{31} = s_{16} = 14$, $s'_{32} = s_{26} = 16$, $s'_{33} = s_{36} = 20$, $s'_{34} = s_{46} = 26$, $s'_{35} = s_{56} = 24$, $s'_{36} = s_{66} = 29$ and $s'_{37} = s_{76} = 30$.

---

Insert Figure 2 About Here

---

In figure 3, we show the RSM diagram based on these starting times, which is similar to the diagram given by Harris and Ioannou (1998). The vertical axis shows the work to be done in the different units while the horizontal axis denotes the time line. They refer to the slope of each line as the unit production rate, i.e. the number of repetitive units that can be accomplished by a resource during a unit of time. Consequently, it can be calculated as the inverse of the duration of that activity at that unit. Total project duration equals 30 days and the resource idle time amounts to 5 days (i.e. the work break between activity 14 and 20, i.e. $FS_{14,20}^{\min} = 5$ in our network of figure 2). The minimal time-lags between activities 1 and 3 at all units (i.e. $FS_{1,3}^{\min} = FS_{7,9}^{\min} = FS_{13,15}^{\min} = FS_{19,21}^{\min} = FS_{25,27}^{\min} = FS_{31,33}^{\min} = 2$) do not affect the resource idle time. The line in bold is the so-called controlling sequence and determines the length of the project duration.

---

Insert Figure 3 About Here

---

**5.2 The horizon-varying approach**

In this section, we illustrate the use of the horizon-varying approach on the project example of figure 4. In this figure, we display the unit network with 10 non-dummy activities. In table 1, we report the activity durations from unit 1 to unit 5. Remark that we assume that crew productivity increases along the units, which denotes a learning effect of crews.

---

Insert Figure 4 & Table 1 About Here

---

In figure 5, we display the complete trade-off profile between the total project duration and work continuity by means of the black bars. This is the result of our horizon-varying approach starting from the critical path length *cpl* = 43 to a project duration of 55 with minimal work discontinuity (*min*). Note that the minimal work discontinuity *min* equals zero since only zero time-lags are involved (*cpm*-case). This trade-off profile can be used as a decision tool to determine an optimal level of resource idle time in the schedule. By assigning costs to both resource idle time and project duration, we can determine the optimal point in the complete profile with an associated project duration and idle time level. In the sequel, we use $c_r$ to denote the cost per unit resource idle time and $c_d$ to denote the cost per time unit that has to be paid during each day of the project duration. Consequently, the total cost of a schedule with total project duration $C_{\max}$ and corresponding resource idle time (*idle*) equals $c_t$

$= c_r * idle + c_d * C_{\max}$. In figure 5, we have reported the total cost $c_t$ by four lines depending on the values for $c_r$ and $c_d$.

---

Insert Figure 5 About Here

---

The optimal project duration and the corresponding level of idle time depend on both the values for $c_r$ and $c_d$. Each unit increase in the project duration involves an extra cost $c_d$ while the total cost will be decreased by $c_r$ times the idle time reduction due to the project duration increase. Consequently, a project duration increase is only beneficial as long as $\frac{c_d}{c_r}$ is smaller than the (negative) slope of the crew idle time curve as displayed in figure 5. As an example, figure 5 shows three values for the slope, i.e. 4 between 43 and 45, 3 between 45 and 49, 2 from 49 onwards. Consequently, 4 different solutions can be optimal, depending on the cost values $c_d$ and $c_r$:

- $\frac{c_d}{c_r} > 4$: It is never beneficial to increase the project duration, and the optimal solution equals the critical path length 43. In figure 5, we have used $c_d$ = 63 and $c_r$ = 14 and the total cost curve (Cost 1) has its lowest point at project duration 43
- $3 < \frac{c_d}{c_r} \leq 4$: It is beneficial to increase the project duration up to 45. This is displayed by the curve labelled "Cost 2" with $c_d$ = 63 and $c_r$ = 18.
- $2 < \frac{c_d}{c_r} \leq 3$: It is beneficial to increase the project duration up to 49. This is displayed by the curve labelled "Cost 3" with $c_d$ = 62.5 and $c_r$ = 25.
- $\frac{c_d}{c_r} \leq 2$: A maximal increase in the project duration leads to the lowest cost. This is displayed by the curve labelled "Cost 4" with $c_d$ = 57 and $c_r$ = 38 with a minimal cost for a project duration of 55.

As a summary, the optimal project duration always coincides with a breakpoint in the trade-off curve between idle time and project duration. In section 6.1, we will show that this optimal point can be found immediately, without enumerating this complete trade-off curve.

**5.3 The Westerschelde Tunnel**

The Westerschelde Tunnel project is a huge project with a groundbreaking boring technique at the Netherlands. This tunnel provides a fixed link between Zeeuwsch-Vlaanderen and Zuid-Beveland, both situated in the Netherlands. It is a bored tunnel with a length of 6.6 kilometres. There are two tunnel tubes and in each tube, there are two road lanes. A detailed description of this project can be found in Vanhoucke and Van Osselaer (2004) or on http://www.westerscheldetunnel.nl.

The project under study involves the construction of the transverse links which connect the two tunnel tubes every 250 metres, resulting in 26 links along the tunnel. Therefore, this project can be represented as a unit network, which will be repeated for 26 times. These connections serve as an emergency exit and account for 10% of the construction budget. Construction is done by means of a freezing technique in order to guarantee watertight transverse links between the tubes. The Westerschelde Tunnel is the first time for the freezing technique to be used on such a large scale.

During the scheduling phase, we analyzed the idle time of two types of resources by comparing two schedules: the earliest start schedule (*ESS*, no minimization of resource idle time) and the work continuity schedule (*WCS*, minimization of resource idle time). The two types of resources are the crews that pass along the 26 units and large freezing machines that are used for the freezing activities at each unit.

The *ESS* clearly results in a lot of resource idle time, both for the freezing machine (within each unit) and for the crews (along the units). The idle time of crews, on the one hand, results from time-lags between the finishing of work at one unit and the start at the next unit. The idle time of the freezing machine, on the other hand, appears within units, due to the earliest starting time of all activities. As an example, the idle time between units 4 and 5 of crew 2 and the idle time of the freezing machine at unit 5 has been indicated at figure 6. The total crew idle time in the *ESS* schedule amount to 165 days while the total freezing idle time over all units is 343 days. The total scheduled project duration of the transverse link subproject equals 380 days.

---

Insert Figure 6 About Here

---

Although the idle time for both the crews and the freezing machines is extremely important, it is completely ignored by the *ESS*. The costs of the crew are generally as follows: An ordinary employee has a cost of – on the average - € 40/manhour while a specialist receives – on the average - € 60/manhour. Consequently, the average cost of one man-hour amounts to € 50. Each crew consists of 3 people that work for 8 hours/day, resulting in a total cost of 50 * 3 * 8 = € 1,200 per day. The efficient use of large *freezing machine* during the construction of the tunnel is also crucial since the company has to pay € 3,000 ($c_r$) for every day that this machine is in operation. Therefore, a hammock activity between two activities was added at every unit in the original *ESS* in order to indicate that all the work in between these activities was performed at a temperature below zero. Consequently, this hammock activity covers a chain of activities that have to be executed at a low temperature. The duration of this hammock activity is variable and equals the total length between the start of the end activity and the start activity of this chain. The *ESS*, however, does not minimize the duration of the hammock activity in any way.

Taking these cost figures into account, we derive the following outline of costs for the *ESS*. The crew idle time takes 165 days, resulting in 165 * € 1,200 = € 198,000 while the cost of the idle time of all the freezing machines equals 343 days * € 3,000 = € 1,029,000. The *WCS* minimizes the resource idle time and results in the following outline of costs. The crew idle time cost amounts to 107 days * € 1,200 = € 128,400 and the idle time of all the freezing machines has a cost of 5 days * € 3,000 = € 15,000. The difference in idle time cost between the two schedules amounts to € 1,083,600. For a detailed description of the project, we refer the reader to Vanhoucke and Van Osselaer (2004).

## 6 COMPUTATIONAL RESULTS

In the previous section, we focused on the trade-off between resource idle time minimization and project duration. In this section, the focus is on the efficiency of the algorithmic approach of section 4. In order to test the efficiency of our horizon-varying approach for scheduling repetitive projects, we have coded it in Visual C++ version 6.0 under Windows NT and run it on a Toshiba personal computer with a Pentium IV, 2 GHz processor using two different testsets. The first testset is the well-known PSPLIB testset (Kolisch and Sprecher 1997), used to report computational results of our procedure and to show the ability to solve large real-sized project scheduling problems. The second set is composed of instances generated by *RanGen* (Demeulemeester et al. 2003) and is used to study the impact of different parameters on the performance of the algorithm.

The generated network instances of both sets contain solely the unit networks (i.e. networks for one unit level) and need to be extended in order to incorporate repetition of activities. To that purpose, we extend the problem instances with repeating activities and corresponding durations by means of two extra parameters: the number of repetitive units $K$ and the rate factor $r_f$. The *number of repetitive units* $K$ is used to denote the number of units in the complete project and equals the number of times the unit network is copied along the stages, resulting in the ladder-like appearance. We make use of a *rate factor* $r_f$ to generate the durations of the repeating activities at the various levels. The rate factor will be used as follows: $d_{ik} = r_f * d_{ik-1}$. Consequently, a rate factor $r_f = 1$ means that all activities have the same durations along the units. A $r_f < 1$ will be used to incorporate learning effects of crews along the units while $r_f > 1$ denotes an increase of the activity duration along the units. This can occur when work becomes more complex when the unit level increases.

The *PSPLIB* dataset contains 480 instances for the 30, 60 and 90 activity networks and 600 instances for the 120 activity networks. Each network has been extended to 5 units and the rate factor has been chosen randomly. Consequently, the total number of problem instances equals 2,040. The results of this first test set are reported in section 5.1. In order to test the impact of different parameters on the effectiveness and efficiency of the procedure, we have constructed the second dataset as follows: Each activity-on-the-node instance contains 30 activities and has been generated with the following settings. The order-strength *OS* (Mastor, 1970) is set at 0.25, 0.50 or 0.75 and is defined as the number of precedence relations (including the transitive ones) divided by the theoretical maximum of number of precedence relations. The number of repetitive units $K$ is set at 5, 10, 15 or 20 and the rate factor $r_f$ is set at 0.8, 0.9, 1.0, 1.1, 1.2 or random. Using 10 instances for each problem class, we obtain a problem set with 720 test instances. The results of this second dataset are reported in section 5.2.

### 6.1 *PSPLIB* instances

Table 2 displays the results for the horizon-varying approach on the *PSPLIB* testset. The table contains information about the CPU time and the number of runs. The number of runs equals the number of iterative call to the recursive search procedure (i.e. the number of repetitions of the while loop of section 4.4) and equals the number of trade-off points on the complete duration/idle time curve between the minimal and maximal project duration. Consequently, the row with the label “average CPU/Run” displays the average CPU time for solving the problem with a given deadline. The column labelled “overall” displays the information for the complete *PSPLIB* set, while the remaining columns distinguish between the 4 testsets with 30, 60, 90 and 120 activities.

---

Insert Table 2 About Here

---

The results illustrate the positive effect of the number of activities on the problem complexity. Nevertheless, projects with up to 120 activities and 5 units (i.e. a network with 602 activities) can be scheduled in less than 5 seconds. The results also indicate the positive effect of the number of activities on the number of runs. Indeed, more activities result in a larger time window between the critical path length and the maximal project duration and, consequently, more runs are needed. The CPU-time per run, however, remains very low, even for projects with a large number of activities.

In section 4.1 we used a labelling step to simulate attraction between activities at the first and last unit. In doing so, we minimized idle time between these units. The value of the labels, which function as a weight of attraction, were chosen to be -1 at the first unit and +1 at the last unit. The values of all other labels, including the labels for the single start dummy and the single end dummy were chosen to be zero. If, however, there is cost information available about the project duration (cd) and idle time of resources (cr), the values cannot be chosen equally. Instead, the trade-off between resource idle time and project duration depends on these cost values and therefore, the label values need to be chosen accordingly.

To that purpose, we replace the unit weights of each activity by the resource idle time cost cr. (i.e. - cr at the first unit and cr at the last unit). Moreover, we assign a negative label to the single start dummy activity with weight -cd and a positive label to the single end dummy with weight cd. In doing so, the project duration is considered as it were a resource with idle time cost equal to cd. Consequently, we implicitly embed the trade-off between idle time and project duration by assigning the appropriate weights to the labels. Depending on the value of $\frac{c_d}{c_r}$, the optimal project duration that results in a minimal total cost will be found without enumerating the complete horizon. In section 5.2, we showed that this optimal point always coincides with a breakpoint in the trade-off curve between idle time and project duration. The column "average CPU/Run" of table 2 illustrates that this can be done in a very efficient way.

### 6.2 *RanGen* instances

In table 3 we report on the computational results for the horizon-varying procedure on the *RanGen* instances for the different settings of the number of units *K*, the rate factor and the order strength *OS*. Again, we display information about the CPU-time (in seconds), the number of runs in the horizon-varying approach, the CPU-time per run and the average number of iterations. The latter equals the average number of times the recursion has been recalled within the recursive search method (step 3). Consequently, it displays the number of times an allowable displacement interval has been calculated in order to shift a set of activities further in time.

The row labeled *'overall'* gives the average results over all 720 instances and illustrates that our procedure is very efficient. Our procedure needs, on the average, 0.049 seconds to schedule a repetitive project with a given deadline. The number of iterations amounts to – on the average – 168,609, with peaks to more than 770,000 for project with 20 units. This illustrates that the recursive search method, which is the body of the algorithm, is very efficient. The remaining rows show more detailed results.

The row labeled *'order strength'* shows that a larger value for the *OS* results in a more complex problem to solve. These results are in line with previous similar research in literature and show that the more dense the network, the more recursion steps are needed and hence, the more difficult the problem (Vanhoucke et al. (2001), De Reyck and Herroelen (1998)). Note that the *OS*-values are only valid on the unit network level. Due to the incorporation of repetitive units, the *OS* of the complete CPM network will have another, unknown value.

The row labeled *'number of units'* clearly shows a positive correlation between the number of units and the problem complexity. Clearly, this stems from the increasing number of activities due to an increasing number of units. Indeed, a repetitive project with 30 unit activities and 20 units contains 30 * 25 + 2 = 602 activities that are subject to the horizon-varying approach. Notice that the total CPU-time denotes 30.920 seconds and repeats the recursive search – on the average – 416.167 times. Consequently, even for problem types with 602 activities in total, the CPU-time only accounts for 0.074 seconds per run.

The row labeled *'rate factor'* shows the influence of the durations of activities along the units on the problem complexity. The results show a positive correlation between the rate factor and the computational effort to solve the problem. This illustrates that learning effects, which is translated in decreasing activity durations, have a positive effect on the problem complexity. The main reason is that the larger the rate factor, the larger the number of combinations in the complete trade-off profile and, consequently, the larger the number of recursive calls in the horizon-varying approach. However, we detect a similar relation for the average CPU-time per run, although average CPU-times are very low.

---

Insert Table 3 About Here

---

In order to test our algorithm for projects in which generalized precedence relations are represented, we have extended the test instances with minimal and maximal time-lags and reported on computational results. The efficiency of our horizon-varying procedure shows, however, similar results as found in table 3. Therefore, we do not report the separate results.

## 7 CONCLUSIONS AND FUTURE RESEARCH

In this paper, we provided a literature summary of project scheduling problem with repeating activities and stressed the importance of so-called work continuity constraints. We have presented an algorithm in order to detect the complete trade-off profile between work continuity and the project deadline of repetitive projects. To that purpose, we have embedded a recursive search procedure into a horizon-varying approach that solves the problem iteratively between two extreme deadlines. The shortest deadline corresponds with a large value of resource idle time while the longest deadline corresponds with a schedule with minimal resource idle time.

The consequences of work continuity constraints in projects have been illustrated by means of three project examples. We have shown that the incorporation of work continuity constraints may involve a trade-off between resource idle time minimization and project duration. Therefore, a careful analysis needs to be made, based on cost figures, to determine the optimal level of resource idle time and total project duration.

The efficiency of the algorithm has been tested on a PC and the results obtained are encouraging. Even repetitive project with up to 30 unit activities and 20 repetitive units can be solved within 0.07 seconds per deadline run. Due to the large number of possibilities in the complete trade-off profile between the project duration and resource idle time, the CPU-times can increase to – on the average – 30 seconds for these projects. Clearly, the number of activities, the order strength and the number of units have a negative effect on the efficiency of the procedure. Still, the procedure can solve large-sized problems in an efficient way. The results reveal a positive correlation between the rate factor and problem complexity.

In our future research, we will extend this problem type with additional features in order to further tighten the gap between the project scheduling literature and real-life project management. More precisely, we would like to investigate time/cost trade-offs in repetitive projects with work continuity constraints. Moreover, we would like to investigate the presence of extra renewable resource constraints that are not the subject of work continuity constraints but that make the scheduling very complex because of limited availability. Finally, we would like to continue with the complete trade-off profile between total project duration and work continuity (see figure 5) and perform some sort of sensitivity analysis on the cost values.

## REFERENCES


Al Sarraj, Z.M., 1990, “Formal development of line-of-balance”, Journal of Construction Engineering and Management, 116 (4), 689-704

Amor, JP, 2002, “Scheduling programs with repetitive projects using composite learning curve approximations”, Project Management Journal, 33 (2), 16-29

Amor, JP and Teplitz, CJ, 1993, Improving CPM's accuracy using learning curves, Project Management Journal, 24 (4), 15-19.

Amor, JP and Teplitz, CJ, 1998, An efficient approximation procedure for project composite learning curves, Project Management Journal, 29 (3), 28-42.

Ashley, D.B., 1980, “Simulation of repetitive unit construction”, Journal of the Construction Division, 106 (CO2), 185-194.

Badiru, A.B., 1995, Incorporating learning curve effects into critical resource diagramming", Project Management Journal, 2 (2), 38-45.

Barrie, D.S. and Paulson, B.C. Jr., 1978, Professional construction management, McGraw-Hill Inc., New York, pp. 232-233.

Bartusch, M., Möhring, R.H. and Radermacher, F.J., 1988, “Scheduling project networks with resource constraints and time windows”, Annals of Operations Research, 16, 201-240.

Birrell, G.E., 1980, “Construction planning – beyond the critical path”, Journal of the Construction Division, 106 (CO3), 389-407.

Callahan, M.T., Quackenbush, D.G. and Rowings, J.E., 1992, “Construction Project Scheduling”, McGraw-Hill, New York, NY.

Carr, R.I. and Meyer, W.L., 1974, “Planning construction of repetitive building units”, Journal of the Construction Division, 100 (3), 403-412.

Chrzanowski, E.N. and Johnston, D.W., 1986, “Application of linear scheduling”, Journal of Construction Engineering and Management, 112 (4), 476-491

De Boer, R., 1998, “Resource-constrained multi-project management – A hierarchical decision support system”, PhD dissertation, Institute for Business Engineering and Technology Application, The Netherlands.

Demeulemeester, E., Vanhoucke, M. and Herroelen, W., 2003, “A random network generator for activity-on-the-node networks”, Journal of Scheduling, 6, 13-34.

De Reyck, B. and Herroelen, W., 1998, “An optimal procedure for the resource-constrained project scheduling problem with discounted cash flows and generalized precedence relations”, Computers and Operations Research, 25, 1-17.

Dressler, J., 1980, “Construction management in West Germany”, Journal of the Construction Division, 106 (4), 447-487.

El-Rayes, K., 2001, “Object-oriented model for repetitive construction scheduling”, Journal of Construction Engineering and Management, 127 (3), 199-205

El-Rayes, K. and Moselhi, O., 1998, “Resource-driven scheduling of repetitive activities”, Construction Management and Economics, 16, 433-446.

Gong, D., 1997, “Optimization of float use in risk analysis-based network scheduling”, International Journal of Project Management, 15 (3), 187-192.

Gorman, J.E., 1972, “How to get visual impact on planning diagrams”, Roads and streets, 115 (8), 74-75

Goto, E., Joko, T., Fujisawa, K., Katoh, N. and Furusaka, S., 2000, “Maximizing Net Present Value for Generalized Resource Constrained Project Scheduling Problem”, Nomura Research Institute, Japan.

Grinold, R.C., 1972, “The payment scheduling problem”, Naval Research Logistics Quarterly, 19, 123-136.

Harris, F.C. and Evans, J.B., 1977, “Road construction – simulation game for site managers”, Journal of the Construction Division, 103 (3), 405-414.

Harris, R.B. and Ioannou, P.G., 1998, “Scheduling projects with repeating activities”, Journal of Construction Engineering and Management, 124 (4), 269-278

Harmelink, D.J., 2001, “Linear Scheduling Model: Float Characteristics”, Journal of Construction Engineering and Management, 127 (4), 255-260

Hegazy, T. and Wassef, N., 2001, “Cost optimization in projects with repetitive nonserial activities”, Journal of Construction Engineering and Management, 127 (3), 183-191

Johnston, D.W., 1981, “Linear scheduling method for highway construction”, Journal of the Construction Division, 107 (2), 241-261.

Kang, L.S., Park, I.C. and Lee, B.H., 2001, “Optimal Schedule Planning for Multiple, Repetitive Construction Process”, Journal of Construction Engineering and Management, 127 (5), 382-390

Mastor, A.A., 1970, “An experimental and comparative evaluation of production line balancing techniques”, Management Science, 16, 728-746.

Mattila, K.G. and Abraham, D.M., 1998, “Resource leveling of linear schedules using integer linear programming”, Journal of Construction Engineering and Management, 124 (3), 232-244.

Moselhi, O. and El-Rayes, K., 1993, “Scheduling of repetitive projects with cost optimization”, Journal of Construction Engineering and Management, 119 (4), 681-697.

O’Brein, J.J., 1975, “VPM scheduling for high-rise buildings”, Journal of the Construction Division, 101 (4), 895-905.

O’Brein, J.J., Kreitzberg, F.C. and Mikes, W.F., 1985, “Network scheduling variations for repetitive work”, Journal of Construction Engineering and Management, 111 (2), 105-116.

Peer, S., 1974, “Network analysis and construction planning”, Journal of the Construction Division, 100 (3), 203-210.

Reda, R.M., 1990, “RPM: repetitive project modeling”, Journal of Construction Engineering and Management, 116 (2), 316-330.

Rowings, J.E. and Harmelink, D.J., 1993, “A multi-project scheduling procedure for transportation projects”, Final Report Part 1, Iowa Department of Transportation, Ames, Iowa.

Rowings, J.E. and Harmelink, D.J., 1994, “A multi-project scheduling procedure for transportation projects”, Final Report Part 2, Iowa Department of Transportation, Ames, Iowa.

Russell, A.D. and Casselton, W.F., 1988, “Extensions to linear scheduling optimization”, Journal of Construction Engineering and Management, 114 (1), 36-52

Russell, A.D. and Wong, W.C.M., 1993, “New generation of planning structures”, Journal of the Construction Engineering, 119 (2), 196-214.

Shoderbek, P.P. and Digman, L.A., 1967, “Third generation PERT/LOB”, Harvard Business Review, 45 (5), 100-110.

Shtub, A., 1991, Scheduling of programs with repetitive projects, Project Management Journal, 22 (4), 49-53.

Shtub, A., LeBlanc, L.J. and Cai, Z., 1996, “Scheduling programs with repetitive projects: a comparison of a simulated annealing, a genetic and a pair-wise swap algorithm”, European Journal of Operational Research, 88, 124-138.

Selinger, S., 1980, “Construction planning for linear projects”, Journal of the Construction Division, 106 (CO2), 195-205

Stradal, O. and Cacha, J., 1982, “Time space scheduling method”, Journal of the Construction Division, 108 (3), 445-457.

Suhail, S.A. and Neale, R.H., 1994, “CPL/LOB: new methodology to integrate CPM and line of balance”, Journal of Construction Engineering and Management, 120 (3), 667-684.

Thabet, W.Y. and Beliveau, Y.J., 1994, “HVLS: horizontal and vertical logic scheduling for multistorey projects”, Journal of Construction Engineering and Management, 120 (4), 875-892.

Vanhoucke, M. and Demeulemeester, E., 2003, “The Application of Project Scheduling Techniques in a Real-Life Environment”, Project Management Journal, 34 (1), 30-42.

Vanhoucke, M. and Van Osselaer, K., 2004, “Work continuity in a real-life schedule: the Westerscheldetunnel”, working paper Ghent University.

Vanhoucke, M., Demeulemeester, E. and Herroelen, W., 2001, “On maximizing the net present value of a project under renewable resource constraints”, Management Science, Vol. 47, 1113-1121.

Vanhoucke, M., Demeulemeester, E. and Herroelen, W., 2000, “A validation of procedures for maximizing the net present value of a project”, Working paper 0030, Departement Toegepaste Economische Wetenschappen, K.U.Leuven, 18 pp.

Whiteman, W.E. and Irwig, H.G., 1988, “Disturbance scheduling technique for managing renovation work”, Journal of Construction Engineering and Management, 114 (2), 191-213.

Yamin, R.A. and Harmelink, D.J., 2001, “Comparison of linear scheduling method (LSM) and critical path method (CPM)”, Journal of Construction Engineering and Management, 127 (5), 374-381.

**FIGURE 1**

**An example project with 6 repeating activities**

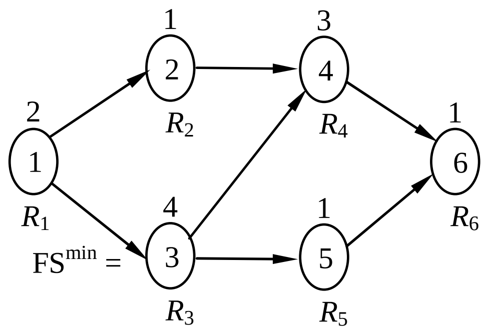

**FIGURE 2**

**A repetitive project network of figure 1 with 6 units**

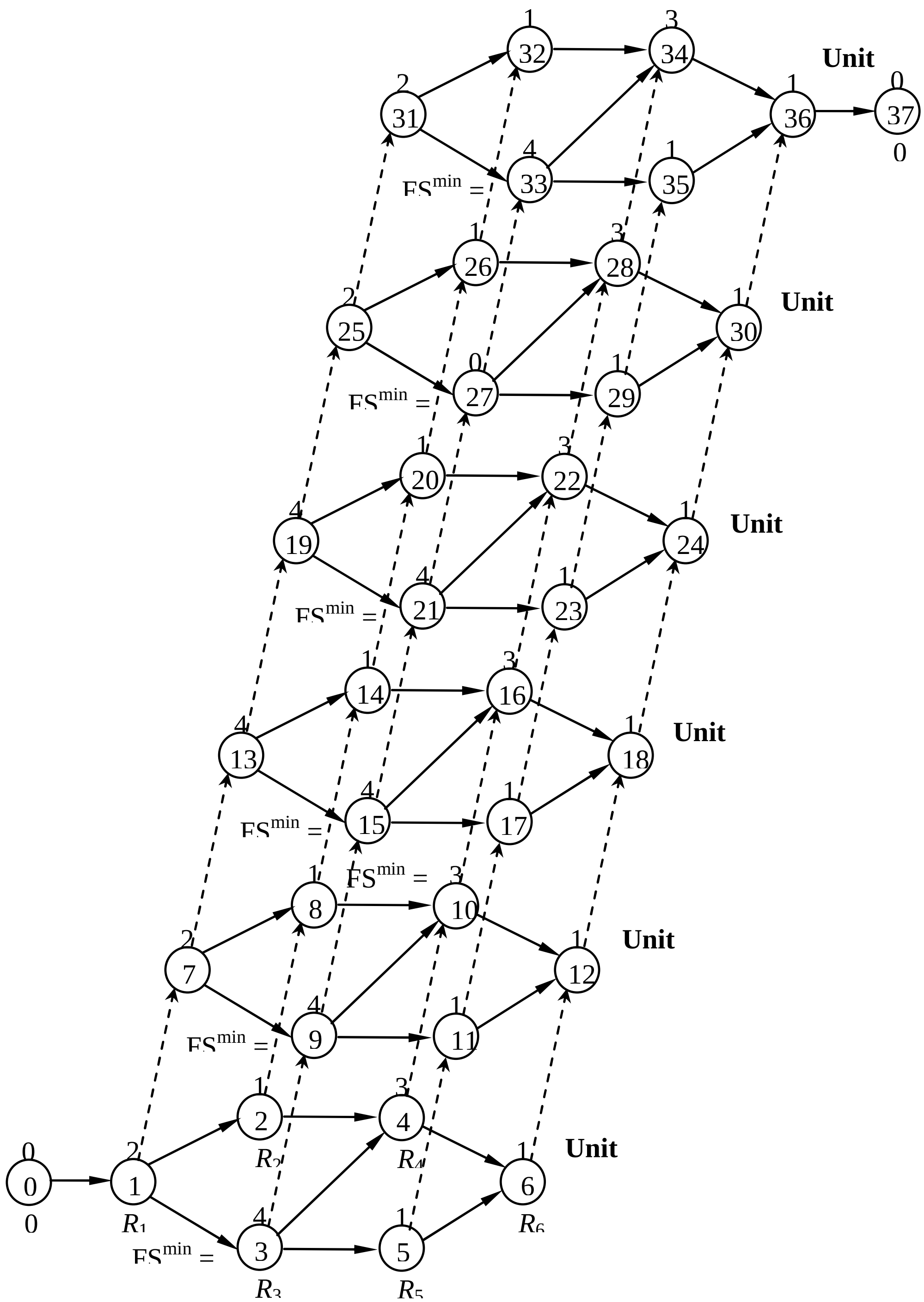

**FIGURE 3**

**RSM diagram for a six units project of figure 2**

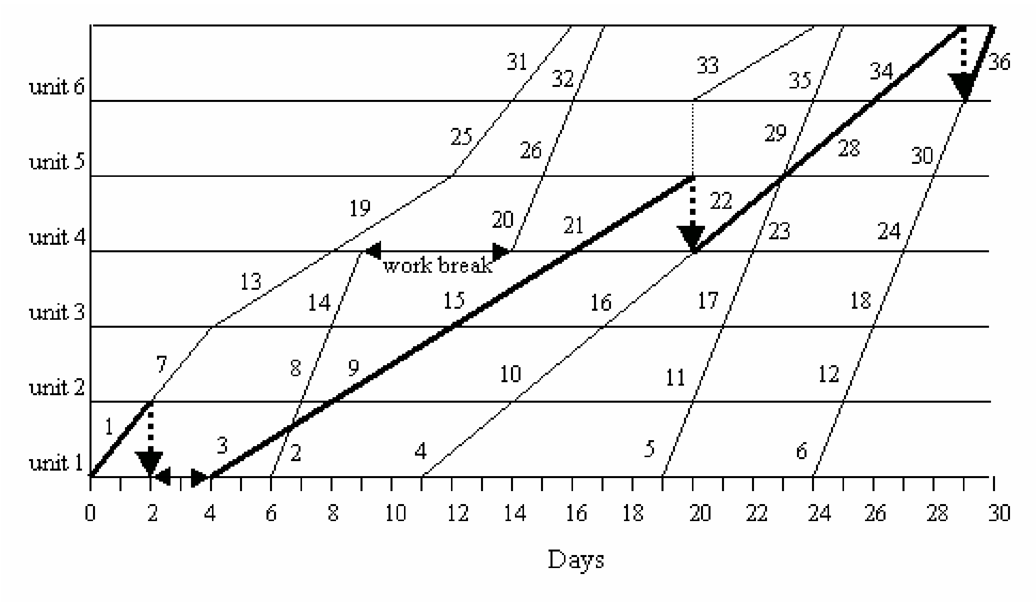

**FIGURE 4**

**An example project with 10 repeating activities**

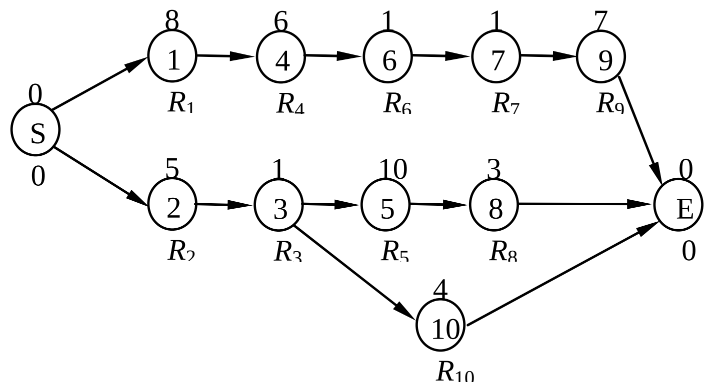

**FIGURE 5**

**Trade-off between work continuity and project deadline and the corresponding total cost**

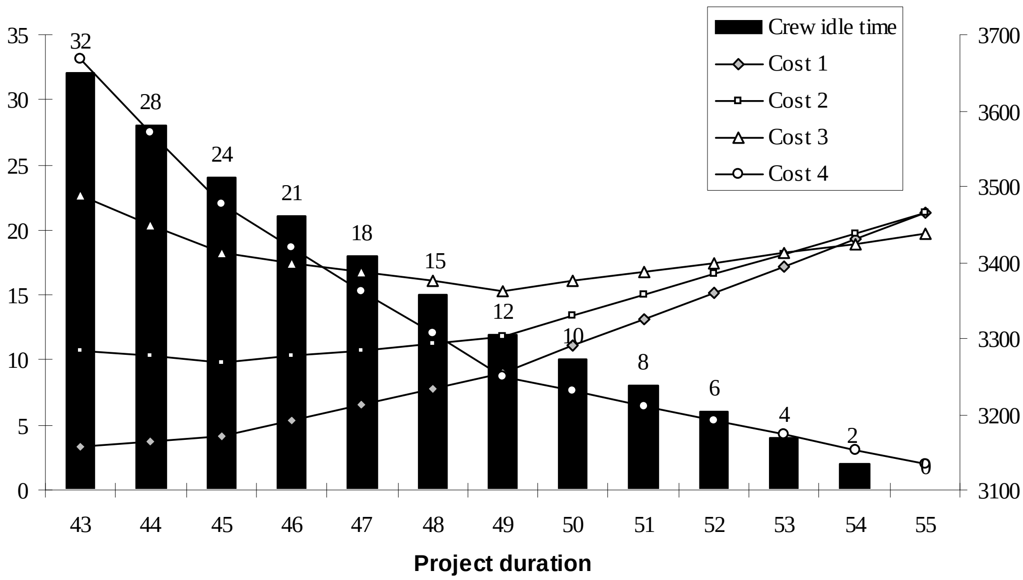

**FIGURE 6**

**Gantt chart obtained by the ESS of the project**

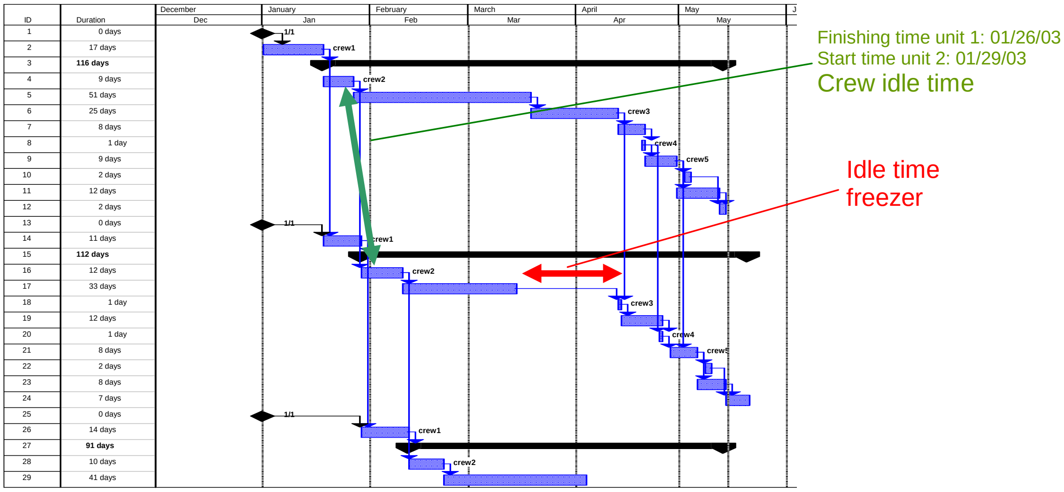

**TABLE 1**

**Activity durations $d_{ik}$ for the project example**

| Unit *k* \ Activity *i* | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 8 | 5 | 1 | 6 | 10 | 1 | 1 | 3 | 7 | 4 |
| **2** | 7 | 4 | 1 | 5 | 8 | 1 | 1 | 2 | 6 | 3 |
| **3** | 6 | 3 | 1 | 4 | 7 | 1 | 1 | 1 | 5 | 2 |
| **4** | 5 | 2 | 1 | 3 | 6 | 1 | 1 | 1 | 4 | 1 |
| **5** | 4 | 1 | 1 | 2 | 5 | 1 | 1 | 1 | 3 | 1 |

# TABLE 2

## Computational results for the horizon-varying approach on the PSPLIB instances

| | **overall** | ***J* 30** | ***J* 60** | ***J* 90** | ***J* 120** |
|---|---|---|---|---|---|
| **CPU time** | | | | | |
| Average | 2.563 | 0.028 | 0.552 | 4.148 | 4.933 |
| St.Dev. | 2.804 | 0.018 | 0.301 | 2.090 | 2.679 |
| Min | 0.000 | 0.000 | 0.110 | 0.461 | 0.801 |
| Max | 22.703 | 0.121 | 2.033 | 12.298 | 22.703 |
| **Run** | | | | | |
| Average | 72.211 | 35.150 | 72.590 | 82.452 | 93.363 |
| St.Dev. | 31.517 | 16.513 | 22.942 | 23.121 | 26.111 |
| Min | 1 | 1 | 21 | 25 | 34 |
| Max | 201 | 105 | 153 | 144 | 201 |
| **Average CPU/Run** | | | | | |
| Average | 0.035 | 0.001 | 0.008 | 0.050 | 0.053 |

## TABLE 3

## Computational results for the horizon-varying approach on the RanGen instances

| | CPU time | | Run | | Average | Iterations |
|---|---|---|---|---|---|---|
| | Average | St.Dev. | Average | St.Dev. | CPU/Run | Average |
| **Overall** | 9.229 | 41.001 | 186.636 | 388.717 | 0.049 | 168,609 |
| ***K*** | | | | | | |
| 5 | 0.034 | 0.037 | 36.494 | 23.617 | 0.001 | 4,050 |
| 10 | 0.644 | 1.018 | 95.644 | 89.173 | 0.007 | 33,472 |
| 15 | 5.320 | 11.071 | 198.239 | 252.886 | 0.027 | 139,688 |
| 20 | 30.920 | 77.340 | 416.167 | 671.095 | 0.074 | 497,228 |
| **Rate Factor** | | | | | | |
| Random | 1.191 | 2.847 | 60.992 | 79.295 | 0.020 | 20,403 |
| 0.8 | 0.143 | 0.187 | 17.075 | 8.938 | 0.008 | 2,779 |
| 0.9 | 0.329 | 0.415 | 32.533 | 14.552 | 0.010 | 6,639 |
| 1.0 | 3.608 | 5.754 | 135.633 | 89.588 | 0.027 | 61,360 |
| 1.1 | 8.802 | 19.390 | 195.758 | 174.902 | 0.045 | 142,755 |
| 1.2 | 41.303 | 91.882 | 677.825 | 743.573 | 0.061 | 777,719 |
| ***OS*** | | | | | | |
| 0.25 | 2.467 | 8.725 | 118.767 | 274.615 | 0.021 | 56,442 |
| 0.50 | 7.292 | 27.418 | 201.875 | 416.491 | 0.036 | 158,564 |
| 0.75 | 17.929 | 64.061 | 239.267 | 445.029 | 0.075 | 290,822 |