Irena BACH , Grzegorz BOCEWICZ, Zbigniew BANASZAK

COONSTRAINT PROGRAMMING APPROACH TO MULTI-PRODUCT SCHEDULING

Abstract

Constraint programming (CP) is an emergent software technology for declarative description and effective solving of large combinatorial problems especially in the area of integrated production planning. In that context, CP can be considered as an appropriate framework for development of decision making software supporting small and medium sized enterprises (SMEs) in the course of projects portfolio prototyping. The problem considered aims at finding a computationally effective approach for scheduling a new projects portfolio subject to constraints imposed by an availability of time-constrained resources. The problem belongs to a class of multi-mode project scheduling problems where the problem of finding a feasible solution is NP-complete. The aim of the paper is to present a CP modeling framework providing a prompt service to a set of routine queries stated both in straight and reverse way, e.g., concerning the projects portfolio makespan implied by a given resources allocation, and the feasible resources allocation guaranteeing an assumed projects portfolio makespan. The way the responses to the routine requests can be guaranteed while may be available in an on-line mode is illustrated in the example enclosed.

1 INTRODUCTION

For Small and Medium Size Enterprises (SMEs) with multiple production orders, an optimal assignment of production steps to available resources is often economically indispensable. The goal is to generate a plan/schedule of production for a given period of time while minimizing of cost that is equivalent to maximization of profit. In that context executives want to know how much a project will cost, what resources are needed, what resources allocation can guarantee project due time completion, and so on. So, SME's needs might be formulated in a form of standard, routine questions, such as: Does the projects portfolio can be completed due to an arbitrary given deadline? Does the execution of the given projects portfolio meet the assumed resources allocation within the arbitrary given period of time? Is it possible to undertake a new project under given (constrained in time) resources availability while guaranteeing disturbance-free execution of the already executed projects? What values and of what variables guarantee the project portfolio will completed with assumed values of a given set of performance indexes?

Most companies, particularly SMEs have to manage various projects which share a pool of constrained resources, taking into account various objectives at the same time. That is well known fact [1],[3], that about 80% of companies have to deal with multiple projects. This corresponds to statistics showing that about 90% of all projects occur in the multiproject context. Since the project management problems belong to the class of NP-complete problems, new methods and techniques addressing the impact of real-life constraints on on-line decision making

are of great importance. Such methods, enhancing on-line project management, and supporting a manager in the course of decision making, e.g., in the course of evaluation whether a new project can be accepted to be processed in a multi-project environment of a manufacturing system at hand or not, could be included into Decision Support Systems (DSS) tools integrated into standard project management software.

Regardless of its character and scope of business activities, a modern enterprise has to build a project-driven development strategy in order to respond to challenges imposed by growing market complexity and globalization. Managers need to be able to use a modern DSS so as to make optimal business decisions from the strategic perspective of enterprise operation. In this context, this contribution covers various issues of decision making within the framework of Constraint Programming (*CP*). The paper can be seen as continuation of our former works concerning projects portfolio prototyping [2], [3] and *CP*-based approach to the project-driven manufacturing.

We first introduce the problem formulation, see the Section 2. Then we present some details of the modeling framework assumed, in particular we describe the formalisms employed, see the Section 3. In the Section 4, the straight and reverse approaches to projects portfolio scheduling and the relevant illustrative examples are discussed. We conclude with some results and lesson learned in the Section 6.

2 PROBLEM FORMULATION

2.1 Illustrative example of decision problem

Tree projects P_1, P_2, P_3, are considered (see the activity networks Fig. 1). Each project consists of 20 operations. Operation times determining the i-th project are specified by the sequence $T_i = (t_{i,j},\ldots,t_{i,k},\ldots,t_{im})$, where: $t_{i,k}$ means the k-th operation time of the i-th project.

Up to two kinds of renewable resources can be allocated to each $O_{i,j}$ operation. So, the $dp_{i,j,k}$ means an amount of the k-th resource allocated to the operation $O_{i,j}$in a unit time.

Therefore, besides of sequences determining the operation times (1)

$$\begin{aligned} T_1 &= (1,\ 2,\ 3,\ 4,\ 4,\ 6,\ 3,2,\ 1,\ 4,\ 3,\ 1,\ 4,\ 3,\ 2,\ 3,\ 1,\ 4,\ 2,\ 4),\\ T_2 &= (2,\ 1,\ 3,\ 5,\ 2,\ 5,\ 2,\ 1,\ 6,\ 3,\ 3,\ 4,\ 6,\ 3,\ 3,2,\ 6,\ 3,\ 1,\ 4), \\ T_3 &= (1,\ 6,\ 4,\ 3,\ 3,\ 6,\ 3,\ 8,\ 5,\ 2,\ 4,\ 3,5,\ 2,\ 4,\ 3,\ 4,\ 6,\ 2,\ 4). \end{aligned} \quad (1)$$

the sequences (2) items of which determine the resources amount required by the relevant operations are considered.

$$\begin{aligned} DP_{1,1} &= (3,\ 1,\ 1,\ 1,\ 1,\ 1,\ 2,\ 1,\ 2,\ 1,\ 2,\ 3,\ 2,\ 1,\ 2\ ,3,\ 2,\ 3,\ 4,\ 3),\\ DP_{1,2} &= (1,\ 2,\ 1,\ 1,\ 2,\ 3,\ 3,1,\ 1,\ \ 2,\ 1,\ 1,\ 1,\ 4,\ 2,\ 1,\ 2,\ 2,\ 1,\ 2),\\ DP_{2,1} &= (2,\ 2,\ 1,\ 1,\ 1,\ 3,\ 1,\ 2,\ 2,\ 2,\ 1,\ 1,\ 2,\ 4,\ 1,\ 2,\ 2,\ 2,\ 1,\ 2),\\ DP_{2,2} &= (2,\ 1,\ 2,\ 3,\ 1,\ 2,\ 1,\ 2,\ 1,\ 1,\ 2,\ 1,\ 2,\ 1,\ 3,\ 2,\ 2,\ 2,\ 1,\ 1), \\ DP_{3,1} &= (2,\ 1,\ 3,\ 1,\ 2,\ 1,\ 2,\ 2,\ 1,\ 1,\ 1,\ 2,\ 1,\ 1,\ 2,\ 1,\ 2,\ 2,\ 1,\ 2),\\ DP_{3,2} &= (1,\ 2,\ 2,\ 1,\ 1,\ 1,\ 2,\ 1,\ 1,\ 4,\ 2,\ 2,\ 2,\ 1,\ 2,\ 3,\ 2,\ 1,\ 3,\ 1). \end{aligned} \quad (2)$$

The total amount of resources (available within assumed time horizon H) is specified by sequences zo_1, zo_2, (3) items of which determine resources' amount available at a given time unit:

$$\begin{aligned} zo_1 &= (15,15,15,15,15,15,\ 5,15,15,15,13,13,13,13,13,13,13,13,13,13,10,10,\ldots,10), \\ zo_2 &= (14,14,14,14,14,14,14,14,17,17,17,17,15,15,15,15,15,15,15,15,10,10,\ldots,10). \end{aligned} \quad (3)$$

Assuming the given above resources availability as well as their allocations the following questions are considered: Does the projects portfolio can be completed due to an arbitrary given deadline 36 units of time? What resources allocation guarantee the projects portfolio will

completed within the period do not exceeding 36 units of time? Response to these questions can be provided under the following assumptions.

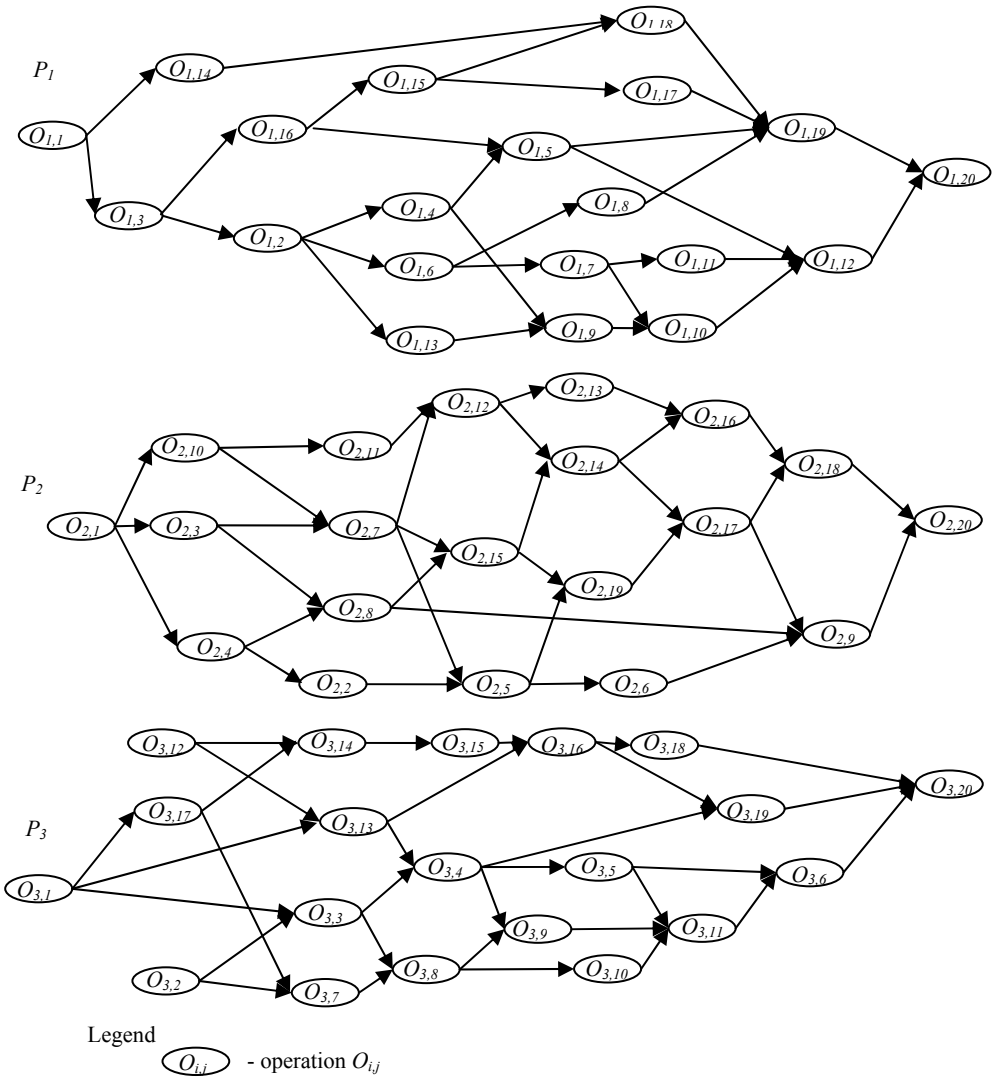


Fig. 1. Activity networks for P_1, P_2, P_3. projects

2.2 Model of decision problem

Consider the discrete time horizon $H = [0, h]$, $H \subset N$, within of which the projects portfolio $P = \{P_1,P_2,...,P_{lp}\}$ has to be completed. Given amount of renewable resources lz and their availability $Zo = (zo_1,zo_2,\ldots,zo_{lz})$; $zo_i = (zo_{i,1},zo_{i,2},\ldots,zo_{i,h})$ – the sequence items of which determine the availability of the i-th resource in a time unit within H. $zo_{i,j}$ – amount of the i-th resource at the

j-th unit of time. Each project P_i consists of lo operations $P_i = \{O_{i,1},O_{i,2},O_{i,3},...,O_{i,loi}\}$, where: $O_{i,j} = (x_{i,j}, t_{ij}, Tp_{i,j}, Tz_{i,j}, Dp_{i,j})$ such that:

$x_{i,j}$ – means the moment the operation $O_{i,j}$ begin, i.e., the time counted from the time horizon beginning H,

t_{ij} – the $O_{i,j}$-th operation time,

$Tp_{i,j} = (tp_{i,j,1}, tp_{i,j,2}, \ldots, tp_{i,j,lz})$ – the sequence of time moments the operation $O_{i,j}$ allocates new amounts of renewable resources: $tp_{i,j,k}$ – the time counted since the moment $x_{i,j}$ the $dp_{i,j,k}$ amount of the k-th renewable resource was allocated to the operation $O_{i,j}$. That means a resource is allocated to operation during its execution period: $0 \le tp_{i,j,k} < t_{ij}$; $k = 1,2,\ldots,lz$.

$Tz_{i,j} = (tz_{i,j,1}, tz_{i,j,2}, \ldots, tz_{i,j,lz})$ – the sequence of moments the operation $O_{i,j}$ releases the subsequent resources: $tz_{i,j,k}$ – the time counted since the moment $x_{i,j}$ the $dp_{i,j,k}$ amount of the k-th renewable resource was released by the operation $O_{i,j}$. That is assumed a resource is released by operation during its execution: $0 < tz_{i,j,k} \le t_{ij}$; $k = 1,2,\ldots,lz$, and $tp_{i,j,k} < tz_{i,j,k}$; $k = 1, 2, \ldots, lz$.

$Dp_{i,j} = (dp_{i,j,1}, dp_{i,j,2},..., dp_{i,j,lz})$ – the sequence of the k-th resource amounts $dp_{i,j,k}$ are allocated to the operation $O_{i,j}$, i.e., $dp_{i,j,k}$ – the amount of the k-th resource allocated to the operation $O_{i,j}$. That assumes: $0 \le dp_{i,j,k} \le zo_k$; $k = 1, 2, \ldots, lz$.

Graphical illustration of the activity $O_{i,j} = (x_{i,j}, t_{ij}, Tp_{i,j}, Tz_{i,j}, Dp_{i,j})$ description .

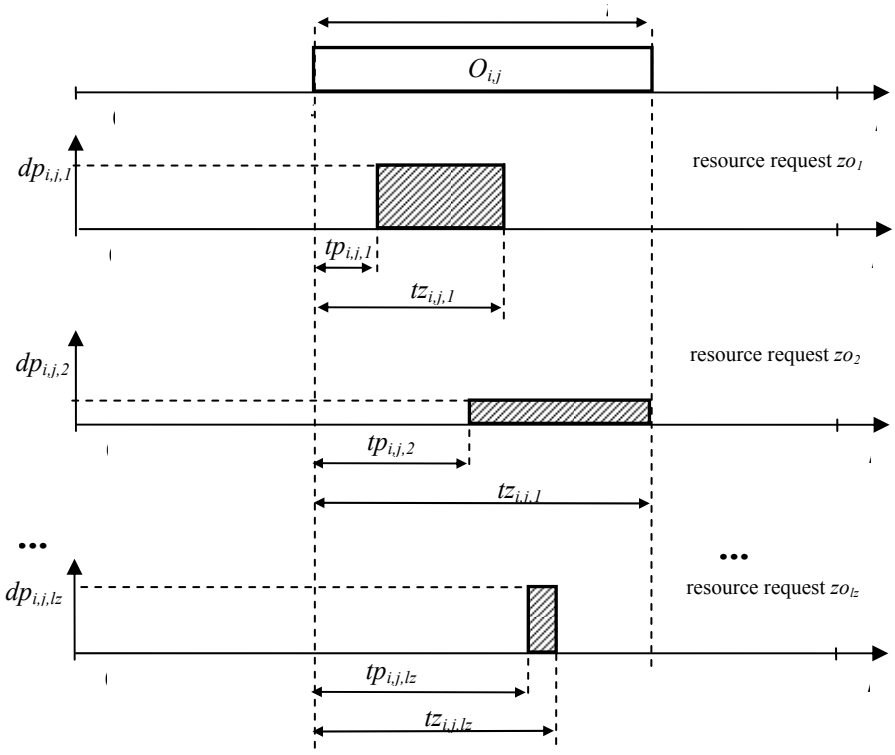


Fig. 2. Graphical illustration of parameters $x_{i,j}$, t_{ij}, $Tp_{i,j}$, $Tz_{i,j}$, $Dp_{i,j}$ specifying activity $O_{i,j} = (x_{i,j}, t_{ij}, Tp_{i,j}, Tz_{i,j}, Dp_{i,j})$

That is assumed the activities are no-preemptive and more than one resource type may be required by every activity in the project and the availability of each type is time-window

limited. For instance the sequence $Dp_{1,2} = (1, 2, 1, 0, 0)$, means that the activity $O_{1,2}$ requires just first three resources and the resource 4th and 5th are do not needed.

Consequently, each operation $O_{i,j} = (x_{i,j}, t_{ij}, Tp_{i,j}, Tz_{i,j}, Dp_{i,j})$ is specified by the following sequences of:

- moments the operations start their execution in the project P_i:
  $X_i = (x_{i,1}, x_{i,2},\ldots, x_{i,lo_i}),\ 0 \le x_{i,j} < h;\ i = 1, 2,\ldots, lp;\ j = 1, 2,\ldots, lo_i,$
- operation times associated to operations in the project P_i: $T_i=(t_{i,1},t_{i,2},\ldots, t_{i,lo_i})$,
- moments the j-th resource is allocated to the k-th operation in the project P_i:
  $TP_{i,j} = (tp_{i,1,j}, \ldots,tp_{i,k,j},\ldots, tp_{i,lo_i,j})$,
- moments the j-th resource is released by the k-th opration in the P_i:
  $TZ_{i,j}= (tz_{i,1,j}, tz_{i,2,j},\ldots , tz_{i,lo_i,j})$,
- amounts of the j-th resources allocated to the k-th operation in the project P_i:
  $DP_{i,j} = (dp_{i,1,j}, dp_{i,2,j},\ldots , dp_{i,lo_i,j})$.

Moreover, the following constraints are assumed as well:

1) ***Operations order constraints:***

The activity networks considered provide the order in which project's operations are executed, where operations state for the nodes and arcs determine an order of operations performance. Consequently, the following constraints are considered:

- the k-th operation follows the i-th one:
  $x_{i,j} + t_{ij} \le x_{i,k}$, (4)
- the k-th operation is follows other operations:
  $x_{i,j} + t_{i,j} \le x_{i,k}, x_{i,j+1} + t_{i,j+1} \le x_{i,k}, \ldots, x_{i,j+n} + t_{i,j+n} \le x_{i,k}$, (5)
- the k-th operation is followed by other operations:
  $x_{i,k} + t_{i,k} \le x_{i,j}, x_{i,k} + t_{i,k} \le x_{i,j+1}, \ldots, x_{i,k} + t_{i,k} + \le x_{i,j+n}$. (6)

For instance, in case of activity networks from Fig. 1 the constraints (4), (5), and (6) are as follows (see Table 1).

Table. 1 Operations order constraints for activity networks from Fig. 1.

Project $_1$	**Project P_2**	**Project P_3**
$x_{1,3} \ge x_{1,1} + t_{1,1}$	$x_{2,3} \ge x_{2,1} + t_{2,1}$	$x_{3,3} \ge x_{3,1} + t_{3,1}$
$x_{1,4} \ge x_{1,1} + t_{1,1}$	$x_{2,4} \ge x_{2,1} + t_{2,1}$	$x_{3,3} \ge x_{3,2} + t_{3,2}$
…	**…**	**…**
$x_{1,20} \ge x_{1,12} + t_{1,12}$	$x_{2,20} \ge x_{2,9} + t_{2,9}$	$x_{3,20} \ge x_{3,6} + t_{3,6}$

2) ***Resource availability rate constraints****:*

The resources requested by an operation usually have to follow some proportion limits, i.e. limits determining the mutual rates the resources can be allocated to the activity $O_{i,j}$. The constraints considered are determined by the formulae (7):

$$dp_{i,j,1} + dp_{i,j,2} + dp_{i,j,3} + \ldots + dp_{i,j,lz} = rm_{i,j}, \quad (7)$$

where: lz – a number of renewable resources, i=1,2,…,lp; j = 1, 2,…, lo_i,
lp- is a number of projects, lo_i - the number of the i-th projects' activities.

So, the sum of resources amount allocated to $O_{i,j}$ is limited by $rm_{i,j}$. In general case, to each P_i the following sequence corresponds $RM_i = (rm_{1,1}, rm_{1,2}, \ldots, rm_{i,loi})$.

3) *Resource conflict constraints*:

In order to avoid deadlocks the constraints providing conflicts resolution, i.e., avoiding the occurrence of closed loop resources request, are considered. The constraints guarantee at any moment within the assumed time horizon H the sum of allocated amounts of a given resource do not exceed its current availability $zo_{i,j}$. So, for each the k-th resource, at the moment $v \in H$, the following inequalities hold (8):

$$\sum_{i=1}^{lp}\sum_{j=1}^{lo_i}\left[dp_{i,j,k} \cdot \bar{1}(v, x_{i,j} + tp_{i,j,k}, x_{i,j} + tz_{i,j,k})\right] \le zo_{k,v} \qquad (8)$$

where: lp – a number of projects, lo_i – a number of operations contained by the i-th project, $dp_{i,j,k}$ – an amount of the k-th resource allocated by $O_{i,j}$,

$\bar{1}(v,a,b)$ – the unit step function of the resource allocation $\bar{1}(v,a,b) = 1(v-a) - 1(v-b)$, where: $1(v)$ - a unit step function defined as follows (see Fig.2)

$$1(v) = \begin{cases} 0 \ dla\ v < 0 \\ 1 \ dla\ v \ge 0 \end{cases}$$

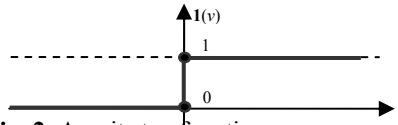


Fig. 2. A unit step function

The constraint (8) limits the summary amount of the k-th resource requested by $O_{i,j}$ Fig. 3 illustrates the case the constraint (8) concerns the activities $O_{1,1}$, $O_{1,2}$, $O_{1,3}$.

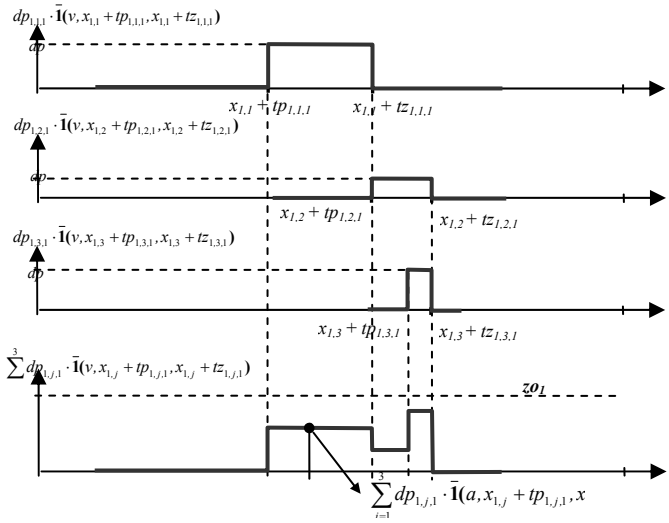


Fig. 3. Illustration of the constraint (8) concerning the unique resource and activities $O_{1,1}$, $O_{1,2}$, $O_{1,3}$.

That can be shown [5], the constraint (8) implies the set of inequalities (9) for the projects portfolio considered.

In constraints (9), the sum of requested resources is calculated only at moments corresponding to the ones $x_{i,j} + tp_{i,j}$, when resources are allocated to subsequent operations. Amount of available resources may change within the time horizon H. So, in order to avoid the amount of allocated resources exceeds the amount of available resources (similarly to (8)) the constraints associated to the moments $vp_{k,i}$ the resources availability change, are introduced (10).

$$\begin{cases} \sum_{i=1}^{lp}\sum_{j=1}^{lo_i}\left[dp_{i,j,k}\cdot 1(x_{1,1}+tp_{1,1,k}, x_{i,j}+tp_{i,j,k}, x_{i,j}+tz_{i,j,k})\right] \le zo_{k,x_{1,1}} \\ \sum_{i=1}^{lp}\sum_{j=1}^{lo_i}\left[dp_{i,j,k}\cdot 1(x_{1,2}+tp_{1,2,k}, x_{i,j}+tp_{i,j,k}, x_{i,j}+tz_{i,j,k})\right] \le zo_{k,x_{1,2}} \\ \dots \\ \sum_{i=1}^{lp}\sum_{j=1}^{lo_i}\left[dp_{i,j,k}\cdot 1(x_{1,lo_1}+tp_{1,lo_1,k}, x_{i,j}+tp_{i,j,k}, x_{i,j}+tz_{i,j,k})\right] \le zo_{k,x_{1,lo_1}} \\ \sum_{i=1}^{lp}\sum_{j=1}^{lo_i}\left[dp_{i,j,k}\cdot 1(x_{2,1}+tp_{2,1,k}, x_{i,j}+tp_{i,j,k}, x_{i,j}+tz_{i,j,k})\right] \le zo_{k,x_{2,1}} \\ \dots \\ \sum_{i=1}^{lp}\sum_{j=1}^{lo_i}\left[dp_{i,j,k}\cdot 1(x_{lp,2}+tp_{lp,2,k}, x_{i,j}+tp_{i,j,k}, x_{i,j}+tz_{i,j,k})\right] \le zo_{k,x_{lp,2}} \end{cases} \quad (9)$$

for: $k = 1,2, \ldots, lz$, where lz – a number of renewable resources.

$$\begin{cases} \sum_{i=1}^{lp}\sum_{j=1}^{lo_i}\left[dp_{i,j,k}\cdot 1(vp_{k,1}, x_{i,j}+tp_{i,j,k}, x_{i,j}+tz_{i,j,k})\right] \le zo_{k,vp_{k,1}} \\ \sum_{i=1}^{lp}\sum_{j=1}^{lo_i}\left[dp_{i,j,k}\cdot 1(vp_{k,2}, x_{i,j}+tp_{i,j,k}, x_{i,j}+tz_{i,j,k})\right] \le zo_{k,vp_{k,2}} \\ \dots \\ \sum_{i=1}^{lp}\sum_{j=1}^{lo_i}\left[dp_{i,j,k}\cdot 1(vp_{k,q}, x_{i,j}+tp_{i,j,k}, x_{i,j}+tz_{i,j,k})\right] \le zo_{k,vp_{k,q}} \end{cases} \quad (10)$$

for $k = 1,2, \ldots, lz$,

where: lz – a number of renewable resources, $vp_{k,i}$ – its the i-th moment the available amount of the k-the resource changes, $zo_{k,(vp_{k,i}-1)} - zo_{k,vp_{k,i}} \neq 0$. For the k-th resource the values $vp_{k,i}$ form the following sequence $Vp_k = (vp_{k,1}, vp_{k,2}, \ldots, vp_{k,q})$ items of which are the moments corresponding to zo_k changes, q – a number of elements in the sequence Vp_k.

2.3 Problem statement

The introduced model provides the formal framework enabling to state the problem considered. Given time horizon H, the projects portfolio P, the set of resources and their availabilities Zo within H. Given are sequences T_i, $TP_{i,j}$, $TZ_{i,j}$, $DP_{i,j}$. The following question should be answered:

- Does a given resources allocation guarantee the project portfolio makespan do not exceed the deadline H? Response to this question results in determination of the sequences: $X_1, X_2, \ldots, X_{lp}$.

- Does there exists such resources allocation guaranteeing the project portfolio makespan do not exceed the deadline H? Response to this question results in determination of the sequences: $X_1, X_2,\ldots,X_{lp}$ as well as the sequences $DP_{1,1}, DP_{1,2}, \ldots,DP_{1,lz}, DP_{2,1}, DP_{2,2}, \ldots,DP_{2,lz}, \ldots, DP_{lp,1}, DP_{lp,2}, \ldots,DP_{lp,lz}$.

The following remarks should be stated: - the problems considered are formulated in terms of variables and their discrete domains as well as sets of constraints; - the sufficient conditions guaranteeing the there exist a admissible solution to the above problems should be known; - the question stated above correspond to the straight and reverse problems of multi-product scheduling.

3 CONSTRAINT SATISFACTION PROBLEM

Constraint programming (*CP*) is an emergent software technology for declarative description and effective solving of large combinatorial problems, especially in the areas of integrated production planning. Since a constraint can be treated as a logical relation among several variables, each one taking a value in a given (usually discrete) domain, the idea of *CP* is to solve problems by stating the requirements (constraints) that specify a problem at hand, and then finding a solution satisfying all the constraints [4]. From this perspective, *CP* can be considered as a pertinent framework for the development of decision making software aimed at supporting SMEs in the course of projects portfolio prototyping. Because of its declarative nature, it is particularly useful for applications where it is enough to state *what* has to be solved instead *how* to solve it.

More formally, *CP* is a framework for solving combinatorial problems specified by pairs: **<a set of variables and associated domains, a set of constraints restricting the possible combinations of the values of the variables>.** So, the constraint satisfaction problem (*CSP*) [4], [8], is defined as follows:

$$CS=((A, D), C) \tag{11}$$

where: $A = \{a_1, a_2,...,a_g\}$ – a finite set of discrete decision variables,
$D = \{D_i \,|D_i=\{d_{i1}, d_{i2},...,d_{ij},...,d_{ih}\}, i = 1,..,g\}$ – a family of finite variable domains, and the finite set of constraints
$C = \{C_i|i=1,..,L\}$ – a finite set of constraints limiting the decision variable values.

The solution to the *CS* is a vector $(d_{1i},d_{2k},...,d_{nj})$ such that the entry assignments satisfy all the constraints *C*. So, the task is to find the values of variables satisfying all the constraints, i.e., a feasible valuation. Generally, the constraints can be expressed by arbitrary analytical and/or logical formulas as well as bind variables with different non-numerical events.

The inference engine consists of the following two components: constraint propagation and variable distribution. Constraints propagation uses constraints to prune the search space. The aim of propagation techniques is to reach a certain level of consistency in order to accelerate search procedures by drastically reducing the size of the search tree. The constraints propagation executes almost immediately. What limits the size of the problem in practical terms is the variable distribution phase, which employs the backtracking-based search and is very time consuming as a result. Consequently, the crucial factor determining the practical usability of *CP/CLP*-based *DSS* is the provision of a variable distribution strategy that guarantees a feasible solution obtained in an on-line mode.

The declarative character of Constraint Programming (*CP*) languages and their high efficiency in solving combinatorial problems offer an attractive alternative to the currently

available systems of computer-integrated management [6], [7], that employ operation research techniques.

4 ILLUSTRATIVE EXAMPLE

For illustration of the *CP* based approach capability let us consider the projects portfolio $P = \{P_1,P_2,P_3\}$ as stated in the Section 2.1. Let us assume the resources are allocated, and released at the moments corresponding to the beginning and completion of operations. Therefore, the sequences $TP_{1,1}$, $TP_{1,2}$, $TP_{2,1}$, $TP_{2,2,}$ $TP_{3,1}$, $TP_{3,2}$ considered are as follows: $P_{1,2} = TP_{1,1} = TP_{2,2} = TP_{2,1} = TP_{3,2} = TP_{3,1} = (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$.
Consequently: $TP_{1,1} = T_1$, $TP_{1,2} = T_1$, $TP_{2,1} = T_2$, $TP_{2,2} = T_2$, $TP_{3,1} = T_3$, $TP_{3,2} = T_3$. The considered sequences: $DP_{1,1}$, $DP_{1,2}$, $DP_{2,1}$, $DP_{2,2}$, $DP_{3,1}$, $DP_{3,2}$ are given in the Section 2.1.

4.1 The straight problem case

For a given set of renewable resources and their availability $Zo = (zo_1, zo_2)$, see the Section 2.1, as well as the time horizon $H = [0, 36]$, $H \subset N$, consider the question: Does the given projects portfolio can be completed within the time horizon H?

Response to this question requires the following sequences calculation: $X_1=(x_{1,1},x_{1,2},\ldots,x_{1,20})$, $X_2= (x_{2,1}, x_{2,2},\ldots, x_{2,20})$, $X_3= (x_{3,1}, x_{3,2},\ldots,x_{3,20})$, where: $0 \le x_{i,j} < 36$; $i = 1,2,3$; $j = 1,2,\ldots,20$.

The sequences X_1, X_2, X_3 sought have to follow constraints imposed by activities order and resources allocation, particularly:

- Activities precedence order constraints:
  For activity networks from Fig. 1 the constraints following (4), (5), (6) are determined as in the Table 1.
- Constraints imposed on resources conflict resolution:
  Taking into account the sequences $TP_{i,j}$ (particularly their zero-value entries) as well as equality of sequences $TZ_{i,j}$ and T_i, the constraints (due to (9)) imposed on shared resources are determined by (12) and (13):

$$\begin{cases} \sum_{i=1}^{3}\sum_{j=1}^{20}\left[dp_{i,j,1}\cdot 1(x_{1,1},x_{i,j},x_{i,j}+t_{i,j})\right] \le zo_{1,x_{1,1}} \\ \cdots \\ \sum_{i=1}^{3}\sum_{j=1}^{20}\left[dp_{i,j,1}\cdot 1(x_{1,20},x_{i,j},x_{i,j}+t_{i,j})\right] \le zo_{1,x_{1,20}} \\ \sum_{i=1}^{3}\sum_{j=1}^{20}\left[dp_{i,j,1}\cdot 1(x_{2,1},x_{i,j},x_{i,j}+t_{i,j})\right] \le zo_{1,x_{2,1}} \\ \cdots \\ \sum_{i=1}^{3}\sum_{j=1}^{20}\left[dp_{i,j,1}\cdot 1(x_{3,20},x_{i,j},x_{i,j}+t_{i,j})\right] \le zo_{1,x_{3,20}} \end{cases} \quad (12)$$

$$\begin{cases} \sum_{i=1}^{3}\sum_{j=1}^{20}\left[dp_{i,j,2}\cdot 1(x_{1,1},x_{i,j},x_{i,j}+t_{i,j})\right] \le zo_{2,x_{1,1}} \\ \cdots \\ \sum_{i=1}^{3}\sum_{j=1}^{20}\left[dp_{i,j,2}\cdot 1(x_{1,20},x_{i,j},x_{i,j}+t_{i,j})\right] \le zo_{2,x_{1,20}} \\ \sum_{i=1}^{3}\sum_{j=1}^{20}\left[dp_{i,j,2}\cdot 1(x_{2,1},x_{i,j},x_{i,j}+t_{i,j})\right] \le zo_{2,x_{2,1}} \\ \cdots \\ \sum_{i=1}^{3}\sum_{j=1}^{20}\left[dp_{i,j,2}\cdot 1(x_{3,20},x_{i,j},x_{i,j}+t_{i,j})\right] \le zo_{2,x_{3,20}} \end{cases} \quad (13)$$

Constraints imposed due to (10) are as follows (14) and (15):

$$\begin{cases} \sum_{i=1}^{3}\sum_{j=1}^{20}\left[dp_{i,j,1}\cdot 1(11, x_{i,j}, x_{i,j}+t_{i,j})\right] \le zo_{1,11} \\ \sum_{i=1}^{3}\sum_{j=1}^{20}\left[dp_{i,j,1}\cdot 1(21, x_{i,j}, x_{i,j}+t_{i,j})\right] \le zo_{1,21} \end{cases} \quad (14) \qquad \begin{cases} \sum_{i=1}^{3}\sum_{j=1}^{20}\left[dp_{i,j,2}\cdot 1(9, x_{i,j}, x_{i,j}+t_{i,j})\right] \le zo_{2,9} \\ \sum_{i=1}^{3}\sum_{j=1}^{20}\left[dp_{i,j,2}\cdot 1(13, x_{i,j}, x_{i,j}+t_{i,j})\right] \le zo_{2,13} \end{cases} \quad (15)$$

Of course, the sequences sought have to follow both the operations order and resource conflict constraints. In the case considered, 120 constraints (12), (13), have been taken into account, each one consisting of 60 components.

That is assumed the moments of beginning of operations ending the projects, i.e., $O_{1,20}$, $O_{2,20}$, $O_{2,20}$ have to follow the constraints (16), Fig. :

$$x_{1,20}+t_{1,20} \le h, \quad x_{2,20}+t_{2,20} \le h, \quad x_{3,20}+t_{3,20} \le h. \quad (16)$$

The stated above constraint satisfaction problem has been implemented in Oz Mozart environment. The first feasible solution:

X_1 = (0, 4, 1, 6, 10, 6, 12, 12, 10, 15, 15, 19, 6, 1, 7, 4, 9, 9, 14, 20),
X_2 = (0, 10, 3, 2, 11, 19, 8, 10, 24, 5, 8, 11, 16, 15, 11, 24, 18, 26, 14, 30),
X_3 = (0, 0, 4, 12, 16, 26, 6, 9, 17, 17, 22, 0, 7, 13, 18, 22, 1, 25, 25, 32),

has been obtained in 137 seconds, after the1993-th step (the processor used: AMD Athlon(tm)XP 2500+ 1.85 GHz and RAM 1,00 GB RAM, Fig. 4).

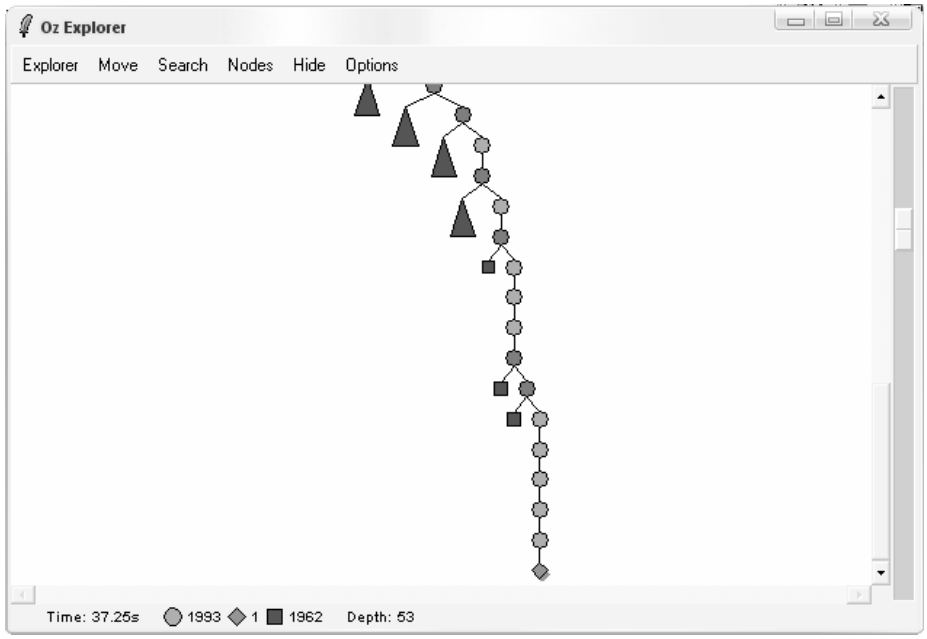
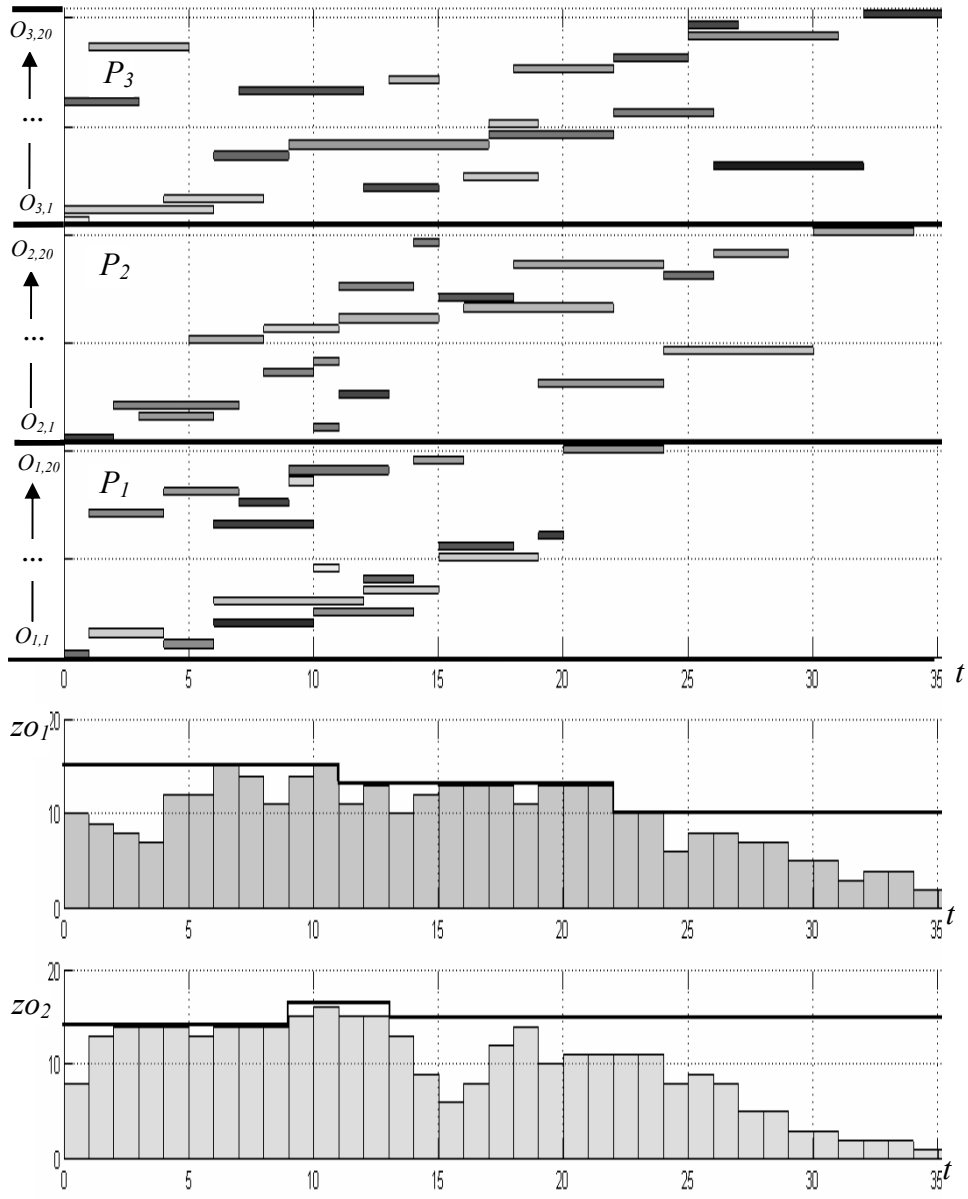


Fig. 4. Searching tree for straight problem and corresponding to its solution see Fig. 7.



Legend:

available amounts of renewable resources

Rys. 5. Gantt's charts of P_1, P_2, P_3 projects execution and resources zo_1, zo_2 allocation for the straight problem.

4.2 The reverse problem case

Given projects portfolio and sequences T_1, T_2, T_3 as well as $TZ_{1,1}$, $TZ_{1,2}$, $TZ_{2,1}$, $TZ_{2,2}$, $TZ_{2,1}$, $TZ_{2,2}$ see the Subsection 4.1. The resources availability limited by constraints (4) are determined by the following sequences :

$RM_1 = (4, 3, 3, 5, 2, 3, 3, 2, 4, 4, 3, 2, 3, 4, 2, 5, 3, 3, 3, 4)$,
$RM_2 = (4, 2, 3, 4, 2, 3, 3, 3, 4, 4, 3, 3, 3, 2, 2, 3, 4, 3, 4, 2)$,
$RM_3 = (5, 2, 3, 4, 2, 3, 3, 3, 4, 4, 4, 2, 2, 4, 2, 3, 3, 3, 4, 4)$.

For a given set of renewable resources and their availability $Zo = (zo_1, zo_2)$, see the Section 2.1, as well as the time horizon $H = [0, 36]$, $H \subset N$, consider the question: Does there exist the resource allocation guaranteeing the projects portfolio completion time will not exceed the assumed time horizon H?
Response to this question requires the following sequences calculation: X_1, X_2, X_3:

$X_1 = (x_{1,1}, x_{1,2}, \ldots, x_{1,20})$, $X_2 = (x_{2,1}, x_{2,2}, \ldots, x_{2,20})$, $X_3 = (x_{3,1}, x_{3,2}, \ldots, x_{3,20})$,

where: $0 \le x_{i,j} < 36$; $i = 1,2,3$; $j = 1,2,\ldots,20$, and

$DP_{1,1} = (dp_{1,1,1}, dp_{1,2,1}, \ldots, dp_{1,10,1})$, $DP_{1,2} = (dp_{1,1,2}, dp_{1,2,2}, \ldots, dp_{1,10,2})$,
$DP_{2,1} = (dp_{2,1,1}, dp_{2,2,1}, \ldots, dp_{2,10,1})$, $DP_{2,2} = (dp_{2,1,2}, dp_{2,2,2}, \ldots, dp_{2,10,2})$,

where: $0 \le dp_{i,j,k} < 4$; $i = 1, 2$; $j = 1, 2, \ldots, 10$, $k = 1,2$.

Of course, the sequences sought have to follow both the activities order and resource conflict constraints. The activities order constraints are given in the Table 1, and the resource availability constraints, determined due to (7) and taking into account sequences RM_1, RM_2, RM_3, are given in the Table 2.

Tab. 2 Resource availability constraints, determined due to (7).

Project P_1	**Project P_2**	**Project P_3**
$dp_{1,1,1}+dp_{1,1,2} = 4$	$dp_{2,1,1}+dp_{2,1,2} = 4$	$dp_{2,1,1}+dp_{2,1,2} = 5$
$dp_{1,2,1}+dp_{1,2,2} = 3$	$dp_{2,2,1}+dp_{2,2,2} = 2$	$dp_{2,2,1}+dp_{2,2,2} = 2$
...	**...**	**...**
$dp_{1,20,1}+dp_{1,20,2} = 4$	$dp_{2,20,1}+dp_{2,20,2} = 2$	$dp_{2,20,1}+dp_{2,20,2} = 4$

Resource conflict constraints follow the formulas: (9), (10).

The stated above constraint satisfaction problem has been implemented in Oz Mozart environment [9]. The first feasible solution below:

$X_1 = (0, 4, 1, 6, 10, 6, 12, 12, 10, 15, 15, 19, 6, 1, 7, 4, 9, 9, 14, 20)$,
$X_2 = (0, 7, 2, 2, 8, 10, 5, 7, 21, 2, 5, 8, 12, 12, 8, 18, 15, 21, 11, 27)$,
$X_2 = (0, 0, 1, 12, 15, 26, 6, 9, 17, 17, 22, 0, 7, 10, 12, 18, 1, 21, 21, 32)$,
$DP_{1,1} = (3, 2, 1, 2, 1, 2, 1, 1, 2, 3, 2, 1, 2, 2, 1, 2, 2, 1, 2, 2)$,
$DP_{1,2} = (1, 1, 2, 3, 1, 1, 2, 1, 2, 1, 1, 1, 1, 2, 1, 3, 1, 2, 1, 2)$,
$DP_{2,1} = (2, 1, 2, 2, 1, 2, 2, 1, 2, 2, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1)$,
$DP_{2,2} = (2, 1, 1, 2, 1, 1, 1, 2, 2, 2, 2, 1, 2, 1, 1, 2, 2, 2, 3, 1)$,
$DP_{3,1} = (3, 1, 2, 1, 1, 2, 2, 1, 1, 2, 1, 1, 1, 2, 1, 1, 2, 1, 1, 2)$,
$DP_{3,2} = (2, 1, 1, 3, 1, 1, 1, 2, 3, 1, 3, 1, 1, 2, 1, 2, 1, 2, 3, 1)$,

has been obtained in 32 ms, after the102-th step (the processor used: AMD Athlon(tm)XP 2500+ 1.85 GHz and RAM 1,00 GB RAM).

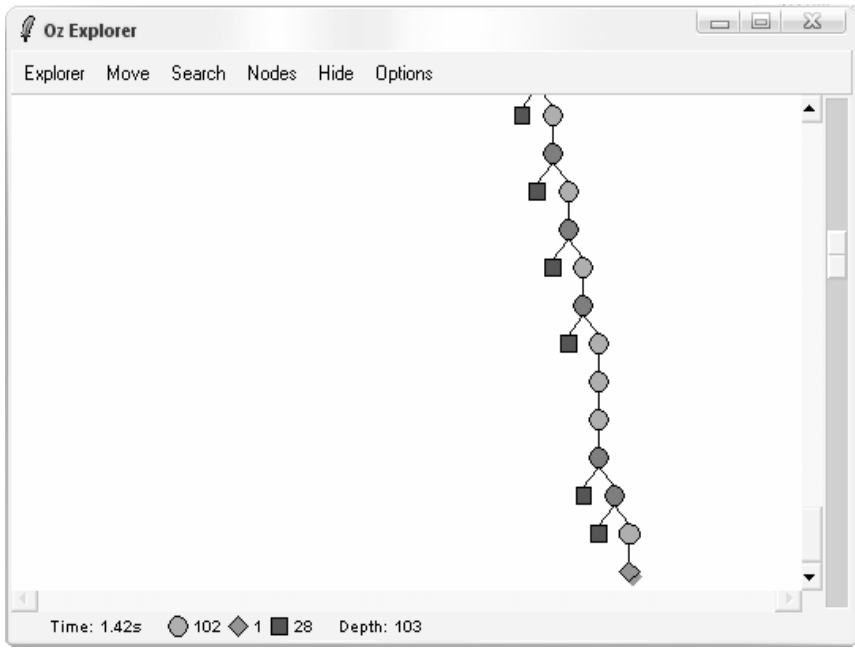


Fig. 8. Searching tree for reverse problem and corresponding to its solution see Fig. 7.

The graphical illustration of the solution obtained is show in Fig. 9.

In order to guarantee the response times will enable for a DSS to be useful in an on-line mode in cases of the real life cases the serialization concept can be implemented. Serialization allows to write much less code that can handle different objects, without knowing in advance what particular instances will be of user's further interest. Therefore the obtained script can be used as an object taken directly from a higher level. In other words serialization results in implementation of the predefined models of constrains involved in the problem in hand.

In considered case of projects scheduling the two kinds of serialization may be used: for unary and cumulative scheduling [11]. However, because the first one is dedicated to the cases the only one kind of resources can be simultaneously allotted to an activity let us concentrate on the second one.

5. SERIALIZATION

In order to illustrate an idea standing behind the concept of serialization let us consider the following example. Given are two discrete renewable resources. The total number of units of the first resource available within the time horizon *{0,...,H}* is limited by 5, and by 2 in the second case. Given activity durations and amounts of resources required be activities (see Table 3). Consider the question: Does there exists such resources allocation guaranteeing the all activities will completed within the assumed time horizon *{0,...,H}*.

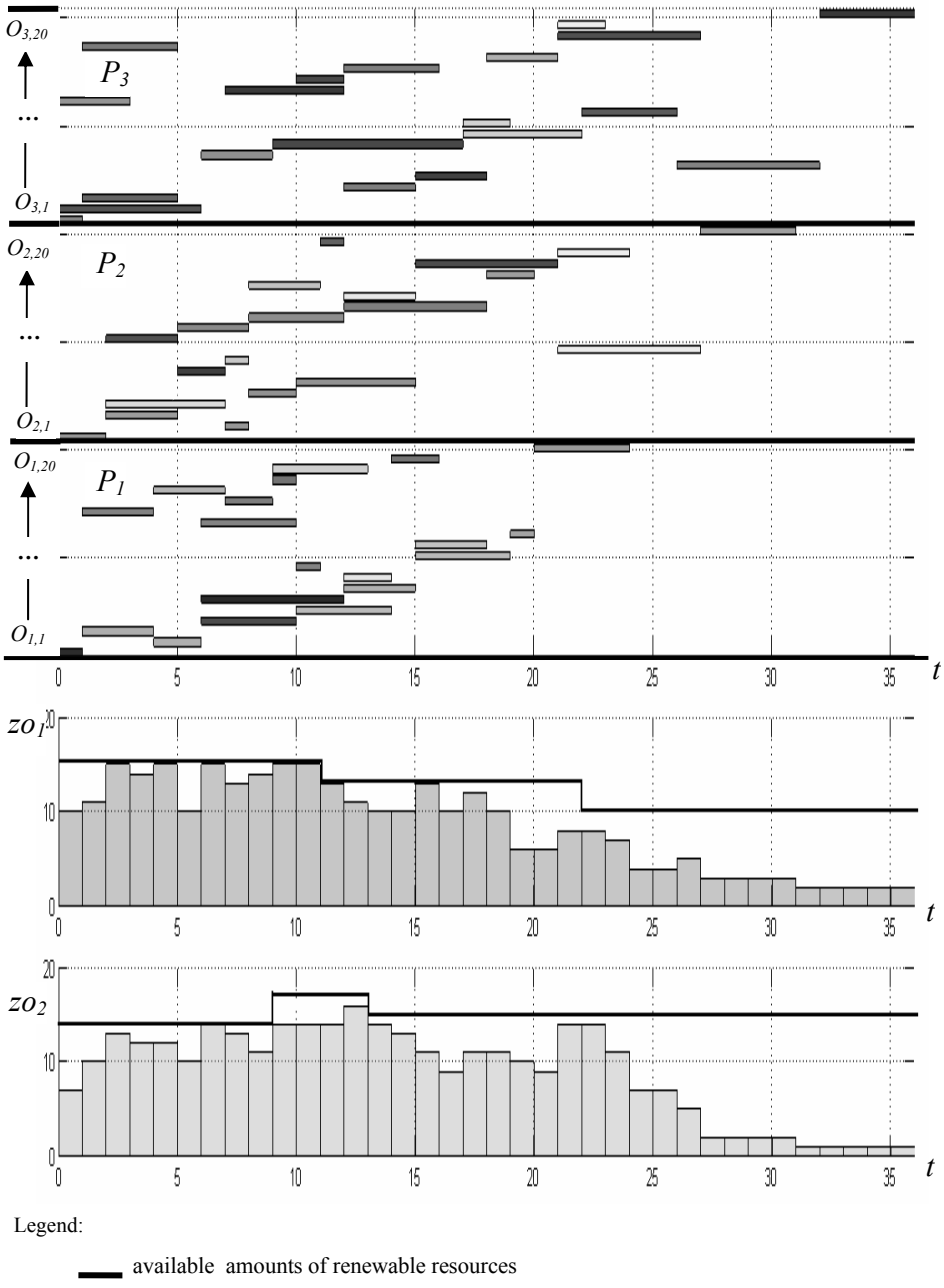


Fig. 9. Gantt's charts of P_1, P_2, P_3 projects execution and resources zo_1, zo_2 allocation for the reverse problem.

An Oz Mozart implementation of the above problem formulated in terms of CSP provides the solution shown in Fig. 10.

Tab. 3 Activity durations and amounts of resources required be activities.

Activity	Resource	Time	Units number of the resource required
$O_{1,1}$	1	5	5
$O_{1,2}$	1	2	3
$O_{1,3}$	2	7	2
$O_{1,4}$	2	4	1
$O_{1,5}$	2	3	1

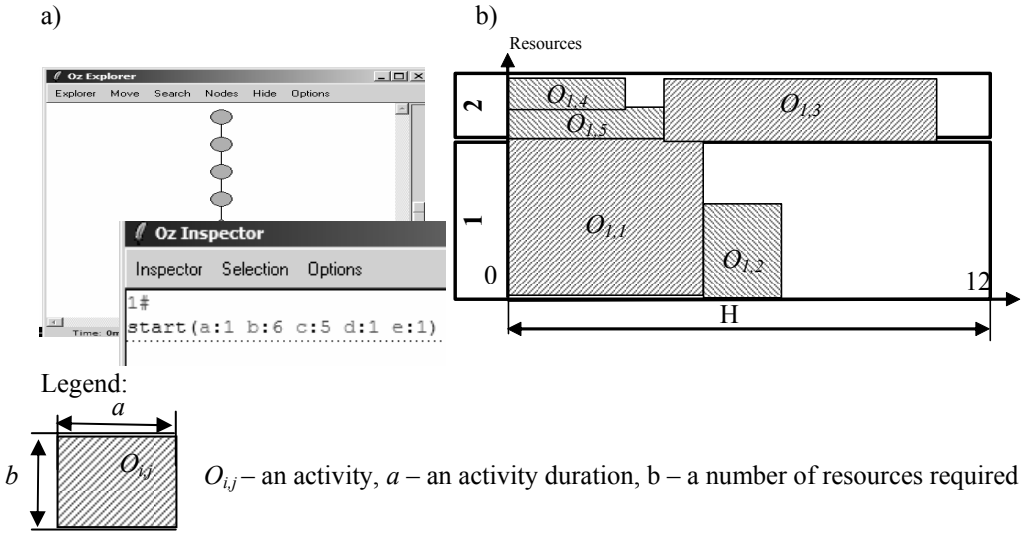


Fig. 10. Serialization effect; a) solution in Oz Mozart, b) Gantt's chart of the solution

Fig. 11 provides the script implementing cumulative scheduling serializator. That should be noted that its usage require just some data declaration as in the case considered: amounts of resources, activities duration, resources required by activities, and numbers of resource units allotted, i.e. decision variables: Cap, Dur, Tasks i Use. Additionally the variable Stat determining starting times of activities is declared as well. That should be noted there is not need to implement the constraints (6), (7), the function Schedule.cumulative implementation is enough. In other words In the course of serialization besides of implementation on "the lower level" of predefined task oriented models some heuristic rules are employed as well. Such tusk specific orientation perspective results in more efficient (faster) data processing, however in the cost of dedicated (i.e., narrowed) domain of applications.

In order to illustrate the benefits following from the implementation of the serialization concept, let us consider the case stated in the Section 4.1, i.e. regarding the straight version of the project portfolio scheduling problem.

```
declare

proc{Wynik Start}

   % deklaracja zmiennych
   % pojemnosci zasobow z1 i z2
   Cap=pz(z1:5 z2:2)
   % czas trwania operacji
   Dur=dur(o11:5 o12:2 o13:7 o14:4 o15:3)
   % przydzial operacji do zasobów
   Tasks=tasks(z1:[o11 o12] z2:[o13 o14 o15])
   % Zuzycie zasobów przez operacje
   Use=use(o11:5 o12:3 o13:2 o14:1 o15:1)
   % Zmienna decyzyjna
   Start={FD.record start [o11 o12 o13 o14 o15] 0#9}
   % koniec defniowania zmiennych
in
   % serializator
   {Schedule.cumulative Tasks Start Dur Use Cap}

   % dystrybucja
   {FD.distribute ff Start}
end
{ExploreOne Wynik}
```

Fig. 11. Oz Mozart's scripts describing Cumulative serializer implementation

Because usage of the Schedule.cumulative function requires the resources amount have to constant within considered the time horizon, the times the resources amount changes are treated as additional (dummy) activity requiring or releasing particular resource. The searching tree providing the solution to the above problem is show in the Fig. 12. The corresponding Gantt's chart is show in Fig. 13.

5 CONCLUDING REMARKS

Proposed approach to projects portfolio prototyping provides the framework allowing one to take into account both: straight and reverse approach to multi-product project-like scheduling. This advantage can be seen as a possibility to response (besides of standard questions: Is it possible to complete a projects portfolio at a scheduled project deadline?) to the questions like: What values and of what variables guarantee the projects portfolio will completed due to assumed values of set of performance indexes?

Proposed approach provides the framework allowing one to take into account both: the sufficient conditions (guaranteeing the admissible solutions there exist) and choosing the best

solution on the basis of chosen evaluation criteria. It can also be considered as a contribution to project-driven production flow management applied in make-to-order manufacturing as well as for prototyping of the virtual organization structures.

REFERENCES

1. Archer, N., and F. Chasemzadech, An integrated framework for project portfolio selection. Int. Journal of Project Management, Vol. 17, No. 4, 1999; 207-216.
2. Banaszak Z., Zaremba M. and Muszyński W., *CP-based decision making for SME*. In: Preprints of the 16th IFAC World Congress (Eds P. Horacek, M. Simandl), P. Zitek, DVD, 2005,Prague, Czech Republic.
3. Banaszak Z., *CP*-based decision support for project-driven manufacturing. In: Perspectives in Modern Project Scheduling, (Józefowska J. and J. Węglarz (Ed)), International Series in Operations Research and Management Science, Vol. 92, Springer Verlag, New York, 2006; 409-437.
4. Barták R. Incomplete Depth-First Search Techniques: A Short Survey, Proceedings of the 6th Workshop on Constraint Programming for Decision and Control, Ed. Figwer J., 2004; 7-14.
5. Bocewicz G., Banaszak Z., Wójcik R.: Design of admissible schedules for AGV systems with constraints: a logic-algebraic approach, In: Agent and Multi-Agent Systems: Technologies and Applications, Nguyen N.T., Grzech A., Howlett R.J., Jain L.C. (Eds.), Lecture Notes in Artificial Intelligence 4496, Springer-Verlag, Berlin, Heidelberg, 2007; 578-587.
6. Kis, T., G. Ërdos, A. Markus and J. Vancza, A Project-Oriented Decision Support System for Production Planning in Make-to-Order Manufacturing. ERCIN News, Vol. 58, 2004; 44-52.
7. Martinez, E. C., D. Duje and G.A. Perez, On performance modeling of project-oriented production. Computers and Industrial Engineering, Vol. 32, 1997; 509-527.
8. Puget J-F.: *A C++ Implementations of CLP,* Proceeding of SPICS 94, 1994.
9. Schulte CH., Smolka G., Wurtz J.: Finite Domain Constraint Programming in Oz, DFKI OZ documentation series, German Research Center for Artificial Intelligence, Stuhlsaltzenhausweg 3, D-66123 Saarbrucken, Germany, 1998.