Marcin KLIMEK*, Piotr ŁEBKOWSKI**

PREDICTIVE SCHEDULING WITH DETERMINED TERMS OF MILESTONE ACHIEVEMENT

Abstract: Predictive scheduling is a new approach used in production planning. The aim of this approach is to develop robust schedules and solutions that are able to absorb disruptions during the schedule execution. In this paper, we present algorithms of predictive scheduling for the Resource-Constrained Project Scheduling Problem (RCPSP), with defined dates of selected activities.

Keywords: predictive scheduling, project scheduling problem, robust scheduling.

1. Introduction

During the execution of production orders, workers often determine the terms of completion of particular production stages. The customers, however, agree with the contractor on the dates of checking the progress of works, known as the milestones. Consequently, the customer is assured of the timely completion of the given project. In order to keep the agreed dates, the contractors must take into account the interference that may occur during the production processes, e.g. machine failures, mistakes made in activity duration estimation etc.

The approach in which we take into account the correctness of the task execution during task sequencing is called either predictive scheduling or proactive scheduling. Under the predictive approach, robust schedules are developed, based on the anticipation of possible production disruptions (Herroelen *et al.* 2004). Despite making the sequence robust, some unexpected interference may occur, causing a default in the agreed terms of milestone attainment. In such cases, it is necessary to apply the reactive scheduling process to reduce the costs of instability.

In this paper, we are describing the problem of predictive production scheduling, with limited accessibility to the resources (RCPSP) and the assumption of

* State School of Higher Vocational Education, Biała Podlaska, Poland
** AGH University of Science and Technology, Krakow, Poland

timely execution of the project milestones. We have defined the objective function and proposed suitable algorithms for particular scheduling stages.

2. Project execution stages

In dynamic production systems, project execution is composed of several stages, presented in Figure 1 (van de Vonder 2006):

– *Planning:* determination of tasks, their durations and resource consumption; definition of sequence relationships between the activities, definition of the terms of completion of the whole project and of its particular stages, or of the milestones which determine the schedule of activities developed during the scheduling process.
– *Scheduling:* definition of the commencement and completion times of the activities that make up the given project, with the allocation of specific resources to each activity; at this stage, predictive scheduling is carried out, both nominal and robust.
– *Execution and Control:* monitoring of primarily timely implementation of the agreed project stages, with concurrent preservation of the predictive schedule stability; at this stage, reactive scheduling is carried out, in response to any production interference.
– *Evaluation:* after the project completion, the sequence assessment is carried out, e.g. by ex-post comparison with the optimum schedule.

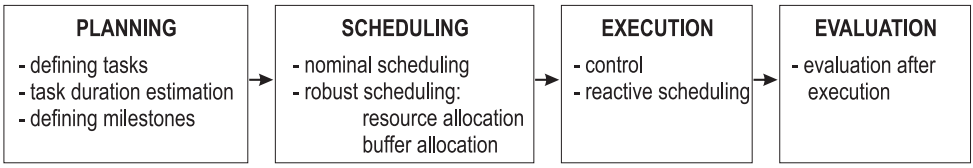

Fig. 1. *Subsequent stages of project execution*

The scheduling stage is of special importance. The development of a production plan and its accurate performance allows for staged accounting with the project owners, improves the production organisation, allows for coordination of internal resources of the business, resource flow between the tasks and orders and timing of material supplies (with contacts with the suppliers and the just-in-time deliveries), as well as enhances the procedures of contractor engagement, contract awarding etc.

The above items indicate that predictive scheduling is essential for the production process organisation. At the first stage of the process of nominal scheduling, an initial schedule is developed, taking into account the efficiency criteria, supplemented with the provision of robustness (e.g. by inserting time buffers in vital sequence locations) during robust scheduling.

3. Problem formulation

The project scheduling problem (where by a "project" we understand a collection of interdependent tasks), with limited availability of resources (RCPSP), consists in finding such time vectors for the commencement of particular tasks (operations and activities) that the resource requirements at each moment are not larger than the available quantity of resources (this quantity is constant, regardless of the load experienced in preceding stages), with the fulfilment of properly defined optimisation criteria (Kostrubiec 2003).

The project schedules in the activity networks will be represented by a non-cyclical, simple, directed graph $G(V, E)$, in which V means a set of nodes corresponding to activities and E is a set of arcs which describe the sequential dependences between the activities. The set V is composed of n activities, numbered from 1 to n, in a topological order, i.e. the predecessor has always a lower number than the successor. We add two dummy operations 0 and $n+1$, with zero durations ($d_0 = d_{n+1} = 0$), representing the project start and finish, respectively.

For an RCPSP, with determined terms of completion of some of the tasks, the constraint conditions can be defined as follows (van de Vonder *et al*. 2005):

– The finish-start and zero-lag precedence relationships occur between the activities: the subsequent operation may start immediately after the completion of the previous one (precedence constraints):

$$\forall (i,j) \in E : s_i + d_i \leqslant s_j \tag{1}$$

– In each moment of time t, the resource consumption does not exceed the available quantities (resource constraints):

$$\forall t \; \forall k \sum_{i \in S_i} r_{ik} \leqslant a_k \tag{2}$$

– For some activities, deadline time of completion is determined (time constraints related to the milestones):

$$z_i \leqslant \delta_i \tag{3}$$

where:

s_i – the time of starting the operation i (decision variable),
d_i – time of the operation i completion,
a_k – quantity of available resources of the type k,
$S - t$ – set of activities executing in time interval $[t-1, t]$,
r_{ik} – the activity i requirement for the type k resource,
z_i – planned time of completion of the activity i,
δ_i – deadline time of the operation i completion, determined exclusively for the activities related to the milestones, that concerns in particular the project completion time δ_{n+1}.

The model determined by equations (1)–(3) uses the milestone system, or the system of critical points that are decisive for the project completion. A timely completion of subsequent stages, or reaching the milestones, will ensure meeting the deadline of work completion. However, the delays usually cause penalties or liquidated damages, while timely project completion may be associated with the payment for the given project stage.

Subsequent critical points related to the tasks whose completion is attached to a specific date ($\delta_j \neq 0$) are marked with km_i. For each i, let the set KM_i contains all the activities whose completion is required for the execution of the given milestone km_i. The total time necessary for the completion of all the activities tkm_i from particular sets of the type KM_i may be determined by formula (4):

$$tkm_i = \sum_{j \in KM_i} d_j \tag{4}$$

Let pb_i mean the level of protection for a km_i, calculated according to formula (5):

$$pb_i = \frac{rez_i + \sum\limits_{j \in KM_i} FS_j}{tkm_i} \tag{5}$$

where:

rez_i – the difference between the deadline time of completion δ_j (determined for the km_i) and the earliest possible time of completion of all the tasks belonging to the set KM_i,

FS_i – time margin after the activity j.

The protection level pb_i thus defined will be used in the objective function for predictive scheduling.

4. Nominal scheduling

4.1. Nominal scheduling for *RCPSP* with milestones

At the nominal scheduling stage, the initial schedule is developed. The schedule can be either imposed after making calculation in a commercial software package, or determined with use of the enterprise's own calculation procedure. Nominal sequencing does not contain any buffers; it is not resistant to possible production disruptions; and it does not take into account the changeability and uncertainty of the production system parameters.

The nominal schedule is most often determined by the application of the algorithms for a deterministic problem. Classical algorithms for solving an RCPSP problem may not be used in the case under consideration. The milestones system is used here, while the defined objective function, proportional to the time consumption, should ensure protection against possible default in the completion dates of particular project stages, taking into account a proper protection level.

The proposed objective function, taking into consideration the observance of the times of completion of all the project stages, is to maximise a weighted total of the levels of milestone protection pb_i:

$$\max\{\sum_{i=1}^{m} pb_i \cdot wm_i\} \tag{6}$$

where wm_i – weight assigned to the milestone km_i.

The value of the weight wm_i depends on the current protection of the milestone km_i, and it is determined on the basis of the ascending milestone sequence with reference to pb_i.

For example, the value may be determined in the following way:

- for the milestone with a minimum protection level pb_i, we assume: $wm_i = m$;
- for the milestone with the k-th pb_i, $wm_i = m - k$;
- for the milestone with the maximum pb_i, $wm_i = 1$.

The weights wm_i assigned to the milestones can also be determined in a different way; however, the following condition should be fulfilled: larger weights wm_i are assigned to less protected milestones km_i.

Owing to the objective function determined by formula (6), we obtain the following:

- equal distribution of the protection buffers based on the levels pb_i, attained by proper definition of the weights wm_i;
- distribution of the time margin, in proportion to the duration of the milestone tkm_i: the larger the tkm_i value, the larger buffering of the milestone km_i.

In the case of an RCPSP problem, when developing a schedule, we apply the decoding procedures, known as Schedule Generation Schemes (*SGS*), which generate the sequence, based on a priority list (or activity list), taking into account the availability of the resources and the sequence interdependence.

For a deterministic problem, the following are used (van de Vonder 2006):

- *serial SGS*: in each iteration, the first non-sequenced activity from the activity list or the priority list is started at the earliest possible commencement time upon the fulfilment of the sequential and resource constraints;
- *parallel SGS*: iteratively at consecutive moments of the time t (at the decision-making points), all the non-sequenced activities are started; the ones which may be started in the sequence arranged on the activity list or the priority list upon the fulfilment of the sequential and resource constraints.

Below we present new algorithms that use the *SGS* procedures when generating a nominal sequence for the defined *RCPSP* problem, with timely achievement of the milestones.

4.2. Priority-list scheduling

In priority-list scheduling, a priority list of activities is created and used to generate the sequence, with the application of schedule generation schemes.

Algorithm 1

Step 1. Calculation of the $zapas_i$: the difference between the deadline for completion δ_j of the closest milestone for the activity i and its earliest possible time of completion, taking into account exclusively duration times of the tasks that precede the activity in the activity network.

Step 2. Calculation of the priority of each activity execution based on formula (7).

$$pr_i = \delta_j \cdot zapas_i \tag{7}$$

Step 3. Creation of the activity priority list, in accordance with the ascending pr_i values (with regarding precedence relationships). Based on that list, the schedule will be determined using *SGS*.

Algorithm 2

Step 1. Determination of the milestone attainment sequence based on the δ_i value; the auxiliary list $LKM = \{km_2, km_3, km_1, \ldots, km_l$ (project completion time) $\}$ is created.

Step 2. Determination of the sequence on the priority list in the sets of subsequent milestones KM_i, e.g. applying either the priorities which are characteristic to the deterministic problem RCPSP, i.e. the Latest Starting Time (LST) list: sorting of the activities in the non-descending order of their latest possible starting times (the best rule for the deterministic RCPSP problem), or the priority rules taking into account the cost of instability (related to the resource consumption of the tasks).

Step 3. Generation of the schedule, using the *SGS*, based on the priority list determined in Steps 1 and 2.

4.3. Meta-heuristic algorithms

The application of meta-heuristics, e.g. of Simulated Annealing (*SA*) and Tabu Search (*TS*) may improve the solution obtained with use of scheduling on the basis of a priority list. The algorithms *TS* and *SA*, which search through the activity list, maximise the objective function determined by formula (6).

5. Robust scheduling

5.1. Robust scheduling for an *RCPSP* with milestones

Robust scheduling is intended to create a sequence which, due to its properties, is resistant to disruptions that may appear in the production process (Al-Fawzan *et al*. 2005).

Two approaches are distinguished here (Kobylański *et al*. 2007):

- schedule quality robustness, where we intend to fulfill the efficiency criteria;
- solution robustness of the whole schedule, where we intend to implement all the sequencing details, in accordance with the plan.

For the defined problem of timely milestone attainment, the schedule quality robustness is reduced to the protection supporting the maintenance of the agreed terms of completion of particular project stages. The objective function, determined by formula (6), is a proper measure of the schedule quality robustness.

In the solution robustness approach, we intend to implement all the sequencing details, in accordance with the plan.

When determining the robustness measures, we make the following assumptions:

- the more resources are involved in the activity, the more susceptible is the activity to disruptions;
- the longer the activity, the larger possibility of its prolongation since longer tasks indicated a larger absolute changeability;
- the disruption of the activity start time in the case of a more resource–consuming activity causes a larger instability cost (or disorganisation of larger quantities of resources).

The proposed measure of the schedule robustness has the form of equation (8).

$$M = \left(\sum_{i=1}^{n} (\sum_{j=1}^{k} r_{ij} \cdot d_i) \cdot \sum_{j=1}^{FS_i} a^{-j} - \sum_{i=1}^{n+1} f_i \cdot \max(0, z_i - \delta_i) \right) \tag{8}$$

where:

a – the parameter affecting the buffer allocation ($a > 1$),
f_i – the marginal cost of exceeding the latest term of completion of the activity i, determined for the activities related to the milestones only,
FS_i – free slack, time buffer for the activity i.

The application of the robustness maximisation measure M as the objective function of robust scheduling leads to a uniform distribution of the time buffers in the schedule, with special protection of the starting time of the activities that are most resource and time consuming (the larger the value of the parameter a,

the more uniform the buffer distribution). Also the dates of completion of the milestones are taken into account by determining the f_i – liquidated damages assigned to the time unit related to the delay of the completion of the activity i.

Robust scheduling is composed of the robust resource allocation and of robust buffer allocation:

– Robust resource allocation: proper assignment of the resources for the performance of particular activities, with view to developing the schedule which is highly resistant to disruptions. The algorithms are most often reduced to the minimisation of the number of additional sequential interdependencies resulting from resource allocation, e.g. *ISH*, *ISH2*, *MABO* (van de Vonder 2006).

– Robust buffer allocation: placement of time buffers before the tasks to make the schedule robust or resistant e.g. to a periodic unavailability of resources or changeability of activity duration. Buffer allocation takes place upon determination of the assignment of resources to tasks.

What is especially essential in scheduling is proper buffer allocation. The time buffers should be entered at vital locations of the nominal schedule that are the most exposed to interference or charged with the largest instability cost. We have below proposed our algorithms for robust buffer allocation applicable to the problem under consideration and to the objective function determined by formula (8), taking into account the schedule robustness quality measure, formula (6).

5.2. Algorithms of Robust Buffer Allocation

Below we present three algorithms (Algorithms 3, 4 and 5) of robust scheduling, proposed for the problem RCPSP defined, with determined terms of milestone achievement.

Algorithm 3

This algorithm inserts the time buffers before particular activities, in the sequence of susceptibility to interference, taking into account the weight of sequential interdependencies in the resource flow network. The buffer placement method also depends on the duration of particular milestones tkm_i. The schedule is buffered in a way enabling as proportional as possible protection of timely completion of all the project stages.

In this algorithm, the coefficient wb_i (formula 10) is calculated. The coefficient is used for determining the sequence of entering the buffers before the activities. The determination of the coefficient wb_i is possible after an analysis of the planned times of commencement and duration of all the tasks that precede the activity i in the resource flow network. We calculate the adjusted time of completion of all the activities j from the set P_i^* – the set of the activities which precede the activity i, directly or indirectly. The adjustment Δd_j of the task j duration is calculated according to formula (9):

$$\Delta d_j = \alpha \cdot d_j \cdot r_j \tag{9}$$

$$wb_i = \frac{\sum_{j \in P_j^*} r_i \cdot (s_j + d_j + \Delta d_j - s_i)}{b_i + 1} \tag{10}$$

where:

b_i – time buffer before the activity *i*,
α – the parameter which can take into account the nature of a given process, i.e. the machine failures, or the coefficient of the actual activity duration.

The algorithm operation consists in the addition of unit buffers during a single loop course until the whole schedule is completed. Before each course is carried out, modified coefficients pb_i are calculated, according to formula (11).

$$pb_i' = \frac{rez_i}{tkm_i} \tag{11}$$

A buffer will be inserted for the milestone with the minimum pb_i. To find the buffered task, the wb_i values are calculated for the activities belonging to the set KM_i. Next, the insertion of the unit buffer before the activity representing the largest wb_i is checked. If the schedule performance time with that buffer does not exceed the times of completion of the key activities, the buffer is be inserted, the schedule is modified and the wb_i and pb_i values are calculated for a new schedule. A new calculation cycle is carried out with the updated wb_i values.

However, when the buffer insertion exceeds the time of milestone attainment, the activity is be added to the list of the tasks which are not buffered. If it is not possible to insert a buffer for a given milestone km_i, the milestone will be omitted in the subsequent runs of the algorithm, that is all the activities belonging to the KM_i will remain unprotected. The course of the algorithm is continued until either all the activities stop to be buffered or the coefficient wb_i assumes the zero value.

Algorithm 4

This algorithm inserts buffers after the activities, protecting a timely commencement of succeeding activities. In the first place, the longest tasks are buffered (under the assumption that longer tasks are characterised by a larger absolute changeability). In each iteration, a new schedule is generated based on the modified (increased) duration times. The duration changes are staged. At each stage, the activity duration time is changed by the maximum dk_i of one unit, calculated according to formula (12), provided that the dk_i is at least equal to 1.

$$dk_i = \beta \cdot d_i - b_i \tag{12}$$

where β – the parameter, an iteration step.

Sequencing is evaluated at each step with use of the objective function determined by formula (8). At subsequent steps, the best solution is recorded. The algorithm will finish its operation, when the activities cannot be buffered any more due to the exceeded time of any of the pre-determined milestone, taking into account the modified task durations.

Algorithm 5

This algorithm is based on the Critical Chain and Buffer Management Method *CC/BM* (Goldratt 1997) related to the Theory of Constraints (*TOC*). The critical chain is defined as a set of the activities that determine the total time of the project completion, taking into consideration the sequential relationships and the resource constraints. In case of unlimited resources, the critical chain definition is equivalent to the critical path definition. In the *CC/BM* method, instead of adding the protection margins to particular tasks, common buffers are created. They are inserted in the strategic places of the project; however, the time of the whole project completion only must be observed, and not the times of completion of individual tasks.

In the buffer management method, it is proposed to insert an additional time margin at the end of the project critical chain (this is known as Project Buffer (*PB*)), as well as additional buffers for the activities outside the chain, known as Feeding Buffers (*FB*). The insertion of additional buffers at the locations where the activities outside the critical chain are connected to the chain provides protection against interference in the course of the critical activity execution. Due to the fact that various resources may be required for the execution of particular processes belonging to the critical chain, the availability of such resources also outside the planned times is the necessary condition for the continuity of the process performance. Signalling of earlier requests for critical resources is possible due to the Resource Buffers (*RB*) that are available in advance with respect to the critical-chain activities.

In the case of the problem with defined milestones, for each checkpoint km_i, critical chains are calculated. Next, for each km_i, the size of the protection margin rez_i (equivalent to the *PB*) is determined. The margin can be calculated as a certain fraction k of the duration of all the activities from the KM_i. The buffering level k is determined on the basis of schedule analysis. After determination of the milestone buffer sizes, *FB*'s and *RB*'s are determined for particular critical chains. A drawback of that approach is the provision of robustness for the schedule quality only. The solution obtained during nominal scheduling is adjusted, while further milestone attainment time protection is reduced to the protection of the tasks belonging to the critical chains.

6. Conclusions

In this paper, we have proposed new algorithms for the problem of predictive production scheduling, with the Resource-Constrained Project Scheduling Problem (RCPSP). Those algorithms are adjusted for making robust the activity sequences characterised by specific deadline times of completion of particular tasks related to milestones. The calculation procedures presented here provide protections against exceeding the milestone completion times, although they also make the schedule robust at the locations that are most exposed to disruptions. Also, the activities whose possible delays would cause the largest instability costs are better protected. The algorithm operations are comparable owing to the defined robustness measures: the predictive scheduling objective functions.

Presently, our work concentrates on testing of the algorithms described here. The preliminary research results indicate the correctness of our procedures and the increased robustness of our schedules.

References

Al-Fawzan M., Haouari M. 2005. *A bi-objective problem for robust resource-constrained project scheduling.* International Journal of Production Economics, Vol. 96, pp. 175–187.

Goldratt E.M. 1997. *Critical chain.* North River Press, Great Barrington.

Herroelen W., Leus R. 2004. *Robust and reactive project scheduling: a review and classification of procedures.* International Journal of Production Research, Vol. 42(8), pp. 1599–1620.

Kobylański P., Kuchta D. 2007. *A note on the paper by M.A. Al-Fawzan and M. Haouari about a bi-objective problem for robust resource-constrained project scheduling.* International Journal of Production Economics, Vol. 107, pp. 496–501.

Kostrubiec A. 2003. *Harmonogramowanie projektów – przegląd modeli.* Inżynieria Zarządzania Przedsięwzięciami, Wydawnictwo Politechniki Gdańskiej, pp. 33–52.

Vonder S. van de 2006. *Proactive-reactive procedures for robust project scheduling.* Katholieke Universiteit Leuven. (Ph.D. thesis).

Vonder S. van de, Demeulemeester E., Herroelen W. 2005. *Heuristic procedures for generating stable project baseline schedules.* [in:] Proceedings of the Third Euro Conference for Young OR researchers and practitioners (ORP3), Valencia, pp. 11–19.