Decision Making in Manufacturing and Services
Vol. 3 • 2009 • No. 1–2 • pp. 49–72

Robust Buffer Allocation for Scheduling of a Project with Predefined Milestones

Marcin Klimek[*], Piotr Łebkowski[**]

Abstract. The paper discusses the problem of robust buffer allocation for Resource-Constrained Project Scheduling Problem (RCPSP) with predefined milestones[1] , for which execution deadlines have been established. To solve the problem, an algorithm is proposed supporting insertion of unit time buffers, with the simultaneous maximisation of new metrics of arrangement robustness. The presented results of experimental research speak for usability of the solutions proposed. The effectiveness is studied with use of test tasks[2] included in the Project Scheduling Problem Library (PSPLIB) with additionally specified project milestones.

Keywords: resource-constrained project scheduling, predictive-reactive scheduling, robust buffer allocation, milestones

Mathematics Subject Classification: 90B35, 90C59

Revised: 19 December 2009

1. INTRODUCTION

The application of project management in production planning has been growing. Make-to-Stock (MTS) processes have been displaced by Make-to-Order (MTO) processes. The Make-to-Order production is primarily used in the production of non-standard products, customised to the ever-changing and sometimes unpredictable requirements of the customer. Each such production order should be treated as a separate project developed in consultation with the customer. The areas in which such projects are implemented include research and development (R&D), public work, construction, IT etc.

[*] Pope John Paul II State School of Higher Vocational Education in Biala Podlaska, Poland. E-mail: marcin_kli@interia.pl

[**] Department of Operations Research and Information Technology, AGH University of Science and Technology, Krakow, Poland. E-mail: plebkows@zarz.agh.edu.pl

[1] The expressions "milestone", "phase of a project" and "project phase" are used interchangeably throughout the paper.

[2] The expressions "task", "activity", "operation" and, occasionally, "job" are used interchangeably throughout the paper.

While implementing a project providing for production as per a customer's order, uncertainty arises connected, for instance, with changing requirements, difficulty in estimating task durations etc. In recent years, the number has been steadily growing of studies including the uncertainty connected with the dynamics of production systems in the changing environment. In numerous survey papers (Aytug *et al.*, 2005; Herroelen and Leus, 2004; Klimek and Łebkowski, 2007; Van De Vonder, 2006; Vieira *et al.*, 2003), the authors have attempted to structure the research in project scheduling under uncertain conditions.

Project scheduling most often includes an analysis of uncertainty pertaining to the activity duration time (Herroelen and Leus, 2004; Kobylański and Kuchta, 2007; Lambrechts *et al.*, 2006; Leus, 2003; Van de Vonder *et al.*, 2005; Van de Vonder *et al.*, 2006; Van De Vonder, 2006). It is assumed that numerous other disturbances (including adverse weather conditions, resource unavailability, irregularities in material supplies etc.) may affect task durations. The problem then reduces to minimising the effect of possible changes on task durations. This approach is also applied in this paper.

Two methods are used in project scheduling under uncertain conditions (Vieira *et al.*, 2003):

1) dynamic scheduling and
2) predictive-reactive scheduling.

In highly changing environments, the planning phase is omitted and dynamic scheduling is used. No schedule is being developed in the dynamic task arrangement, also known as online scheduling. The allocation of tasks to appropriate resources (machinery) is carried out as soon as resources of a given type (a machine) are ready to process a new task. All information available at the time is used. Most often, the task processing queue is being created based on a specified criterion (dispatching rules). Priorities are attached to tasks waiting for processing and the task with the top priority at a given time is allocated to the resources. The criteria used include Shortest Processing Time (SPT) and Earliest Due Date (EDD).

Where the number of disturbances is not large, predictive-reactive scheduling guarantees better results. The method comprises two, often separate phases:

1) predictive scheduling phase, connected with the production planning phase, and
2) reactive scheduling phase, connected with the production plan execution phase.

In actual systems, both phases should be used jointly (Vieira *et al.*, 2003).
In the production planning phase, a predictive schedule, also known as a baseline schedule, is developed. Because task arrangement is determined before the production process starts, this phase is also known as offline scheduling. The development of a predictive schedule prior to the project execution plays an important role in production planning. Its advantages include the possibility of schedule-driven production control and the possibility of monitoring the execution timeliness for individual project phases.

The plan having been developed, in the course of the production process in actual production systems, it is necessary to monitor the production process in order to react to events disturbing the predictive schedule. In this phase, known as reactive scheduling, changes are introduced in the task arrangement enabling the problems arising in the baseline schedule (e.g., connected with machine failures) to be solved. The task arrangement is subject to changes on an ongoing basis, during the production process; therefore, this phase is also known as online scheduling. The advantages of schedule development put emphasis on the importance of production system stability, which may be achieved through the implementation of a specified plan included in the schedule. Accordingly, among the objectives of reactive scheduling, there is schedule stability, that is as precise as possible execution of the predictive schedule. The larger the number of changes in the schedule, the larger the nervousness of those involved in the production, which may disrupt the production process.

In this paper, predictive-reactive scheduling is used. The study focuses on the predictive scheduling phase, which starts with the development of a nominal schedule, to be later made robust during proactive scheduling, also known as robust scheduling. A robust schedule is designed to prevent instabilities in nominal schedules through, for instance, factoring in the knowledge of variability and uncertainty of production system parameters. The development of a proactive schedule includes the insertion of time buffers and resource buffers at key points in the task arrangement. The method facilitates the inclusion of statistical knowledge concerning potential disturbances in the production plan, gathered through the analysis of earlier production runs.

The following techniques are used in proactive scheduling (Van De Vonder, 2006):

– redundancy-based techniques,
– contingent scheduling,
– development of Partial Order Schedules (POS) (Policella *et al.*, 2004),
– arrangement sensitivity analysis (Hall *et al.*, 2004).

In this paper, redundancy-based techniques are used, which reduces to the generation of schedules including time buffers (Van de Vonder *et al.*, 2005; Van de Vonder *et al.*, 2006) or resource buffers (Lambrechts *et al.*, 2006). Time or resource buffers are inserted in order to render the schedule robust to possible disturbances in production processes, with use of the knowledge on the uncertainty of production parameters. Buffer insertion protects the given arrangement against a delay, if any, in the execution of a given task, but it simultaneously increases the total time of execution of all tasks and contributes to the deterioration of schedule quality. A relation does exist between the schedule quality and its robustness (Leus, 2003; Van de Vonder *et al.*, 2006).

For the project scheduling problem, different redundancy-based methods are used to render an arrangement robust, including:

– Critical Chain/Buffer Management (CC/BM) method or Critical Chain Scheduling/Buffer Management (CCS/BM) method (Goldratt, 1997);
– methods of estimating buffer size based on the statistical knowledge for the problem of minimising weighted cost of schedule instability: time buffer allocation algorithms with variable task execution times (Van de Vonder, et al., 2005; Van de Vonder *et al.*, 2006; Van De Vonder, 2006), time or/and resource buffer allocation algorithms with variable resource availability (Lambrechts *et al.*, 2006).

Basic information on the methods referred to above follows.

The CC/BM method is a frequently used approach to project scheduling. The idea of critical chain, proposed by Goldratt (1997) is based on the Theory Of Constraints (TOC). It has commonly been applied in the area of project management, both in theory and in practice (e.g., in the management of IT or construction projects). A new project management philosophy emerged, known as Critical Chain Project Management (CCPM), which has in practice displaced the project management methodology based on such techniques as Performance Evaluation Review Technique/Critical Path Method (PERT/CPM).

The CCPM methodology introduces the idea of a critical chain, defined as the set of activities determining the total project execution time, with the arrangement relations and resource-related constraints factored in the determination. With unlimited resources, the definition of a critical chain coincides with that of a critical path. The critical chain method supports a new way to manage a project. No milestones are used. Instead of security margins for individual tasks, common buffers are inserted at strategic points along the project execution path. It is the protection of the deadline for the entire project execution that becomes particularly important, and not timely execution of individual tasks. The additional time buffer is introduced in the first place, at the end of the project critical chain; this buffer is referred to as the Project Buffer (PB). The introduction of project buffers supports the protection of timely project execution.

For activities out of the critical chain, feeding buffers (FB) are introduced. In the PERT/CPM methodology, for tasks out of the critical path, a time play is introduced, which – according to Parkinson's Law – is wasted. In the CCPM methodology, those time reserves are removed from all tasks and retained in the form of feeding buffers. Such buffers are inserted at the points where a non-critical chain connects to the critical chain, thus protecting the progress of critical activities against disturbances. A feeding buffer is a common time buffer for an entire group of tasks comprising a non-critical chain.

Earlier availability (availability in advance) of resources for critical activities is achieved through the introduction of Resource Buffers (RB). Resource buffers are inserted so as to guarantee the availability of resources for tasks included in the critical chain in appropriate quantities and on appropriate times.

An advantage of the CC/BM method is that it supports easy monitoring of the project execution progress through tracking the current use ratio of the critical chain buffer (project buffer). This procedure is known as the Buffer Management. The CC/BM method focuses on the protection of tasks along the critical chain. Tasks out of the critical chain are not buffered. Non-critical tasks are executed at appropriate times. Non-critical activities of the project are commenced on the ALAP (As Late As Possible) basis, with time provided for buffers. The ALAP approach shifts capital expenditure in time, thus increasing the project Net Present value (NPV).

The CC/PM method has primarily been developed for projects executed by humans. Its objective is to mitigate the human-related risks of untimely project execution. There exist production environments in which the stability of execution of all elements of production plan is of importance, given, for instance, the requirement to deliver materials exactly at the task commencement time.

Therefore, it is reasonable to develop buffer arrangement algorithms which would guarantee production stability. The critical chain method is not applied to the problem considered also because of the scheduling objective set in the paper: in addition to timely execution of the entire project, meeting other contractual deadlines for individual project phases is also important.

The CC/BM method is not suitable in production systems where not only the timely execution of the entire project, but also schedule stability is important. Algorithms are developed designed to estimate sizes of buffers protecting both the schedule and the timely project completion. In project scheduling, the most often analysed problem is uncertainty related to activity durations. The algorithms for buffer size definition include:

– algorithms using information on Fixed Relative Deviation (FRD) (Herroelen and Leus, 2004; Leus, 2003; Van de Vonder *et al.*, 2005; Van de Vonder *et al.*, 2006),
– algorithms using information on weights attached to activities and arrangement-related relations among activities: Resource Flow Dependent Float Factor (RFDFF) and RFDFF+ (Herroelen and Leus, 2004; Leus, 2003; Van de Vonder *et al.*, 2005; Van de Vonder *et al.*, 2006),
– algorithms using information on weights attached to activities and knowledge of activity duration variability: Starting Time Critically (STC) and STC+ (Herroelen and Leus, 2004; Leus, 2003; Van de Vonder *et al.*, 2005; Van de Vonder *et al.*, 2006),
– integer programming algorithms which arrange time buffers evenly, thus maximising – for all tasks – free slack (FS) (Al-Fawzan and M. Haouari, 2005) or minimum ratio of FS to activity duration (Kobylański and Kuchta, 2007).

In this paper, a new RCPSP model is proposed, with predefined project phases (milestones), for which execution deadlines are defined. For the RCPSP problem with milestones, robustness metrics are developed and an algorithm of robust buffer allocation is proposed; an algorithm which insets unit time buffers so as to maximise arrangement robustness. The algorithm has been tested with use of test tasks which were sourced from the PSPLIB library and for which milestones were additionally defined.

2. STATING OF THE PROBLEM

A project is a set of inter-related tasks (operations, activities) executed with use of resources in order to achieve set objectives. Table 1 presents symbols standing for parameters and variables used in defining the project scheduling problem with predefined milestones.

Networks (graphs) are used to graphically represent project scheduling problems. For the considered RCPSP problem, the AON network, known also as the operation network, is used, suitable for scheduling problems including time optimisation criterion. With use of the operation network, a project is represented as an acyclic, connected, simple, oriented graph $G(V, E)$. The tasks are numbered from 1 to n in such a way that the predecessor's number is always less than the successor's.

Table 1. *Parameters and variables – symbols*

$G(V,\ E)$	–	graph describing a project in the AON (Activity-On-Node) representation,
V	–	set of vertices corresponding to tasks,
E	–	set of edges describing the arrangement relations between tasks,
s_i	–	start time of the activity i (decision variable),
d_i	–	duration of the activity i,
a_k	–	number of available resource of the type k,
r_{ik}	–	demand of the activity i for the type k resource,
z_i	–	planned completion time for the activity i,
δ_i	–	deadline for the activity i; precisely – the deadline for the execution of the earliest project phase during which the activity has to be executed,
tm_i	–	deadline for the execution of the milestone i,
m	–	number of milestones ($n > m > 0$).

The scheduling problem for a resource-constrained project consists in finding the task start (or end) time vector for an accepted optimisation criterion. Additionally, the following constraints are defined:

– no-delay end-start relations between operations occur - the next operation can start immediately upon the end of the previous one having ended (arrangement constraints):

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

– at any time t, the use of resources by operations does not exceed the available quantities (resource constraints), resources (e.g., workforce of machinery) are renewable, that is the quantity of given resource is constant irrespective of their respective loads during earlier periods:

$$\sum_{i \in S_t} r_{ik} \leqslant a_k \qquad \forall t, \forall k \tag{2}$$

The most common optimisation criterion used for the problems considered is the minimisation of the total execution time for the entire project. Given the uncertainty encountered in the course of project execution, the authors suggest that milestones (perhaps contractually specified project phases defined by the project's principal) should be defined, which would reduce the risk of unsuccessful or untimely execution of the project. Predefined milestones enable the customer (principal) to monitor the progress of project execution. Any delay in the execution of individual milestones results in a contractual penalty being imposed. On the other hand, timely execution of a given milestone may trigger the payment of consideration for the completion of a given project phase.

Milestones are defined by way of setting time constraints for individual activities. For each task, a deadline is defined connected with the deadline for the closest project phase during which a given task is to be performed (3).

$$z_i < \delta \tag{3}$$

Figure 1 presents an example of an AON-type network with defined times of monitoring the execution of project phases.

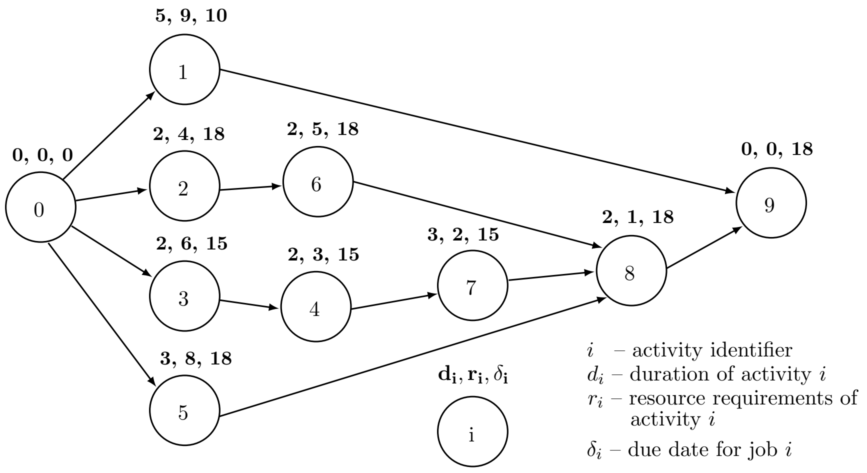
5, 9, 10
1
0, 0, 0
0
2, 4, 18
2
2, 5, 18
6
2, 6, 15
3
2, 3, 15
4
3, 2, 15
7
3, 8, 18
5
2, 1, 18
8
0, 0, 18
9
$\mathbf{d_i}, \mathbf{r_i}, \delta_\mathbf{i}$
i
i – activity identifier
d_i – duration of activity i
r_i – resource requirements of activity i
δ_i – due date for job i

Fig. 1. *An example of an AON-type network for a project with single resource and predefined deadlines for some tasks*

Subsequent phases of a project (milestones) are connected with tasks which have the same execution time defined ($\delta_j \neq 0$). Let denote the set of tasks directly connected with the i-th milestone by M_i. M_i includes all activities with the same deadline δ_j:

$$M_i = \{j : \delta_j = tm_i, j \in V\} \tag{4}$$

The sets M_i are pairwise disjoint, that is each activity lies in one of the sets M_i only:

$$M_i \cap M_j = \phi, \qquad \forall i \neq j, \qquad i, j = 1, 2, ..., m \tag{5}$$

The deadlines tm_i for the arranged project milestones are determined based on the predefined task deadlines δ_j in such a way that the sequence is strictly increasing, that is the following inequalities hold:

$$tm_i < tm_{i+1}, \qquad \forall i \in < 1, m) \tag{6}$$

Let KM_i denote the set of all activities upon the performance of which the performance of the milestone km_i is conditional:

$$KM_i = \{j : j \in M_i \vee (j \in P_k, k \in M_i)\} \tag{7}$$

where P_k stands for the set of all (direct or indirect) predecessors of activity k.

The schedule looked for should take into consideration deadlines for individual milestones. For the project illustrated in Fig. 1, the minimum total execution time for the entire project is 12. The requirement of timely execution of all project phases lengthens the total project execution time. Job 1 should be completed by the absolute deadline ($\delta_1 = 10$). Given resources availability requirement, during the execution of Job 1, no other activity may be performed, with the exception of Job 8, which – in turn – may start no sooner than at $t = 7$. Thus, the minimum project execution time is the aggregate of the duration of Job 1 and duration of Jobs 3, 4, 7 and 8 (along the critical path) and equals 14 ($t_1 + t_3 + t_4 + t_7 + t_8 = 5 + 2 + 2 + 3 + 2 = 14$). An example of a schedule for the execution time 14 is presented in Fig. 2.

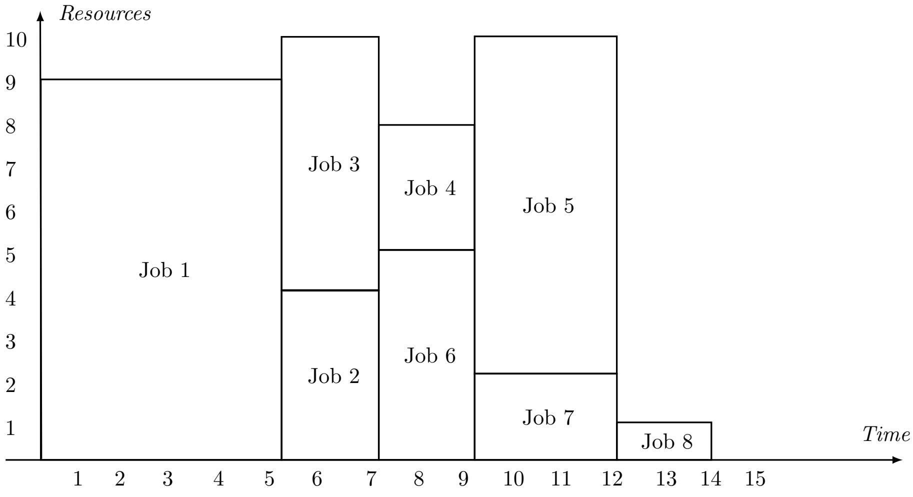

Fig. 2. *Schedule taking into consideration deadlines for individual activities*

Minimisation of the project duration and other objective functions commonly used for the RCPSP problem are not suitable objective functions for the considered problem of timely execution of all phases of a project. It is necessary to define a new objective function. Such a definition is proposed below as formula (9).

The protection (security) level for timely execution of all phases of a project has a decisive effect on the quality of a schedule. Let pb_i denote the protection level for the timely execution of the i-th phase of a project. Then the security level may be computed according to formula (8):

$$pb_i = \frac{rez_i}{\sum\limits_{j \in KM_i} d_j} \tag{8}$$

where: rez_i stands for the difference between the absolute completion deadline δ_j (defined for the i-th milestone) and the earliest possible completion time for all tasks comprising the set KM_i.
For the project illustrated in Fig. 1, we thus obtain:

$tm_1 = 10, M_1 = \{1\}$, $KM_1 = \{1\}$, $tkm_1 = 5$, $rez_1 = 5, pb_1 = 1$
$tm_2 = 15, M_2 = \{3,4,7\}$, $KM_2 = \{3,4,7\}$, $tkm_2 = 7$, $rez_2 = 3, pb_2 = 3/7$
$tm_3 = 18, M_3 = \{2,5,6,8,9\}$, $KM_3 = \{1,2,3,4,5,6,7,8,9\}$, $tkm_3 = 21$, $rez_3 = 4, pb_3 = 4/21$

The authors-proposed objective function for nominal scheduling, taking into consideration the protection of timely execution of all phases of a project, is maximisation F_n defined as a weighted average of time buffers rez_i, as in formula (9).

$$F_n = \sum_{i=1}^{m} rez_i \cdot wm_i \tag{9}$$

where wm_i stands for the weight ascribed to the i-th phase of a project.

The value of the weight wm_i depends on the current (at any given time) level of protection of the i-th milestone and is determined based on the list of milestones sorted in the increasing order. Weights are ascribed in such a way that the less the level of milestone protection, the larger the weight ascribed to that milestone. In this paper, milestones are arranged in the descending order of levels pb_i (multiple milestones with the same level pb_i are arranged in the ascending order of the respective numbers of project phases); then the weights are computed as follows:

- for the milestone with the minimum protection level $pb_i : wm_i = m_2$;
- for the milestone with the k-th pb_i (according to the order assumed): $wm_i = (m-k)2$;
- for the milestone with the maximum protection level $pb_i : wm_i = 1$.

For instance, in the case of the problem illustrated in Fig. 1, indices pb_i are arranged as follows: pb_3, pb_2, pb_1. Consequently, $wm_1 = 1, wm_2 = 4, wm_3 = 9$. The value of the objective function F_n is now computed as follows:

$$F_n = \sum_{i=1}^{3} rez_i \cdot wm_i = 5 \cdot 1 + 3 \cdot 4 + 4 \cdot 9 = 53 \tag{10}$$

Using the objective function F_n defined by formula (9) guarantees:

- uniform distribution of security buffers according to the levels pb_i owing to the appropriate definition of the weights wm_i,
- distribution of the time buffer proportionally to the durations of individual milestones – the longer the execution time of all tasks comprising a given project phase, the larger the buffer ascribed to that phase (milestone).

The maximum possible protection of milestone deadlines is material, given the uncertainty (connected, e.g., with possible occurrence of production disturbances) observed in actual production systems. The creation of a nominal schedule with large time buffers rezi (with large values of the objective function F_n) would also support the insertion of time buffers protecting the tasks most vulnerable to production disturbances in the phase of robust scheduling.

3. SCHEDULING IN A SCHEDULING PROBLEM FOR A PROJECT WITH MILESTONES

The phase of nominal scheduling includes the arrangement (determination of task starting times) with use of the schedule quality criterion (in the case discussed, the maximisation of protection of timely execution of milestones). The next scheduling phase is devoted to making the schedule robust to possible production disturbances; therefore, this phase is known as robust scheduling.

The robust scheduling has with growing frequency been used in task arrangement. A robust schedule is developed in the phase of project planning, prior to project execution, and is defined as an arrangement which, owing to its properties, is robust to disturbances which may occur during the production process. Jensen (2001) defines a robust schedule as an arrangement whose quality remains acceptable even when unpredictable events have occurred. Al-Fawzan and Haouari (2005) have limited the scope of the notion of robustness by suggesting that schedule robustness should be understood as the schedule's ability to counteract such minor increases in task durations which may be caused by uncontrollable factors. This paper follows his approach.

The robust scheduling phase for the RCPSP problem includes the following two optimisation steps (Van De Vonder, 2006):

1) robust resource allocation – appropriate allocation of resources for the purposes of performing individual activities in order to develop a schedule which would be as robust to disturbances as possible; the relevant algorithms include those minimising the number of additional arrangement-related relations resulting from resource allocation, e.g., ISH, ISH2 (Policella *et al.*, 2004) or MABO (Van De Vonder, 2006),
2) robust buffer allocation – performed with the fixed resource allocation to tasks and consisting in the insertion of resource or time buffers before (or after) tasks in order to make the schedule robust to temporary unavailability of resources and variability of activity duration.

Robust allocation of time buffers is subject of analysis in his paper. The appropriate localisation of buffers is material in the context of, inter alia, the project execution cost. For a less disturbed schedule, lower storage cost is observed or a lower cost of loan financing the purchase of materials supplied just-in-time. Time buffers should be inserted at critical points of the nominal schedule, those most exposed to disturbances, and with the largest projected instability cost. Individual tasks are commenced as under the CC/BM methodology, in line with the ALAP principle, but with task and milestone buffers included in the schedule.

Time buffers are designed to prevent minor increases (or, more generally, oscillations) in task durations, where such increases are triggered by uncontrollable factors. Other production disturbances are excluded from considerations as it has been assumed that the majority of disturbances affect task durations. In the course of project execution, task durations may lengthen for reasons difficult to identify during project planning, such as errors in duration estimates, adverse weather conditions, failures etc.

The need to buffer tasks also results from the principle, currently applied in practice, of estimating task durations in project execution, for instance, under the CCPM methodology. Each project is a unique undertaking, harmonised with the customer's specifications. Frequently, no statistical data is available on durations of identical tasks. However, information may be available concerning the execution of similar tasks. Based on such information, an analysis of task complexity or other input, in the production planning phase, duration is estimated for each task, in line with the reasonable assessment of execution feasibility. Minimum estimate values, known as aggressive estimates, are assumed. This is justified by the results of current research into the field of project management. These results indicate that assuming safe estimates is less effective than assuming aggressive estimates. Where safe estimates are used, the execution of a task, as a rule, consumes the entire time available anyway (as predicted by Parkinson's Law), and even delays occur with a frequency slightly lower than when aggressive estimates are used (the student syndrome). Assuming shorter task durations enables the development of a schedule with time buffers securing timely execution of milestones. It is also often possible to insert additional time buffers securing task arrangement at points most vulnerable to disturbances. The following assumptions were adopted for buffer arrangement optimisation:

– the larger is the number of resource types used by an activity, the more is the activity vulnerable to disturbances; e.g., the probability is larger of temporary resource unavailability (machine failure, an employee's sickness etc.);
– the longer is activity duration, the larger is the probability that it lengthens: tasks with longer execution times reveal larger absolute variability;
– a disturbance in the commencement of a more resource-consuming activity generates larger instability cost (as it disorganises the operation of a larger number of resource types).

The assumptions adopted result from the fact that in actual production systems, the longest and most resource-consuming tasks are critical for project execution - they are more difficult to estimate and more vulnerable to disturbances.

The buffer allocation algorithms described in Section 2 cannot be applied to the problem of timely execution of milestones. Schedule buffering should be performed in such a way that predefined proportions should be followed as best as possible in protecting timely execution of all project milestones. The authors have adopted the following assumptions for buffer allocation in the problem considered:

– these activities may be buffered only which are connected with milestones for which $rez_i > 0$;

– buffer insertion must not result in contractual deadlines of individual milestones being missed.

Under the assumptions adopted, the number of different buffer arrangements depends on the sizes rez_i of individual milestones. The larger is rez_i, the larger is the number of possible distributions of time buffers for individual tasks.

Buffer allocation is designed to determine the list of buffers after activities $B = (B_1, B_2, \ldots, B_n)$ or before activities $b = (b_1, b_2, \ldots, b_n)$, so that the resulting proactive arrangement is as robust to disturbances as possible. Thus it is of key importance that the level of schedule robustness be determined, that is its ability to absorb disturbances occurring during the operation of a production system. The following two types of robustness are defined:

– quality robustness – under this approach, the key task is to meet the performance criterion, which is referred to as stability of makespan; anticipating production disturbances is designed to minimise the deviation of the actual makespan from the planned one;
– solution robustness – this approach endeavours to execute all details of the arrangement as planned; the deviations are taken into consideration concerning, for instance, starting and ending times of individual tasks, manner of resource allocation to individual activities etc.; anticipating production disturbances is designed to minimise, e.g., the total deviation of actual task starting times from the planned ones or the number of activities in the actual arrangement rearranged when compared with the planned arrangement.

For objective functions of robust scheduling, quality robustness metrics and solution robustness metrics are used. The metrics used in the study of the RCPSP problem cannot be used, as they do not cover contractual deadlines of milestones. Accordingly, the authors have developed robustness metrics dedicated to the considered problem of timely execution of agreed milestones.

For the considered RCPSP problem with milestones, quality robustness is meant as the ability to meet all agreed deadlines of milestones. The problem of achieving quality robustness consists in protecting deadlines of individual milestones and thus of the entire project. The metric of arrangement quality robustness, appropriate for the problem considered, is the objective function of nominal scheduling defined by formula (9). Therefore, the initial schedule developed during nominal scheduling is robust in terms of arrangement quality.

The objective function for robust scheduling is the measure of robustness of arrangement details. With the robustness of arrangement details, the schedule stability is connected, with the proviso that robustness is the metric of arrangement quality in the planning phase, while schedule stability is such a metric during the implementation of the plan (in reactive scheduling). Robust scheduling may be described as endeavour to minimise the effect of disturbances onto the stability indicators of the implemented arrangement. Also in this paper, the authors suggest that solution robustness should be measured by way of simulation and defined as a schedule stability indicator determined through computational experiments with random generation of production disturbances.

In the studies concerning the RCPSP problem, different reactive-scheduling objective functions are used. Some of them are also used in the development of stability indicators for the problem of timely execution of milestones.

For a stochastic model (Lambrechts *et al.*, 2006; Van de Vonder *et al.*, 2005; Van de Vonder *et al.*, 2006; Van De Vonder, 2006), the minimisation of the weighted instability cost is the objective function:

$$\Delta(S^R, S^0) = \sum_{i=1}^{n} w_i \cdot |s_i^R - s_i^0| \tag{11}$$

where:
S^R – the realized schedule,
S^0 – the planned schedule,
s_i^0 – stands for the planned starting time for the operation i; it serves as the decision variable during the development of a predictive schedule,
s_i^R – stands for the actual starting time for the operation i.

Now Al-Fawzan and Haouari (2005) consider a bi-objective function enabling an analysis to be performed of the relation between schedule robustness to disturbances and stability on the one hand and its optimality expressed in terms of project execution time on the other. A parameter is defined specifying relative importance of individual optimisation criteria and subject to the planner's decisions.

Following Al-Fawzan and Haouari's approach, the RCPSP problem with milestones may be defined as the problem of maximising a bi-objective function admitting a following general description:

$$F_\alpha = \alpha \cdot \mathit{Stab} + (1 - \alpha) \cdot \mathit{Qual} \qquad 0 \leqslant \alpha \leqslant 1 \tag{12}$$

where:
F_α – stands for a function of arrangement stability and quality,
α – stands for a parameter specifying the importance of schedule robustness metrics with respect to its quality,
Stab – stands for the metric of arrangement stability,
Qual – stands for the metric of schedule performance (quality).

For the resource-constrained scheduling problem with predefined deadlines of key activities, considered herein, the objective function should reflect both the quality criterion *Qual* and the arrangement stability criterion *Stab*. The authors propose to minimise the following bi-objective functions (Klimek and Łebkowski, 2008a):

$$F_1 = \alpha \cdot \sum_{i=1}^{n} (\sum_{j=1}^{K} r_{ij} \cdot \left|s_i^R - s_i^0\right|) + (1-\alpha) \cdot \sum_{i=1}^{m} f_i \cdot \max_{j \in M_i}(0, z_j - tm_i)) \tag{13}$$

$$F_2 = \alpha \cdot \sum_{i=1}^{n} \left|s_i^R - s_i^0\right| + (1-\alpha) \cdot \sum_{i=1}^{m} f_i \cdot \max_{j \in M_i}(0, z_j - tm_i)) \tag{14}$$

where f_i stands for the penalty function defining the penalty for missing the deadline of the milestone km_i), and reflecting, e.g., cost connected with milestone execution delay by 1 unit of time.

Under the approach using the function F_1, it is assumed that the larger the number of resources undergoing changes, the less the stability of the production system – a delay in the commencement of a more resource-consuming activity causes larger instability of the production system. Under the approach using the function F_2, it is assumed that all disturbances have the same effect on production stability, irrespective of the activity disturbed.

The objective functions F_1 and F_2 enable an analysis of the RCPSP problem for stability (schedule robustness) and performance (quality robustness). The coefficient α lies in the interval $\langle 0, 1 \rangle$ and determines the importance (weight) of the stability of production plan execution for a given project (taking into consideration a number of factors, including the nature of the project, industry profile etc.):

- when $\alpha = 0$, the problem consists in the timely execution of all milestones (quality criterion);
- when $\alpha = 1$, the problem consists in rendering the schedule robust to possible disturbances in order to minimise the total deviation of the actual task staring times from the planned ones (stability criterion).

For an individual milestone, the function f_i depends on the effect of delays on project execution. In this paper, the penalty f_i is such a transformation of the cost related to the delay in execution of the milestone i which renders similar values of the metrics of stability $Stab$ and quality $Qual$.

The objective functions F_1 and F_2 proposed above may be used for both predictive scheduling (in the course of which they are determined by way of simulation) and reactive scheduling. They both serve as comprehensive criteria for schedule assessment.

4. BUFFER ALLOCATION ALGORITHM FOR THE PROBLEM OF TIMELY MILESTONE EXECUTION

In this section, the buffer allocation algorithm $BufR$ is presented, developed by the authors and executing the procedure of unit buffer insertion with view to optimising static metrics of robustness. The $BufR$ algorithm uses an auxiliary objective function of proactive scheduling (schedule robustness metric); the function is computable without production system simulation.

The use of this auxiliary objective function should lead to making the schedule robust in such a way that the instability cost (see formulae (13) and (14)) of the actual arrangement is as low as possible.

A schedule developed in the nominal scheduling phase provides for time redundancy protecting the schedule against untimely execution of milestones. The redundancy may partially be used for time buffers in order to render the arrangement robust at the points most exposed to disturbances and thus generating the highest instability cost. The metric of solution robustness should reflect a specific nature of a given production system. In this paper, the authors assume that a task performed with/by a larger number of resources and more time-consuming is more exposed to disturbances.

The authors propose the following general form of the metric of solution robustness:

$$R = \left(\sum_{i=1}^{n} w_i \cdot \sum_{j=1}^{bB_i} \frac{1}{j^a} + \sum_{i=1}^{m} \big(wm_i \cdot \sum_{j=1}^{bm_i} \frac{1}{j^a} \big) - \sum_{i=1}^{m} fm_i \cdot \max_{j \in M_i} \big(0, z_j - bm_i - tm_i \big) \right) \quad (15)$$

where:

a – stands for a parameter determining buffer allocation ($a > 0, a = 2$ has been assumed in the computations),

bB_i – stands for the time buffer of the task i(before the activity b_i or after the activity B_i),

w_i – stands for the weight ascribed to the activity i,

wm_i – stands for the weight ascribed to the i-th milestone,

fm_i – stands for the cost relating to missing the deadline for the i-th milestone, it may include, for instance, actual contractual penalties for delays,

bm_i – stands for the time buffer of the i-th milestone, i.e., the number rez_i of time units provided for the protection of timely execution of the i-th project phase:

$$bm_i = \begin{cases} 0, & \text{if } rez_i \leqslant 0, \\ \lceil \xi \cdot rez_i \rceil, & \text{if } rez_i > 0 \end{cases} \quad (16)$$

ξ – stands for the parameter specifying what portion of the time redundancy rez_i has been applied towards the time buffer bm_i of the i-th milestone.

The size of bm_i depends on the buffering strategy. The development of a strategy appropriate for production environments characterised by different variability of task durations has been subject to experimental analysis discussed herein.

The weights w_i and wm_i ascribed to individual activities and milestones may be based on an analysis of costs connected with the instability of project execution and may include storage cost, additional organisational cost and contractual penalties for delays.

For the metric R, such task time buffer allocation is preferred in which the buffers bB_i are as little as possible. The even distribution of buffers throughout the schedule is striven at (the idea akin to the one proposed by Kobylański and Kuchta (2007). For instance (the importance of instability cost for a given activity having been neglected): an increase in the time buffer bB_i before an activity for which $bB_i = 0$ instead of before an activity for which $bB_i = 2$, increases the value of the objective function by the factor of 3^a. The parameter a controls the evenness of buffer distribution (the larger is a, the more even is the distribution).

The general form of the robustness metric R is used to develop the following metrics R_1, R_2 and R_3:

$$R_1 = \left(\sum_{i=1}^{n} (\sum_{j=1}^{k} r_{ij}) \cdot \sum_{j=1}^{bB_i} \frac{1}{j^a} + \sum_{i=1}^{m} (r_{\max} \cdot \sum_{j=1}^{bm_i} \frac{1}{j^a}) - \sum_{i=1}^{m} sumr_i \cdot \max_{j \in M_i} \left(0, z_j - bm_i - tm_i\right) \right) \tag{17}$$

$$R_2 = \left(\sum_{i=1}^{n} d_i \cdot \sum_{j=1}^{bB_i} \frac{1}{j^a} + \sum_{i=1}^{m} (d_{\max} \cdot \sum_{j=1}^{bm_i} \frac{1}{j^a}) - \sum_{i=1}^{m} sumd_i \cdot \max_{j \in M_i} \left(0, z_j - bm_i - tm_i\right) \right) \tag{18}$$

$$R_3 = \left(\sum_{i=1}^{n} \left(\sum_{j=1}^{bB_i} \frac{1}{j^a} \right) + \sum_{i=1}^{m} (\sum_{j=1}^{bm_i} \frac{1}{j^a}) - \sum_{i=1}^{m} n_i \cdot \max_{j \in M_i} \left(0, z_j - bm_i - tm_i\right) \right) \tag{19}$$

where:

n_i – stands for the number of tasks executed in the i-th project phase, i.e., the number of elements in the set KM_i;

r_{max} – stands for the maximum aggregate demand for resources from project activities;

$sumri$ – stands for the demand for resources from the activities comprising the execution of the i-th milestone;

d_{max} – stands for the maximum duration of project activities;

$sumd_i$ – stands for the aggregate duration of the activities comprising the execution of the i-th milestone.

Maximisation of R_1, R_2 and R_3 renders the arrangement robust. The metric R_1 protects primarily the most resource-consuming activities. The metric R_2 is designed to prefer buffer insertion after activities with longer durations. The use of the metric R_3 renders all tasks equally important. The metric R_1 has been designed for the objective function defined by formula (13), while the metrics R_2 and R_3 – for objective function (14). All these metrics include milestone deadlines; missing the deadlines is "penalised".

The $BufR$ algorithm inserts unit buffers one after another. In a single step of the algorithm, adding a time buffer to each activity is tested. A time buffer is increased for the job for which the robustness metric R used (i.e., R_1, R_2 or R_3) takes the largest value. The algorithm stops when adding such a buffer does not enhance the schedule robustness.

The $BufR$ algorithm supports the insertion of buffers both before and after an activity, with the proviso that a given proactive schedule inserts buffers either exclusively after activities or exclusively before them.

The $BufR$ algorithm is supported by the metrics R_1, R_2, R_3, which have been designed for the considered problem RCPSP with milestones and for the objective functions of the predictive-reactive scheduling used herein (see formulae (13) and (14)). However, the $BufR$ algorithm is versatile: it may operate with other robustness metrics based on other assumptions with respect to variability of production parameters; it may also reflect considerations of various production environments. Fig. 3 illustrates the run of the algorithm $BufR$ maximising the metric R.

```
//Step 1: Initialization $S_0$ (actual schedule with time buffers)
//as nominal schedule $S_{nom}$ without time buffers
$S_0$ = $S_{nom}$
//Iteration steps: in one step 1-unit buffer is inserted
repeat
bestR := 0   //best value of robustness measure in actual step
for job 1 to n
  buffer(job) = buffer(job) + 1
  Modify$S_0$
  if (R($S_0$) > bestR) then
    bestJob := job
    bestR := R($S_0$)
  end if
  buffer(job) = buffer(job) − 1
  Modify$S_0$
end for
if (R($S_0$) < bestR) then
  buffer(bestJob) = buffer(bestJob) − 1
end if
until (R($S_0$) >= bestR)
//no improve after adding buffer in actual step
```

Fig. 3. *Flowchart of the operation of the buffer allocation algorithm BufR*

5. CONSTRUCTION OF TEST TASKS

This section discusses the rules for the construction of test tasks for the RCPSP problem with milestones. Experiments are conducted with use of the PSPLIB library 0, most commonly applied in research into the RCPSP problem and generated with use of the project generator ProGen/max. From the PSPLIB library, test instances have been selected comprising 30 or 90 tasks, with a single execution mode (single-mode RCPSP), with renewable resources and minimisation criterion for project duration. A total of 960 instances (480 30-task instances and 480 90-task ones) have been used.

For projects sourced from the PSPLIB library, the authors suggest using the LOSM procedure of unequivocal definition of individual milestones, as this procedure guarantees inspection of project progress over the entire execution period through an even distribution of inspection times tm_i:

$$tm_{i+1} - tm_i = \frac{tm_m}{m}, \qquad \forall i \in < 1, m) \tag{20}$$

where tm_m stands for the entire project deadline ($tm_m = \delta_{n+1}$).

The LOSM procedure runs as follows:

Step 1. a) Development of a schedule H for the RCPSP problem with use of priority algorithm with random task priorities (a random number generator used to generate task priorities is initialised with the seed identical to the one used for the given test problem by ProGen). Computation of the project duration Cmin for the random schedule H thus developed.

b) Determination of milestone deadlines $(tm_1, tm_2, \ldots, tm_m)$ according to the formula:

$$\forall i \in \langle 1, m) : tm_i = i \cdot \left\lceil \frac{C_{\min} \cdot (1+\tau)}{m} \right\rceil \tag{21}$$

where τ stands for a parameter describing the project duration relative lengthening (expressed as percentage) which may be used for task buffering.

Step 2. Creation of the task list L and supplementing it with all inconspicuous/minor tasks arranged in the ascending order of their respective starting times specified in the schedule H generated in Step 1. When multiple tasks have the same starting time, they are added to the list L in the ascending order of their numbers.

Step 3. Downloading the subsequent task j from the list L and including it in the set M_i if the following condition holds true:

$$s_j^H + d_j < (1 - \beta_i) \cdot tm_i \tag{22}$$

where:

s_j^H – stands for the starting time of the activity j in the schedule H,

β_i – stands for a parameter controlling the possibility of buffering the i-th milestone.

If condition (22) is not met, the task j is added to the next set $M_i + 1$ and the then current set of milestones is changed to the next one: $i := i + 1$. If $i = m$, the procedure goes to Step 4; otherwise, Step 3 is repeated for the next activity in the list L.

Step 3 is executed for all tasks considered in the order defined in the list L. Step 3 supports the generation of the sets $M_i (i = 1, 2, \ldots, m - 1)$. First, the set $M_1 (i := 1)$ is created. In a single run of Step 3, a single task is allocated to the appropriate set M_i. Step 3 is rerun until $i = m$.

Step 4. Adding, to the set M_m, the remaining tasks included in the list L and not yet allocated to any of the sets $M_i (i = 1, 2, \ldots, m - 1)$.

For the purposes of computational experiments, four project phases (milestones: $m = 4$) are defined for each test problem. The parameters tm_i are defined as the common deadlines for tasks included in the same set M_i, as well as the tasks included in individual sets M_i according to the LOSM procedure (in Step 1: the parameter $\tau = 30\%$, in Step 3 $\beta_1 = 10\%, \beta_2 = 10\%$ and $\beta_3 = 10\%$).

Beside the definition of conventional milestones, the definition of task duration variability is also material in simulations. Project execution is inescapably connected with the uncertainty of task duration estimations, related to numerous factors, including a large number of uncontrollable factors, unique nature of the tasks involved etc. The authors have assumed that the planned task durations may be subject to factors which are not identifiable in the planning phase, such as duration estimate errors, adverse weather conditions, failures etc. The analysis of task durations for real projects (especially those executed by human resources) reveals that the probability curve for task ending takes the shape of the β curve. The cumulative beta distribution is used in project planning to model possible ending times at the given expected ending time and its variability.

In this paper, the authors adopt the same parameters describing task duration variability which have been used in other studies (Herroelen and Leus, 2004; Lambrechts *et al.*, 2006; Leus, 2003; Van de Vonder *et al.*, 2005; Van de Vonder *et al.*, 2006; Van De Vonder, 2006). For each task, the actual duration is randomly selected from the discretised positively skewed beta distribution with the parameters 2 and 5, and with the expected (average) value equal to the planned task duration. The unequivocal parameter definition requires that minimum and maximum values should also be defined. Two cases are considered – of large and small task duration variability. For each task $i (i = 1, 2, \dots, n)$:

– with high variability, the minimum value is 0.25 d_i, while the maximum value is 2.85 d_i;
– with low variability, the minimum value is 0.75 di, while the maximum value is 1.625 d_i.

6. RESULTS OF COMPUTATIONAL EXPERIMENTS

In this section, the effectiveness of the proposed robust buffer allocation is analysed. Experiments were performed on a computer with a Pentium 1.7 GHz processor, supported by a program implemented in C# in the Visual Studio.NET environment.

A nominal schedule was generated with use of the Simulated Annealing (SA) metaheuristics, with the objective function defined by formula (9). The best values of parameters of the algorithm and best solution search techniques were determined experimentally (Klimek and Łebkowski, 2008b). The next phase comprised robust allocation of resources. Resources allocation was executed with use of the ISH^2 (Iterative Sampling Heuristic) algorithm (Policella, et al., 2004; Policella, 2005). The use of that algorithm enables the reduction of the number of additional arrangement relations (known as synchronisation points) in the project activity network.

The ISH^2 algorithm allocates the activity j just analysed to these resources (chains) in which the last activity immediately precedes the activity j. If the demand of the activity j for resources exceeds the demand of its predecessors, the remaining chains are selected by the ISH algorithm, which allocates an activity demanding a given type of resources ($r_{jk} > 1$) so as to maximise the number of common chains shared with the most recent activities in the available chains.

The objective functions F_1 and F_2 defined by formulae (13-14) serve as assessment criteria for robust buffer allocation algorithms. The functions F_1 and F_2 are determined by way of simulation for durations generated from the beta distribution with a large or small variability. In the project execution phase, the simplest right-shift rescheduling method is used, consisting in shifting the disturbed activities to the right with resources allocation unchanged.

The following parameters are tested in the $BufR$ algorithm:

- robustness metric used (R_1, R_2 or R_3),
- various values of ξ ($0.25, 0.5, 0.75$),
- buffer insertion before activities (b buffers) or after activities (B buffers).

For each project, three production process scenarios are generated: random task durations sourced from the distribution with appropriate parameter values (selected separately for large variability and for small variability). Projects (their test instances) have different buffering possibilities. A large variability of instability cost with respect to the objective function of nominal scheduling renders it more difficult to analyse the efficiency of the buffer allocation algorithm $BufR$ against the parameters used (large values of standard deviations for the instability costs obtained are observed).

Tables 2 and 3 set forth the results of the operation of the $BufR$ algorithm with small and large variability of durations.

Table 2. *Average values of the objective functions F_1 and F_2 for selected values of the parameters of the $BufR$ algorithm – 30-job projects (J30 set)*

Algorithm	Small duration variability				Large duration variability			
	$\overline{F_1}$		$\overline{F_2}$		$\overline{F_1}$		$\overline{F_2}$	
	$\alpha = 0.25$	$\alpha = 0.75$	$\alpha = 0.25$	$\alpha = 0.75$	$\alpha = 0.25$	$\alpha = 0.75$	$\alpha = 0.25$	$\alpha = 0.75$
$R_1, \xi = 0.25, b$	75.5	72.0	3.7	4.6	4,467.9	2,066.7	184.8	99.9
$R_1, \xi = 0.50, b$	71.5	112.8	3.9	7.5	4,032.2	2,091.5	168.9	105.5
$R_1, \xi = 0.75, b$	101.5	213.2	5.9	13.7	3,906.8	2,265.8	165.2	117.7
$R_2, \xi = 0.25, b$	78.9	81.1	3.8	4.8	4,508.1	2,092.6	186.6	100.7
$R_2, \xi = 0.50, b$	74.1	122.4	4.0	7.7	4,085.3	2,122.1	170.8	106.4
$R_2, \xi = 0.75, b$	102.2	215.4	5.8	13.7	3,922.9	2,268.9	165.8	117.6
$R_3, \xi = 0.25, b$	77.3	74.1	3.7	4.4	4,491.9	2,071.7	186.0	99.4
$R_3, \xi = 0.50, b$	74.0	121.3	3.9	7.6	4,055.1	2,108.0	170.0	105.7
$R_3, \xi = 0.75, b$	103.4	219.1	5.9	13.7	3,936.8	2,281.8	166.4	118.0
$R_1, \xi = 0.25, B$	67.7	81.1	3.5	5.2	4,220.1	2,040.0	174.9	100.6
$R_1, \xi = 0.50, B$	76.9	136.2	4.3	8.9	3,903.3	2,117.8	164.2	109.0
$R_1, \xi = 0.75, B$	112.5	247.4	6.6	16.0	3,824.4	2,310.3	162.7	122.4
$R_2, \xi = 0.25, B$	**61.8**	**68.1**	**3.1**	**4.3**	4,162.0	**2,003.0**	172.8	**99.0**
$R_2, \xi = 0.50, B$	70.0	118.0	3.9	7.9	3,881.0	2,088.9	163.0	107.4
$R_2, \xi = 0.75, B$	107.9	233.9	6.4	15.4	**3,800.3**	2,282.4	**161.7**	121.2
$R_3, \xi = 0.25, B$	69.1	78.4	3.6	5.0	4,228.3	2,027.2	175.6	100.0
$R_3, \xi = 0.50, B$	77.3	139.7	4.4	9.3	3,889.4	2,114.1	164.0	109.1
$R_3, \xi = 0.75, B$	112.9	248.5	6.7	16.2	3,839.4	2,316.4	163.4	122.9
Nominal	162.7	399.2	10.1	26.6	3,850.1	2,531.8	166.0	138.9

Table 3. *Average values of the objective functions F_1 and F_2 for selected values of the parameters of the $BufR$ algorithm – 90-job projects (J90 set)*

Algorithm	Small duration variability				Large duration variability			
	$\overline{F_1}$		$\overline{F_2}$		$\overline{F_1}$		$\overline{F_2}$	
	$\alpha = 0.25$	$\alpha = 0.75$	$\alpha = 0.25$	$\alpha = 0.75$	$\alpha = 0.25$	$\alpha = 0.75$	$\alpha = 0.25$	$\alpha = 0.75$
$R_1, \xi = 0.25, b$	482.1	379.4	20.4	21.3	35,469.8	15,680.6	1,315.9	683.5
$R_1, \xi = 0.50, b$	456.3	634.7	21.5	37.6	33,702.9	16,377.4	1,258.5	745.3
$R_1, \xi = 0.75, b$	660.9	1,322.4	34.3	78.8	33,247.7	17,722.1	1,254.2	840.0
$R_2, \xi = 0.25, b$	508.1	421.2	21.2	22.6	36,069.2	15,993.7	1,335.5	694.9
$R_2, \xi = 0.50, b$	466.3	653.6	21.6	37.6	34,177.3	16,542.5	1,274.9	749.6
$R_2, \xi = 0.75, b$	655.5	1,304.2	33.7	76.9	33,383.1	17,732.0	1,258.7	838.4
$R_3, \xi = 0.25, b$	522.9	427.4	22.0	22.5	35,658.1	15,772.1	1,325.1	684.3
$R_3, \xi = 0.50, b$	502.1	756.8	23.4	42.3	33,751.9	16,479.5	1,264.3	748.2
$R_3, \xi = 0.75, b$	699.0	1,422.1	35.7	82.4	33,504.9	17,876.4	1,264.9	845.2
$R_1, \xi = 0.25, B$	401.8	394.4	17.6	22.4	33,982.9	15,460.9	1,260.6	685.2
$R_1, \xi = 0.50, B$	463.6	722.2	22.3	42.4	32,756.8	16,365.7	1,227.7	758.5
$R_1, \xi = 0.75, B$	707.9	1,491.5	37.5	89.5	32,711.3	17,869.6	1,239.5	861.4
$R_2, \xi = 0.25, B$	**362.1**	**330.9**	**15.5**	**18.6**	33,955.5	**15,409.5**	1,258.2	**682.2**
$R_2, \xi = 0.50, B$	421.0	625.9	20.2	37.3	32,841.6	16,282.6	**1,227.1**	752.1
$R_2, \xi = 0.75, B$	671.4	1,386.8	35.8	84.4	**32,618.9**	17,709.7	1,235.1	853.6
$R_3, \xi = 0.25, B$	433.5	429.8	19.1	24.2	34,028.7	15,453.9	1,263.5	684.1
$R_3, \xi = 0.50, B$	501.4	833.4	24.4	48.6	32,886.8	16,495.6	1,232.8	764.1
$R_3, \xi = 0.75, B$	742.3	1,582.5	39.6	95.2	32,947.7	18,007.3	1,248.9	868.3
Nominal	1,104.4	2,690.3	64.7	171.2	33,047.8	19,434.0	1,270.1	977.4

$\overline{F_1}, \overline{F_2}$ – stand for the average value of the objective function (instability cost) F_1 and F_2, respectively.

For the majority of parameter configurations, the use of buffer allocation enhances arrangement stability when compared with the execution of a nominal schedule without time buffers. The larger the importance of stable execution of individual tasks (the larger the value of α), the larger the importance of time buffers.

For the robustness metrics proposed, the insertion of buffers after activities proves more effective than their insertion before activities. When buffers are inserted before activities, the metric R_1 is most effective. When buffers are inserted after activities, the metric R_2 is most effective. For the majority of the objective functions analysed, the most effective algorithm is the algorithm $BufR$ optimising the metric R_2 with $\xi = 0.25$, inserting buffers after activities. This means that it is effective to first insert buffers protecting the schedule against the lengthening of most time-consuming tasks.

The parameter ξ has major effect on the effectiveness of the $BufR$ algorithm. Schedules generated with $\xi = 0.25$ prove most stable. It is advisable to identify the most effective value of parameter ξ for each project, as this value has an enormous effect on the robustness of a proactive schedule.

A more detailed analysis of the effect of parameter ξ on production stability was performed for a test task J30_1_3 with $F_n = 392$. One hundred duration disturbance scenarios were generated from the beta distribution. Table 4 and Fig. 5 present results of experiments run for the $BufR$ algorithm with the following parameters: metric R_2 and buffer insertion after activities with small task duration variability. In simulations was analyzed the objective function F_2 with $\alpha = 0.25$.

Table 4. *The average value of F_2 for selected values of parameter ξ with small task duration variability*

ξ	$\overline{F_2}$	ξ	$\overline{F_2}$	ξ	$\overline{F_2}$	ξ	$\overline{F_2}$
0.00	16.50	0.25	5.9	0.50	14.5	0.75	23.1
0.05	16.50	0.30	7.1	0.55	18.8	0.80	28.3
0.10	5.34	0.35	8.2	0.60	19.5	0.85	36.5
0.15	5.34	0.40	8.2	0.65	19.5	0.90	40.4
0.20	5.76	0.45	11.6	0.70	21.6	0.95	50.5

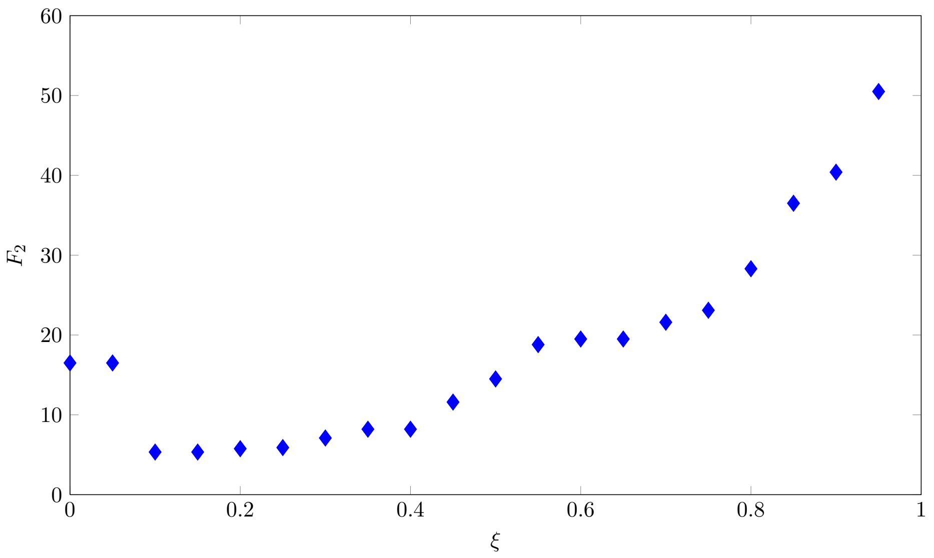

Fig. 4. *The instability cost F_2 as a function of parameter ξ describing what proportion of the time reserve rez_i has been applied towards the time buffer bm_i*

The parameter ξ exerts an enormous impact on production stability. Its optimum value depends on numerous factors, including the specific nature of the test task involved and the variability of task durations.

7. SUMMARY

The paper discusses a scheduling model for resource-constrained projects with predefined deadlines for certain tasks connected with project milestones. The model proposed may prove very useful for the execution of large production, construction or development orders, because progress inspection times (points) are commonly defined for large projects. Timely execution of milestones mitigates the risk of unsuccessful execution of the entire project.

For the model developed, the authors also develop an algorithm for robust allocation of time buffers. The results of computational experiments confirm the effectiveness of the algorithm proposed.

The subject of further research will include the development of effective algorithms of reactive scheduling for the problem defined.

REFERENCES

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

Aytug, H., Lawley, M., McKay, K., Mohan, S., Uzsoy, R. (2005). *Executing production schedules in the face of uncertainties: A review and some future directions*. European Journal of Operational Research, 161(1), pp. 86-110.

Goldratt, E.M. (1997). *Critical chain*. Great Barrington: The North River Press.

Hall, N.G., Posner, M.E. (2004). *Sensitivity Analysis for Scheduling Problems. Journal of Scheduling*, 7(1), pp. 49–83.

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

Jensen, M.T. (2001). *Improving robustness and flexibility of tardiness and total flow-time job shops using robustness measures*. Applied Soft Computing, 1, pp. 35–52.

Klimek, M., Łebkowski P. (2007). *Predictive-Reactive Project Scheduling. in Innovations technologies in economics and innovative management* (ed. J. Duda), Uczelniane Wydawnictwa Naukowo-Dydaktyczne Akademii Górniczo-Hutniczej, Kraków, pp. 198–206.

Klimek, M., Łebkowski P. (2008a). *Miary odporności harmonogramów* [Schedule Robustness Metrics, in Polish]. in Komputerowo Zintegrowane Zarządzanie (ed. R. Knosala), Oficyna Wydawnicza Polskiego Towarzystwa Zarządzania Produkcją, Opole, Vol. I, pp. 569–577.

Klimek, M., Łebkowski P. (2008b). *Algorytmy metaheurystyczne dla problemu harmonogramowania projektu z kamieniami milowymi* [Metaheuristics Algorithms for Scheduling Problem for Projects with Milestones, in Polish], Zeszyty Naukowe Politechniki Śląskiej, Series: Automatyka, Fasc. 150, pp. 63–72.

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, 107, pp. 496–501.

Kolisch, R., Sprecher A. (1997). PSPLIB – a project scheduling library, European Journal of Operational Research, 96, pp. 205–216.

Lambrechts, O., Demeulemeester, E. , Herroelen, W. (2006). *Proactive and reactive strategies for resource-constrained project scheduling with uncertain resource availabilities*. Report KBI_0606, K. U. Leuven.

Leus, R. (2003). The generation of stable project plans, PhD thesis at K. U. Leuven, Belgium.

Policella, N., Oddi, A., Smith, S., Cesta, A. (2004). *Generating robust partial order schedules.* in Proceedings of CP2004, Toronto, Canada.

Policella, N. (2005). *Scheduling with Uncertainty – A Proactive Approach using Partial Order Schedules.* PhD thesis at La Sapienza Universita, Rome.

Van de Vonder, S., Demeulemeester, E., Herroelen, W., Leus, R. (2005). *The use of buffers in project management: The trade-off between stability and makespan.* International Journal of Production Economics, 97, pp. 227–240.

Van de Vonder, S., Demeulemeester, E., Herroelen, W., Leus, R. (2006). *The trade-off between stability and makespan in resource-constrained project scheduling*, International Journal of Production Research, 44(2), pp. 215–236.

Van De Vonder, S. (2006). *Proactive-reactive procedures for robust project scheduling*, PhD thesis at K. U. Leuven, Belgium.

Vieira, G.E., Herrmann, J.W., Lin, E. (2003). *Rescheduling manufacturing systems: a framework of strategies, policies and methods*, Journal of Scheduling, 6(1), pp. 35–58.