A Project Scheduling Approach To Production Planning with Feeding Precedence Relations

Arianna Alfieri, Marcello Urgo, Tullio Tolio

A Project Scheduling Approach To Production Planning with Feeding Precedence Relations

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2009-IJPR-0674.R1
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	20-Dec-2009
Complete List of Authors:	Alfieri, Arianna; Politecnico di Torino, Production Systems and Economics Department Urgo, Marcello; Politecnico di Milano, Manufacturing and Production Systems Division, Department of Mechanical Engineering Tolio, Tullio; National Research Council, Institute of Industrial Technologies and Automation
Keywords:	PRODUCTION PLANNING, PROJECT SCHEDULING
Keywords (user):	FEEDING PRECEDENCE RELATIONS

International Journal of Production Research
Vol. 00, No. 00, 00 Month 200x, 1–29

RESEARCH ARTICLE

A Project Scheduling Approach To Production Planning with Feeding Precedence Relations

Arianna Alfieri[a], Tullio Tolio[b] and Marcello Urgo[c]*

[a]*Production Systems and Economics Department, Politecnico di Torino, Torino, Italy;* [b]*Institute of Industrial Technologies and Automation, National Research Council, Milano, Italy;* [c]*Manufacturing and Production Systems Division, Department of Mechanical Engineering, Politecnico di Milano, Milano, Italy*

()

In Manufacturing-to-Order or Engineering-to-Order systems producing complex and highly customized items, each item has its own characteristics,which are often tailored for a specific customer. Project scheduling approaches are suitable for production planning in such environments. **However, when we consider the production of complex items, the distinct production operations are often aggregated into activities representing whole production phases. In such cases, the planning and scheduling problem works on the aggregate activities, considering that, in most cases, such activities also have to be manually executed.** Moreover, simple finish-to-start precedence relations no longer correctly represent the real production process, but overlapping among activities should be allowed. In this paper, a project scheduling approach is proposed for production planning in Manufacturing-to-Order systems. The Variable Intensity formulation is used to allow the effort committed to the execution of activities to vary over time. Feeding precedences are developed to model generalized precedence relations when the execution mode of activities is not known a priori. Two mathematical formulations of these precedence relations are proposed. The formulations are applied both to random generated instances and to an industrial system producing machining centers and are compared in terms of computational efficiency.

Keywords: production planning, project scheduling, feeding precedence relations.

1. Introduction

The use of project scheduling approaches for production planning have been frequently addressed in the scientific literature (Klein 2000, Márkus *et al.* 2003). In particular, when hierarchical planning approaches are used, project scheduling can serve as a planning tool at certain aggregate levels (Neumann and Schwindt 1998, Neumann *et al.* 2003). Project scheduling approaches for production planning are especially important and useful when particular kinds of production environments are considered. In Manufacturing-to-Order (MTO) or Engineering-to-Order (ETO) systems producing complex and highly customized items, for example, each item has its own characteristics, which are often tailored for a specific customer. The design, production and delivery to the customer of each product is then a *one of a kind* activity that can be easily modeled as the execution of a project. In the production plant, different projects are executed together, competing for the same production resources (machines, workers, etc.).

*Corresponding author. Email: marcello.urgo@mecc.polimi.it

ISSN: 0020-7543 print/ISSN 1366-588X online

DOI: 10.1080/00207540xxxxxxxxx
http://www.informaworld.com

In production planning, project scheduling approaches must usually refer to a medium or long time horizon to provide an adequate view on the future activities. As in hierarchical planning and scheduling approaches, it is appropriate for a production planning approach to work on an aggregate level without considering all the scheduling details. Hence, distinct production operations can be grouped into aggregate activities, and machines and workers are considered together to constitute production resources. The aggregate activities often represent whole production phases, and their duration can be significant. Moreover, planning on an aggregate level can also be an unescapable choice when a detailed production process is not known or is difficult to define at the production planning level.

The use of a project scheduling approach at the aggregate production planning level has been described by Hans (2001), where it is called the *aggregate (tactical) capacity planning* problem or *resource loading* problem. At this aggregate level, a planner can plan production subject to capacity requirements over a time horizon of several weeks to several months, and thus can quote reliable due dates to customers.

Though it takes an aggregate view of the production activities, project scheduling approaches still consider precedence relations among aggregate activities together with resource loads, and hence they are more likely to provide feasibility at the detailed scheduling phase than the usual rough cut capacity planning approaches when applied to these kinds of production systems (i.e., MTO and ETO). The inadequacy of existing hierarchical planning approaches is often due to the fact that material-oriented (MRP/MRP II systems) or capacity-oriented (HPP systems) issues are kept separate (Zijm 2000). The integration of capacity planning and material coordination achieved by project scheduling approaches to production planning can provide a solution to these problems in particular types of production environments.

In Neumann and Schwindt (1998) and Neumann *et al.* (2003), a three-level hierarchical multi-project planning approach is presented to deal with make-to-order systems. At the first level of the proposed hierarchical approach, a portfolio of long-term projects (orders) is to be planned and executed within a medium/long planning horizon. Each project has a work breakdown structure consisting of aggregate activities to be scheduled subject to scarce key resources, and their duration is estimated by the critical-path length of the corresponding subprojects plus a time buffer. The resource requirement for an aggregate activity is computed as the ratio between the total workload of the corresponding subproject and its estimated duration. A fixed execution mode is considered, and the execution constraints are modeled using generalized precedence relations.

In contrast, project scheduling at an aggregate level leads to some difficulties in the definition of precedence relations among the activities (Váncza *et al.* 2004). Considering a single manufacturing operation, it is easy to define a finish-to-start precedence relation representing technological constraints. However, when the operations are grouped into aggregate activities, the finish-to-start relations might no longer correctly represent the real production process. In these cases, a common approach to model the precedence relations more accurately is to use Generalized Precedence Relations (GPRs) (Elmaghraby 1977, Elmaghraby and Kamburowski 1992) that allow a certain amount of overlap among activities. GPRs have been extensively considered in the literature on project scheduling to model complex precedence structures in activity networks (Demeulemeester and Herroelen 1997, Neumann and Schwindt 1997, De Reyck and Herroelen 1999, Klein 2000).

A further issue in MTO/ETO systems is the presence of manually executed activities. FFor these activities, the concepts of *unary* resources and activity durations need to be reassessed. A single worker, in fact, can be assigned to different activities in the same time period, and, at the same time, more workers can be assigned to the same activity. Hence, either the resource used in each time period, or the duration of the activity, are not univocally defined, which makes the traditional scheduling methods no longer suitable. In the literature, the Variable Intensity formulation of the Resource Constrained Project Scheduling problem is proposed to deal with such cases. In this formulation, an *intensity* variable is introduced to define the effort dedicated to process each activity in each time period (Leachman *et al.* 1990, Kis 2005). Resources (considered continuously divisible) are allocated to activities in quantities that vary over time.

A further criticality arises when using the variable intensity formulation together with generalized precedence relations. Due to the introduction of the intensity variable, an infinite number of execution modes are allowed for the activities. Hence, the execution progress of each activity is not fixed but depends on its execution mode, and GPRs can no longer exhaustively describe the overlap among activities (Kis 2006, Tolio and Urgo 2007).

The concept of *Feeding Precedence Relation*, introduced by Kis (2005), has been developed precisely to overcome the difficulties described above. However, Kis only defines a single type of feeding precedence relation, to constrain an activity to start only after a certain percentage of its predecessor activity has been completed. This type of feeding precedence relation has been modeled through binary variables called the *execution mask* and solved through an ad-hoc Branch-and-Cut algorithm. With respect to the existing literature, and especially to the work of Kis (2005), our contribution relies in the definition of another three types of feeding precedence relations, so as to consider all the types of generalized precedence relations where the execution mode of the activities can vary over time. The proposed precedence relations constrain an activity to 1) start before a given percentage of the execution of its successor activity has been completed; 2) start after a given percentage of the execution of its predecessor activity has been completed (already introduced in Kis (2005)); 3) finish before a given percentage of the execution of its successor activity has been completed; 4) finish after a given percentage the execution of its predecessor activity has been completed. We will refer to these relations as *feeding precedences*. Feeding precedences provide a different perspective on the role of precedence relations between pairs of activities, by considering both their start and finish time and the progression of their execution.

Moreover, we developed two mathematical formulations to model such feeding precedence relations in a resource constrained project scheduling problem with variable intensity activities. One formulation (Formulation B) uses the idea execution masks (Kis 2005), while the other (Formulation A) is based on the typical variables used in the time-indexed formulation of scheduling problem, which, to the best of our knowledge, has not been presented in any previous work related to Variable Intensity execution of activities. The two formulations were tested on randomly generated instances and also applied to an industrial system producing machining centers. Given a model of the production process with feeding precedence relations between activities, a project scheduling approach is used to plan the production on a medium or long time horizon.

Section 2 deals with activity aggregation and illustrates how Generalized Precedence Relations can be used to model overlapping activities. The difficulties in using Generalized Precedence Relations for Variable Intensity formulations are discussed in Section 3, in which *feeding precedence* relations are introduced. Section 4 reports the two mathematical programming formulations for our problem, **Section 5 shows their equivalence and Section 6 points out some remarks on the definition of feeding precedence relations. A discussion of the parame-**

ters used to characterize networks appears in Section 7. The application of the developed formulations to randomly generated instances and to the industrial case is presented in Section 8. Section 9 concludes the paper and provides suggested directions for future research.

2. Generalized Precedence Relations and aggregate activities

Working on an aggregate level of detail is very common in production planning approaches (Neumann *et al.* (2003), Váncza *et al.* (2004)). When dealing with medium and long planning horizons, the number of detailed operations to be considered can be prohibitively large. Hence, planning aggregate manufacturing activities instead of single manufacturing operations can provide a consistent reduction in the dimension of the planning problem. Based on a detailed description of the production process and of the characteristics of the different operations, similar resources are aggregated into resource groups and manufacturing/assembling operations into aggregate activities.

Notice that an aggregate level is also used when a detailed production process is not known or is difficult to define at the production planning phase. As an example, consider the production of instrumental goods like machining centers. When these items are produced, a large amount of components, together with ancillary parts, are assembled onto the machine structure. Due to their functionalities or to technological reasons, some components need to be assembled in a well defined sequence. Others, in contrast, can be assembled at any time within a certain time window during the assembly process. Hence, at the production planning level, it is not desirable to define the detailed assembling sequence for all the small parts. It is more appropriate to provide a start and finish time for the whole assembly phase, thus leaving the definition of the scheduling details to the shop floor level.

Aggregation, however, can also have undesirable effects on production planning approaches. When single operations are considered, simple finish-to-start precedence relations are enough to define the constraints affecting the execution of the different operations. On the contrary, when a finish-to-start precedence is defined between two aggregate activities, it forces all of the operations in the predecessor activity to be completely executed before any operation in the successor activity can start. Clearly, this behavior over-constrains the original problem, and a certain overlapping between the two aggregate activities should be allowed to overcome such over-constraining.

Four different GPRs can be defined to link start-times and finish-times of pairs of activities: Start-to-Start (SS), Finish-to-Finish (FF), Start-to-Finish (SF) and Finish-to-Start (FS).

For each of the aforementioned GPRs, further extensions can be introduced by considering a maximal or a minimal time lag between activities. A minimal time lag $SS^{min}_{ij}(l_{min})$ specifies that activity j can start only if the execution of its predecessor i started at least l_{min} time units before (Fig. 1(a)). Instead, a maximal time lag $SS^{max}_{ij}(l_{max})$ specifies that activity j should be started, at the latest, l_{max} time units after activity i has started (Fig. 1(b)). GPRs refers to *indivisible* activities with a fixed executed mode. In such conditions, once an activity starts, its progress execution at a certain time is completely defined.

Figure 1

However, when these assumptions (indivisibility and fixed execution mode) do not hold, the fraction of an activity executed at a certain time depends on the effective execution mode and the overlapping between activities according to a certain percentage of their execution can be no longer described through GPRs.

3. GPRs and Variable Intensity execution

In the *Variable Intensity* formulation, the execution of the activities is described by a set of continuous variables x_{it}. The value of x_{it} represents the fraction of activity i executed in time bucket t. The variable intensity formulation describes the execution of activities when the amount of work performed in a time bucket is not given but depends on the amount of resources devoted. The variable intensity formulation is suitable to model human workers when more than one activity is processed by a group of workers in a single time bucket. Notice that the variable intensity formulation also allows an infinite number of execution modes, and the time needed to completely process an activity is not a priori defined. This time, instead, is strictly related to the value of the intensity execution variables and ranges between a minimal and a maximal duration (expressed in time buckets). The minimal and maximal durations are related to the minimum and maximum amounts of resources that it is possible to allocate to each activity in each time bucket. Since the durations of the activities are not defined a priori, the percentage of an activity executed in a time interval does not completely depend on its duration in terms of time buckets. Moreover, when preemption is allowed, the maximum duration is also not constrained.

Figure 2

In these cases, the generalized precedence relations with minimum and maximum time lags are not appropriate to define precedence relations based on the percentage execution of the activities. Hence, different precedence relations must be defined to to take into consideration the execution of the activity according to the values of the intensity variables. Four distinct cases can be defined:

- *%Completed-to-Start (CtS) precedence:* successor activity j can start its processing only when, in time bucket t, the percentage of predecessor activity i that has been processed is greater than or equal to q_{ij} (Fig. 2(a)).
- *%Completed-to-Finish (CtF) precedence:* successor activity j can be completed only when, in time bucket t, the percentage of predecessor activity i that has been processed is greater than or equal to q_{ij} (Fig. 2(c)).
- *Start-to-%Completed (StC) precedence:* the percentage execution of successor activity j, in time bucket t, can be greater than g_{ij} only if the execution of predecessor activity i has already started (Fig. 2(b)).
- *Finish-to-%Completed (FtC) precedence:* the percentage execution of successor activity j, in time bucket t, can be greater than g_{ij} only if the execution of predecessor activity i has been completed (Fig.2(d)).

These precedence relations are called *feeding precedence relations* and their use to sequence aggregate activities is illustrated through the examples provided in Figure 3 and 4.

Figure 3

A network of activities is given together with precedence relations (Fig. 3(a)). When simple Finish-to-Start precedence relations are considered and an aggregation is performed (Fig. 3(b)), a single precedence relation between two original operations might enforce a precedence relation between two aggregate activities.

In such cases, feeding precedence relations are more suitable to represent the proper relations between aggregate activities. In fact, as illustrated in Figure 4, there exists a set of original operations (belonging to aggregate activity j) that can be executed even if predecessor aggregate activity i has not yet been completed. The amount of resources needed to process the set of original operations, compared to the resources needed to execute the whole aggregate activity j, represents the percentage of j that can be executed even if i has not been completed (i.e., g_{ij}).

An overlapping between the execution of the two aggregate activities i and j is therefore allowed. This overlapping is not defined on a temporal basis but it refers to a certain fraction of the predecessor or successor activity having been completed.

Figure 4

Feeding precedence relations can also be useful in lot streaming problems. In such problems, a lot is processed on several machines and can be partitioned into sublots of possibly different sizes to be transferred between two successive workstations. In such cases, feeding precedences can be used to constrain the maximum or minimum sizes of sublots.

4. Problem formulation

The production planning problem in a Manufacturing-to-Order or Engineering-to-Order environment can be formally represented through a mathematical formulation of feeding precedence relations and variable intensity execution.

In this section, two alternative discrete-time formulations (i.e., the planning horizon is divided into discrete time buckets) are presented for the makespan minimization. The makespan (i.e.,maximum completion time) reflects the objective of finishing the production of all items in the shortest time, given the available resources.

The two formulations use a common set of parameters describing the problem data:

J: set of activities;
T: set of time buckets;
K: set of resources;
$\mathcal{T}$: set of precedence relations;
$\mathcal{T}_1 \in \mathcal{T}$: subset of precedence precedence relations of type %Completed-to-Start;
$\mathcal{T}_2 \in \mathcal{T}$: subset of precedence precedence relations of type Start-to-%Completed;
$\mathcal{T}_3 \in \mathcal{T}$: subset of precedence precedence relations of type %Completed-to-Finish;
$\mathcal{T}_4 \in \mathcal{T}$: subset of precedence precedence relations of type Finish-to-%Completed.
B_j: maximum percentage of work that can be done on activity j in a time bucket;
b_j: minimum percentage of work that can be done on activity j in a time bucket;
i_p: predecessor activity for precedence relation $p \in \mathcal{T}$;
j_p: successor activity for precedence relation $p \in \mathcal{T}$;
q_p: percentage of work needed on activity i_p to allow activity j_p to start or finish in a given time bucket, $p \in (\mathcal{T}_1 \cup \mathcal{T}_3)$;
g_p: percentage of work reachable on activity i_p only if activity j_p has started or finished in a given time bucket, $p \in (\mathcal{T}_2 \cup \mathcal{T}_4)$;
r_j: release time of activity j;
d_j: due date of activity j;
Q_{ik}: amount of resource k needed to completely process activity i;
R_{kt}: total amount of resource k available in time bucket t.

4.1 *Formulation A*

The first mathematical formulation (hereafter called *Formulation A*) explicitly takes into account the time bucket when each activity starts and the time bucket when each activity finishes. Variables representing time buckets in which each activity starts and finishes, and variables representing the amount of work done in each given time bucket, for each activity, are used to model feeding precedences.

The length of the time buckets reflects the maximal level of detail suitable at the planning phase and must be representative of the smallest decision period in the production plan. In fact, once the time horizon is divided into time buckets, no control action can be taken within a time bucket.

No processing time is provided for each activity, since the amount of activity processed in a time bucket depends on the resources dedicated to it, although it ranges between a minimum an maximum amount. In each time bucket we can process a fraction of each activity. This fraction can also assume the value 0, so preemption is allowed.

The following variables are defined:

C_{max}: maximum completion time (makespan);
f_{jt}: binary variables assuming value 1 if activity j finishes in time bucket t, 0 otherwise;
s_{jt}: binary variables assuming value 1 if activity j starts in time bucket t, 0 otherwise;
x_{jt}: continuous positive variables representing the percentage of work done on activity j during time bucket t;
η_{jt}: binary variables assuming value 1 if activity j is processed in time bucket t, 0 otherwise.

The planning problem is formulated as follows:

$$\min \quad C_{max}$$

subject to:

$$C_{max} \geq \sum_{t=r_j}^{d_j} t \cdot f_{jt} \qquad \forall j \tag{1}$$

$$\sum_{t=r_j}^{d_j} s_{jt} = 1 \qquad \forall j \tag{2}$$

$$\sum_{t=r_j}^{d_j} f_{jt} = 1 \qquad \forall j \tag{3}$$

$$\sum_{t=r_j}^{d_j} t \cdot s_{jt} \geq r_j \qquad \forall j \tag{4}$$

$$\sum_{t=r_j}^{d_j} x_{jt} = 1 \qquad \forall j \tag{5}$$

$$x_{jt} \leq B_j \eta_{jt} \qquad \forall j, t \tag{6}$$

$$x_{jt} \geq b_j \eta_{jt} \qquad \forall j, t \tag{7}$$

$$s_{jt} \leq \eta_{jt} \qquad \forall j, t \tag{8}$$

$$f_{jt} \leq \eta_{jt} \qquad \forall j, t \tag{9}$$

$$f_{jt} \leq \sum_{h=r_j}^{t} x_{jh} \qquad \forall j, t \tag{10}$$

$$\sum_{h=r_j}^{t} s_{jh} \geq x_{jt} \qquad \forall j, t \tag{11}$$

$$\sum_{i} Q_{ik} x_{it} \leq R_{kt} \qquad \forall k, t \tag{12}$$

$$s_{jt} \leq \sum_{h=1}^{t-1} x_{ih} - q_p + 1 \qquad \forall p \in \mathcal{T}_1, i = i_p, j = j_p, \forall t \tag{13}$$

$$(1 - \sum_{h=1}^{t} f_{jt}) \geq q_p - \sum_{h=1}^{t-1} x_{ih} \qquad \forall p \in \mathcal{T}_3, i = i_p, j = j_p, \forall t \tag{14}$$

$$\sum_{h=1}^{t} x_{jh} \leq g_p + (1 - g_p) \sum_{h=1}^{t-1} s_{ih} \qquad \forall p \in \mathcal{T}_2, i = i_p, j = j_p, \forall t \tag{15}$$

$$\sum_{h=1}^{t} x_{jh} \leq g_p + (1 - g_p) \sum_{h=1}^{t-1} f_{ih} \qquad \forall p \in \mathcal{T}_4, i = i_p, j = j_p, \forall t \tag{16}$$

Constraint (1) simply defines the makespan as the completion time of the last completed activity. Each activity must have a unique start and a unique finish time bucket, and these two requirements are assured by constraints (2) and (3), respectively. Moreover, no activity can start before its release date (constraint (4)), and, in the time horizon between its release date and due date, each activity must be completely processed (constraint (5)).

If activity j is processed in time bucket t, the amount of work done on it must respect the maximum and minimum thresholds B_j and b_j (constraints (6) and (7)). These constraints are mainly due to technological and/or economical reasons, since, depending on the type of activity and on the required resources, it can be infeasible or non-economical to process an activity more or less than a given amount.

Constraints (8) and (9) forbid an activity to effectively start or finish (s and f equal to 1) in a time bucket where it is not processed ($\eta = 0$).

Constraints (10) and (11), on the other hand, represent the obvious fact that an activity cannot finish if it has not been completed and cannot be processed if is not started (i.e., they link the x variables to f and s variables, assuring consistency in the plan). The total amount of resources used by the activity processing must not exceed, for each resource and in each time bucket, the total amount available in the time bucket. This is assured by constraint (12).

Constraints (13) to (16) represent the feeding precedences between two activities. Constraint (13) represents the *%Completed-to-Start* relations. Given a certain time bucket t, for all the *%Completed-to-Start* precedence relations ($p \in \mathcal{T}_1$), until the predecessor activity $i = i_p$ has been cumulatively processed for at least a percentage q_p, the successor activity $j = j_p$ cannot start ($s_{jt} < 1$). Constraint (15) represents the *Start-to-%Completed* precedences. Given a certain time bucket t, for all the *Start-to-%Completed* precedence relation ($p \in \mathcal{T}_2$), assures that, the cumulative execution of activity $j = j_p$ can be greater than percentage g_p only if activity $i = i_p$ has started. In fact, $\sum_{h=1}^{t} x_{jh}$ can be greater than g_p only if $\sum_{h=1}^{t-1} s_{ih} = 1$. Constraints (14) and (16) work in a similar way (just substituting *start* with *finish* in the previous descriptions) for the other types of relations. In particular, constraint (14) represents *%Completed-to-Finish* relations while constraint (16) represents *Finish-to-%Completed* relations.

4.2 *Formulation B*

The second mathematical formulation (hereafter called *Formulation B*) does not explicitly consider the start time and finish time bucket of the activities. Instead, an *execution mask* z_{jt} is defined for each activity j. It is possible to process activity j in time bucket t only if the execution mask z_{jt} assumes the value 1. The execution masks z_{jt} have value 1 at $t = 0$ and are constrained to have non-increasing shapes. Hence, the mask z_{jt} assumes value 0 only after activity j has been completed, and can be used to model precedence relations, i.e., that the successor activity can be processed only if the execution mask of the predecessor activity has the value 0. Appropriate execution masks $z_{p,t}$ representing feeding precedence are defined for each precedence relation p.

- *%Completed-to-Start* and *%Completed-to-Finish* precedences: the execution mask $z_{p,t}$ associated to these types of precedence relation assumes value 1 while the fraction of the predecessor activity $i = i_p$ is smaller than q_p. When, in time bucket t, the percentage processing of predecessor activity $i = i_p$ becomes greater than or equal to q_p, the execution mask $z_{p,t}$ assumes value 0 for $t \geq t + 1$.
- *Start-to-%Completed* and *Finish-to-%Completed* precedences: the execution mask $z_{p,t}$ associated to these types of precedence relation assumes value 1 while the processed fraction of the successor activity $j = j_p$ is smaller than g_p. When, in time bucket t, this fraction becomes greater than or equal to g_p, the execution mask $z_{p,t}$ assumes value 0 for $t \geq t + 1$.

In Formulation B, execution masks z_{jt} and $z_{p,t}$ are Boolean variables used to model the execution of activities thus playing the roles of the variables s_{jt} and f_{jt} in Formulation A.

$$min \qquad C_{max}$$

subject to:

$$C_{max} \geq t \cdot z_{jt} \qquad \forall j, t \tag{17}$$

$$\sum_{t=r_j}^{d_j} x_{jt} = 1 \qquad \forall j \tag{18}$$

$$x_{jt} \leq B_j \eta_{jt} \qquad \forall j, t \tag{19}$$

$$x_{jt} \geq b_j \eta_{jt} \qquad \forall j, t \tag{20}$$

$$x_{jt} \leq B_j z_{jt} \qquad \forall j, t \tag{21}$$

$$z_{j,t-1} \geq z_{jt} \qquad \forall j, t \tag{22}$$

$$\sum_{i} Q_{jk} x_{jt} \leq R_{kt} \qquad \forall k, t \tag{23}$$

$$z_{p,t-1} \geq z_{p,t} \qquad \forall p \in \mathcal{T}, \forall t \tag{24}$$

$$x_{jt} \leq B_j(1 - z_{p,t}) \qquad \forall p \in \mathcal{T}_1, i = i_p, j = j_p, \forall t \tag{25}$$

$$\sum_{h=1}^{t-1} x_{ih} \geq b_i - z_{p,t} \qquad \forall p \in \mathcal{T}_2, i = i_p, j = j_p, \forall t \tag{26}$$

$$(1 - \sum_{h=1}^{t} x_{jh}) \geq b_j z_{p,t} \qquad \forall p \in \mathcal{T}_3, i = i_p, j = j_p, \forall t \tag{27}$$

$$x_{it} \leq B_i z_{p,t} \qquad \forall p \in \mathcal{T}_4, i = i_p, j = j_p, \forall t \tag{28}$$

$$\sum_{h=1}^{t-1} x_{ih} \geq q_p(1 - z_{p,t}) \qquad \forall p \in (\mathcal{T}_1 \cup \mathcal{T}_3), i = i_p, \forall t \tag{29}$$

$$(1 - \sum_{h=1}^{t} x_{jh}) \geq (1 - g_p) z_{p,t} \qquad \forall p \in (\mathcal{T}_2 \cup \mathcal{T}_4), j = j_p, \forall t \tag{30}$$

Constraint (17) defines the makespan as the last possible time bucket when some activity can still be processed. After it, all masks will be 0 and all activities must be completed. Constraints (18), (19) and (20) are identical to constraints (5), (6) and (7) in Formulation A. In particular, constraint (18) assures that each activity j is completely processed within its release date r_j and due date d_j, while constraints (19) and (20) assure that when x_{jt} is greater than zero ($\eta_{jt} = 1$), i.e., activity j is processed in time bucket t, and the percentage of activity j processed in time bucket t is greater than the minimum amount b_j but does not exceed the maximum amount B_j. These constraints avoid the fragmentation of each activity execution, while respecting economic and feasibility criteria. The resource constrains are assured by (23).

Constraint (21) defines the z variables assuring that an activity can be processed (i.e., $x_{jt} > 0$) only if its execution mask z_{jt} assumes value 1. Constraint (24) assures that the execution masks are non-increasing functions of t while constraints (25)-(30) model feeding precedences. In particular, constraints (25) to (28) define the different type of precedence relations using the execution masks $z_{p,t}$, while the behavior of the execution mask $z_{p,t}$ is controlled by constraints (29)-(30).

Figure 5

The definition of the execution masks for feeding precedences depends on the type of the precedence relation. For *%Completed-to-Start* and *%Completed-to-Finish* precedence relations ($p \in (\mathcal{T}_1 \cup \mathcal{T}_3)$), the execution mask is associated to the predecessor activity $i = i_p$: while the executed fraction of the predecessor activity is less than q_p, the mask must have value 1. When the executed fraction is greater than q_p the mask can assume either value 1 or 0 (Figure 5(a) and 5(c)). This behavior is defined by the constraint (29). The execution mask constrains the execution of the successor activity $j = j_p$. For *%Completed-to-Start* precedence relations, constraint (25) prevents the successor activity $j = j_p$ from starting if the value of the mask is 1 (x_{jt} can not > 0). For *%Completed-to-Finish* precedence relations, constraint (27) assures that, until the mask assumes the value 1, the successor activity can not be completed. Hence, when the mask assumes the value 0, the successors $j = j_p$ should have completed at least a fraction b_j, that is, the minimum fraction processable in a single time bucket.

When *Start-to-%Completed* and *Finish-to-%Completed* precedence relations are considered, on the other hand, the execution mask refers to the successor activity $j = j_p$. In these cases, constraint (30) forces the execution mask $z_{p,t}$ to assume value 0 as soon as the executed fraction of the successor activity $j = j_p$ reaches g_p (Figure 5(b) and 5(d)). For the *Start-to-%Completed* relations, constraint (26) assures that, if the execution mask assumes value 0, then the executed fraction of the predecessor activity $i = i_p$ must be at least b_i. Since b_i is the minimum fraction processable in a single time bucket, then the predecessor activity should at least be started. For *Finish-to-%Completed* relations, constraint (28) imposes that, when the execution mask assumes the value 0, it is no longer possible to process the predecessor activity $i = i_p$ ($x_{it} \leq 0$); hence, this activity should already have been completed.

The idea of using execution masks to model *%Completed-to-Start* feeding precedence relations was proposed in Kis (2006). Constraints (21), (22), (24), (25) and (29) are already included in Kis's formulation. The remaining constraints are new and refer to the other types of feeding precedence relations introduced in this paper.

4.3 *Computational Complexity*

As stated in the previous paragraph, the proposed problem is an extension of the RCPSVP proposed in Kis (2005), where simple finish-to-start precedence relations were considered. In the proposed formulations, the same problem is considered, but, in addition, feeding precedence relations are introduced, thus our problem contains the RCPSVP as a special case, since %Completed-to-Start feeding precedence relations with $q = 1$ are equal to finish-to-start precedence relations. In Kis (2005), Kis demonstrates that the RCPSVP is NP-hard in the strong sense, since it contains the Preemptive Flowshop Scheduling Problem as a special case; hence, as extension of RCPSVP, our problem is likewise NP-hard in the strong sense.

5. Formulations equivalence

Formulations A and B are equivalent in the sense that they reach the same optimal solution in terms of objective function value. However, this does not imply that the they provide the same solution in terms of the values of the variables. If more than one solution is available with the same objective function values, the two formulations may find two different optimal activity execution profile.

However, their equivalence in terms of the objective function value is assured by the equivalent behavior of the precedence constraints. In fact, constraints (1) and (17) simply define the makespan using the respective variables of each formulation; constraints (5), (6), (7), (12) in Formulation A are identical to (18),(19), (20), (23), respectively, in Formulation B, and constraints (2)-(4), (8)-(11) in Formulation A and (21),(22) and (24) in Formulation B only serve to link the variables to ensure the correct behavior of the precedence constraints. Hence, the equivalence is demonstrated by focusing on precedence constraints.

5.1 *%Completed-to-Start*

In Formulation A, if at a certain time $t-1$ the activity i has been processed for less than q_p, the right-hand side of constraint (13) is less than 1. Thus, the left-hand side should be less than 1, s_{jt} cannot assume value 1, activity j is not allowed to begin and, due to constraint (11) it cannot be processed at all. In Formulation B, under the same hypothesis (at a certain time $t-1$ the activity i has been processed for less than q_p), the left-hand side of constraint (29) is less than q_p and the right-hand side can be less than this value only if $z_{p,t} = 1$. If $z_{p,t} = 1$, then according to constraint (25), $x_{j,t} \leq 0$. Thus, activity j cannot be executed in time bucket t. However, since the mask $z_{p,t}$ is non-increasing in t (due to constraint (24)), it must assume the value 1 in all the time buckets before t, so activity j cannot be executed before t and, consequently, it can not begin.

If, on the other hand, at a certain time $t-1$, activity i has been processed for more than q_p, the right-hand side of constraint (13) in Formulation A is greater than or equal to 1. In this case s_{jt} is no longer constrained ($s_{jt} \in [0,1]$) and activity j can begin. In Formulation B, under the same hypothesis, constraint (29) allows $z_{p,t}$ to assume either value 0 or 1. Thus, according to (25), $0 \leq x_{jt} \leq b_j$ and activity j can begin in t.

5.2 *Start-to-%Completed*

If, at a certain time $t-1$, activity i has not begun, the right-hand side of constraint (15) in Formulation A assumes the value g_p and the left-hand side (fraction of activity j executed) cannot be greater than this value. Under the same hypothesis, in Formulation B, the left-hand-side of constraint (26) assumes value 0 (activity i has not begun). Constraint (26) thus forces the value of $z_{p,t}$ to be 1. Therefore, the right-hand side of the constraint (30) assumes the value $1-g_p$. To satisfy constraint (30), the left-hand side must not be less than $1-g_p$ and hence the fraction of activity j processed until time bucket t cannot be greater than g_p.

If, on the other hand, at a certain time $t-1$, activity i has already begun, the right-hand side of constraint (15) in Formulation A assumes the value 1 and the left-hand side is no longer constrained. Hence, the processed fraction of activity j can be greater than g_p. In Formulation B, if activity i has already begun, it must be processed for at leas a fraction b_j. Due to constraint (26), $z_{p,t}$ is allowed to assume the value 0. Considering the constraint (30), if $z_{p,t}$ is allowed to assume the value 0, then the value of the right-hand side can be also 0, and the left-hand side is no longer constrained, so the fraction of activity j processed until time bucket t can be greater than g_p.

5.3 *%Completed-to-Finish*

In Formulation A, constraint (14) assures that, if at a certain time $t-1$, activity i has been processed for less than q_p, then the right-hand side of the constraint is greater than 0. Then, the left-hand side should also be greater than 0; hence f_{jt} cannot assume the value 1, i.e., activity j is not allowed to finish (according to constraint(10)). Under the same hypothesis, in Formulation B, the left-hand side of constraint (29) must be less than q_p and hence $z_{p,t}$ is constrained to assume the value 1. Therefore, the right-hand side of constraint (27) assumes the value b_j. Since the left-hand side of constraint (27) represents the fraction of activity j not yet processed, and this fraction should be greater than b_j, activity j cannot be completed.

If, on the other hand, at a certain time $t-1$ activity i has been processed for more than q_p, the right-hand side of constraint (14), in Formulation A, is less than 0, so the left side is no longer constrained, f_{jt} can assume the value 1, and hence, activity j can be completed. Under the same hypothesis, in Formulation B, the right-hand side of constraint (29) is greater or equal to q_p and $z_{p,t}$ is allowed to assume the value 0. If $z_{p,t}$ can assume value the 0, then the right-hand side of constraint (27) can be 0, i.e., the fraction of activity j not yet processed can be 0, and hence, activity j can be completed.

5.4 *Finish-to-%Completed*

If, at a certain time $t-1$, activity i has not been completed, then the right-hand side of constraint (16), in Formulation A, assumes the value g_p, and hence the fraction of activity j executed cannot be greater than this value. Under the same hypothesis, in Formulation B, the value of $z_{p,t}$ in constraint (30) must be 1, implying that activity j cannot be executed for a percentage greater than g_p. In fact, it is possible to process activity i (i.e., $x_{it} > 0$) only if the value of $z_{p,t}$ is 1 in constraint (28). Since the mask $z_{p,t}$ is non-increasing (as stated by constraint (24)) in t, until activity i has not be finished, $z_{p,t}$ must be 1; otherwise it is not possible to process i anymore. Hence, activity i cannot be finished.

If, on the other hand, at a certain time $t-1$, activity i has been completed, then the right-hand side of constraint (16), in Formulation A, assumes the value 1 and the left-hand side is no longer constrained. Hence the fraction of activity j processed can be greater than g_p. In Formulation B, if activity i has already been completed, then $z_{p,t}$ can assume the value 0. If $z_{p,t}$ can assume the value 0, then the right-hand side of constraint (30) is no longer constrained and the fraction of activity j processed until time bucket t can be greater than g_p.

6. On the definition of the feeding precedence relations

The feeding precedence relations described above can sometimes demonstrate somewhat pathological behavior. When *%Completed-to-Finish* precedence relations are used, both mathematical formulations allow a high percentage of the successor activity j to be executed (e.g., 99.99%) and then wait until q_{ij} of the predecessor activity i has been executed to finish j. In the case of *Start-to-%Completed*, activity i can start and be processed for only for a very small percentage (e.g., 0.01%) to allow activity j to be completed for more than g_{ij}. Although possible, such pathological behaviors can be avoided through an appropriate calibration of the parameters b_j, together with a proper structure of the precedence relations among the aggregate activities.

In fact, b_j can be used to model work organization (a single worker cannot work alone) or technological issues (if an activity is executed in a time bucket, then a minimum amount of working hours should be devoted to it) thus making the probability of processing an activity for a extremely small fraction quite unlikely. The mathematical formulations also allow us to define more than one precedence relation between the same pair of activities. This can be used to shape the mutual execution of a pair of activities to assure compatibility with the reality. As an example, it can be stated that successor j can start only when the percentage executed of i is $\geq q$, but, at the same time, the execution of activity i can be more than g only if activity j has already started (Figure 6).

Figure 6

7. Morphological and resource-related issues

A project scheduling problem can be represented by means of an acyclic directed graph $G = V, U$ using an *activity-on-node* representation. Each activity is represented by a node in the set V while each arc in the set G represents a precedence constraint between two activities. It is common practice, in the project scheduling literature, to characterize a problem through morphological and resource-related measures of its graph representation. In (Tavares *et al.* 1999, Vanhoucke *et al.* 2004) several complexity measures are proposed to describe the morphological structure of a network while in (Demeulemeester *et al.* 2003) resource-related measures are presented. Some of the morphological and resource-related measures considered in the above cited papers are briefly described in the following, as they will be used in Section 8 to characterize the complexity of the networks we experimented with.

Among the morphological indices presented in (Tavares *et al.* 1999, Vanhoucke *et al.* 2004), we consider the following:

- I_1: *Size of the Problem.* This index is equal to the number of nodes (i.e., activities) in the network and it is a measure of the size of the network.
- I_2: *Serial or Parallel Indicator.* It measures how close a network is to a serial or parallel directed graph. When all activities are in parallel, $I_2 = 0$, while when all the activities are serially connected, $I_2 = 1$. Real networks contain a number of activities that can be executed in parallel and a number of serial precedences: the closer to 1 is the value of I_2, the larger the number of serial connections with respect to the parallel components of the network.

Among the resource-related measures presented in (Demeulemeester *et al.* 2003), we consider:

- RU: *Resource density.* RU measures, for each activity, the number of resources it uses (not the quantity used). The value of this index varies between 0, if the activity needs no resource, to the maximum number of resources available, if the activity uses all the available resources. RU can only assume integer values.
- RC: *Resource constrainedness.* It computes, for each resource, the ratio between the average quantity (over all activities that use the resource) required for the resource and its total availability. RC is zero if no activity uses the resource, while it approaches 1 if all activities, requiring the resource, demand for a quantity close to the total availability. If RC is bigger than 1, the problem is resource-infeasible, since, on average, more of the resource is required than the available quantity.

In Section 8, the performance of Formulations A and B are tested on random generated instances and on an industrial case. In the following comparison, 1) the random instances were generated by fixing the above described morphological and resource-related indices so that they well represent industrial problems typical of the system we are considering (Manufacturing-to-Order and Engineering-to-Order), but with significant differences among the various classes of instances, and 2) the industrial case instances were classified according to the morphological and the resource aspects of the activity network.

8. Computational experiments

The two mathematical formulations presented in Section 4 were tested using both random generated instances and instances drawn from an industrial application. The two mathematical formulations were solved by using CPLEX 10.0 on a XEON workstation (clock: 3.0 Ghz, RAM: 4.00 Gb). A preliminary computational test was carried out to investigate the possible influence of the CPLEX settings (in particular the generation of different type of cuts) on the solution time. The results showed no particular effect of such settings on the solution time; moreover, the solution time obtained with the standard settings of CPLEX was always among the best ones. Hence, we experimented with the standard CPLEX settings, since the solution time with standard settings can be considered a strong indication of the difficulty of solving the instances.

8.1 *Random generated instances*

The random instances were generated using RanGen2 (Vanhoucke *et al.* 2008), an activity network instance generator for project scheduling problems based on the indicators described in the previous section. RanGen2, however, generates instances for classical resource constrained project scheduling problem, i.e., instances with fixed activity durations and finish-to-start precedence relations. To use variable intensity formulation for activity execution and feeding precedences between activities, the generated instances were modified in the following way:

- A given fraction of finish-to-start precedence relations is randomly chosen and transformed to feeding precedences with 50% overlap (i.e., q_{ij} and g_{ij} are equal to 0.5).
- The duration L_j of activity j is considered as the minimum duration, i.e., $B_j = 1/L_j$. The minimum fraction of activity processable in each time bucket is not constrained ($b_j = 0$).

A set of problem instances were generated using the generation parameters reported in Table 1. The roles of I_1, I_2, RU and RC are as described in the previous section, and Res indicates the number of resources in each instance.

Table 1

For each combination of the values for I_1, I_2, Res, RU and RC in Table 1, 2000 instances were generated. Then, to assure the complete randomness of the test instances, for each class of instances, a set of 100 instances was sampled to be used for the experiments. Given an instance, a certain percentage $\%Prec$ of the existing precedence relations are changed to feeding precedence relations of the same type *PrecType*. Different types of feeding precedences are tested in a separate way, i.e., in each experiment, only one type of feeding precedence is considered (besides the usual finish-to-start). The feeding precedence relation types are coded as 1 (*CtS*), 2 (*CtF*), 3 (*StC*) and 4 (*FtC*). Then the mathematical formulations A and B (*Model*) are used to solve the instances. The factors used in the computational tests are reported in Table 2.

Table 2

8.2 *Results*

In the experimental tests, a maximum solution time of 1000 seconds was set for each experiment. If it is not solved to optimality within 1000 seconds, an experiment is considered a *failure*. Given the dimension of the instances, 1000 seconds can be considered a suitable threshold to identify failures in the solution of the instances. The result summary (Table 3) reports the average solution time (*AvTime*) and the percentage of failures (*PercFail*) in solving the instance to optimality given

the number of activities (I_1) and the formulation (*Model*).
The results show that, as expected, the solution time increases with the number of activities. The behavior of the two formulations is however quite different. Formulation A has a significantly longer solution time, and hence, a larger percentage of failures than formulation B.

Table 3

Given the considerable influence of the number of activities and the mathematical formulation on the performance, both in terms of the solution time and the percentage of failures (Table 3), we investigated the joint influence of all the parameters used to generate instances has been investigated.

A first analysis was carried out to analyze the influence of the generation parameters on the percentage of failures. A preliminary qualitative analysis is reported in Figure 7. This confirms that the main factors influencing the number of failures are the number of activities (I_1), the mathematical formulation used (*Model*) and the the shape of the activity network (I_2). In addition, the value of RC has a slight influence, causing the problem to be more difficult to solve as the RC value increases. The remaining factors (the amount and type of feeding precedences, *%Prec* and *PrecType*), on the other hand, did not show any significant influence.

To complete the analysis, the interaction between pairs of factors is investigated through the Interaction Plot shown in Figure 8. The graph shows a clear interaction between the number of activities (I_1) and the formulation used (*Model*). In particular, Formulation A is strongly influenced by the value of I_1 (for 60 activities, the percentage of failure is consistent) while when Formulation B is used, the influence of (I_1) is significantly less.

Figure 7

Figure 8

A second analysis is carried out to investigate the influence of the generation parameters on the time needed to solve a problem to optimality. Clearly, in this analysis, only the experiments that did "not fail" (i.e., for which it was possible to reach optimality within 1000 seconds) were considered.

The graph of the main effects (Figure 9) confirms that the influencing factors are, also for the solution time, the dimension of the problem (I_1) and the mathematical formulation (*Model*). The resources load (RC) has a slightly greater influence while the type of precedence relations (*PrecType*) shows an interesting pattern: test cases with precedence type 2 seems less difficult to solve than those of types 1, 3 and 4.

The analysis of the interactions between factors (Figure 10) indicates that the dimension of the problem (I_1) magnifies the effects of all the other factors. In fact, when $I_1 = 60$ the influence of RC, $\%Prec$ and $PrecType$ is more evident. However, when the formulation B is used ($Model = 1$) the influence of I_1 is strongly decreased. The influence of feeding precedence type (*PrecType*) shows the same pattern seen in the Main Effect Plot (Figure 9), i.e., the solution time for problems with only feeding precedences of type 2 is shorter than for the other types of precedence relations. This behavior has interaction with I_1, RC and $Model$. More precisely, it becomes most evident when $I_1 = 60$, $RC = 0.5$ and Formulation A is used ($Model = 0$).

Figure 9

Figure 10

This influence can be better observed in Table 4, where the average solution time and the average percentage of failures are reported, for only the experiments employing Formulation A, according to the number of activities in the problem (I_1) and the type of precedence relation used ($PrecType$).

Table 4

In all the experiments, both formulations gave the same results in terms of makespan. However, the computational tests on the randomly generated instances provide a clear picture of the performance of the two proposed formulations in solving instances with different characteristics. In particular, it can be argued that the performance of Formulation A is strongly influenced by the number of activities in the scheduling problem, both for the number of failures and the solution time. Formulation B, instead, was able to solve to optimality the vast majority of the instances in a reasonable time. Moreover, the analysis shows that, when Formulation B is used, the number of activities in the instances has almost no influence on the solution time, thus demonstrating that Formulation B also outperforms Formulation A in terms of robustness.

The results obtained with the two formulations have also been compared in terms of solution structure, i.e., how many pieces an activity is preempted on average and what percentage is processed in each time bucket. The two formulations showed, on average, the same number of preemptions, but while Formulation A tends to preempt less as the processing approaches the due date, Formulation B tends to preempt more evenly. Moreover, Formulation A splits activities in such a way that the percentage processed in each time bucket (whether the activity is preempted or not) is always the same. For example, if an activity uses 6 time buckets, 1/6 of it will be processed in each. On the other hand, the solutions found by Formulation B also processed different percentages in the time buckets used (e.g., 6 time buckets used, processed percentages equal to 0.1, 0.2, 0.2, 0.2, 0.1, 0.2). These characteristic makes Formulation A more suitable for ad-hoc algorithms based on column generation techniques (dynamic programming can be more easily used to find solutions, in terms of the values of x).

8.3 *Industrial application*

To demonstrate the viability of the developed method, it was applied to production planning in a real industrial environment that produces machining centers. A machining center is a CNC (Computer Numerical Controlled) machine integrated with an automatic tool changer, and it often has equipment for pallet or part handling.

Even if standard machining centers are available, customers often ask for modifications tailored to their specific needs. This is a common practice for European (and in particular Italian) machining center manufacturers. After the customized parts have been completely designed, a large set of components is assigned to external suppliers, while only high precision manufacturing activities for critical components are executed internally. At the end, all the parts and ancillary components are assembled together, tested and then partially disassembled and delivered to the customer.

To model the production process, the bill of materials of a set of machining center types has been analyzed. Components were grouped into functional units and, for each group, a manufacturing or assembling operation has been considered. The work content was estimated for each operation, and proper precedence constraints were defined among them to represent the technological constraints affecting the production process. Hence, considering the resources involved, the operations have been grouped to obtain a reduced set of aggregate production activities: *Structure Preparation*, *Structure Painting*, *Assembling Autonomous Components*, *Assembling*, *Wiring*, *Testing*, *Metrological Testing*, *Disassembling and Delivery*.

Given the aggregate production activities, feeding precedence constraints were used to model the production process correctly. The need of feeding precedence relations is motivated by the fact that finish-to-start precedence relations among aggregate activities impose unnecessary constraints with respect to the real manufacturing process.

In fact, the *Assembling* phase contains a certain number of sub-phases dealing with the separate assembling of single autonomous components such as the electrical cabinet or the spindle head. The assembling of such components need not be completely processed before the machine assembling activity starts. Rather, it is desirable that these activities be completed at the latest before the subassembly is installed onto the machining center (Figure 11). In such a case, a Finish-to-%Completed precedence constraint can be used to allow the assembling of different autonomous components to be completed at the latest after a certain percentage of the machining center assembling has been executed. This percentage represents the percentage of the assembling activity that can be carried out even if the considered subassembly is not yet ready to be installed in the machining center. In a similar way, the wiring and testing phases should not wait for the completion of the whole assembling phase to start. The wiring phase can start as soon as components thah need to be wired together are installed in the machining center. furthermore, in this case, a suitable approach to allow the wiring activity to start at the earliest after a certain percentage of the assembling activity has been completed. Hence a %Completed-to-Start precedence constraint can be used to allow the cabling phase to start as soon as the components that need to be cabled together are installed onto the machining center.

Figure 11

Formulations A and B were tested to plan the production of a subset of the production orders, drawn from the industrial case, with the objective of minimizing the total duration of the production activities (i.e., the makespan). The machining centers to produce, corresponding to the selected production orders, have the same number of activities and the same structure of precedence relations. They differ in terms of processing times and percentages used in feeding precedence relations (q_{ij} and g_{ij}).

For each machining center, three feeding precedence relations are used to correctly represent the production process: a Finish-to-%Completed between the *Assembling Autonomous Components* and the *Assembling* phases, and two %Completed-to-Start between the *Assembling* and *Wiring* and the *Assembling* and *Testing* phases. Given the detailed precedence constraints structure between production operations, the percentages to be used in the feeding precedence relations have been calculated, for each machining center type, according to the procedure described in Figures 3 and 4. The value of g_{ij} for the Finish-to-%Completed precedence ranges between 0.21 and 0.25 while the values of q_{ij} for the %Completed-to-Start precedences range between 0.65 and 0.78 according to the different types of machining centers.

All the described production phases are mainly processed by human workers. Their behavior can be correctly modeled using the variable intensity formulation allowing a variable resource utilization. The workers are grouped into seven different types according to their particular skills ($Res = 7$) and each production phase requires only one type of workers ($RU = 1$).

The value of RC in the randomly generated instances considered a constant availability of resources. In the industrial case, however, this hypothesis does not hold. In fact, the availability of resources depends on the request of the other orders not considered in the experiments. The value for RC has therefore been calculated through an average availability over the time horizon considered. Moreover, as described in Section 7, the value of RC depends on the amount of resources requested by all the activities. In the considered industrial case, the amount of resources requested depends on the type of machining centers to be produced. Hence, different values for RC are obtained for the different industrial instances considered.

In Table 5, the values of the parameters characterizing the industrial case are reported. Notice that, in contrast to the randomly generated instances, the different types of feeding precedence relations are mixed together in the same instance. Notice also that, since the parameters were directly derived from the industrial data, there is no discretion about them. For this reason, we did not investigate the sensitivity of the results to the parameter values, as we did for the random generated instances.

Table 5

In Table 6 the results of the experiments on the industrial case are reported. It can be observed that, for each instance, the solution times are smaller when Formulation B is used. Moreover, in two instances (*IC1* and *IC2*), Formulation A failed, i.e., it was not able to solve the problem to optimality within the time limit of 1000 seconds. Considering only the successful cases, Formulation A was solved to optimality in an average time of 6.78 seconds while the average solution time for Formulation B was 2.55 seconds. These results are in line with those obtained using the randomly generated instances.

Table 6

9. Conclusion and further research

This paper has addressed the problem of production planning in Manufacturing-to-Order or Engineering-to-Order systems producing complex and highly customized items. A project scheduling approach has been proposed using variable intensity formulations to allow the effort committed to the execution of activities to vary over time. Feeding precedence relations were developed to model generalized precedence relations when the execution mode of activities is not known a priori and their possible utilizations have been described through the application to a real industrial case.

Two alternative mathematical formulations were proposed and tested on randomly generated instances and on real instances drawn from an industrial case in order to show the application of the approach. The results of the computational tests, both on randomly generated and industrial instances, highlighted the different performance levels and the main characteristics of the two mathematical formulations. In particular, the tests allowed us to evaluate their different levels of sensitivity to the parameters defining the characteristics of the production planning problem, such as the number of activities and the load of the resources.

The computational experiments were carried out using a commercial software (Ilog CPLEX) to solve the mathematical formulations. The use of a commercial software might reduce the effort required to introduce the proposed approach to a firm. However, the numerical results clearly showed that this is a viable approach only with small instances. In fact, the use of commercial software might be impractical for very large problems (i.e., large numbers of activities). In particular, the use of Formulation A seems to be impractical for problems with more than 30 activities, while Formulation B performs better, and is able to solve problems with up to 60 activities. Moreover, beyond their different performances in terms of CPU time, the two formulations also have different characteristics in terms of solution structure (i.e., values of the relevant variables) which can be exploited to develop ad-hoc solution algorithms.

The application to the real industrial case was judged positively by the management of the company, since the obtained results were considered very helpful in devising the base production plan. because the models do not account for uncertainty, it frequently happened that, in practice, the base plan sometimes had to be partially modified. However, the robustness of the plan with respect to temporal allocation of resources, allowed better management of changes, when they happened.

The improvement of the solution performance of the proposed formulations, through the exploitation of the different characteristics in tailored solution algorithms, and the extension of the approach to consider uncertainty, will be subjects of future research.

Acknowledgments

The authors would like to thank MCM S.p.A for their support in the definition of the real case. This research was partially funded by the "XVII Executive Programme of Scientific and Technological Cooperation between the Republic of Hungary and the Republic of Italy".

References

De Reyck, B. and Herroelen, W., 1999. The multi-mode resource-constrained project scheduling problem with generalized precedence relations. *European Journal of Operational Research*, 119 (2), 538–556.

Demeulemeester, E., Vanhoucke, M. and Herroelen, W., 2003. RanGen: A Random Network Generator for Activity-on-the-Node Networks. *Journal of Scheduling*, 6 (1), 17–38.

Demeulemeester, E.L. and Herroelen, W.S., 1997. A Branch-And-Bound Procedure for the Generalized Resource-Constrained Project Scheduling Problem. *Operations Research*, 45 (2), 201–212.

Elmaghraby, Salah, E. and Kamburowski, J., 1992. The Analysis of Activity Networks under Generalized Precedence Relations (GPRs). *Management Science*, 38 (9), 1245–1263.

Elmaghraby, S.E.E., 1977. *Activity Networks*. John Wiley & Sons.

Hans, E., 2001. Resource Loading by Branch-and-Price Techniques. Thesis (PhD). University of Twente.

Kis, T., 2005. A branch-and-cut algorithm for scheduling of projects with variable-intensity activities. *Mathematical Programming*, 103 (3), 515–539.

Kis, T., 2006. RCPS with Variable Intensity Activities and Feeding Precedence Constraints. *In*: *Perspectives in Modern Project Scheduling.*, 105–129 Springer.

Klein, R., 2000. Project scheduling with time-varying resource constraints. *International Journal of Production Research*, 38 (16), 3937–3952.

Leachman, R.C., Dincerler, A. and Kim, S., 1990. Resource-Constrained Scheduling of Projects with Variable-Intensity Activities. *IIE Transactions*, 22 (1), 31–40.

Márkus, A., Váncza, J., Kis, T. and Kovács, A., 2003. Project scheduling approach for production planning. *Annals of the CIRP*, 52 (1), 359–362.

Neumann, K. and Schwindt, C., 1998. A capacitated hierarchical approach to make-to-order production. *European Journal of Automation*, 32, 397–413.

Neumann, K., Schwindt, C. and Zimmermann, J., 2003. *Project scheduling with time windows and scarce resources - temporal resource-constrained project scheduling with regular and nonregular objective functions.* 2nd ed. Springer Berlin.

Neumann, K. and Schwindt, C., 1997. Activity-on-node networks with minimal and maximal time lags and their application to make-to-order production. *OR Spectrum*, 19 (3), 205–217.

Tavares, L.V., Ferreira, J.A. and Coelho, J.S., 1999. The risk of delay of a project in terms of the morphology of its network. *European Journal of Operational Research*, 119 (2), 510–537.

Tolio, T. and Urgo, M., 2007. Planning of Machining Centres Production: the Role of Precedence Modelling. Montecatini, Italy, 10 – 12 September 2007.

Váncza, J., Kis, T. and Kovács, A., 2004. Aggregation: the Key to Integrating Production Planning and Scheduling. *Annals of the CIRP*, 53 (1), 374–376.

Vanhoucke, M., Coelho, J., Debels, D. and Tavares, L.V., On the morphological structure of a network. , 2004. , Technical report 9, Vlerick Leuven Gent Management School.

Vanhoucke, M., Coelho, J., Debels, D., Maenhout, B. and Tavares, L.V., 2008. An evaluation of the adequacy of project network generators with systematically sampled networks. *European Journal of Operational Research*, 187 (2), 511–524.

Zijm, W.H.M., 2000. Towards intelligent manufacturing planning and control systems. *OR Spektrum*, 22, 313–345.

Table 1. Parameter values for instance generation

I_1	10	30	60
I_2	0.25	0.50	
Res	4		
RU	1		
RC	0.25	0.50	

Table 2. Experimentation factor values

Factor	Type	Levels	Values
I_1	fixed	3	10; 30; 60
I_2	fixed	2	0.25; 0.50
RC	fixed	2	0.25; 0.50
$\%Prec$	fixed	2	0.2; 0.4
$Model$	fixed	2	A(0); B(1)
$PrecType$	fixed	4	1; 2; 3; 4

Table 3. Average aggregate results

I_1	Model	AvTime	% Failures	Model	AvTime	% Failures
10	A	1.7023	0.12	B	0.1779	0.00%
30	A	115.1464	16.88	B	5.8271	1.94%
60	A	255.0025	93.38	B	56.7425	12.19%

Table 4. Influence of precedence type for instances of Formulation A

I1	PrecType	Average Time	% Failures
10	1	2.79	0.00%
10	2	0.86	0.25%
10	3	1.82	0.00%
10	4	1.33	0.25%
30	1	151.19	13.00%
30	2	69.19	11.00%
30	3	106.73	11.50%
30	4	140.14	32.00%
60	1	262.22	94.75%
60	2	187.79	91.50%
60	3	261.62	89.50%
60	4	461.20	97.75%

Table 5. Industrial case parameters

Parameters	Values
I_1	30
I_2	0.22
RC	(0.13,0.19)
$\%Prec$	0.25
$Model$	A(0); B(1)
$PrecType$	mixed

Table 6. Industrial case results

Instance	$I1$	$I2$	RC	$\%Prec$	$Model$	$Time$	$Model$	$Time$
IC1	30	0.222	0.18849	0.25	A	–	B	9.73
IC2	30	0.222	0.18808	0.25	A	–	B	7.69
IC3	30	0.222	0.1654	0.25	A	10.05	B	1.05
IC4	30	0.222	0.15207	0.25	A	7.08	B	1.19
IC5	30	0.222	0.17784	0.25	A	9.97	B	1.00
IC6	30	0.222	0.15207	0.25	A	6.52	B	1.11
IC7	30	0.222	0.16363	0.25	A	5.08	B	0.95
IC8	30	0.222	0.15538	0.25	A	4.39	B	1.25
IC9	30	0.222	0.13906	0.25	A	2.66	B	0.75
IC10	30	0.222	0.15538	0.25	A	8.55	B	0.80

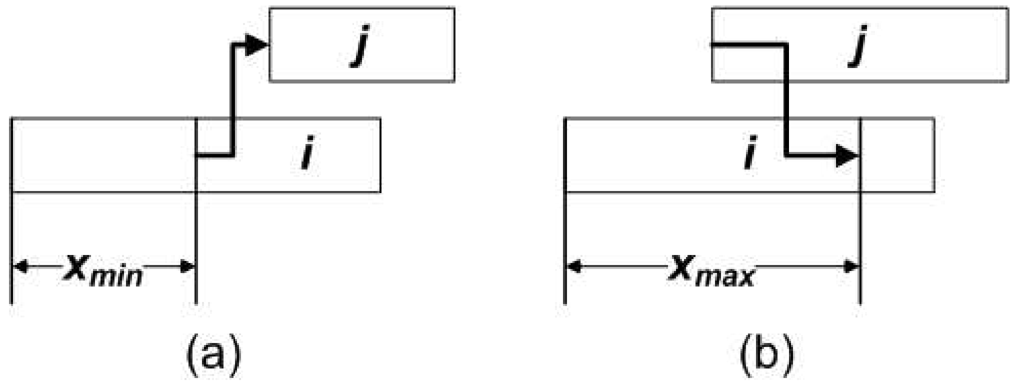

Figure 1. Generalized precedence relations with time lags.
180x68mm (72 x 72 DPI)

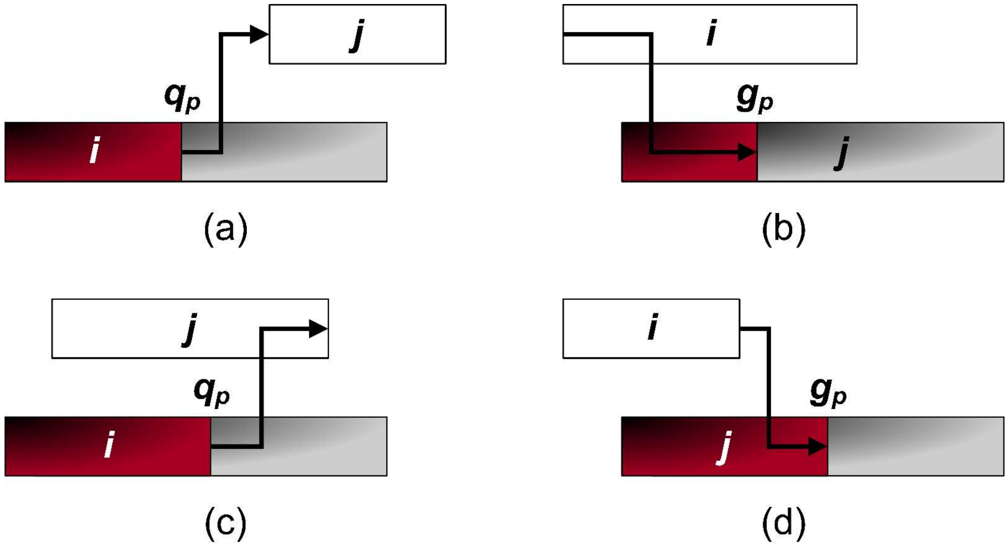

Figure 2. Feeding precedence relations.

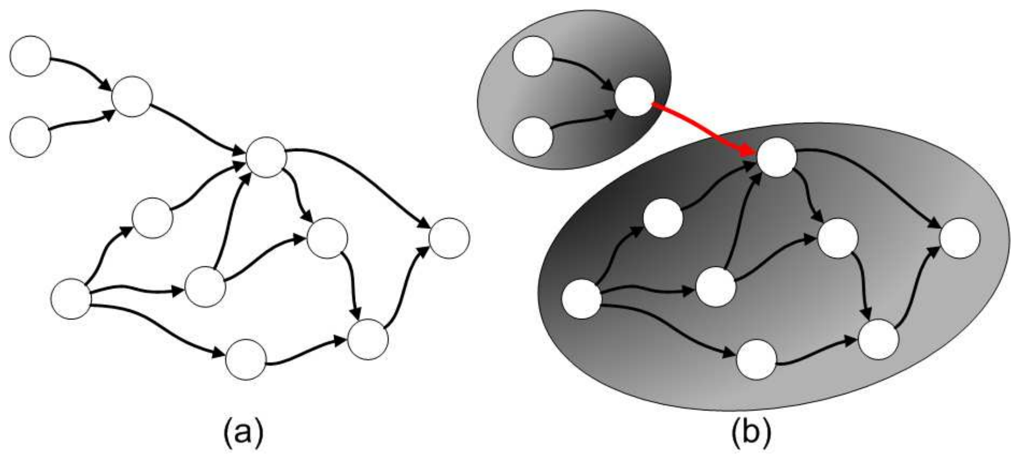

Figure 3. Aggregation of activities.
328x146mm (72 x 72 DPI)

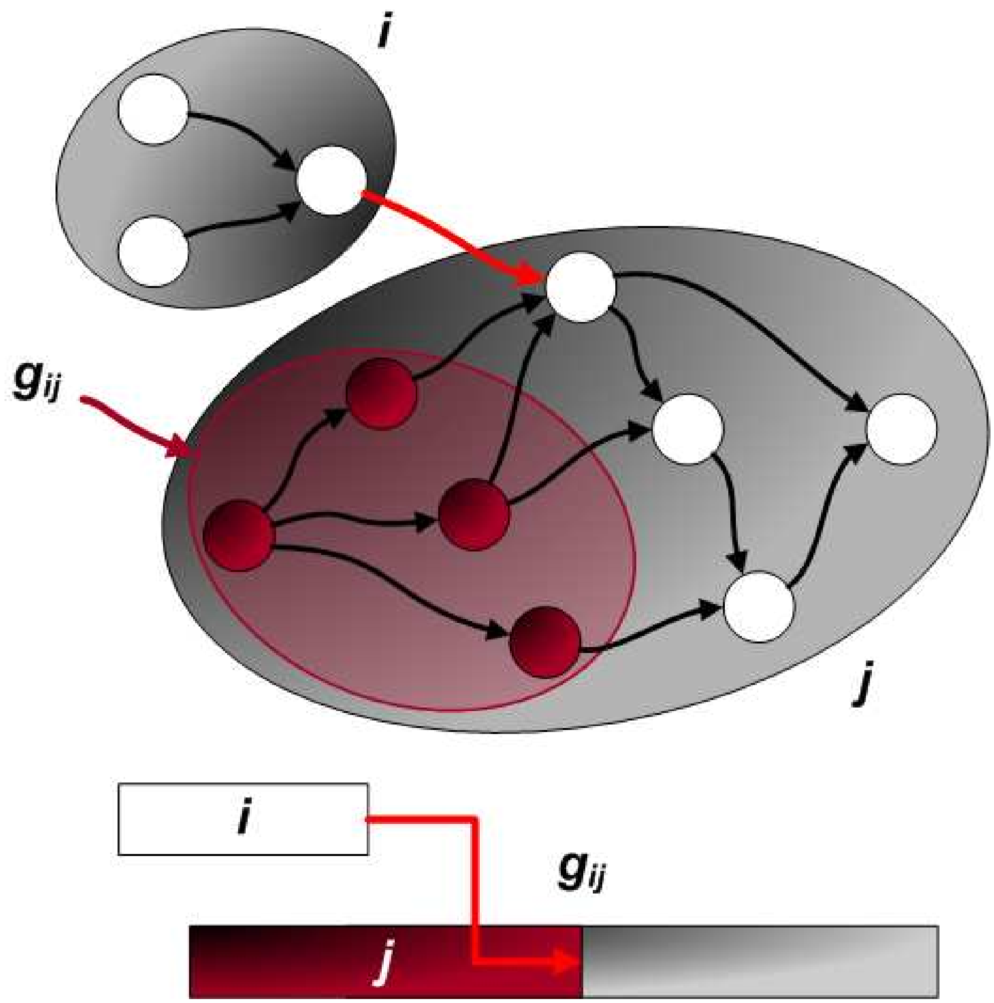

Figure 4. Feeding precedence on aggregate activities.
187x188mm (72 x 72 DPI)

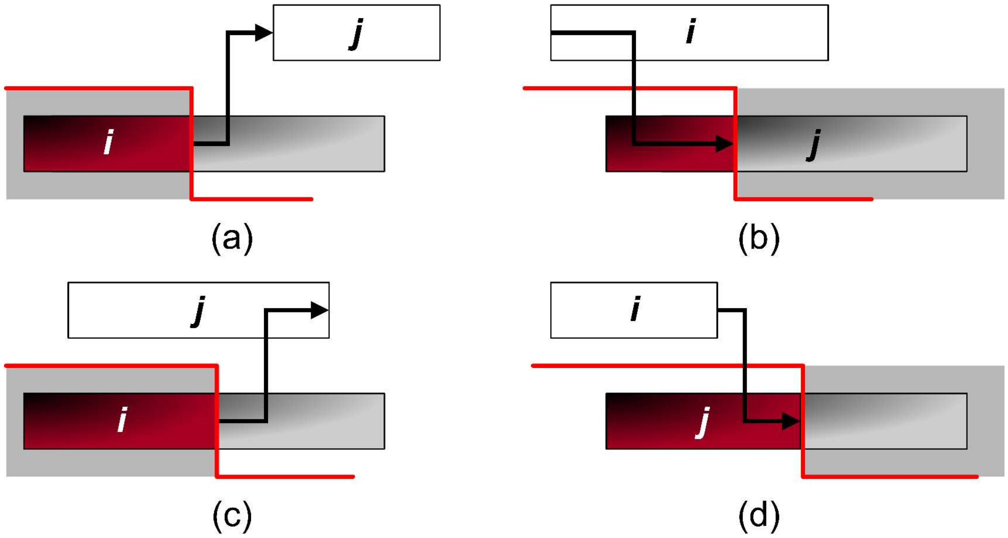

Figure 5. Execution masks for feeding precedence relations.

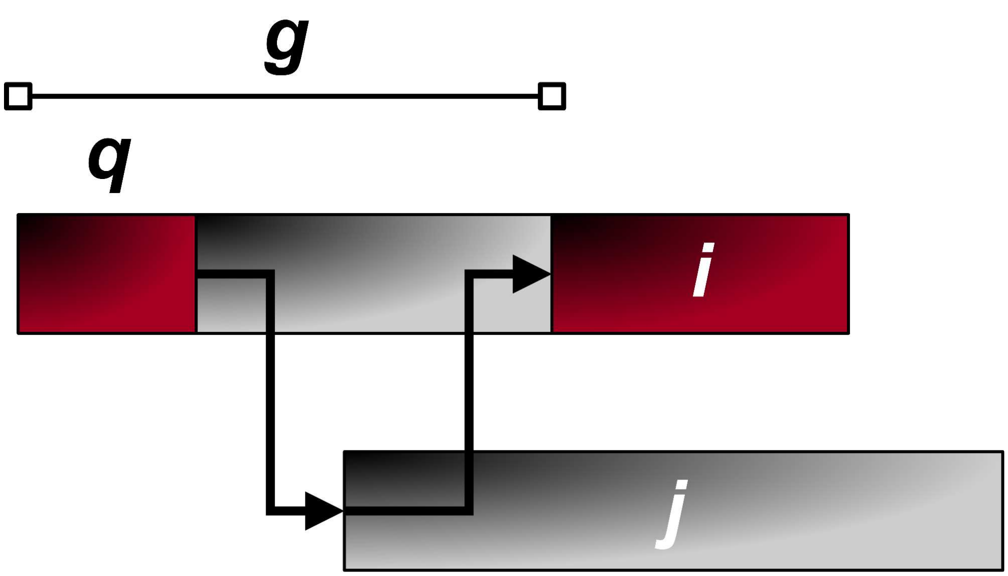

Figure 6. Multiple precedence constraints.

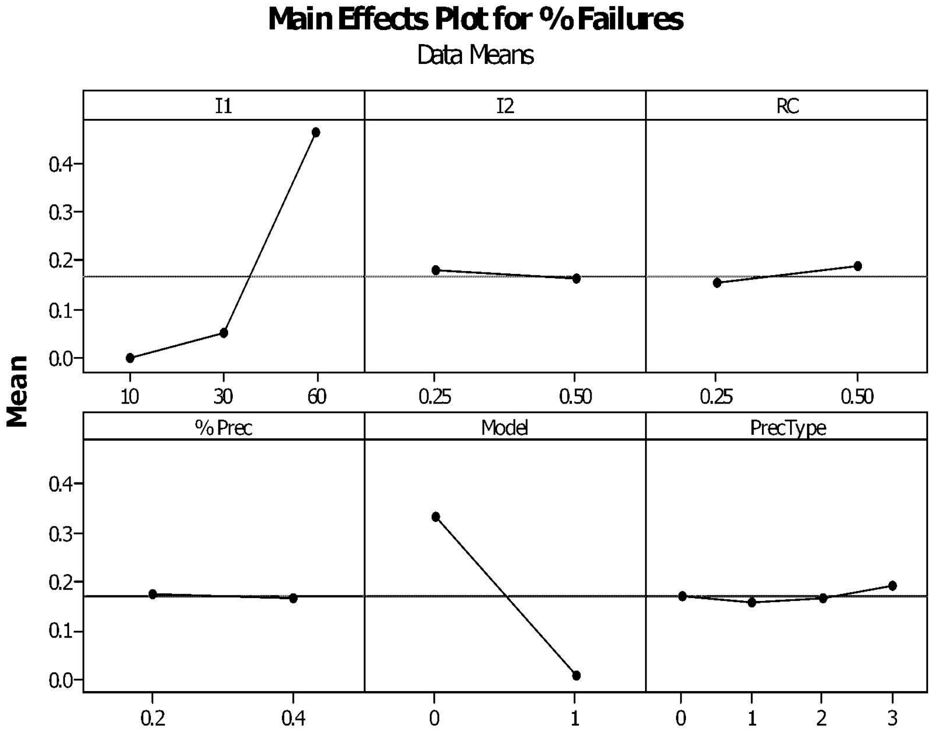

Figure 7. Main Effects Plot for mean failures percentage.

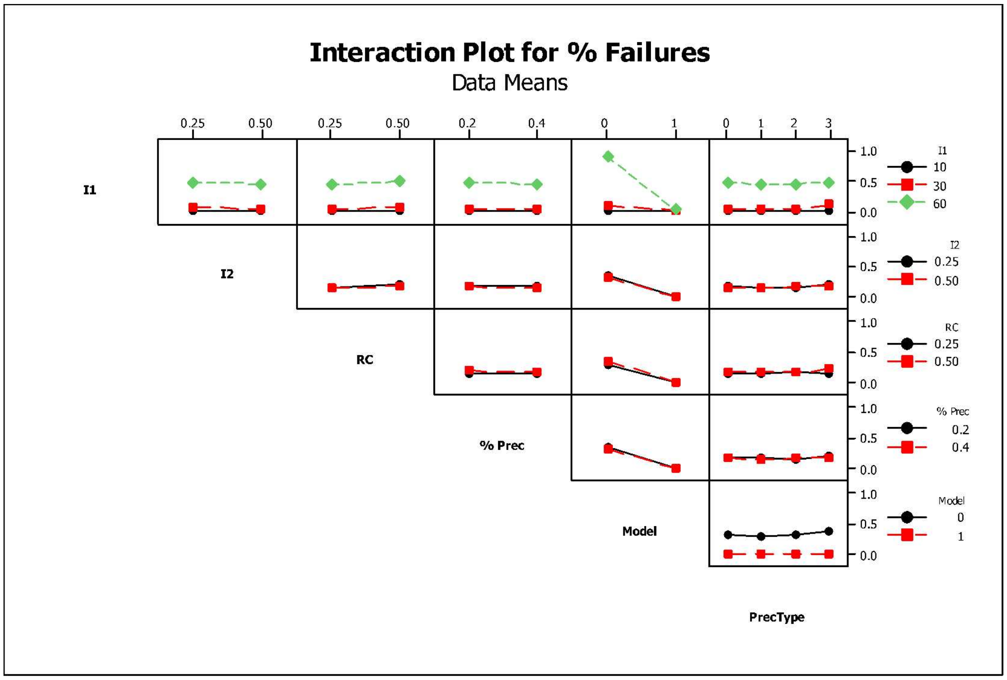

Figure 8. Interaction Plot for mean failures percentage.

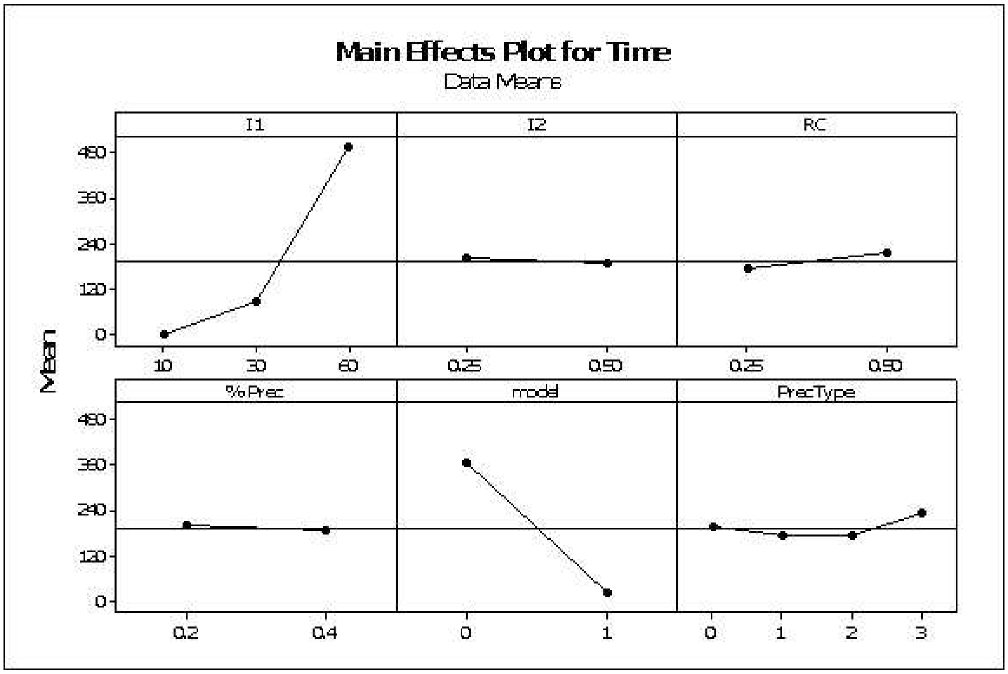

Figure 9. Main Effects Plot for mean solution time.

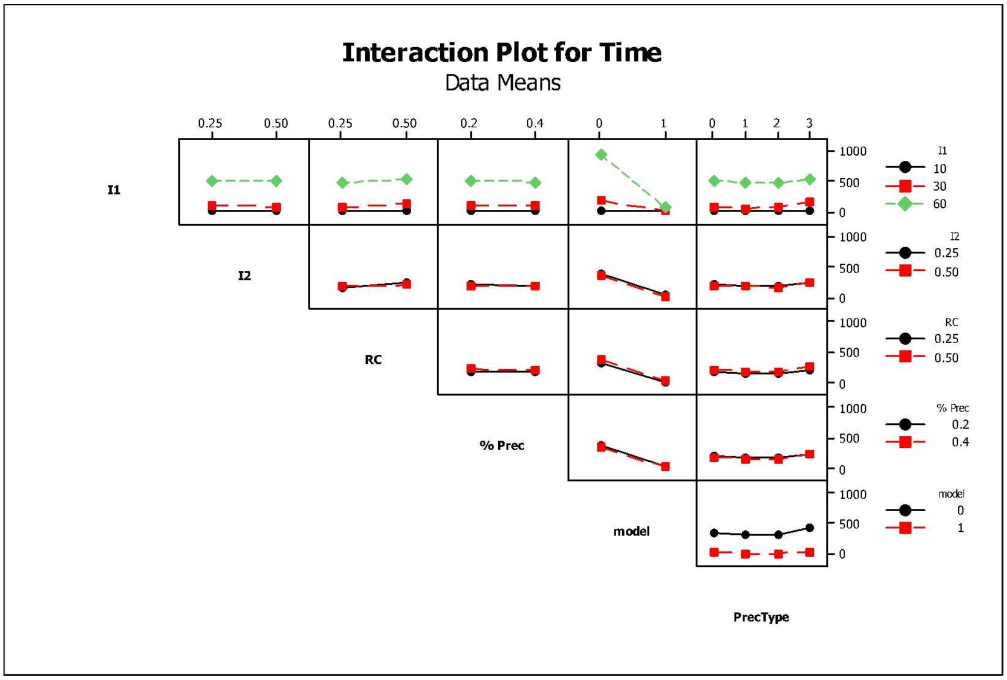

Figure 10. Interaction Plot for mean solution time.

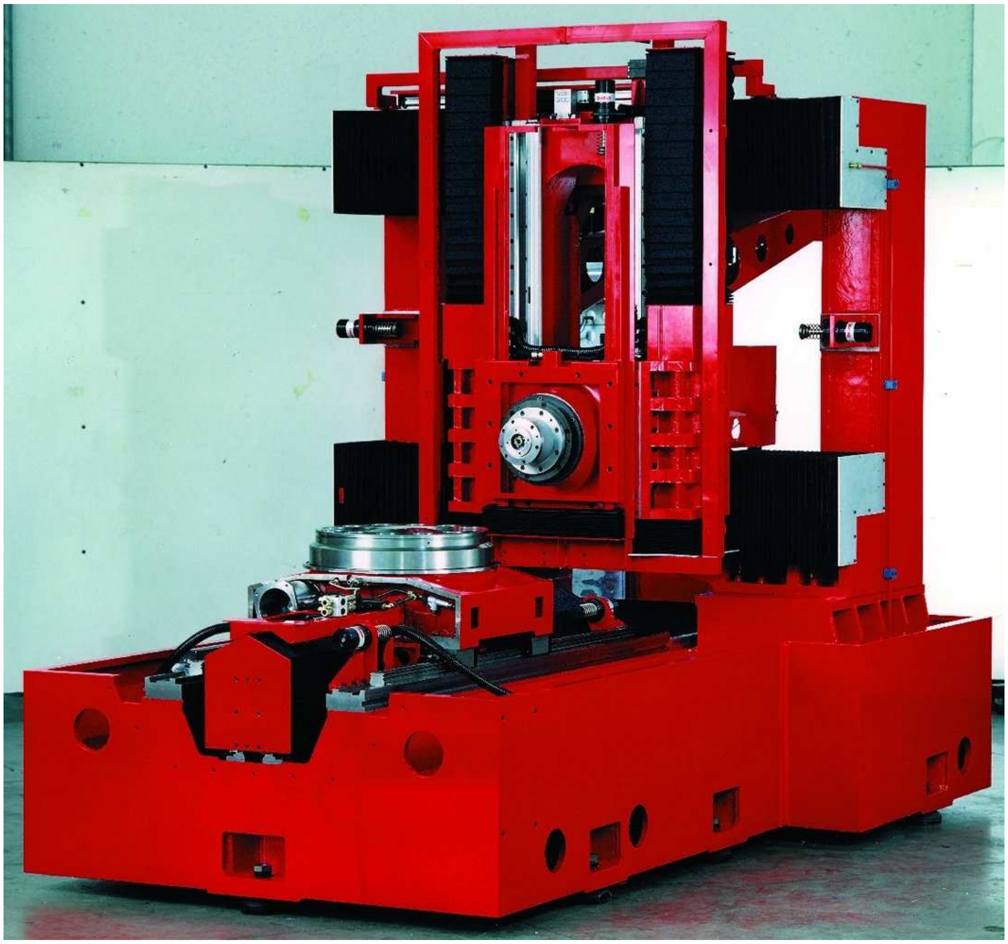

Figure 11. Machining Center structure with preassembled components installed.
386x361mm (72 x 72 DPI)