PhD Paweł Sitek
Technical University of Kielce
Prof. Zbigniew Banaszak
Wyższa Szkoła Informatyki i Zarządzania Copernicus we Wrocławiu

CP-APROACH TO DECISION SUPPORT IN JOB SHOP SCHEDULING PROBLEMS WITH MANPOWER RESOURCES

In this paper we present the job-shop scheduling problems with manpower resources representation. The objective considered here is the minimization of the makespan. Allocation manpower to the jobs is more complicated than allocating machines, as different jobs and operation require different manpower. This problem is strongly NP-hard. We proposed the CP-approach to decision support in this environment. The most important features of CP are declarative problem modeling allowing a clean separation between the problem statement (variables and constraints) and the resolution of the problem (the constraint solving algorithms) propagation of the effects of decisions, and efficient search for feasible and optimal solution.

ZASTOSOWANIE METODY PROGRAMOWANIA Z OGRANICZENIAMI DO WSPOMAGANIA DECYZJI DLA PROBLEMÓW HARMONOGRAMOWANIA ZADAŃ W SYSTEMIE GNIAZDOWYM PRZY UWZGLĘDNIENIU ZASOBÓW SIŁY ROBOCZEJ

W pracy przedstawiono problem harmonogramowania zadań w systemie gniazdowym przy uwzględnieniu zasobów siły roboczej przy minimalizacji długości uszeregowania. Alokacja zasobów siły roboczej do zadań jest zagadnieniem bardziej skomplikowanym niż przydział zadań do maszyn, ponieważ różne zadania mogą wymagać do ich wykonania różnej liczby ludzi. Zadanie to należy do zadań silnie NP-trudnych. W pracy zaproponowano metodę programowania z ograniczeniami do wspomagania decyzji w przedstawionym problemie harmonogramowania zadań.

1. INTRODUCTION

Scheduling is the process of designing a procedure for a particular objective, specifying the sequence or time for each item in this procedure. Typical scheduling problems are railway time-tabling, project scheduling, production scheduling (job-shop and flow-shop), scheduling medical man shifts in a hospital. Application examples of scheduling

in computer systems are in FMS (Flexible Manufacturing Systems), robot activity scheduling, scheduling in networks and hard real-time scheduling. Further, there is a number of related problems, e.g., resource allocation in a job-shop scheduling. The scheduling problems with the resource-constrained availability are a very interesting area research for SME (Small and Medium-Sized Enterprises). In this kind of enterprises very common is using external resources. External resources can have radical influence whether or not the overall schedule is workable. One such limiting resource is manpower. Manpower resources often vary with time due to holidays and sickness, more thorough representation is therefore required in the job-shop environment. Some papers [1],[2],[3] deal with general issues concerning the modeling and resource-constrained project scheduling. Authors present a survey of models and algorithms for the discrete-continuous project-scheduling. The heuristic methods are proposed. In this paper we present the job-shop scheduling problems with manpower resources representation. Constraint programming approach to the job-shop scheduling problems with manpower resources representation is proposed. The objective considered here is the minimization of the makespan. This problem is strongly NP-hard.

2. CP-APPROACH TO DECISION SUPPORT IN JOB-SHOP SCHEDULING

The classical job-shop scheduling problem is defined as follows [4]. There are $j \in J$ jobs to be processed through $s \in S$ machines. Each job must pass through each machine exactly once. The processing of a job on a machine is called an operation $o \in O_j$ and requires a duration called the processing time p_{jos}. Technological constraints demand that each job should be processed through machines in a specific order. Each job should have a release time and deadline. The general problem in the above environment is to find a sequence in which jobs pass through the machines, which is compatible with the technological constraints and optimal with some performance criterion (very often makespan C_{max}). A variant of the described problem is the flow-shop scheduling problem, which arising if all jobs share the same processing order.

Job-shop scheduling problem is a prototypical problem for a number of problems arising in several disciplines. Apart from other problems in production scheduling such as assembly line balancing and flexible manufacturing systems, job-shop scheduling has a close correspondence to project scheduling, time-tabling of lectures, etc.

In the classic job-shop scheduling problem resources outside the schedule are not take into account but it is a serious problem in decision making process in practical scheduling problems. External resources can have radical influence whether or not the overall schedule is workable. In practice there are a number of other factors that must be taken into account in a typical job-shop but the manpower resources are the most important. Solving the job-shop scheduling problem with manpower resources one can support decision making process and answer basic support questions (fig.3):

- If we have limited manpower resources, how long will this work take?
- If we have unlimited manpower resources, how long will this work take?
- How much manpower will we allocate in order to meet our delivery schedule?

- Can we use some of our excess manpower to start the production of another production without impact on our promised delivery of the current production?

2.1 Illustrative example

Consider the job-shop scheduling example, where $j \in \{a,b,c\}$ jobs to be processed through $s \in \{s1,s2, s3\}$ machines. Each job must pass through each machine exactly ones. The processing of a job *a* on machine *s1* is an operation *a* (denoted *aa*), on machine *s2* is an operation *b* (denoted *ab*), on machine *s3* is an operation *c* (denoted *ac*) etc. Technical requirements demand that each job should be processed in a specific order. For this example order for each job can be stated as follow *j=a, {aa, ab, ac}; j=b, {bb, ba, bc}; j=c, {ca, cb, cc}*. Processing time p_{ab} is a duration of the processing operation *b* in job *a*. There are processing times for the illustrative example: $p_{aa}=4$, $p_{ab}=3$, $p_{ac}=4$, $p_{ba}=5$, $p_{bb}=4$, $p_{bc}=$, $p_{ca}=3$, $p_{cb}=4$, $p_{cc}=6$. The problem in this example is to find a sequence in which jobs *j=a, j=b, j=c* pass through the machines *s1, s2, s3,* which satisfy technological constraints and optimal with makespan. Firstly in this example, manpower resources were not taken into account The result is the optimal schedule (fig.1). Then the manpower requirement for the schedule calculated in the previous example has been taken into account (fig. 2). For each operation exception last of each job the manpower was two workers, for the last operation of each job manpower was one worker. The basic question is If we have limited manpower resources (for instance 3 workers), how long will this work take ?

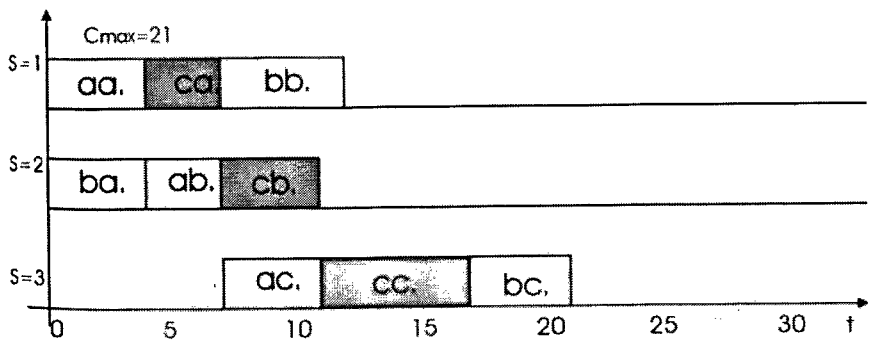

Fig. 1 Optimal schedule for illustrative example ($C_{max}=21$)

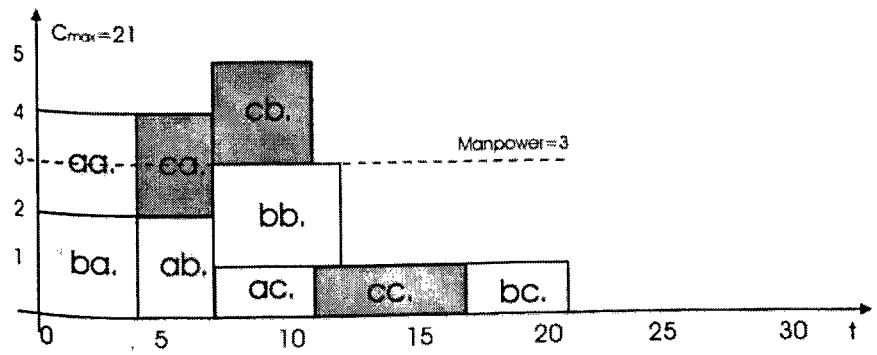

Fig. 3 Optimal schedule for illustrative example (C_{max}=21, manpower requirements are from 1 to 5)

2.2. Constraint programming (CP)

Constraint Programming (CP) is a problem solving method that was developed out of Logic Programming and Artificial Intelligence. The diversity of scheduling problems, the existence of many specific constraints (precedence, resource, capacity, etc.) in each problem and the efficient constraint based scheduling algorithms [5] make constraint programming a method of choice for the resolution of complex practical problems. In constraint programming approach to decision support in scheduling problems (Fig.3) the problem to be solved is represented in terms of decision variables and constraints on these variables.

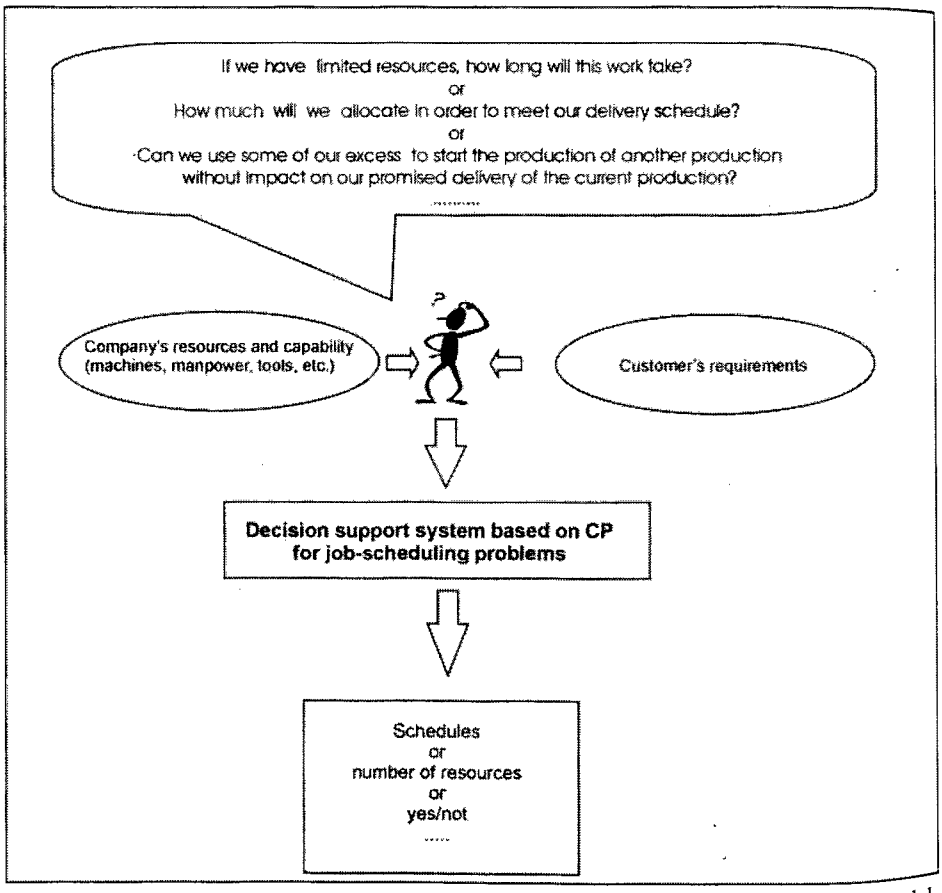

Fig. 3 Decision support system based on constraint programming for job-shop scheduling problems

2.3 Constraint programming modeling-CP model for job-shop scheduling with manpower resources representation

The problem model in CP contains declarations of all decision variables, including their domains. Additionally, a set of constraints called *conceptual constraints* imposed on those variables is determined. Another type of constraints, *actual constraints*, is used for increasing the effectiveness of searching for a problem solution. Frequently, such constraints are provided by the CP software producers as ready predicates attached to the system. While modeling problems in CP environment, variables 0-1, which are common in the MP – Mathematic Programming environment, can be used. A knowledge of the specificity of the problem variables and how they relate to one another makes the process more effective. Thus we can say that in CP the problem is viewed through its variables, the approach, which is different from that in MP modeling. Here, the problem is considered as a whole, the logic is modeled through allocation variables and transformed into linear constraints. Furthermore, MP does not make use of the variables' characteristics and relations at the stage of modeling. Basic problem solving techniques in CP are: constraint propagation, a variety of variables distribution methods, backtracking, etc. The knowledge both on the CP implementation environment, described in Chapter 4, as well as on the job-shop system was used while building the model. The job-shop system makes it possible to arrange freely particular jobs and operations on the machines, whereas within a given job it defines a fixed order of operations. Allocation of operations to machines is also defined. A knowledge of data structures and relations between them helps determine decision variable X_{jo} as a moment which begins the operation o of job j (in MP it would be $X_{jos}r$), and two dependent variables U_{jots}, K_{jor} (after specifying the X_{jo} their values are determined automatically – domain propagation), which in the MP environment would have a form of independent variables.

Table 1 Decision variables

X_{jo}	*a decision variable, which means the moment the operation begins* $o \in O_j$, *for job* $j \in J$, *on machine* $s \in S$, *using resource* $r \in R$;
U_{jots}	$U_{jots} = \begin{cases} 1 & \text{when at the moment } t, \text{ the operation } o \text{ of the job } j \text{ is performed on machine } s \\ 0 & \text{in all other cases} \end{cases}$
K_{jotr}	$K_{jotr} = \begin{cases} A_{j,o,r} & \text{when at the moment } t \text{ the operation of the job } j \text{ is performed using} \\ & \text{the resource type } r \\ 0 & \text{in all other cases} \end{cases}$

Basic constraints occurring in the job-shop scheduling and resource allocation are (1) sequence constraint, which enables the subsequent operation to start after the previous one has ended, (2) the constraint which prevents simultaneous access to a machine (at the t moment only one operation can be performed on machine s, and (3) limited resources in the system, necessary to perform further jobs. Constraints (2) and (3) are declarative and have a form of a loop.

$$X_{jo}+p_{jo} <= X_{j,o+1} \text{ for } j \in J \text{ i } o \in \{O_j - \text{final operation}\} \qquad (1)$$

Where: p_{jo} – period of time operation o is performed for job j on machine s

For $s \in S$		
	For $t = 1..T$	
		$\sum_{j \in J} \sum_{o \in O_j} U_{jots} \leq 1$

(3)

Dla t=1..T		
	Dla r ∈R	
		$\sum_{j \in J} \sum_{o \in O_j} K_{j,o,t,r} \leq N(r)$

Where: $N(r)$ number of accessible *r-type* resources

A_{jor} number of *r-type* resources necessary to perform operation o for job j

4. IMPLEMENTATION OF CP-APPROACH TO DECISION SUPPORT IN JOB-SHOP SCHEDULING

We propose Oz [6] as a platform to decision support in job-shop scheduling which integrates algorithms from OR and CP to achieve a combination of a high-level constraint language with efficient OR techniques. Oz is a concurrent constraint language providing for object-oriented, functional and constraint programming.

The unique advantages of Oz are [7]:*Expressiveness, Programmable Search. Modularity, Openness*. For the purpose of the implementing of model (1) .. (3) the MOZART package environment and programming language Oz were used. The implementation of the model in Oz had several stages. In the first stage appropriate data structures were proposed. In order to ensure scaling properties, which in an important factor due to the implementation flexibility and easiness in carrying out calculation experiments, list structures for both parameters and decision variables were used (fig. 4). The next step involved programming the model's constraints, which, due to the declarativity of Oz/MOZART environment, become the part of the program, which solved them. The manpower resource requirements for each operation can be interpreted as a next parameter of an operation (fig. 5). Our experiences at the applying of language Oz to the above type of problems [8] induced us to search for more effective solutions in implementation of search algorithms. Therefore we used propagators what does Oz deliver for stronger propagation employed for capacity constraints. A unary resource is one that cannot be shared by two activities; i.e., as soon as an activity requires a resource, for a time interval, no other activity can use it during the same time interval. Unary resources can be used to model a variety of applications. A typical example of a unary resource is a machine in a job-shop scheduling application.

For unary resources Oz provides two propagators employing edge-finding to implement capacity constraints. The propagator *Schedule.serialize* is an improved version of an algorithm described in [9]. A single propagation step has complexity $O(n^2)$ where n is the number of jobs the propagator is reasoning on, i. e. the number of jobs on the resource considered by the propagator. Because the propagator runs until propagation

For unary resources Oz provides two propagators employing edge-finding to implement capacity constraints. The propagator *Schedule.serialize* is an improved version of an algorithm described in [9]. A single propagation step has complexity $O(n^2)$ where n is the number of jobs the propagator is reasoning on, i. e. the number of jobs on the resource considered by the propagator. Because the propagator runs until propagation reaches a fixed-point, we have the overall complexity of $O(k*n^3)$ when k is the size of the largest domain of a job's start time (at most $k*n$ values can be deleted from the domains of job variables).

The propagator *Schedule.taskIntervals* provides stronger propagation than *Schedule.serialize*. While a single propagation step has complexity $O(n^3)$, the overall complexity is $O(k*n^4)$.

```
%NAME pe RESERVED
declare W
fun {W}
   NUMBER_Of_Workers=10
   Jobs=[a b c d e f g h x j ]        %List of jobs
   Operations=[a b c d e f g h x j ] %List of operations
%table of processing times
   Processing_Times=[ 62 24 25 84 47 38 82 93 24 66
                      47 97 92 22 93 29 56 80 78 67
                      45 46 22 26 38 69 40 33 75 96
                      85 76 68 88 36 75 56 35 77 85
                      60 20 25 63 81 52 30 98 54 86
                      87 73 51 95 65 86 22 58 80 65
                      81 53 57 71 81 43 26 54 58 69
                      20 86 21 79 62 34 27 81 30 46
                      68 66 98 86 66 56 82 95 47 78
                      30 50 34 58 77 34 84 40 46 44
                      0 ]
%table of manpower requirements for each operation
   Manpower_requirements=     [1 1 1 1 1 1 1 1 1 1
                              1 1 1 1 1 1 1 1 1 1
                              1 1 1 1 1 1 1 1 1 1
                              1 1 1 1 1 1 1 1 1 1
                              1 1 1 1 1 1 1 1 1 1
                              1 1 1 1 1 1 1 1 1 1
                              1 1 1 1 1 1 1 1 1 1
                              1 1 1 1 1 1 1 1 1 1
                              1 1 1 1 1 1 1 1 1 1
                              0 ]
%table of allocation operations of each Job to machines
   Machines = machines(
            0:[ ah bx cf dx ef fx gf hg xf ja]
            1:[ aj be ca dh eg ff gj hh xh je]
            2:[ ag bb cd dg ej fd gb hf xj jd]
            3:[ ad bh ch de ed fa gx hj xg jb]
            4:[ ae bf cg da ee fe gg ha xx jh]
            5:[ ac ba cj dc eh fc ga hc xc jf]
            6:[ af bj cc df ex fg gd hb xb jj]
            7:[ aa bg cb dj ec fj gc hx xe jc]
            8:[ ab bc cx db ea fh gh hd xd jg]
            9:[ ax bd ce dd eb fb ge he xa jx]
                  )
                                    .....
```

Fig.4 Data structures in Oz for Example_3 (MT10)

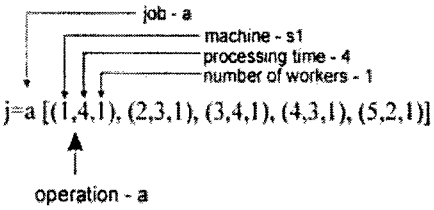

Fig.5. Description of job data structure

Table 2. Computational examples

Example_1	**Example_2**
j∈ {a,b,c,d,e} o∈ {a,b,c,d,e} s∈ {s1,s2,s3,s4,s5}	j∈ {a,b,c,d,e} o∈ {a,b,c,d,e} s∈ {s1,s2,s3,s4,s5}
j=a [(1,4,1), (2,3,1), (3,4,1), (4,3,1), (5,2,1)]	j=a [(1,4,2), (2,3,1), (3,4,1), (4,3,1), (5,2,1)]
j=b [(5,4,1), (4,5,1), (3,4,1),(2,3,1), (1,4,1)]	j=b [(5,4,1), (4,5,1), (3,4,1),(2,3,1), (1,4,2)]
j=c [(1,3,1), (5,4,1), (2,8,1), (3,3,1), (4,6,1)]	j=c [(1,3,2), (5,4,1), (2,8,1), (3,3,1), (4,6,1)]
j=d [(1,8,1), (2,4,1), (3,4,1), (4,3,1), (5,4,1)]	j=d [(1,8,2), (2,4,1), (3,4,1), (4,3,1), (5,4,1)]
j=e [(5,3,1), (4,8,1), (3,4,1), (2,3,1), (1,6,1)]	j=e [(5,3,1), (4,8,1), (3,4,1), (2,3,1), (1,6,2)]
Example_3 (MT10)	
j∈ {a,b,c,d,e, f,g,h,x,j} o∈ {a,b,c,d,e,f,g,h,x,j} s∈ {s1,s2,s3,s4,s5,s6,s7,s8,s9,s10} – jobs such how on fig. 1	

Table 3 Computational results

Example	Manpower	C_{max}	Processing times (ms)
Example_1	∞	33	< 100
Example_1	5	33	< 100
Example_1	3	37	< 100
Example_2	∞	33	< 100
Example_2	5	33	< 100
Example_2	3	44	< 100
Example_3 (MT10)	∞	943	2333
Example_3 (MT10)	7	945	106903

After the complete implementation of the model (2.3) into Oz/MOZART environment computation experiments were carried out. The parameters of computational examples are presented in table 2. For each of the examples optimal solution was searched for due to the makespan – C_{max} (optimal schedule). Computation experiments were carried out with the different number of manpower resource. The results of calculation (makespans, processing times) have been presented in table 3. Computation experiments were started on the computer PIV 1,4 GHz, RAM 512 under Windows XP. The received schedules for the Example_2 as Gantt charts have been shown in fig. 6 and fig.7.

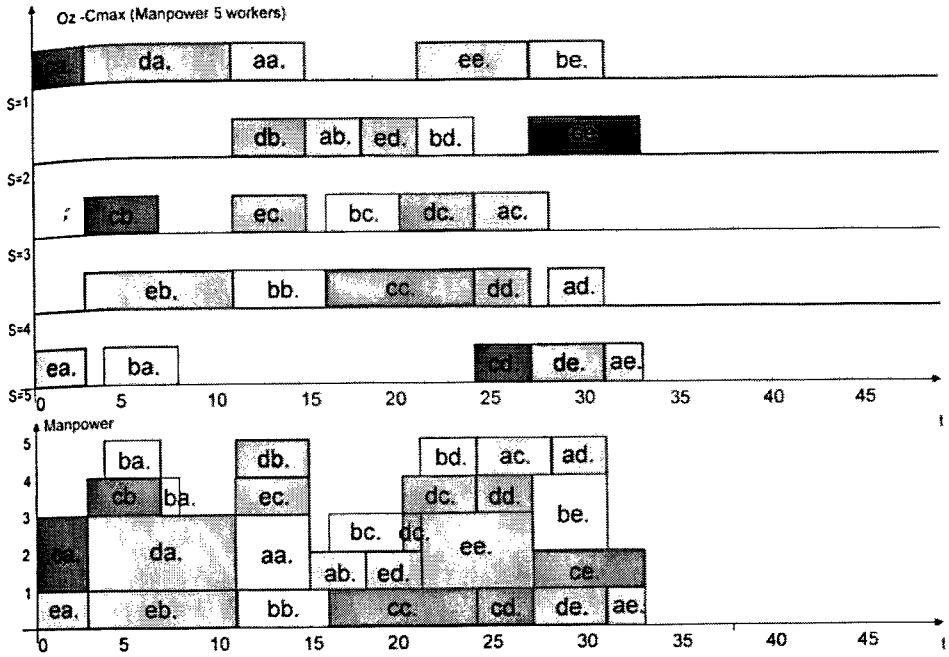

Fig.6. Gantt charts for Example_2 (manpower N=5), C_{max}=33.

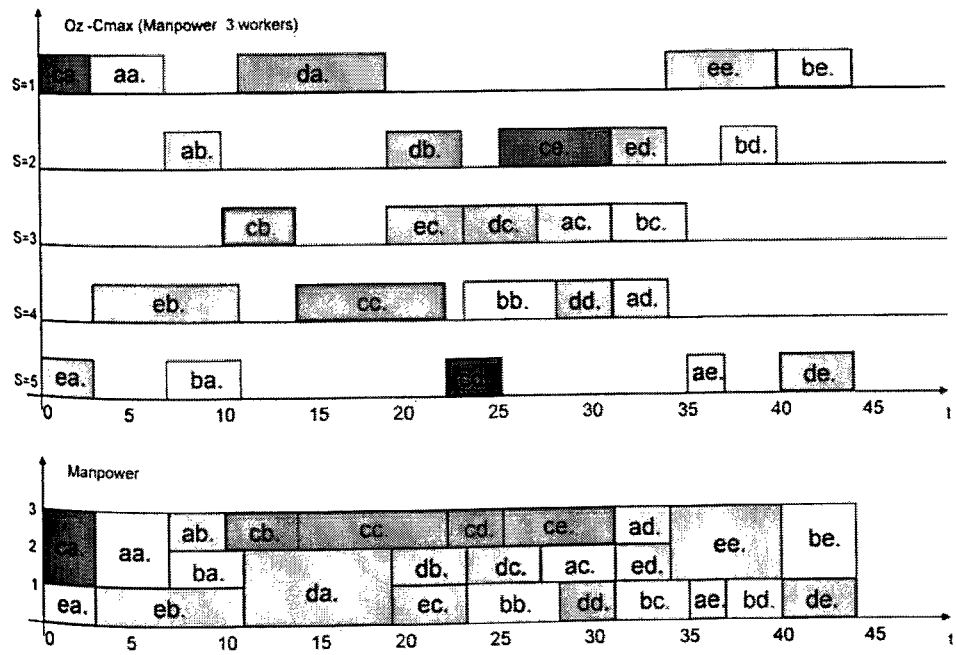

Fig. 7. Gantt charts for Example_2 (manpower N=3), C_{max}=44.

5. CONCLUSIONS

Allocation manpower to the jobs is more complicated than allocating machines, as different jobs and operation require different manpower. Therefore, CP approach is unusually interesting when referred to generally understood scheduling problems with manpower or other outer resources. The constraint propagation reduces significantly the domains of decision variables of the modeled problems. This feature, together with backtracking-based methods, makes CP methodology very effective. In addition, the CP implementation environment, e.g. Oz/MOZART possesses very strong propagation strategies (propagators: *Schedule.taskIntervals, Schedule.serialize*). The proposed approach can be considered as a contribution to job-shop scheduling problems with manpower resources applied in SME where this kind of resources can have influence for production and delivery schedules. That is especially important in the context of cheap, fast and user friendly decision support in SME.

REFERENCES

1. J.Józefowska *Discrete-continuous project scheduling,* Project Driven Manufacturing, WNT, Warszawa, pages 7-22, 2003.
2. J.Józefowska, R. Różycki, J.Węglarz *Uncertainty in discrete-continuous project scheduling* Project Driven Manufacturing, WNT, Warszawa, pages 23-33, 2003.
3. M.Mika, G.Waligóra, J.Węglarz *Metaheuristic approach to the multi-mode resource-constrained project scheduling problem with discounted cash flows and progress payments* WNT, Warszawa, pages 34-53, 2003.
4. S. French *Sequencing and Scheduling: An Introduction to the Mathematics of Jobshop*. Chichester:Ellis Horwood,1982.
5. C.Le Pape *Three Mechanisms for Managing Resource Constraints in a Library for Constraint-Based Scheduling* Proceedings INRIA/IEEE Conference on Emerging Technologies and Factory Automation, 1995.
6. www.mozart-oz.org/
7. J. Würtz *Constraint-Based Scheduling in Oz,*. Operations Research Proceedings 1996:218-223,Berlin, Heidelberg, New York,Springer-Verlag
8. P.Sitek, Z. Banaszak, W. Muszyński *Application of CLP to Prototyping Shop Orders in Small and Medium-Sized Enterprises* MMAR 11th IEEE International Conference on Methods and Models in Automation and Robotics. Międzyzdroje 2005, pages 1091–1096.
9. P. Martin, D. B. Shmoys. *A new approach to computing optimal schedules for the job shop scheduling problem.* In International Conference on Integer Programming and Combinatorial Optimization, Vancouver, pages 389-403, 1996.