A discrete time reactive scheduling model for new order insertion in job shop, make-to-order industries

Marta Castilho Gomes, Ana Paula Barbosa-Povoa, Augusto Queiroz Novais

A discrete time reactive scheduling model for new order insertion in job shop, make-to-order industries

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2009-IJPR-0075.R1
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	10-Oct-2009
Complete List of Authors:	Gomes, Marta; Instituto Superior Tecnico, CESUR - Department of Civil Engineering and Architecture Barbosa-Povoa, Ana Paula; Instituto Superior Técnico, CEG-IST - Department of Engineering and Management Novais, Augusto; Instituto Nacional de Engenharia, Tecnologia e Inovacao, Department of Process Modelling and Simulation
Keywords:	JOB SHOP SCHEDULING, MAKE TO ORDER PRODUCTION, INTEGER PROGRAMMING
Keywords (user):	REACTIVE SCHEDULING

A discrete time reactive scheduling model for new order insertion in job shop, make-to-order industries

M. C. Gomes, A. P. Barbosa-Póvoa and A.Q. Novais

Keywords: job shop scheduling, make-to-order production, integer programming, reactive scheduling

Authors' affiliations:

Marta Castilho Gomes (Corresponding author)

CESUR – Department of Civil Engineering and Architecture
Instituto Superior Técnico
Av. Rovisco Pais
1049-001 Lisboa
Portugal

e-mail: marta.gomes@ist.utl.pt

Telephone: + 351 218 418 305
Fax: + 351 218 409 884

Ana Paula Barbosa-Póvoa

CEG – IST – Department of Engineering and Management
Instituto Superior Técnico - Campus do TagusPark
Av. Professor Cavaco Silva
2780-990 Porto Salvo
Portugal

e-mail: apovoa@ist.utl.pt

Telephone: + 351 214 233 507
Fax: + 351 214 233 568

Augusto Queiroz Novais

DMS – Department of Process Modelling and Simulation
Instituto Nacional de Engenharia, Tecnologia e Inovação
Est. Paço do Lumiar
1649-038 Lisboa
Portugal

e-mail: augusto.novais@ineti.pt

Telephone: + 351 210 924 643
Fax: + 351 217 167 016

A discrete time reactive scheduling model for new order insertion in job shop, make-to-order industries

Abstract

This research presents a new reactive scheduling methodology for job shop, make-to-order industries. An integer linear programming formulation previously developed by the authors to schedule this type of industries is extended to address the problem of inserting new orders in a predetermined schedule, which is important in order-driven industries. A reactive scheduling algorithm is introduced to iteratively update the schedules.

Numerical results on realistic examples of job shops of different sizes illustrate the effectiveness of the approach. In each case, different alternatives for inserting a set of new orders in an initial schedule are optimally generated, enabling the user to choose the most convenient one. Solutions are characterized by measures of scheduling efficiency as well as stability measures that assess the impact of rescheduling operations in a previously defined scheduling solution.

Keywords: job shop scheduling, make-to-order production, integer programming, reactive scheduling.

1. Introduction

Industrial environments are dynamic in nature and disturbances in the daily operation of the plant such as machine failures, processing time delays, arrival of new orders and unavailable material may cause a predetermined schedule to become inefficient or even infeasible. Thus, in real-time, the ability to rapidly react to such unexpected events and revise the schedule in an effective way is as important as the scheduling problem itself. Starting in the 1950s, a great deal of effort has been spent developing methods to generate optimal or near-optimal production schedules. However, interest in reactive scheduling techniques, an issue of major importance in practice, intensified only in the last decade and the first review papers in the field appeared a few years ago (Raheja and Subramaniam 2002; Vieira *et al.* 2003).

A review of the literature in the discrete parts manufacturing field shows that most approaches are either heuristic or based on artificial intelligence (AI) techniques. Examples of the first kind of approach are the work of Jain and Elmaraghy (1997), Fang and Xi (1997), Rangsaritratsamee *et al.* (2004), Li *et al.* (1993) and Suwa and Sandoh (2007). The first three make use of genetic algorithms, the fourth develops an heuristic algorithm that adapts ideas from MRP systems and the last one uses tabu search to generate schedules. Smith (1995), Goldman and Boddy (1997), Sun and Xue (2001) and Wong *et al.* (2006) are examples of AI-based reactive scheduling systems described in the literature. The work of Liao *et al.* (1996) is an example of a mathematical programming formulation for reactive scheduling. They present a scheduling tool (with rescheduling capabilities) for the semiconductor industry, based on an integer programming formulation. However, the model is solved by an approximate, near-optimal solution method based on Lagrangian relaxation and network flow techniques; optimal solution was not attempted.

The following lines review recent examples of reactive scheduling approaches based on mathematical programming formulations for the process industries. Roslöf *et al.* (2001, 2002) present a Mixed Integer Linear Programming (MILP) based algorithm that can be used in different instances: to improve an existing non-optimal schedule, to reschedule jobs in the case of changed operational parameters or to insert newly arrived jobs in a production program. Méndez and Cerdá (2003) introduce a MILP formulation for reactive scheduling of multiproduct batch plants. The approach allows multiple rescheduling operations at the same time: insertion of new order arrivals, reassignment of existing batches to alternative units due to equipment failures and the reordering and time-shifting of old batches at the current processing sequences. Janak *et al.* (2006) address reactive scheduling of a large-scale industrial polymer compounding plant based on a MILP continuous-time formulation for short-term scheduling and a rolling horizon medium-term scheduling framework. The unexpected events considered are unit shutdown and the addition or

modification of orders. Ferrer-Nadal *et al.* (2007) present a MILP-based approach for reactive scheduling of multistage, multipurpose batch plants that incorporates the concept of recipe flexibility (some tasks are allowed to modify recipe parameters as an additional rescheduling action). Events of unit breakdown, late order arrival and the need of maintenance tasks are dealt with in the case study presented.

Contrary to these works, where features of the predetermined schedule are maintained and hence the scope of rescheduling is intentionally limited, Vin and Ierapetritou (2000) address the problem of efficient reactive scheduling in multiproduct batch plants with no constraints imposed upon rescheduling alternatives. The situations modelled are machine breakdown and rush order arrival; the objective function includes a term for profit maximization and a penalty term used to minimize the deviations from the original schedule. Relvas *et al.* (2007) present a similar MILP-based approach to reactive scheduling of a complex oil supply system composed of a pipeline and associated end-of-pipe tank farm. Various system perturbations that require rescheduling were modelled, including variation on clients' demands, imposition on product sequence, unpredicted pipeline stoppages and batch volume modifications.

The present paper introduces a new strategy for reactive scheduling of job shop, make-to-order industries that relies on an extension of an integer linear programming model previously developed by the authors for scheduling this type of industries (Gomes, Barbosa-Póvoa and Novais 2005). The problem addressed is that of inserting new orders in a predetermined schedule, which is important and challenging in order-driven industries. Artigues and Roubellat (2002) is an example of other approach to this problem: the authors propose an exact polynomial algorithm to insert a job operation in an existing schedule for multiresource job shop with sequence dependent setup times. Also, Caricato and Grieco (2008) develop a constraint programming approach to insert a new order in a production plan once it has already started. This is applied to an industrial case study where production is composed of several phases each of which can be considered as an independent stage in a generic hybrid flow shop.

A commonly used classification divides mathematical programming models for scheduling into discrete and continuous time models. Discrete time models divide the scheduling horizon into a finite number of intervals with equal and predefined duration and allow the beginning and ending of operations to take place only at the boundaries of these time periods. In continuous time models there is no previous division of the scheduling horizon, and timing decisions are explicitly represented as a set of continuous variables defining the exact times at which the events take place. The proposed approach consists of a Generalized Job Shop scheduling model of discrete type combined with a Reactive Scheduling Algorithm that uses a dynamic scheduling horizon to successively solve the model when new orders arrive. These are added from a given insertion point (in time) onwards by freely rescheduling operations in the previous schedule, as in Vin and

Ierapetritou (2000). Nevertheless, here the extent of schedule changes is controlled by defining the sets of orders that may be rescheduled and those that cannot. By differently assigning those sets, several scenarios for new order insertion may be defined, allowing the best balance between meeting the new order due dates and the disruption of the previously defined schedule to be chosen.

Numerical examples are used to illustrate and further clarify the proposed approach. Order demands of several units to be produced and processing routes through several machine groups where different operations may take place in parallel lead to schedule dimensions (in terms of the total number of operations) considerably higher than the majority of the papers reviewed above (mathematical programming approaches to reactive scheduling). Optimal solutions of the models were successfully obtained offering promising perspectives for the development of this strategy.

2. Problem definition

Figure 1 depicts the type of job shop manufacturing environment modelled. Orders may follow different processing routes. Each order has an associated demand (a number of discrete units to be produced) and due date. Machine groups, represented by rectangles in figure 1, consist of a few homogeneous machines that may process several units in parallel with an upper bound on the total number of units being processed simultaneously – the machine group capacity. Machine groups have buffers (depicted by circles in figure 1) where units wait for processing in the group. Intermediate buffers have finite capacity, while input buffers of a processing route (b_1 and b_7 in figure 1), as well as stocks of finished order units (b_9 and b_{10} in figure 1) are considered of infinite size. Finite buffers have to be considered when products are physically large and buffer room between successive machines is of limited capacity.

The number of units in an order can be divided into smaller sized lots which are loaded onto machine groups throughout the scheduling horizon. Setup times are assumed to be negligible and material handling facilities for the transport of products in the shop are assumed to be non-restrictive. This means the only constraints imposed on the problem of scheduling orders are the finite capacities of the buffers and machine groups.

Given the machine group and buffer capacities, the problem modelled and solved in this paper consists in obtaining a production schedule for an initial set of orders (scheduling problem) and then adapting it to insert new orders arrived after production of the initial ones has started (reactive scheduling problem). Scheduling objectives are to finish the units of each order as close as possible to the corresponding due date and to minimize storage in intermediate buffers.

Different rescheduling policies can be studied that may go from a total reschedule of the old orders (exception made to those undergoing processing) to a non-reschedule of the old orders.

Figure 1. Manufacturing environment modelled

3. Reactive scheduling methodology

3.1 Generalized Job Shop Scheduling Model

In this section the integer programming model developed by Gomes *et al.* (2005) for scheduling job shop, make-to-order industries is generalized with a view to application in a dynamic context. The scheduling problem can be described generically as follows.

Given:

a) a set of orders, each corresponding to a processing route and having an associated demand (number of units) and due date;

b) buffer and machine group capacities;

c) initial conditions, i.e. the number of units in the intermediate and final buffers at the start of the scheduling horizon as well as the units loaded onto the machine groups before the scheduling horizon;

the scheduling problem for a discrete time horizon between *0* and *T* is solved so as to satisfy a given scheduling objective.

In this paper we develop a reactive scheduling strategy for the job shop problem illustrated in figure 2. The scheduling horizon is progressively shifted to take the arrival of new orders into account (**dynamic scheduling horizon**) and the generalized job shop scheduling model solved for successive scheduling horizons ΔT_k (e.g. ΔT_1 , ΔT_2 and ΔT_3 in figure 2) such that:

a) the **"new orders"** arrived between $t_{(k-1)}^{start}$ and t_k^{start} are scheduled from scratch;

b) for the **"old orders"**, i.e. orders already scheduled in $\Delta T_{(k-1)}$, operations scheduled to start from instant t_k^{start} onwards may be rescheduled while operations already finished or still in progress at t_k^{start} cannot be changed.

Figure 2: Dynamic scheduling horizon for reactive scheduling

Each time interval ΔT_k is arbitrarily chosen by the user, the only constraints being $t_k^{start} > t_{(k-1)}^{start}$ and $t_k^{end} \geq t_{(k-1)}^{end}$. The length and limits of the scheduling horizons are independent from each other. Figure 3 represents the discrete treatment of time considered, where scheduling horizon ΔT_k

is divided into intervals of equal length. For modelling purposes, ΔT_k is considered as the set of interval limits t_k^{start}, $t_k^{start}+1$, $t_k^{start}+2$, …, $t_k^{end}-2$, $t_k^{end}-1$, t_k^{end}., as depicted in Figure 3.

Figure 3. Scheduling horizon (discrete time treatment)

Using the above definitions the scheduling problem presented by Gomes *et al.* (2005) is generalized so as to account for the dynamic scheduling horizon. The following sets, parameters and variables are defined.

Indices

i	order
j	processing route
m	machine group
t	time
v	an integer

Sets

I	set of orders
I^{new}	set of new orders
I^{old}	set of old orders to be rescheduled

$I = I^{old} \cup I^{new}$

J	set of processing routes
I_j	set of orders that follow processing route j

$I = \bigcup_{j \in J} I_j$ and $I_j \cap I_{j'} = \varnothing \quad \forall\, j,j' \in J$

M	set of machine groups
M_j	set of machine groups in processing route j
ΔT	scheduling horizon

$\Delta T = \{ t^{start}, t^{start}+1, t^{start}+2, \ldots, t^{end}-2, t^{end}-1, t^{end} \}$

Set elements specifically defined for each processing route j:

f_j	first machine group in processing route j	$f_j \in M_j$
l_j	last machine group in processing route j	$l_j \in M_j$

Parameters

b_{mt} capacity of the buffer of machine group m available between instants t-1 and t

c_{mt} capacity of machine group m available between instants t and $t+1$

d_i due date for order i

e_i earliness penalty per unit of order i per time unit

g_i tardiness penalty per unit of order i per time unit

h_{im} penalty for time spent in the buffer of machine group m per unit of order i per time unit

n_i penalty for not finishing a unit of order i at the end of the scheduling horizon

p_{im} processing time of units of order i in machine group m

q_i demand for order i (number of units)

u_{imv} number of units of order i loaded onto machine group m v time units before the start of the scheduling horizon

x_{im} number of units of order i lodging in the buffer of machine group m at the start of the scheduling horizon

y_i number of finished units of order i at the start of the scheduling horizon

Variables

U_{imt} number of units of order i loaded onto machine group m for processing at instant t;

X_{imt} number of units of order i in the buffer preceding machine group m between instants t-1 and t;

Y_{it} number of finished units of order i between instants t-1 and t.

Note that the set of orders I is divided into two sets: orders to be scheduled for the first time (I^{new}) and those already scheduled and that may be rescheduled (I^{old}). Another division of I is due to the definition of processing routes (set J). I_j is the set of orders that follow processing route j, some of which may belong to I^{new}, others to I^{old}. M is the unordered set of machine groups while M_j is the ordered set of machine groups in processing route j. This means that for each machine group $m \in M_j$ the model refers its preceding and succeeding elements as $m-1$ and $m+1$ respectively, which correspond to the machine groups before and after that machine group in processing route j. Whereas the base scheduling model of Gomes *et al.* (2005) considered constant machine groups and buffer capacities, in the generalization to the reactive scheduling context these capacities become time-dependent and parameters u_{imv}, x_{im} and y_i are used to link successive scheduling solutions.

In terms of variables, Figure 4 illustrates the association of X and Y variables with intervals (of unit length) and that of U variables with interval limits (instants).

Until they are finished, units of order i go (even if instantaneously) through the buffers that precede the machine groups in the corresponding processing route; this is accounted for by the X_{imt} variables. Before processing has started, units are considered to be lodging in the buffer that precedes the first machine group of the processing route. However, the stock of finished units of an order is not associated with any machine group; this is the reason why final buffers are described by a new set of variables Y_{it} . Finally variables U_{imt} describe the units of order i loaded onto machine group m for processing at instant t.

Figure 4. Representation of the model variables.

Based on the above definitions and characteristics the Generalized Job Shop (GJS) scheduling model can now be defined:

Model GJS: *Generalized Job Shop scheduling model*

$$\text{Min} \quad Z = \sum_{j\in J}\sum_{i\in I_j}\left\{\sum_{t\in\Delta T}\left[\psi_{it}\cdot\left(Y_{i,t+1}-Y_{it}\right)+\sum_{m\in M_j\setminus\{f_j\}} h_{im}\cdot X_{imt}\right]+h_i\cdot\left(q_i-Y_{i(t^{last}+1)}\right)\right\} \tag{1}$$

Subject to:

$$X_{imt^{start}} = q_i \qquad \forall\, i\in I^{new},\ m=f_j:\ i\in I_j \tag{2}$$

$$X_{imt}^{start} = 0 \qquad \forall\, i\in I^{new},\ m\in M_j\setminus\{f_j\}:\ i\in I_j \tag{3}$$

$$Y_{it}^{start} = 0 \qquad \forall\, i\in I^{new} \tag{4}$$

$$U_{im(t^{start}-v)} = 0 \qquad v=1,\ldots,P_{im} \qquad \forall\, i\in I^{new},\ m\in M_j:\ i\in I_j \tag{5}$$

$$X_{imt^{start}} = x_{im} \qquad \forall\, i\in I^{old},\ m\in M_j:\ i\in I_j \tag{6}$$

$$Y_{it}^{start} = y_i \qquad \forall\, i\in I^{old} \tag{7}$$

$$U_{im(t^{start}-v)} = u_{imv} \qquad v=1,\ldots,P_{im} \qquad \forall\, i\in I^{old},\ m\in M_j:\ i\in I_j \tag{8}$$

$$X_{im(t+1)} = X_{imt} + U_{i(m-1)(t-p_{i(m-1)})} - U_{imt} \qquad \forall\, i\in I,\ m\in M_j\setminus\{f_j\}:\ i\in I_j,\ t\in\Delta T \tag{9}$$

$$X_{im(t+1)} = X_{imt} - U_{imt} \qquad \forall\, i\in I,\ m=f_j:\ i\in I_j,\ t\in\Delta T \tag{10}$$

$$Y_{i(t+1)} = Y_{it} + U_{im(t-p_{im})} \qquad \forall\, i\in I,\ m=l_j:\ i\in I_j,\ t\in\Delta T \tag{11}$$

$$\sum_{\substack{j\in J:\\ m\in M_j}}\sum_{i\in I_j}\sum_{\tau=t-P_{im}+1}^{t} U_{im\tau} \le c_{mt} \qquad \forall\, m\in M,\ t\in\Delta T \tag{12}$$

$$\sum_{\substack{j\in J:\\ m\in M_j\setminus\{f_j\}}}\sum_{i\in I_j} X_{imt} \le b_{mt} \qquad \forall\, m\in M,\ t\in\Delta T\cup\{t^{last}+1\} \tag{13}$$

Variables X_{imt}, U_{imt}, Y_{it} are integer. (14)

The objective function (1) incorporates the main decisions that influence scheduling in the make-to-order manufacturing environment modelled: to observe the order due dates and to reduce in-process inventory. Orders are composed of several units produced independently; ideally they should all be finished at the corresponding due date but it is possible to finish part of the units before and part of the units after the due date (i.e. the due date is a "soft constraint"). This is accounted for by the first term in (1). Ψ_{it} is the earliness/tardiness penalty coefficient for order i at instant t, which is the product of the time span (regarding the due date) by the earliness/tardiness coefficients per unit of order i per time unit:

$$\psi_{it} = \begin{cases} e_i(d_i - t) & t \le d_i \\ g_i(t - d_i) & t > d_i \end{cases} \qquad \forall\, i \in I,\ t \in \Delta T. \qquad (15)$$

$Y_{i,t+1} - Y_{it}$ is the number of units of order i whose processing was finished at instant t:

$$Y_{i,t+1} - Y_{it} = U_{im(t-p_{im})} \qquad \forall\, i,\ m = l_j:\ i \in I_j,\ t \in \Delta T. \qquad (16)$$

The second term is the cost of keeping (uncompleted) units in intermediate buffers and the third one a penalty for the orders not fully completed at the end of the scheduling horizon (if this term was to be excluded from the objective function, the minimization of this function would imply a value of zero for the X and Y variables, i.e. no production would take place).

Equations 2 to 8 define conditions at the start of the scheduling horizon. Equation (2) states that orders in I^{new} are released at t^{start} and so the number of units of each order i in the first buffer of the corresponding processing route is q_i. Equations (3) and (4) define that the intermediate and final buffers are empty and equation (5) guarantees that no units are loaded before the start of the scheduling horizon. Equations (6), (7) account for the orders in I^{old}, to be rescheduled, at the start of the scheduling horizon and that may have units lodging in any buffer. Equation (8) accounts for the units loaded onto machine groups and still in processing.

Flow balance equations 9-11 are the "core constraints" of the model that establish the relationship between the number of units in buffers in adjacent time intervals. Equation 9 defines the number of units of an order i in an intermediate buffer between instants t and $t+1$ as the algebraic sum of the number of units in the buffer in the previous interval plus the number of units that the preceding machine group finished processing at instant t and hence added to the buffer minus the number of units loaded onto the next machine group at instant t, and hence were withdrawn from the buffer. This balance is different in the cases of the first buffer in a processing route (equation 10) since there is no preceding machine group, and also in the final buffer (equation 11) because there is no subsequent machine group. Note that p_{im}, the processing time of units of order i in machine group m, is a constant: units may be considered as being processed "in parallel",

so the processing time is independent of the number of units that start being processed in a given machine group and time instant.

Finally, the machine capacity constraints (12) ensure that the total number of units undergoing processing in a machine group between instants t and $t+1$ does not exceed the available capacity. In addition, buffer capacity constraints (13) limit the total number of units lodging in the buffer of a machine group, between instants $t-1$ and t, to the available capacity. The initial buffers of processing routes are not included in this sum since they are unlimited in size.

3.2 Reactive Scheduling Algorithm

Figure 5 depicts the reactive scheduling methodology proposed. The integer linear programming model for job shop scheduling derived above is embedded in a Reactive Scheduling Algorithm that shifts the scheduling horizon and solves the model iteratively to insert the new orders that continuously arrive at the shop.

In each iteration of the algorithm the user defines the scheduling horizon as well as the subset of "old orders" (i.e. orders already scheduled) that may be rescheduled, leaving the others unchanged or "fixed". By defining the subset of "reschedulable" old orders the extent of schedule changes is controlled by the user. Continuity between scheduling solutions for these orders is ensured by computing model parameters x_{im}, u_{imv} and y_i. Available machine group and buffer capacities are obtained after subtracting the capacity used by the "fixed" old orders, and as a consequence are dependent on the time instant t. A detailed description of the Reactive Scheduling Algorithm is presented below.

Figure 5: Reactive Scheduling Algorithm

Reactive Scheduling Algorithm:

Initialization step ($k = 1$):

The first scheduling solution is obtained by solving the GJS model from scratch for a set of orders I and a scheduling horizon ΔT_1 defined by the user with:

$$I^{new} = I$$

$$I^{old} = \varnothing$$

$$\Delta T = \Delta T_1$$

Cycle (to be repeated):

A new scheduling solution is obtained from the last one by inserting a set of new orders. This comprises the following steps:

a) Increase k ($k = k+1$);

b) Divide the set of orders I in the last iteration into sets I^{resch} and I^{fixed} where the former is the set of orders that can be rescheduled and the later the set of orders that remain unchanged.

$$I = I^{resch} \cup I^{fixed}$$

c) Define a set of new orders to be inserted in the schedule, I^{new}.

d) Define a new scheduling horizon $\Delta T_k = \{t_k^{start}, t_k^{start}+1, \ldots, t_k^{end}-1, t_k^{end}\}$.

e) Store conditions at the start of the new scheduling horizon for orders in I^{resch}:

e1) Parameter x_{im} stores units lodged in buffers:

$$x_{im} = X_{imt_k^{start}} \qquad \forall\, i \in I^{resch},\ m \in M_j:\ i \in I_j$$

e2) Parameter u_{imv} stores units whose processing in a given machine group started v time units before t_k^{start} and is still on-going:

$$u_{imv} = U_{im(t_k^{start}-v)} \qquad v = 1,\ldots, P_{im} \qquad \forall\, i \in I^{resch},\ m \in M_j:\ i \in I_j$$

e3) Parameter y_i stores finished units:

$$y_i = Y_{it_k^{start}} \qquad \forall\, i \in I^{resch}$$

f) Compute machine group capacities available in ΔT_k by subtracting from the total machine group capacity the units of orders in I^{fixed} still undergoing processing:

$$c_{mt} = c_m - \sum_{\substack{j \in J: \\ m \in M_j}} \sum_{i \in (I_j \cap I^{fixed})} \sum_{\tau=t-P_{im}+1}^{t} U_{im\tau} \qquad \forall\ t \in (\Delta T_{(k-1)} \cap \Delta T_k)$$

$$c_{mt} = c_m \qquad \forall\ t \in (\Delta T_k \setminus \Delta T_{(k-1)})$$

g) Compute buffer capacities available in ΔT_k by subtracting from the total buffer capacity the units of orders in I^{fixed} lodged in buffers:

$$b_{mt} = b_m - \sum_{\substack{j \in J: \\ m \in M_j \setminus \{F_j\}}} \sum_{i \in (I_j \cap I^{fixed})} X_{imt} \qquad \forall\ t \in (\Delta T_{(k-1)} \cap \Delta T_k)$$

$$b_{mt} = b_m \qquad \forall\ t \in (\Delta T_k \setminus \Delta T_{(k-1)})$$

h) Reassign sets I, I^{old} and ΔT:

$$I^{old} = I^{resch}$$

$$I = I^{old} \cup I^{new}$$

$$\Delta T = \Delta T_k.$$

i) Solve the GJS model for I and ΔT .

j) If $\boldsymbol{k}$ did not reach the iteration limit: go to a).

4. Computational study

In this section, computational results for numerical examples are presented. Example 1 (a small size job shop) is thoroughly explored in terms of results with the purpose of clarifying the reactive scheduling methodology. This includes visual representation of solutions, a powerful aid to compare and apprehend differences between rescheduling solutions. A large size job shop (example 2) is further introduced to show the capabilities of the proposed approach regarding problem size.

4. 1 Example 1: Description

Example 1 has two processing routes depicted in figure 6. There are five machine groups in the shop; machine groups M1 and M4 are common to both processing routes. Table 1 displays the processing times in each machine group. For machine groups common to both routes processing

times may be different (M1) or equal (M4) in both routes. These machine groups display a total capacity of 30 units while the other (M2, M3 and M5), assigned to only one route, can process at the most 20 units simultaneously. No more than 5 units are allowed to wait in intermediate buffers when production is running.

Due dates and demand (number of units) for the set of orders to be produced, as well as the corresponding processing route, are presented in Table 2. Orders O1 to O7 are scheduled first and a pair of new orders (O8 and O9) arrives requiring schedule adaptation i.e. reactive scheduling. The total number of units is 215.

Parameter values $g_i = 20$, $e_i = 1$, $h_{im} = 0.1$ and $n_i = 10^7$ for all orders i and machine groups m were used in the objective function (they are equal to the ones used in Gomes *et al.* 2005). A very large value for n_i is used to force production of all units in the scheduling horizon and leave no uncompleted orders at the end.

Both the Generalized Job Shop model and the Reactive Scheduling Algorithm were implemented in the General Algebraic Modelling System (GAMS, 1998); the GJS model was solved with CPLEX version 8.1.0 on a 3 GHz Pentium IV with 512MB of RAM running Windows XP Professional.

Experiments with various features of the solver to speed up the solution process showed a significant increase when the simplex dual algorithm (default setting) was replaced by the simplex primal to obtain the initial relaxed solution of the problem.

Application of the Reactive Scheduling Algorithm to example 1 consisted of scheduling **old orders** O1 to O7 first and then inserting **new orders** O8 and O9, which arrive at instant 15, henceforth called the **insertion point**. The GJS model is solved again and a second scheduling solution, or rescheduling solution, is obtained that schedules the new orders as well as the subset of the old ones allowed to be rescheduled. Time distribution of the order due dates is depicted in Figure 7.

Table 1. Processing times in example 1.

Table 2. Order data in example 1.

Figure 6. Processing routes in example 1.

Figure 7. Due dates in example 1. Orders above the time line follow processing route I, the ones below it follow processing route II.

4.2 Example 1: Generation of scenarios and performance measures

The results obtained for new order insertion with different combinations of old orders to be rescheduled are shown in table 3. Scenarios 1 and 8 represent the two extreme situations of insertion of new orders:

a) no rescheduling (**scenario 1**);

b) rescheduling allowed of every old order (**scenario 8**).

Between these, several intermediate scenarios were created where rescheduling of one, two or three of the old orders is allowed. A scheduling horizon of 60 time units was used for the first scheduling and the interval [15,60] for the second, which would ensure completion of all orders in any of the scenarios. The results shown in table 3 are organized in three sections: objective function related measurements, number of operations and observed changes in the first, or original, schedule.

In order to compare values of the objective function Z between different scenarios, a consistent basis needs to be established, since the set of orders over which the summation in equation (1) (objective function definition) is performed varies with the scenario.

The following alternative Z functions are considered:

Z^{old} : summation over the set of old orders {O1, …, O7}

Z^{resch} : summation over the set of reschedulable old orders (scenario dependent)

Z^{new} : summation over the set of new orders {O8, O9}

Therefore, for the first and second solutions of the GJS model, the objective function and the corresponding scheduling horizons are as follows:

First scheduling:	$Z = Z^{old}$	$\Delta T = [0,60]$
Second scheduling (rescheduling):	$Z = Z^{resch} + Z^{new}$	$\Delta T = [15,60]$

In Table 3, Z^* stands for the optimal value of Z in the second scheduling and Z^{new} for the contribution of new orders. While Z^{new} can be compared between scenarios, Z^* and Z^{resch} cannot, since the number of reschedulable old orders varies. With a view to overcome this limitation, a new value of Z^{old} is computed after the second scheduling (for $\Delta T = [0,60]$). This value is also shown in Table 3 together with Z^{total}, which quantifies the global rescheduling solution and is given by the sum of Z^{old} and Z^{new}. To further characterise the solutions, table 3 also displays the number of scheduled operations (their total and division into old and new orders), the observed changes in the first schedule (which identifies the orders actually changed) and the number of changes in the schedule. These changes may consist of new operations, removed operations or quantity changes, and are defined as follows:

- New operation: U_{imt} (schedule) = 0 and U_{imt} (reschedule) > 0
- Removed operation: U_{imt} (schedule) > 0 and U_{imt} (reschedule) = 0
- Quantity change: U_{imt} (schedule), U_{imt} (reschedule) > 0 and U_{imt} (schedule) ≠ U_{imt} (reschedule)

where U_{imt} (schedule) and U_{imt} (reschedule) are the values of variable U_{imt} for the old order i in the first and second scheduling solutions, respectively.

All combinations or scenarios display a trade-off between the values of Z^{new} and Z^{old}: a better insertion (and hence lower values of Z^{new}) is obtained at the cost of a degradation of Z^{old}. However, the magnitude of the decrease in Z^{new} may not equal the increase in Z^{old} and hence the quality of the global scheduling solution (assessed by Z^{total}). In the limit, when rescheduling of all orders is allowed (**scenario 8**), only orders O1 (finished at the insertion point) and O5 (close to the insertion point) are left unchanged; all other remaining orders are rescheduled. This gives rise to the best new order insertion (a 33% decrease in Z^{new} with respect to scenario 1) but also the highest deterioration of Z^{old} (an 88% increase). The solution is overall the best, with an improvement in Z^{total} of 1.4% regarding scenarios 1 to 3, but also the one displaying the most operations and schedule changes. **Scenarios 3** (rescheduling of order O7) **and 5** (rescheduling of orders O4 and O7) stand out as those displaying a finer balance between new order insertion and schedule changes; the former if we want changes to be negligible and the latter when moderate changes are allowed.

Table 4 shows model performance regarding the (first) scheduling solution and the smallest and largest scenarios for rescheduling (scenarios 1 and 8). The number of variables, constraints and non-zero elements of the models are shown as well as the number of iterations and computation times. CPU times needed to solve the models to optimality were considerably less than one second.

4.3 Example 1: Visual representation of solutions

Figures 8 to 10 display load diagrams for machine group M1, common to both processing routes, in scenarios 1, 5 and 8. A distinction is made between:

a) Fixed capacity, used by operations that cannot be rescheduled, either because the orders are fixed or the operations were started before the insertion point.

b) Capacity used by operations of reschedulable old orders that start at the insertion point or afterwards.

c) Capacity used by the new orders.

The profiles of the capacity used by the new orders and the reschedulable old orders are different in the three diagrams. Allowing for more orders to be rescheduled results in greater flexibility in capacity management and more units of the new orders can be started earlier.

Figure 11 depicts the Gantt chart for the initial schedule (old orders), figure 12 displays the solution after new order insertion in scenario 1 and figures 13A and 13B the solution obtained in scenario 8. Operations are represented by bars in different grey shades so as to make the distinction between adjacent machine groups clearer. Each bar displays the number of units loaded onto the machine group as well as the corresponding order (in brackets). The total number of units loaded at one time is limited by the machine group capacity.

When new orders are optimally inserted with no changes in the previous schedule (figure 12), a total of 40 operations is required to process these orders. Operations for new orders are shown in a different grey shade and pattern to differentiate from old order operations. In machine groups M1 and M3 new order operations start immediately after old order operations are finished, indicating optimization of resource use; the same is not visible in the other machine groups due to processing route constraints. The number of units per operation for new orders is on average lower than for the old ones, being under 5 in several cases (for the old orders this is a multiple of 5 between 5 and 20).

In figures 13A and 13B the bar colour and pattern for new order operations is equal to that used in figure 12. Regarding the old orders, distinction is made between old order operations left unchanged and old order operations different from the initial schedule either in time allocation or number of units processed. Unchanged operations are marked with the same colour and pattern used in figure 12. Division of production into more operations with a lower unit content per operation is visible in every machine group both for old and new orders, compared with scenario 1; for new orders, the number of scheduled operations more than doubles (88 operations). Order O4 is the only exception to this operation division trend. A comparison of the time span of new order production in both scenarios shows that rescheduling of old orders allows new orders to be started earlier in all machine groups but also to be finished later in some of them. The first effect, however, is stronger since a 33% decrease in Z^{new} is observed, and thus the overall insertion of the new orders is better in scenario 8.

Table 3. Summary of results for example 1.

Table 4. Model performance for example 1.

Figure 8. Capacity of machine group M1 used in scenario 1 (no orders rescheduled).

Figure 9. Capacity of machine group M1 used in scenario 5 (orders O4 and O7 rescheduled).

Figure 10. Capacity of machine group M1 used in scenario 8 (all orders may be rescheduled).

Figure 11. Scheduling of old orders in example 1.

Figure 12. Insertion of new orders without rescheduling of the old ones in example 1.

Figure 13A. Insertion of new orders with full rescheduling of the old ones in example 1.

Figure 13B. Insertion of new orders with full rescheduling of the old ones in example 1 (continued).

4.4 Example 2: Description

A larger example is now introduced with result analysis provided in condensed form.

Figure 14 displays the job shop in schematic form, which comprises 14 machine groups and 4 processing routes. Table 5 displays the processing times and table 6 machine groups and buffer capacities. Table 7 presents the order data: due date, demand and processing route for each order. Orders O1 to O10 are scheduled first and three new orders (O11 to O13) must be inserted in the production scheduling. The number of units to produce is 710 in total. Costs in the objective function are the same as in example 1.

4.5 Example 2: Results

A scheduling horizon of 170 time units is needed to finish production of the old orders. New orders can be scheduled from instant 40 onwards (insertion point) and a new scheduling horizon [40, 240] is needed in any of the scenarios constructed to complete all orders (limits of scheduling horizon are rounded to a multiple of 10). Results and model performance are summarized in table 8.

Four scenarios were constructed for new order insertion: the extreme scenarios (no rescheduling and rescheduling of all orders allowed) and two intermediate scenarios (rescheduling of 3 and 7 orders allowed). Although the corresponding integer programming models display tenths of thousand variables and constraints, solving them to obtain (and confirm) optimal solutions took just a few seconds. The optimal value of the objective function (Z^*), number of operations scheduled and number of changes in the first schedule are also shown. Figure 15 shows division of the objective function into the Z^{resch} and Z^{new} terms. Allowing for a larger number of old orders to be rescheduled leads to an increase in Z^{resch} and Z^* but also a better insertion of the new orders since the corresponding Z^{new} value decreases.

Table 5. Processing times in example 2.

Table 6. Machine groups and buffer capacities in example 2

Table 7. Order data in example 2

Table 8. Summary of results for example 2

Figure 14. Processing routes in example 2.

Figure 15. Division of the objective function in example 2.

4.6 Discussion of model performance

On the one hand, the size of NP-hard scheduling problems for which results with exact methodologies can be obtained has been consistently increasing with the expansion of computer capacity. In the case of mixed-integer linear programming approaches, another factor adds to this effect: the development of algorithms for solving this type of models which have been incorporated into increasingly efficient software tools nowadays available to researchers as pointed out by Bixby *et al.* (2000) and Gupta and Stafford (2006). On the other hand, in discrete time models scheduling constraints have only to be monitored at specific and known time points, which reduces the problem complexity and makes the model structure simpler and easier to solve, particularly when resource and inventory limitations are taken into account (Méndez *et al.* 2006). In fact, the solution of the linear relaxation is generally tight to the integer solution which has a dramatic effect on the number of nodes to be solved and hence the computation time of the branch-and-bound search.

Both effects explain the fact that for example 2 the Reactive Scheduling Algorithm was able to generate the rescheduling scenarios in a few seconds although the MILP models have thousands of variables (all of discrete nature) and constraints (table 8). Response time is a critical issue for reactive scheduling since in most of the real-world industrial environments it is necessary to quickly adapt the on-going schedule after the occurrence of unexpected events.

A major disadvantage of discrete time models is that their performance depends on the time unit used. In most of the real cases processing times need to be rounded in order to avoid obtaining a very large model (which will happen if the discretisation interval used is small). In continuous time models processing times can be accurately represented because continuous variables are used for the timing decisions. A reactive scheduling approach for make-to-order industries based on a continuous time formulation has also been developed by the authors in Gomes (2007), Gomes, Barbosa-Póvoa and Novais (2006, 2007).

5. Conclusions and future work

In this paper a new strategy for reactive scheduling of job shop, make-to-order industries is presented. It combines an integer linear programming model (discrete time model) and a Reactive Scheduling Algorithm that shifts the scheduling horizon and solves the model iteratively to take account of new orders that keep arriving at the shop. Continuity between solutions is guaranteed by storing conditions in the previous solution at the start of a new scheduling horizon. When inserting new orders the extent of schedule changes is controlled by the user through definition of the subset of orders that may be rescheduled, leaving the others unchanged. Different scenarios for inserting a set of new orders in an on-going schedule can thus be constructed and compared making the

Reactive Scheduling Algorithm appropriate to be included in a decision support system for scheduling job shop, make-to-order industries.

The approach is illustrated with realistic examples of a small and a large size job shop where orders have an associated due date and demand (several discrete units to be produced). In both cases, an initial schedule and different alternatives for inserting new orders were generated using commercial mathematical programming software. Optimal solutions obtained in very low CPU time show the usefulness of the proposed approach. Result analysis included computing changes to the initial schedule and, for the small size example, depicting solutions in graphical form (Gantt charts and machine load diagrams). The former is a novel aspect of this work, since measures of scheduling efficiency are well described in the scheduling literature but the impact of disruptions induced by rescheduling has seldom been addressed (Rangsaritratsamee *et al.* 2004). Visual representation of solutions is almost absent from the scheduling literature for discrete parts manufacturing. We opted to include it since it conveys more information about a solution than numerical performance measures and therefore has a deeper impact in illustrating how the Reactive Scheduling Algorithm works.

In the future, and regarding application of this work to a specific production environment, the definition of the objective function implies determining inventory costs (of uncompleted and finished units) and penalties for failing to meet due dates, which will depend both on the product type and the client importance. The impact of the cost/penalty used for each term both on the solution generated and the computational effort should also be assessed. This sensitivity analysis would be more comprehensibly undertaken on the basis of a system that was accepted by the scientific community as a credible standard. With this work the authors aimed at the setting up of such a system, which explains their concern in adjusting the penalties for tardiness, earliness, inventory and processed orders, to values experienced and reflecting priorities of actual operating plants. On-going exploratory work suggests that given the number of criteria involved, the use of two nested optimization cycles might be preferable to the generation of Pareto curves. In which case the inner cycle would be the system as is, and the outer one the optimization of the criteria themselves, given a previous indication of the aspiration level or range for each of them.

Constant changes in production schedules induce instability, sometimes labeled “shop floor nervousness”, and are undesirable in dynamic environments. Future developments of the integer programming model should therefore intensify control upon the rescheduling process such as adding constraints to limit changes in the operations previously scheduled, using different weights for old and new orders in the objective function and including a penalty term in the objective function for changes in the schedule. Inserting new orders in the production schedule is a significant problem in order-driven industries. In the future, the Reactive Scheduling Algorithm should cope

with other important situations in this type of industry like changes in order specifications (due date, demand) and order cancelling.

References

Artigues, C. and Roubellat, F. (2002). An efficient algorithm for operation insertion in practical jobshop schedules. *Production Planning and Control*, 13, 175-186.

Bixby, R.E., Fenelon, M., Zonghao, G., Rothberg, E. and Wunderling, R. (2000). MIP: theory and practice - Closing the gap. In: Powell, M.J.D. and Scholtes, S. (Eds.), *System Modelling and Optimization: Methods, Theory and Applications* (Kluwer Academic Publishers), 19-50.

Caricato, P. and Grieco, A. (2008). An online approach to dynamic rescheduling for production planning applications. *International Journal of Production Research*, 46, 4597–4617.

Fang, J. and Xi, Y. (1997). A rolling horizon job shop rescheduling strategy in the dynamic environment. *International Journal of Advanced Manufacturing Technology*, 13, 227-232.

Ferrer-Nadal, S., Méndez, C. A., Graells, M. and Puigjaner, L. (2007). Optimal reactive scheduling of manufacturing plants with flexible batch recipes. *Industrial and Engineering Chemistry Research*, 46, 6273-6283.

GAMS – A User's Guide (1998). GAMS Development Corporation, USA.

Goldman, R.O. and Boddy, M.S. (1997). A constraint-based scheduler for batch manufacturing. *IEEE Expert*, 12, 49-56.

Gomes, M.C., Barbosa-Póvoa, A. and Novais, A.Q. (2005). Optimal scheduling for flexible job shop operation. *International Journal of Production Research*, 43, 2323-2353.

Gomes, M.C., A. Barbosa-Póvoa and A.Q. Novais (2006). Optimal reactive scheduling of multipurpose, make-to-order industries. *Proceedings of the 16th European Symposium on Computer Aided Process Engineering and 9th International Symposium on Process Systems Engineering* (Amsterdam: Elsevier B.V.), 1587-1592.

Gomes, M.C. (2007). Reactive scheduling in make-to-order production systems: An optimization based approach. *PhD Thesis in Systems Engineering*, Instituto Superior Técnico, Technical University of Lisbon.

Gomes, M.C., A. Barbosa-Póvoa e A.Q. Novais (2007). A continuous time formulation for reactive scheduling of job shop, make-to-order industries with recirculation and assembly. *Proceedings of ORP3 Conference - The OR Peripatetic Post-Graduate Programme*, 235-244.

Gupta, J.N.D. and Stafford Jr., E.F. (2006). Flowshop scheduling research after five decades. *European Journal of Operational Research*, 169, 699-711.

Harjunkoski, I. and Grossmann, I.E. (2002). Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. *Computers and Chemical Engineering*, 26, 1533-1552.

Jain, A.K. and Elmaraghy, H.A. (1997). Production scheduling/rescheduling in flexible manufacturing. *International Journal of Production Research*, 35, 281-309.

Janak, S.L., Floudas, C.A., Kallrath, J. and Vormbrock, N. (2006). Production scheduling of a large-scale industrial batch plant. II. Reactive scheduling. *Industrial and Engineering Chemistry Research*, 45, 8253-8269.

Li, R.-K., Shyu, Y.-T. and Adiga, S. (1993). An heuristic rescheduling algorithm for computer-based production scheduling systems. *International Journal of Production Research*, 31, 1815-1826.

Liao, D.-Y., Chang, S.-C., Pei, K.-W. and Chang, C.-M. (1996). Daily scheduling for R&D semiconductor fabrication. *IEEE Transactions on Semiconductor Manufacturing*, 9, 550-560.

Méndez, C.A. and Cerdá, J. (2003). Dynamic scheduling in multiproduct batch plants. *Computers and Chemical Engineering*, 27, 1247-1259.

Méndez, C.A., Cerdá, J., Grossmann, I.E., Harjunkoski, I. and Fahl, M. (2006). State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Computers and Chemical Engineering*, 30, 913-946.

Nemhauser, G.L. and Wolsey, L.A. (1999). *Integer and Combinatorial Optimization* (New York: John Wiley and Sons).

Pan, J.C-H. (1997). A study of integer programming formulations for scheduling problems. *International Journal of Systems Science*, 28, 33-41.

Raheja, A.S. and Subramaniam, V. (2002). Reactive recovery of job shop schedules – A review. *International Journal of Advanced Manufacturing Technology*, 19, 756-763.

Rangsaritratsamee, R., Ferrell, W.G. and Kurz, M.B. (2004). Dynamic rescheduling that simultaneously considers efficiency and stability. *Computers and Industrial Engineering*, 46, 1-15.

Relvas, S., Matos, H. A., Barbosa-Póvoa, A. P. F. D. and Fialho, J. (2007). Reactive scheduling framework for a multiproduct pipeline with inventory management. *Industrial and Engineering Chemistry Research*, 46, 5659-5672.

Roslöf, J., Harjunkoski, I., Björkqvist, J., Karlsson, T. and Westerlund, T. (2001). An MILP-based reordering algorithm for complex industrial scheduling and rescheduling. *Computers and Chemical Engineering*, 25, 821-828.

Roslöf, J., Harjunkoski, I., Westerlund, T. and Isaksson, J. (2002). Solving a large-scale industrial scheduling problem using MILP combined with a heuristic procedure. *European Journal of Operational Research*, 138, 29-42.

Sawik, T. (2000). Mixed integer programming for scheduling flexible flow lines with limited intermediate buffers. *Mathematical and Computer Modelling*, 31, 39-52.

Smith, S.F. (1995). Reactive scheduling systems. In: Brown, D.E. and Scherer, W.T. (eds.), *Intelligent Scheduling Systems* (Boston: Kluwer Academic Publishers), 155-192.

Sun, J. and Xue, D. (2001). A dynamic reactive scheduling mechanism for responding to changes of production orders and manufacturing resources. *Computers in Industry*, 46, 189-207.

Suwa, H. and Sandoh, H. (2007). Capability of cumulative delay based reactive scheduling for job shops with machine breakdowns. *Computers and Industrial Engineering,* 53, 63–78.

Vieira, G.E., Herrmann, J.W. and Lin, E. (2003). Rescheduling manufacturing systems: a framework of strategies, policies and methods. *Journal of Scheduling*, 6, 39-62.

Vin, J.P. and Ierapetritou, M.G. (2000). A new approach for efficient rescheduling of multiproduct batch plants. *Industrial and Engineering Chemistry Research*, 39, 4228-4238.

Wong, T. N., Leung, C. W., Mak, K. L. and Fung, R. Y. K. (2006). Integrated process planning and scheduling/rescheduling - an agent-based approach. *International Journal of Production Research*, 44, 3627–3655.

Table 1. Processing times in example 1

Processing route	Machine group					Total processing time
	M1	M2	M3	M4	M5	
I	6	3	-	2	4	15
II	5	-	6	2	-	13

Table 2. Order data in example 1

Order	Demand	Due date	Processing route
O1	20	15	I
O2	35	25	I
O3	15	30	I
O4	15	35	I
O5	30	20	II
O6	15	28	II
O7	25	32	II
O8	25	35	I
O9	35	37	II

Table 3. Summary of results for example 1

Scenario	No. of reschedulable old orders	Z^{*}†	Z^{total}	Z^{old}	Z^{new}	No. of operations			Changes in the first schedule				
						Total	Old orders	New orders	Changed orders	No of changes	No. of new operations	No. of removed operations	No. of quantity changes
1	None	10960.0	14849.0	3889.0	10960.0	104	64	40	None	0	0	0	0
2	O4	11660.0	14849.0	4549.0	10300.0	118	68	50	O4	12	8	4	0
3	O7	12660.5	14849.0	4789.0	10060.0	105	64	41	O7	6	3	3	0
4	O2, O6	11793.0	14848.5	3888.5	10960.0	114	64	50	O2	2	1	1	0
5	O4, O7	13300.9	14789.4	6589.4	8200.0	122	76	46	O4, O7	34	19	7	8
6	O3, O4, O7	13860.7	14749.2	5829.2	8920.0	134	84	50	O3, O4, O7	43	27	7	9
7	O4, O6, O7	13502.4	14789.4	6109.4	8680.0	128	85	43	O4, O7	43	28	7	8
8	All (O1 to O7)	14637.2	14637.2	7317.2	7320.0	141	88	53	O2, O3, O4, O6, O7	64	34	10	20

† Z^{*}: optimal value of the objective function in the second scheduling (rescheduling), where $Z^{*} = Z^{resch} + Z^{new}$. For solution comparison, Z^{total} must be used ($Z^{total} = Z^{old} + Z^{new}$). Z^{old}, Z^{resch}, Z^{new}: summation of equation (1) over the set of old orders, reschedulable old orders and new orders, respectively.

Table 4. Model performance for example 1

Solution description	Scheduling horizon	No. of variables	No. of constraints	No. of non zeroes	No. of iterations	CPU time (sec)†	Objective function
Scheduling	[0,60]	3860	2888	16017	477	0.12	3889.0
Rescheduling (scenario 1)	[15,60]	846	942	3427	56	0.05	10960.0
Rescheduling (scenario 8)	[15,60]	3850	2796	15603	341	0.17	14637.2

† For proven optimal solution (optimality gap = 0).

Table 5. Processing times in example 2

Production sequence	Machine group							
	m1	m2	m3	m4	m5	m6	m7	m8
I	3	4	1	4	4	-	2	2
II	3	-	1	-	5	5	2	2
III	-	-	-	-	-	5	-	-
IV	-	-	4	7	-	-	-	-

Production sequence	Machine group							Total time
	m9	m10	m11	m12	m13	m14	m15	
I	-	-	-	-	-	-	-	20
II	-	-	-	-	-	-	-	18
III	7	8	8	7	8	6	-	49
IV	-	8	6	-	-	6	9	40

Table 6. Machine groups and buffer capacities in example 2

Machine group	Machine capacity	Buffer capacity
m1	20	500
m2	45	100
m3	45	80
m4	30	110
m5	10	90
m6	25	60
m7	15	75
m8	15	80
m9	20	90
m10	30	90
m11	35	90
m12	15	90
m13	15	90
m14	40	90
m15	10	90

Table 7. Order data in example 2

Order	Demand	Due date	Processing route
O1	60	100	I
O2	35	50	I
O3	30	87	I
O4	25	22	II
O5	25	80	II
O6	65	60	III
O7	40	90	III
O8	25	50	III
O9	85	40	IV
O10	40	70	IV
O11	95	70	I
O12	85	100	III
O13	100	90	IV

Table 8. Summary of results for example 2

Scenario	No. of reschedulable old orders	No. of variables	No. of constraints	CPU time (sec)†	Objective function	No. of operations	No. of changes in first schedule
1	0	8687	10228	0.5	365900.0	185	0
2	3	17373	15094	1.8	446547.5	296	118
3	7	28699	21454	6.5	485413.5	472	268
4	10	37385	26320	5.7	486813.5	490	283

† For proven optimal solution (optimality gap = 0).

List of figure captions:

Figure 1. Manufacturing environment modelled.

Figure 2. Dynamic scheduling horizon for reactive scheduling.

Figure 3. Scheduling horizon (discrete time treatment).

Figure 4. Representation of the model variables.

Figure 5. Reactive Scheduling Algorithm.

Figure 6. Processing routes in example 1.

Figure 7. Due dates in example 1. Orders above the time line follow processing route I, the ones below it follow processing route II.

Figure 8. Capacity of machine group M1 used in scenario 1 (no orders rescheduled).

Figure 9. Capacity of machine group M1 used in scenario 5 (orders O4 and O7 rescheduled).

Figure 10. Capacity of machine group M1 used in scenario 8 (all orders may be rescheduled).

Figure 11. Scheduling of old orders in example 1.

Figure 12. Insertion of new orders without rescheduling of the old ones in example 1.

Figure 13A. Insertion of new orders with full rescheduling of the old ones in example 1.

Figure 13B. Insertion of new orders with full rescheduling of the old ones in example 1 (continued).

Figure 14. Processing routes in example 2.

Figure 15. Division of the objective function in example 2.

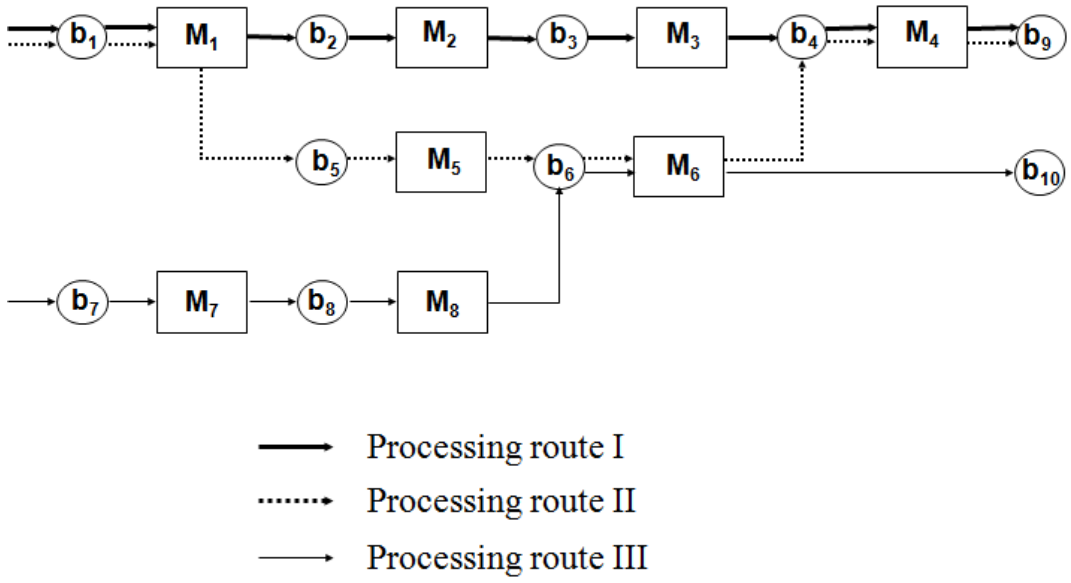

Figure 1. Manufacturing environment modelled

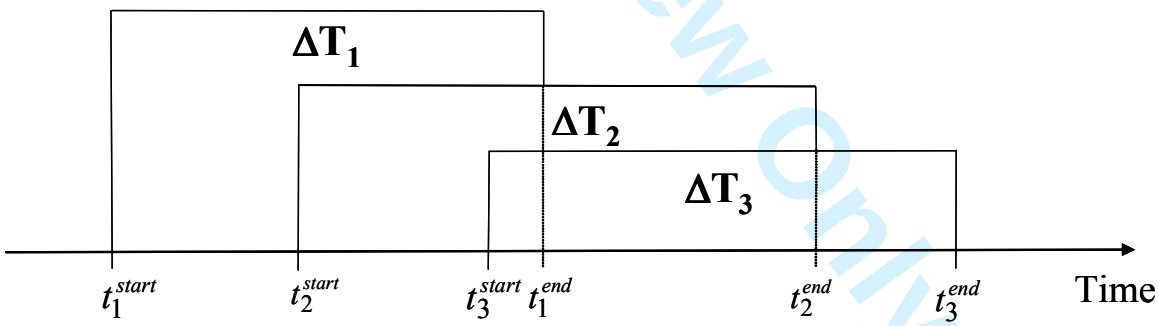

Figure 2. Dynamic scheduling horizon for reactive scheduling

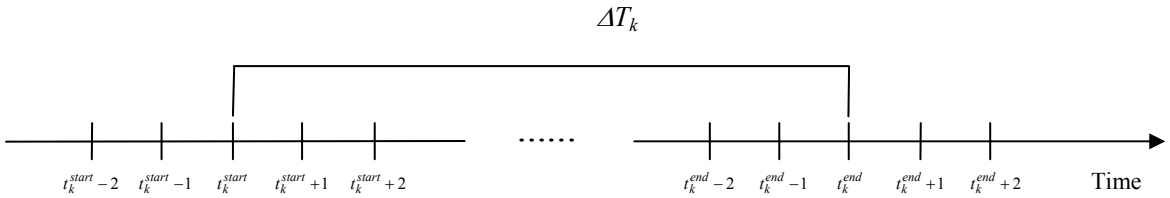

Figure 3. Scheduling horizon (discrete time treatment)

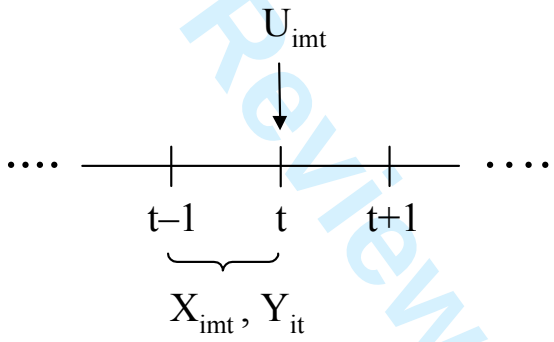

Figure 4. Representation of the model variables.

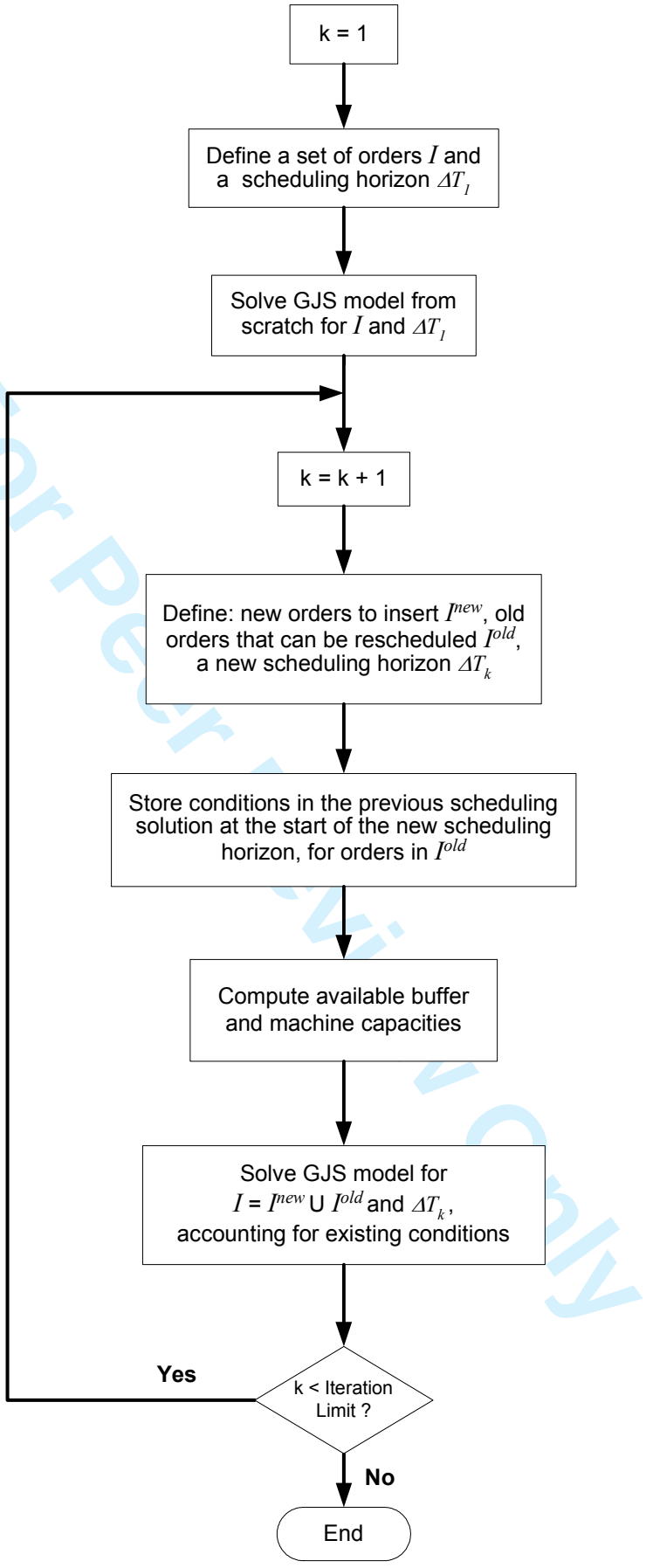

Figure 5. Reactive Scheduling Algorithm

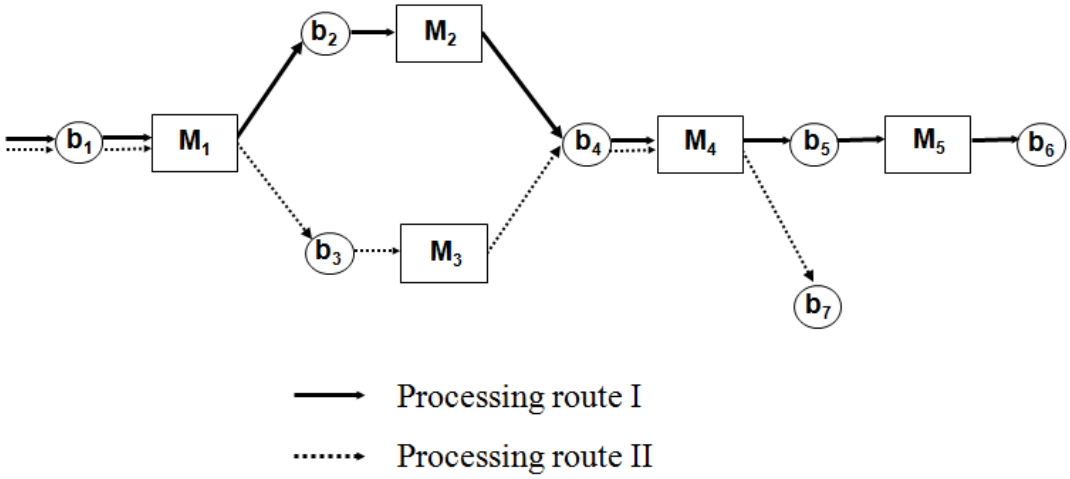

Figure 6. Processing routes in example 1

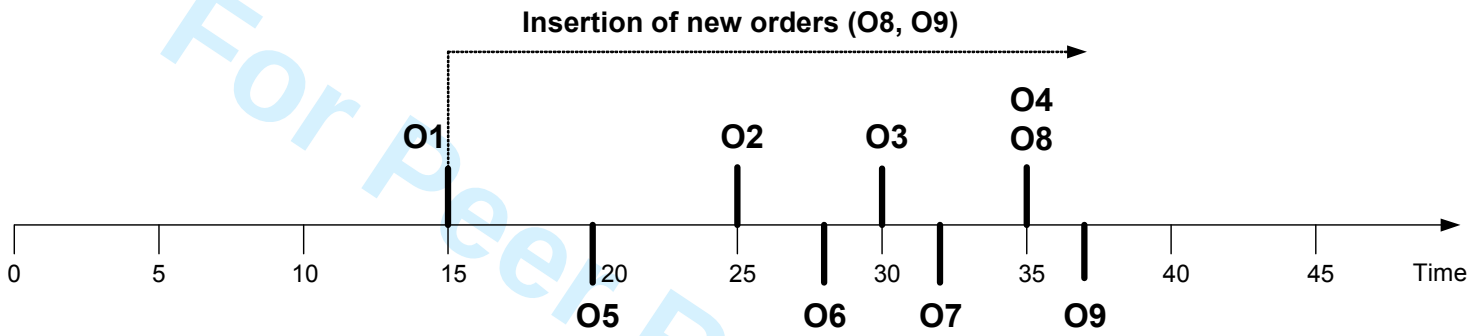

Figure 7. Due dates in example 1. Orders above the time line follow processing route I, the ones below it follow processing route II.

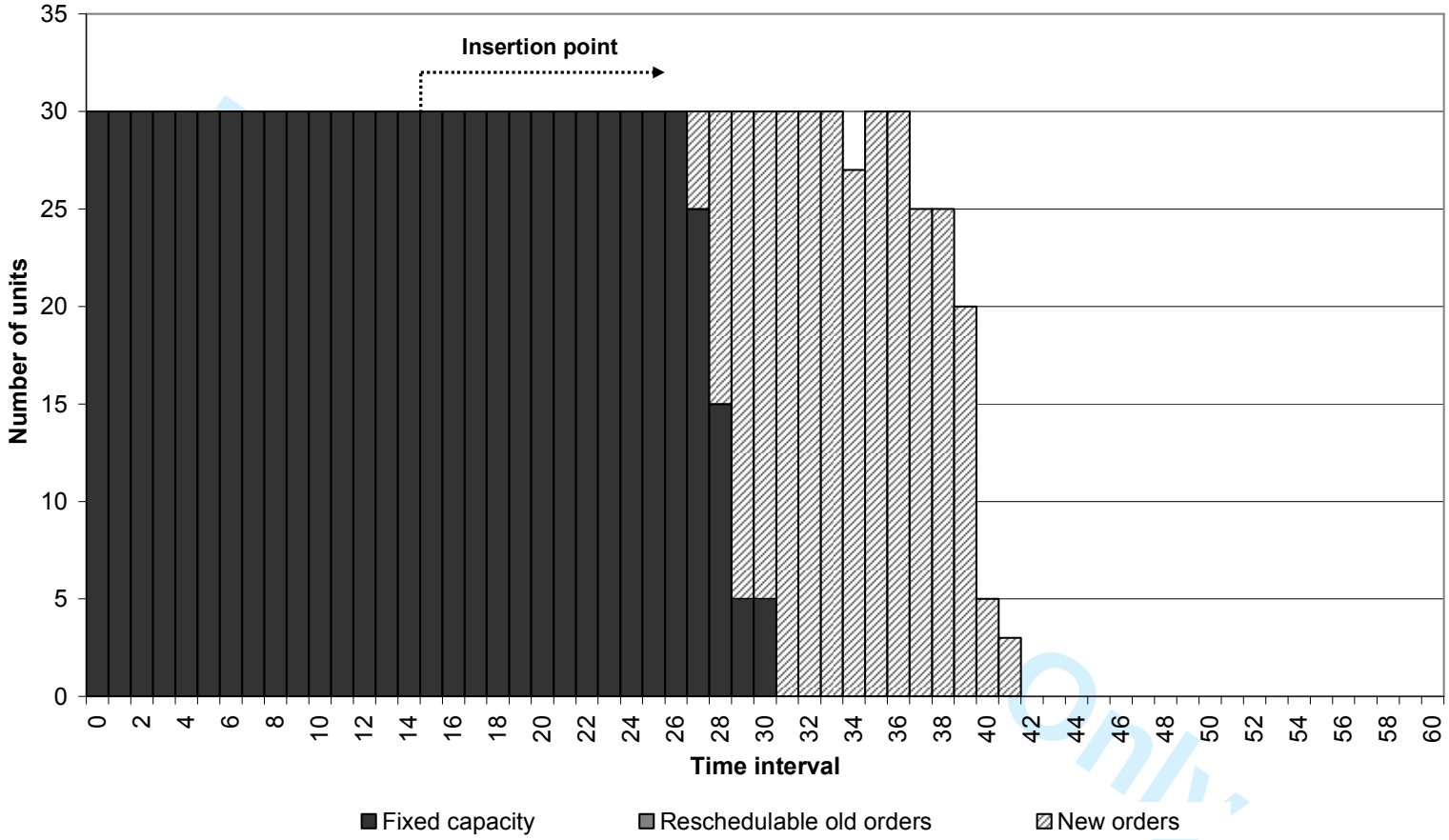

Figure 8. Capacity of machine group M1 used in scenario 1 (no orders rescheduled)

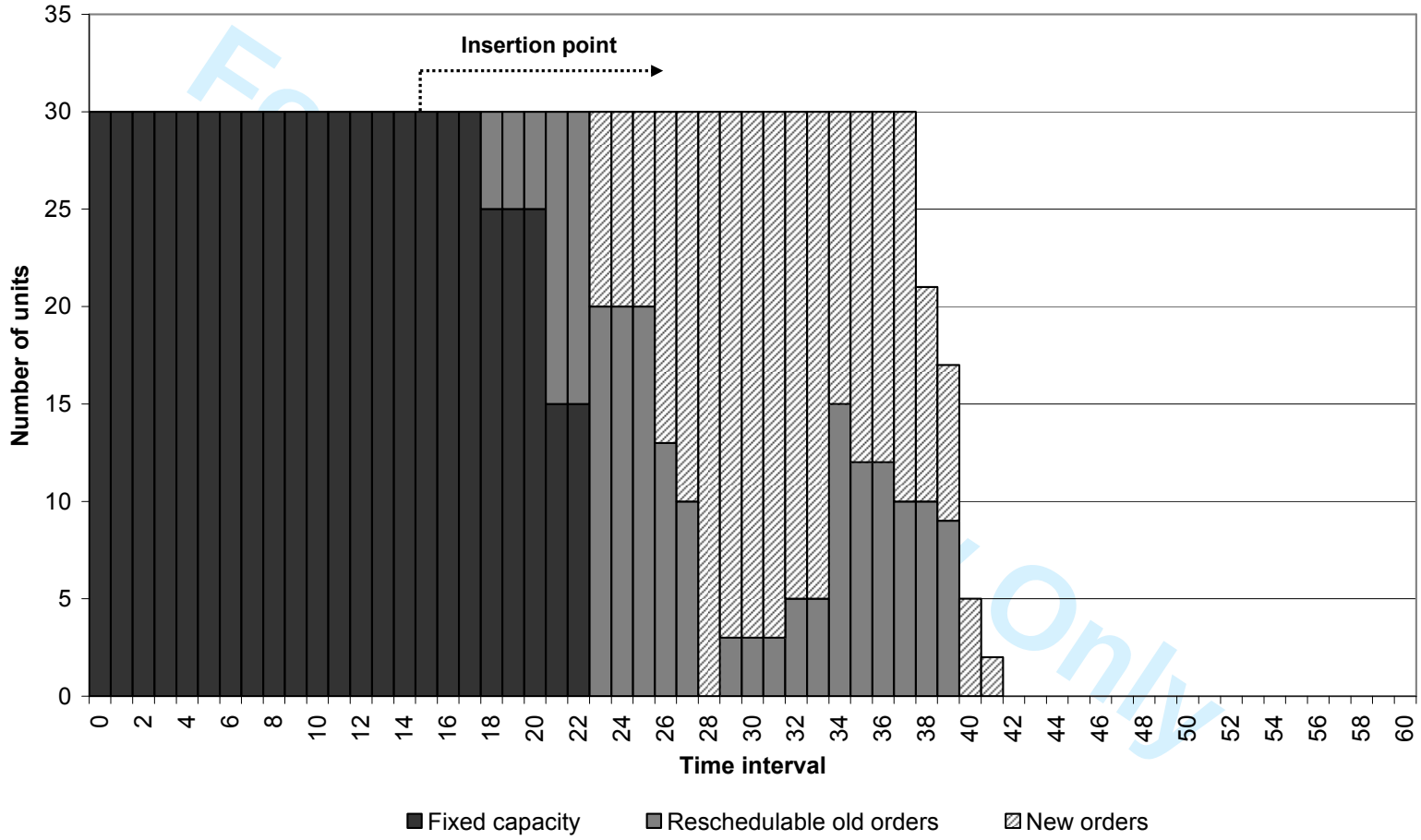

Figure 9. Capacity of machine group M1 used in scenario 5 (orders O4 and O7 rescheduled)

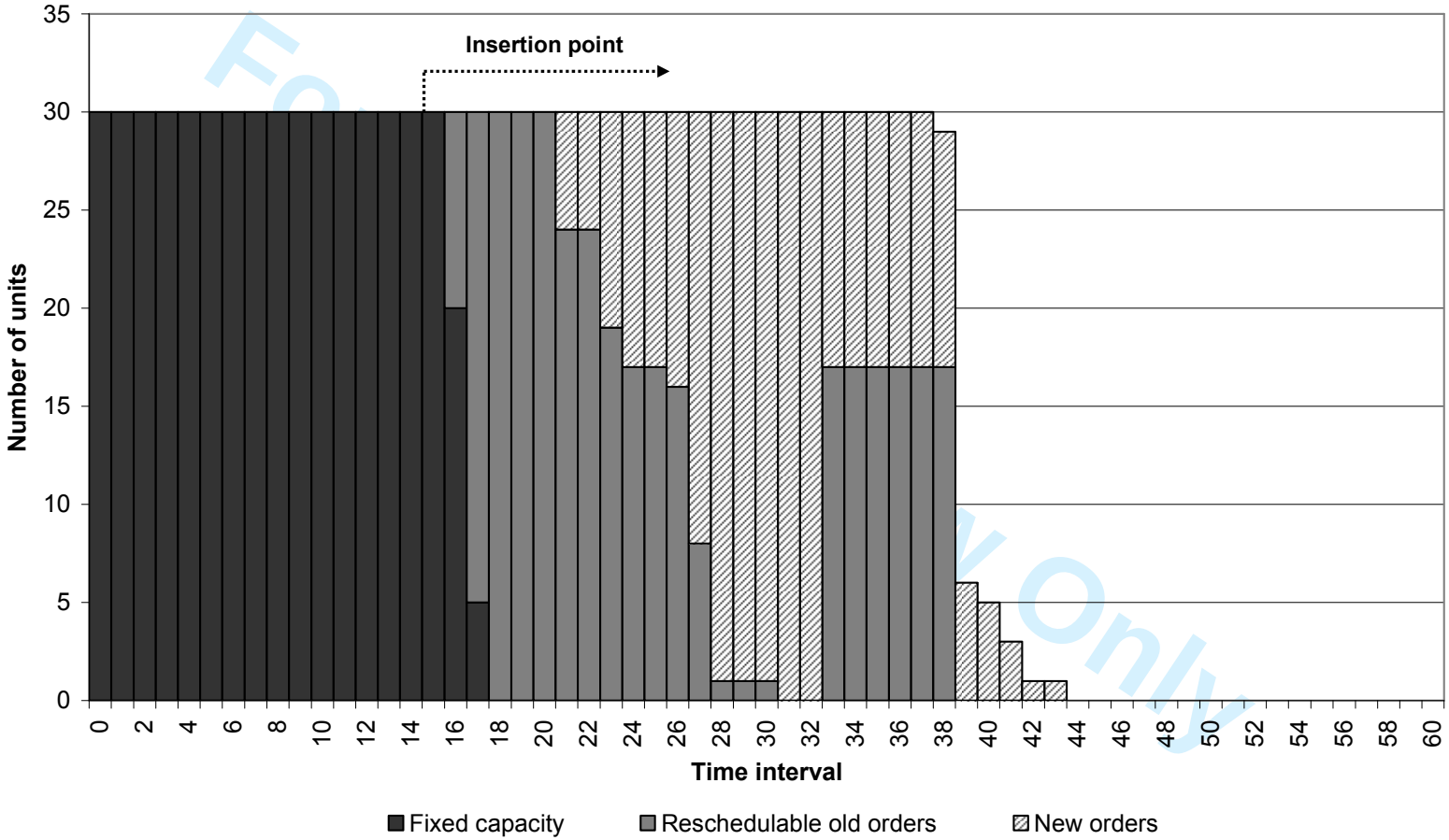

Figure 10. Capacity of machine group M1 used in scenario 8 (all orders may be rescheduled)

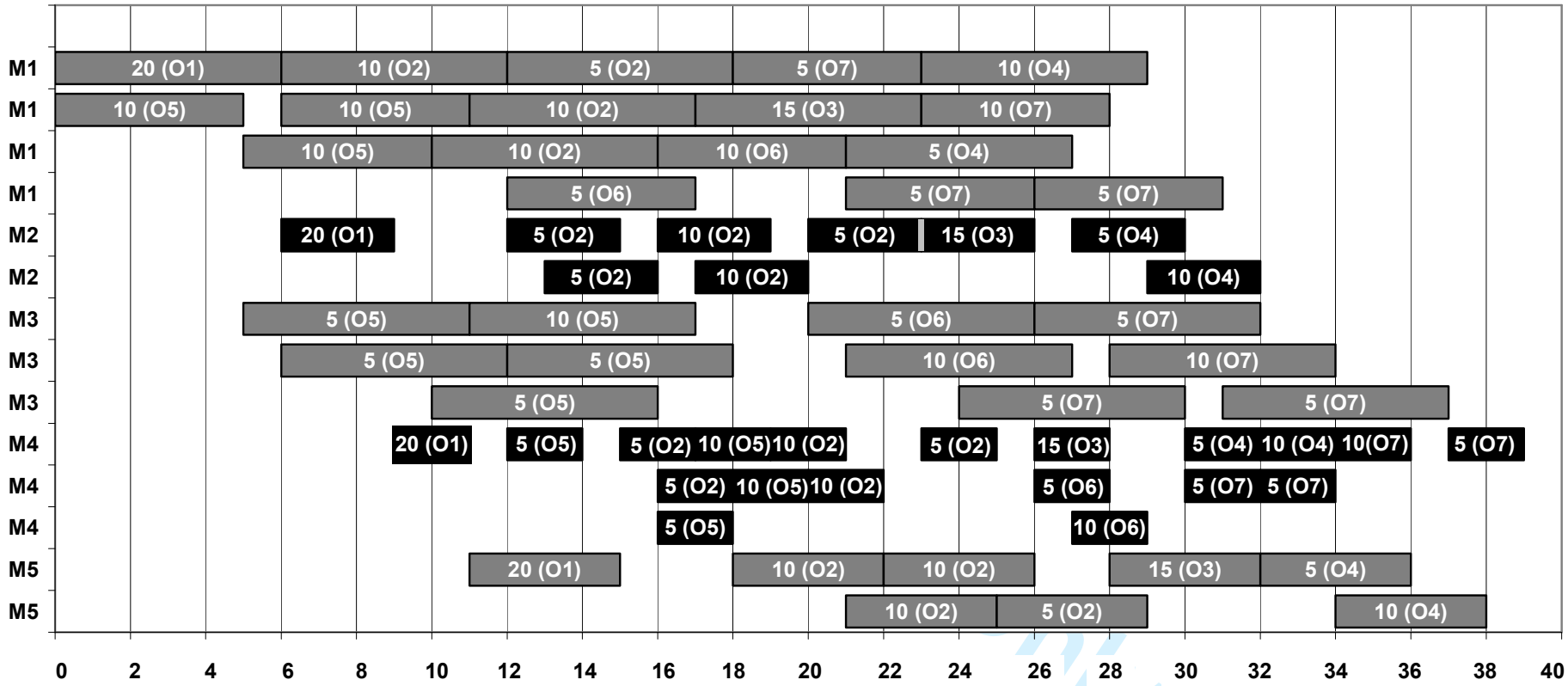

Figure 11. Scheduling of old orders in example 1

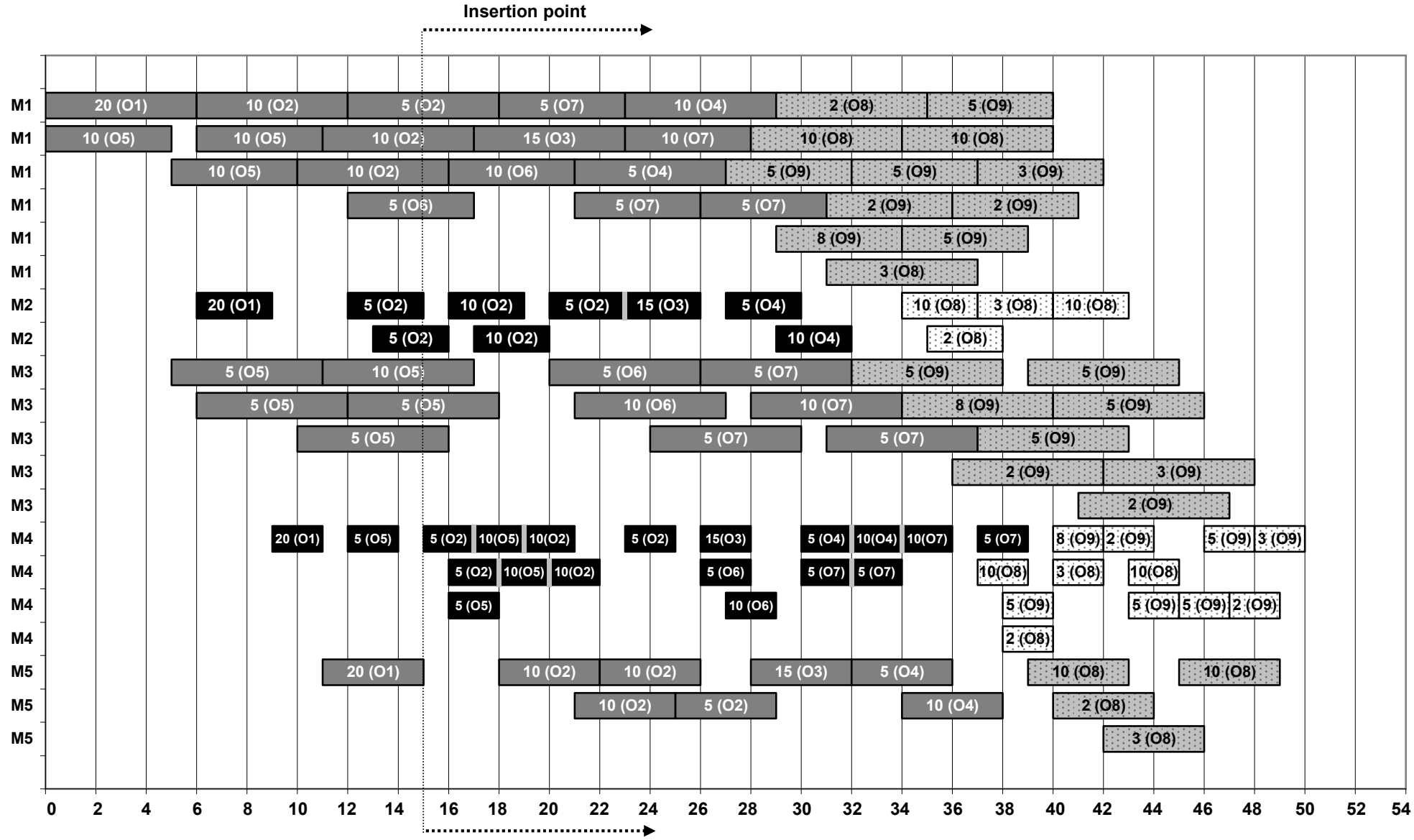

Figure 12. Insertion of new orders without rescheduling of the old ones in example 1

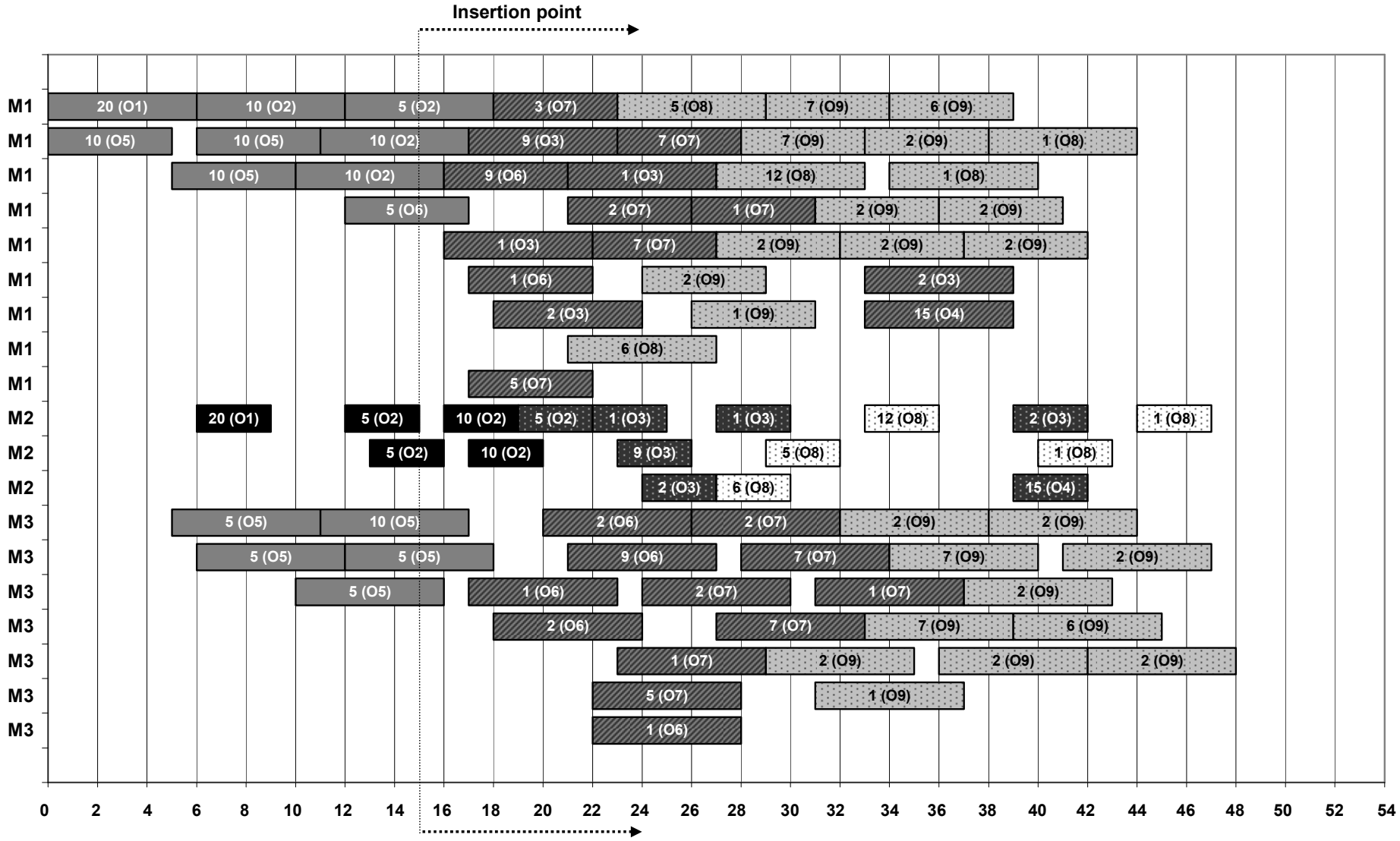

Figure 13A. Insertion of new orders with full rescheduling of the old ones in example 1

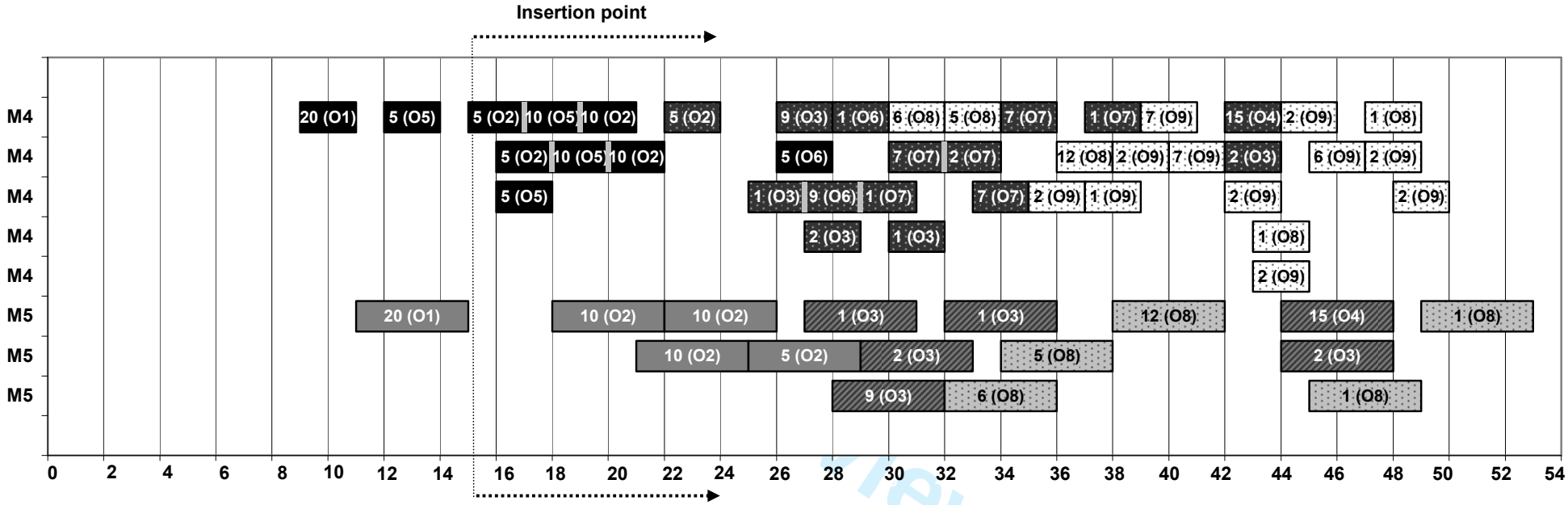

Figure 13B. Insertion of new orders with full rescheduling of the old ones in example 1 (continued)

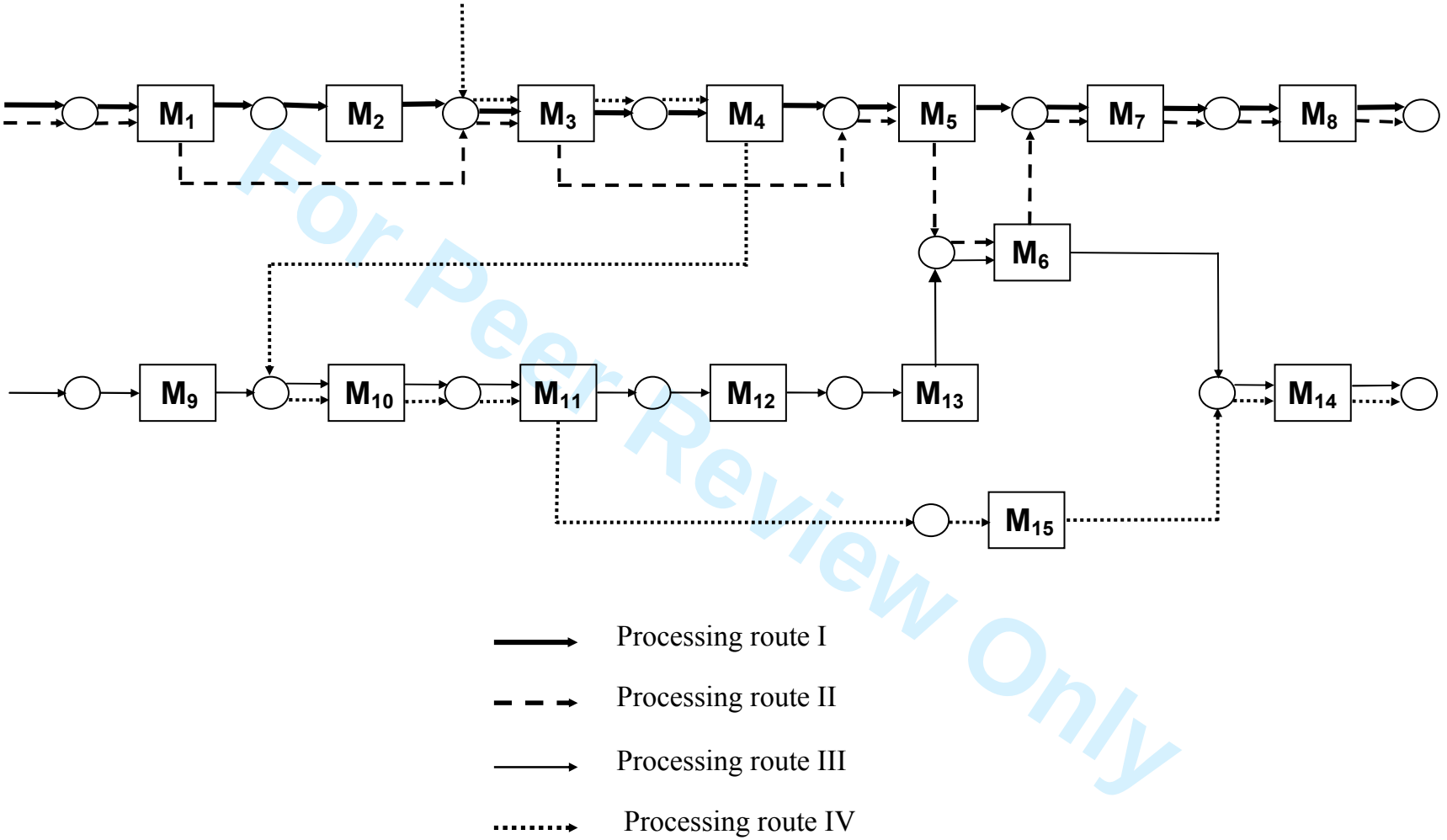

Figure 14. Processing routes in example 2

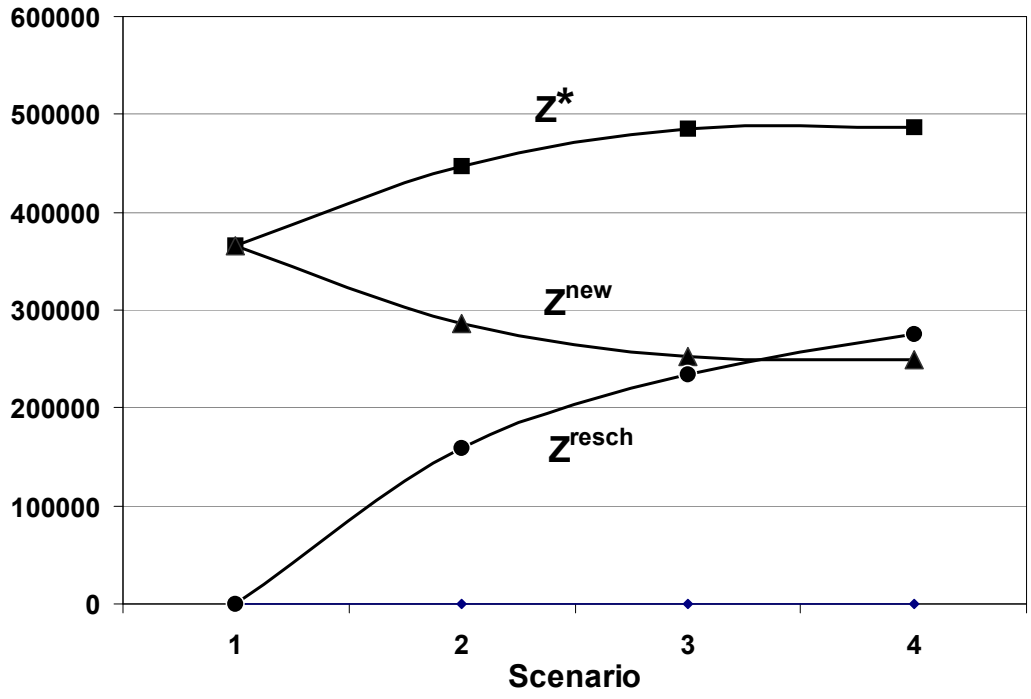

Figure 15. Division of the objective function in example 2