14

A Declarative Framework for Constrained Search Problems in Manufacturing

Sitek Pawek and Wikarek Jaroslaw
Technical University of Kielce
Poland

1. Introduction

Today's highly competitive business environment makes it an absolute requirement on behalf of the managers to continuously make the best decisions in the shortest possible time. '*Learning from mistakes'* has left its place to '*one strike and you're out*' reality. That is, there is no room for mistake in making decisions in this global environment. Success depends on quickly allocating the organizational resources towards meeting the actual needs and requirements of the customer. Decision problems involve various numeric and non-numeric constraints, some of which are conflicting with each other. Occasionally, decision-makers do not have complete information on the situation. Thus they perform *'what-if'* and goal-seeking analyses involving constraints. In order to succeed in such an unforgiving environment, managers and decision makers need integrated '*intelligent*' decision support systems (DSS) that are capable of using a wide variety of models along with data and information resources available to them at various internal and external repositories. In this chapter we present the use of constraint logic programming as a tool for such decision support systems in constrained search problems, focusing on the model representation and analyses.
The original contribution of our approach consists of a declarative framework for constrained search problems, developed within the constraint logic programming (CLP) paradigm together with relational SQL database, and the development of a constraint logic solver for scheduling problems with external resources and resource dependent processing times in different production organization environments.

2. Constrained search problems

Constrained search problems (e.g., scheduling, planning, resource allocation, placement, routing) appear frequently at different levels of decision in manufacturing. They are usually characterized by technical, environmental or manpower constraints, which make them unstructured, and in most of the cases are difficult to solve (NP-complete). Traditional mathematical programming approaches (linear programming, integer and mixed integer programming) are deficient in the following ways: their representation of constraints is artificial (commonly using 0-1 variables), their computing time in the presence of many constraints is very long (due to combinatorial explosion), and they cannot process various constraints applied to the main problem. Thus, the most used approach consists in

developing specific software, written in a procedural language like PASCAL, BASIC or C, to solve each particular problem. However, the use of procedural languages brings the following well known disadvantages: the development time of the programs is very long and the programs are very complex, hence difficult to maintain and adapt to rapid changes of requirements.
Unlike traditional approaches, CLP provides for a natural representation of heterogeneous constraints and allows domain-specific heuristics to be used on top of generic solving techniques.

3. Declarative programming – SQL, CLP

Declarative programming is a term with two distinct meanings, both of which are in current use. According to one definition, a program is *'declarative'* if it describes *what* something is like, rather than *how* to create it. For example, HTML, XML web pages are declarative because they describe *what* the page should contain — title, text, images — but not *how* to actually display the page on a computer screen. This is a different approach from imperative programming languages such as PASCAL, C, and Java, which require the programmer to specify an algorithm to be run. In short, imperative programs explicitly specify an algorithm to achieve a goal, while declarative programs explicitly specify the goal and leave the implementation of the algorithm to the support software (for example, an SQL *select* statement specifies the properties of the data to be extracted from a database, not the process of extracting the data).
According to a different definition, a program is *'declarative'* if it is written in a purely functional programming language, logic programming language, or constraint programming language. The phrase "declarative language" is sometimes used to describe all such programming languages as a group, and to contrast them against imperative languages.
These two definitions overlap somewhat. In particular, constraint programming and, to a lesser degree, logic programming, focus on describing the properties of the desired solution (the *what*), leaving unspecified the actual algorithm that should be used to find that solution (the *how*). However, most logic and constraint languages are able to describe algorithms and implementation details, so they are not strictly declarative by the first definition.
Constraint Logic Programming (CLP) is a declarative modelling and procedural programming environment that integrates qualitative /heuristic knowledge representation of logic and quantitative/algorithmic reasoning into a single paradigm. Unlike traditional approaches, CLP provides for a natural representation of heterogeneous constraints and allows domain-specific heuristics to be used on top of generic solving techniques. The main issue for the constrained-based approach is CSP (Constraint Satisfaction Problem). In artificial intelligence and operation research, constraint satisfaction is the process of finding a solution to a set of constraints. Such constraints express allowed values for variables. A solution is therefore an evaluation of these variables that satisfies all constraints. Constraint Satisfaction Problems (on finite domains) are typically solved using a form of search. The most used techniques are variants of backtracking, constraint propagation and local search.
CLP as a declarative modelling and procedural programming environment is increasingly realized as an effective tool for decision support systems (Bisdorff & Laurent, 1995; Lamma et al., 1997; Lee & Lee 1996). Constraint Logic Programming is suitable for Decision Support Systems (DSS) because (Liao et al., 2002; Ryu, 1998):

- CLP is a very good tool for the development of knowledge base that has expertise and experience represented in terms of logic, rules and constraints. This tool allows the knowledge base to be built in an incremental and accumulating way (it is suitable for ill-structured or semi-structured decision analysis problems).
- Constraints naturally represent decisions and their inter-dependencies. Decision choices are explicitly modelled as the domains of constraint variables.
- CLP can serve as a good integrative environment for the decision analysis that has different kinds of model.

Decision analysis requires a number of computational facilities which this tool can provide.

4. Declarative framework for constrained search problems

There is a growing need for decision support tools capable of assisting a decision maker in the constrained search problems in manufacturing. The most important of them are scheduling problems and scheduling problems with resource allocation. 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 make constraint logic programming a method of choice for the resolution of complex practical problems. In constraint programming approach to decision support in scheduling problems, the problem to be solved is represented in terms of decision variables and constraints on these variables (Pape, 1995).

Depending on the particular applications, the variables of scheduling problems (job-shop, flow-shop, open-shop, and project shop) can be:

- The start time and the end time of each operation.
- The set of resources assigned to each operation (if this set is not fixed).
- The capacity of a resource that is assigned to an operation (e.g. the number of workers from a given team assigned to operation).
- The processing times (constant, variable increasing/decreasing function of starting times or allocated resources, etc.).

The constraints of a scheduling problem include:

- Temporal and precedence constraints which define the possible values for the start and end times of operations and the relations between the start and end time of two operations.
- Resource constraints which define the possible set of resources for each operation.
- Capacity constraints which limit the available capacity of each resource over time.
- Problem-specific constraint which correspond to particular features of operations and resources.

Additional variables and constraints can be included to represent optimization criteria, preferences of the user of scheduling system, etc.

4.1 Assumptions of DSS based on declarative framework

The presented in (section 3) advantages and possibilities of CLP environment for decision support make it interesting for decision support in constrained search problems. Building decision support system for scheduling, covering a variety of production organization forms, such as job-shop, flow-shop, project, multi-project etc., is especially interesting.

The following assumptions were adopted in order to design the presented scheduling processes of decision support system (see Fig. 1.):

- Problem-specific constraint which correspond to particular features of operations and resources.
- The system should possess data structures that make its use possible in different production organization environments (see Fig. 2.).
- The system should make it possible to schedule the whole set of tasks/jobs simultaneously, and after a suitable schedule has been found, it should be possible to add a new set of tasks later, and to find a suitable schedule for both sets without the necessity to change initial schedules.
- The decisions of the systems are the answers to appropriate questions formed as CLP predicates.
- The system should regard:
  - additional resource types apart from machines, e.g. people, tools, etc,
  - temporary inaccessibility of all resource types,
  - resource or time depending processing times, etc.

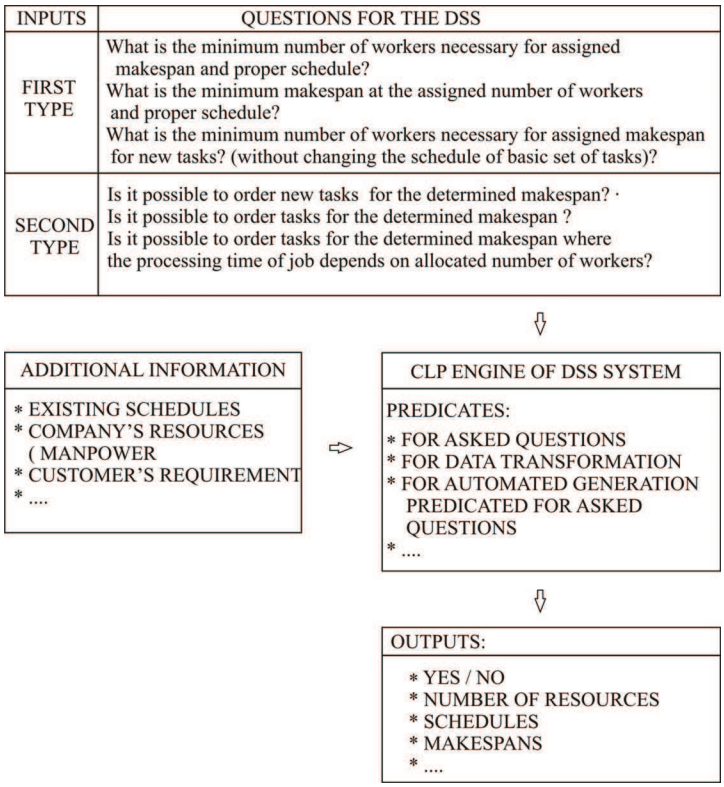

Fig. 1. Concept of DSS for scheduling problems based on declarative framework.

The range of the decisions made by the system depends on data structures and asked questions. Thus, the system is very flexible as it is possible to ask all kinds of questions (write all kinds of predicates). In this version of DSS the questions which can be asked are the following:

- What is the minimum number of workers necessary for assigned makespan and proper schedule? (predicate *opc_d(L,C)*).
- What is the minimum makespan at the assigned number of workers and proper schedule? (predicate *opc_g(L,C)*).
- Is it possible to order new tasks (both orders and projects) for the determined makespan? (predicate *opc_s(L,C)*).
- What is minimum makespan at the assigned number of workers for new tasks? (predicate *opcd_g(L,C)*).
- What is the minimum number of workers necessary for assigned makespan for new tasks? (without changing the schedule of basic set of tasks) (predicate *opcd_d(L,C)*).
- Is it possible to order tasks for the determined makespan ? (predicate *opcd_s(L,C)*).
- Is it possible to order tasks for the determined makespan where the processing time of task depends on allocated number of workers? (predicate *opcd_s1(L,C)*).
  L – number of workers (manpower), $C=C_{max}$ – makespan

These questions are just examples of questions that the present system can be asked. New questions are new predicates that need to be created in CLP environment. Two types of questions are asked in the system:

- About the existence of the solution (eg., is it possible to carry out a new task in the particular time?, etc.).
- About a particular kind of the solution: find a suitable schedule fulfilling the performance index, find the minimum scheduling length-makespan, find the minimum number of workers to carry out the task, etc.

The foregoing questions can include a random set of additional renewable resources (in this case, workers only) and refer a random number of production organization forms (job-shop, flow-shop, open-shop, project etc.). Additionally, the presented decision support system model implements an extra functionality which is resource dependent processing times. Scheduling problems literature gives the processing time as constant and defined before the tasks are realized. In practical applications the time is significantly dependent on the amount of the allocated resources for their realization. These dependencies are usually non-linear and can be presented as a relationship (relational database table) or function. The system implemented the possibility of changing the time of task/job realization in relation to the allocated number of workers. The functionality above does not call for the change of predicates; it requires suitably prepared data describing the problem and included in the relational database. The proposed structure of the relational database (see Fig. 2.) and the way CLP predicates are built allow the system to generate both schedules with determined parameters for different production organization forms, but also include allocation of additional resources (in general case resource sets) and effects they may have on the realized tasks.

4.2 Data structures

Data structures were designed in such a way that they could be easily used to decision problems in a variety of scheduling environments, which is job-shop, flow-shop, project or multi-project. The obtained flexibility resulted from the use of a relational data model.

Figure 2 presents the ERD (Entity Relationship Diagram) of the database that was designed to meet the requirements of cooperation with CLP environment and to have the following possibilities:

- Storing the data for scheduling problems and resource allocation for different types of production organisation.
- Storing information about additional resources (e.g. labour force, tools or AGV vehicles).
- Saving the content and parameters of CLP predicates calls.
- Generating ready scripts for a CLP engine on the basis of the existing data.
- Saving the results obtained with a CLP solver, necessary for further calculations, visualisations or creating reports.
- Saving data about other problems within the family of constrained search problems.

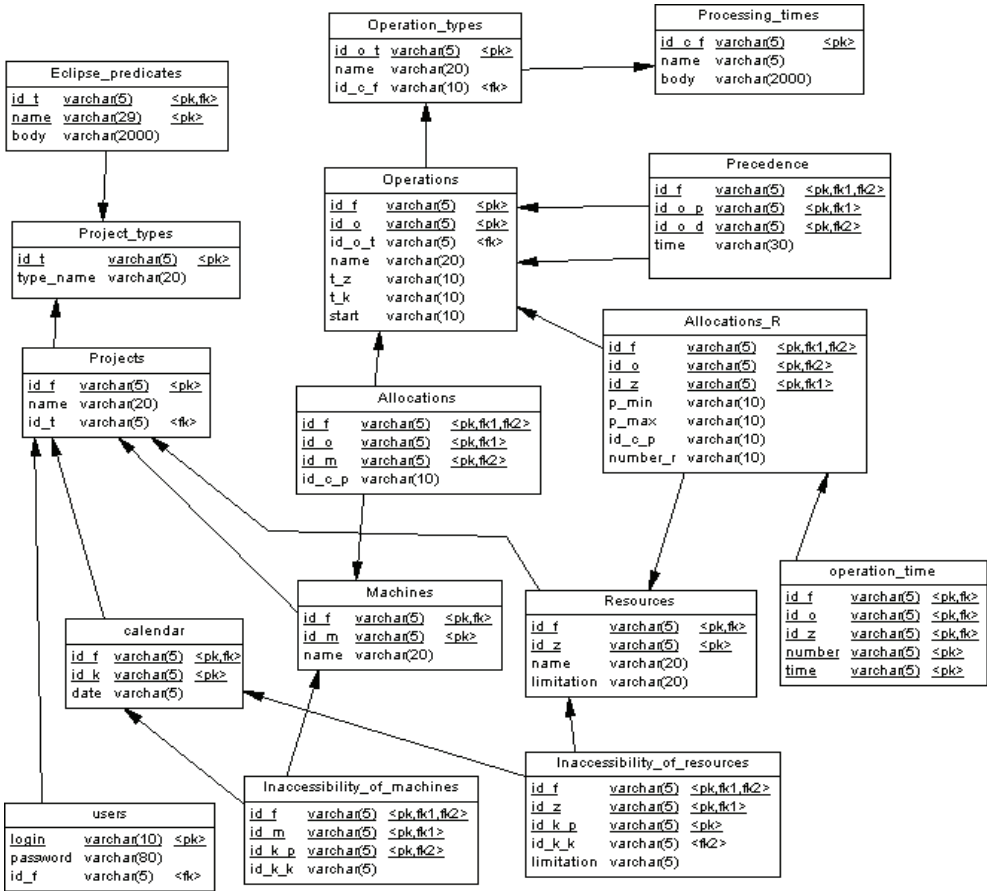

Fig. 2. Schema of database of DSS for production scheduling problems (Entity Relationship Diagram).

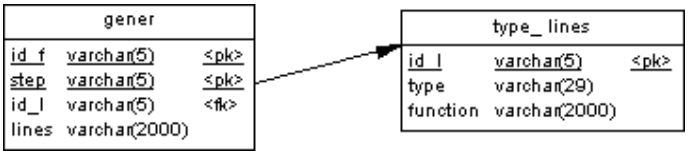

Fig. 2b. Schema of the part of database of DSS for an automatic generation CLP predicates (Entity Relationship Diagram).

Table 1 shows the description of database structure.

Table name	Table description	Column	Column description
Project_types	The types of possible projects for realization	id_t	project_type_id
		type_name	project_type_name
Projects	The specification of separate projects in enterprises	id_f	project_id
		name	project_name
		id_t	project_type_id
Processing_times	The list of functions of time calculation	id_c_f	function_id
		name	function_name
		body	function_body
Opertaion_types	The list of operation types	id_o_t	operation_type_id
		name	operation_type_name
		id_c_f	function_id
Operations	The list of operations to be realized	id_f	project_id
		id_o	operation_id
		id_o_t	operation_type_id
		name	operation_name
		t_z	release time
		t_k	critical time
		start	start time
Precedence	Defines the sequence of the realized operations	id_f	project_id
		id_o_p	operation_id
		id_o_d	operation_id
		time	time between operations
Machines	The specification of available machines for the operation realization	id_f	project_id
		id_m	machine_id
		name	machine_name
Allocations	The allocation of operation to machines	id_f	project_id
		id_o	operation_id
		id_m	machine_id
		id_c_p	parameters_of_function
Resources	The specification of renewable/external resources	id_f	project_id
		id_z	resource_id
		name	resource_name
		limitation	resource_limitation

Table name	Table description	Column	Column description
Allocations_R	The allocation of renewable/external/ additional resources to operations	id_f	project_id
		id_o	operation_id
		id_z	resource_id
		p_min	min number of allocated resource
		p_max	max number of allocated resource
		id_c_p	parameters_of_function
		number_r	number of allocated resource
Calendar	The specification of planning/scheduling periods	id_f	project_id
		id_k	period_number
		date	starting_date
Inaccessibility_ of_machines	The specification of inaccessibility of machines	id_f	project_id
		id_m	machine_id
		id_k_p	number of initial period
		id_k_k	number of final period
Inaccessibility_ of_resources	The specification of limitation/inaccessibility of resources	id_f	project_id
		id_z	resource_id
		id_k_p	number of initial period
		id_k_k	number of final period
		accessibility	number of accessible resources
Type_lines		id_l	line generation type
		type	type description
		function	function (in script language)
Gener	Describes the process of model generation for Eclipse	id_f	project_id
		step	number of generation step
		id_l	line generation type
		lines	line to be made
Eclipse_predicates	The codes for the ready predicates of Eclipse	id_f	project_id
		name	name of predicate
		body	code of predicate
Users		login	login
		password	password
		id_f	project_id

Table 1. Description of database structure.

5. Implementation of DSS based on declarative framework

We propose ECL^{i}PS^e (http://www.cs.kuleuven.ac.be, 2008, Apt & Wallace, 2007) and SQL database as a platform to decision support in scheduling problems. ECL^{i}PS^e is a software

system - based on the CLP paradigm - for the development and deployment of constraint programming applications. It is also ideal for developing aspects of combinatorial problem solving, e.g. problem modelling, constraint programming, mathematical programming, and search techniques. Its wide scope makes it a good tool for research into hybrid problem solving methods. ECL^{i}PS^e comprises several constraint solver libraries, a high-level modelling and control language, interfaces to third-party solvers, an integrated development environment and interfaces for embedding into host environment. The ECL^{i}PS^e programming language is largely backward-compatible with Prolog and supports different dialects.

The novelty of the proposed approach is in the integration of the CLP methodology with a commonly used relational database model. The scripts started by a CLP engine are generated automatically on the basis of data in the database (numerical values and CLP predicates). The proposed solution makes it possible to easily develop the system (developing and saving in the database the content of additional CLP predicates) and to integrate it with other computer systems based on a relational SQL database (Fig. 3.). Owing to the developed database structure (see Fig. 2.) solving other problems of the constrained search problems class is possible. In order to ensure an automatic generation of the production scheduling problem model in the form of a script with CLP predicates, two additional tables were added to the database (Fig. 2b). The *gener* table describes the model schema as lines containing the model's identity (*id_f*), generating step (*step*) and the identity of the line type that is to be written in the CLP script (*id_l*) with its source (*lines*). The type of the generated line is determined from the entry in the table *type lines*. The model (CLP script) distinguishes lines created among others as inserting CLP predicate (line in the *eclipse_predicates* table), inserting data after SQL statement, inserting a comment, heading, etc; thus the relation between tables *gener* and *type_lines* is 1:N type (Fig. 2b.).

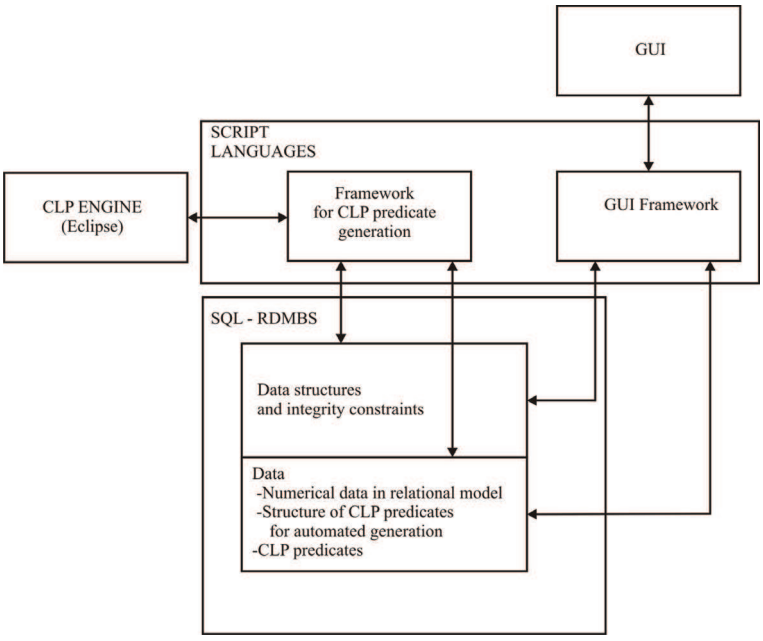

Fig. 3. Implementation of the declarative framework of DSS.

6. Illustrative examples

After the complete implementation of the DSS into ECLiPSe and SQL environments, computation experiments were carried out. The job-shop scheduling problem with manpower resources (Example 1) and project –building house (Example 2) were considered. The proposed illustrative examples cover a wide range of scheduling problems encountered in the SMEs (Small and Medium Sized Enterprises). The examples are selected in such a way that they how two extremely different forms of production organization; repetitive production in the job-shop environment and the unique production including the project. The presented methodology makes solving scheduling problems possible also in indirect methods of production organization. Moreover, the examples are larded with problems of constrained resources (e.g. manpower, specialized machines, etc.) and the dependence of particular jobs processing time on the amount of the allocated resources, for instance

6.1 Example 1- the job shop scheduling

In the classical scheduling theory job processing times are constant (Example_1a). However, there are many situations where processing time of a job depends on the starting time of the job in queue or the amount of allocated additional resources (e.g. people) (Example_1b) etc. The parameters of computational examples are presented in table 2. The job data structures are shown in Fig. 4a and Fig. 4b

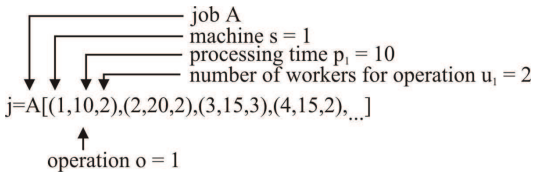

Fig. 4a. Description of task (job) data structure for job-shop computational example (Example_1a) – the constant processing times.

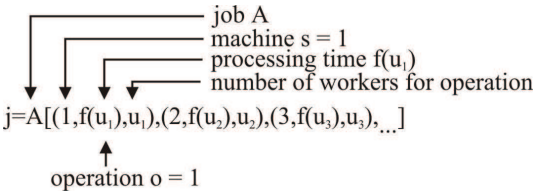

Fig. 4b. Description of task (job) data structure for job-shop computational example (Example_1b) – the processing times depend on allocated number of workers.

j∈{A,B,C,D,E}, o∈{1,2,3,4,5}, s∈{1,2,3,4,5}
j=A [(1,10,2), (2,20,2), (3,15,3), (4,15,2), (5,15,1)]
j=B[(1,10,1), (2,20,1), (3,15,2), (4,15,1), (5,20,1)]
j=C[(5,15,2), (4,20,2), (3,15,1), (2,10,2), (1,20,2)]
j=D[(1,10,3), (3,15,2), (2,20,2), (4,20,1), (5,10,2)]
j=E[(5,15,2), (4,10,1), (3,15,2), (2,10,2), (1,20,1)]

Table 2. (Example_1**) –** constant processing times

For the computational example (Example_1a) the following questions (write following predicates) were asked:

- *opc_g(_,_)* (see Fig. 5.).
- *opc_d(_, 120)* (see Fig. 6.).
- *opc_s(4,155)* (see Fig. 7.).
- *opc_s(4,180)* (see Fig. 8.).
- *opc_g(5,_)* (see Fig. 9.).

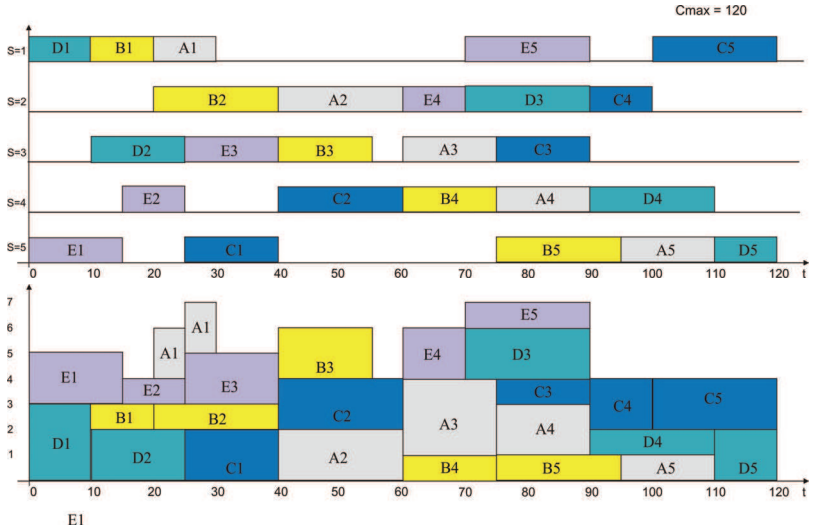

Fig. 5. Gantt's charts for the answer to the question implemented in predicate *opc_g(_,_)*, C_{max}*=120, L=7* (Example_1a).

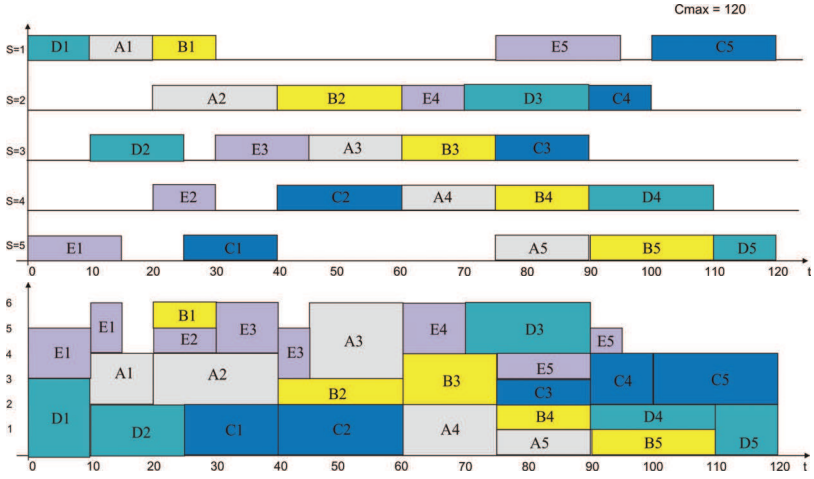

Fig. 6. Gantt's charts for the answer to the question implemented in predicate *opc_d(_,120)*, L_{min}*=6*, C_{max}*=* C_{max} * =120* (Example_1a).

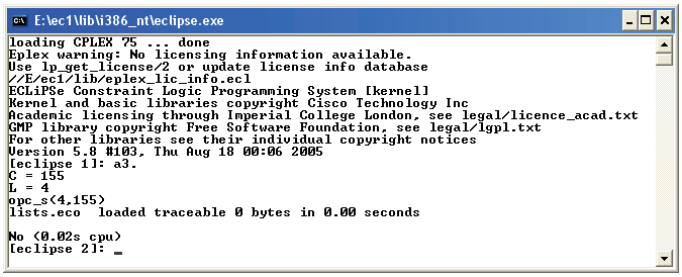

Fig. 7. Answer to the question implemented in predicate *opc_s(4,155)* – ***No*** (Example_1a).

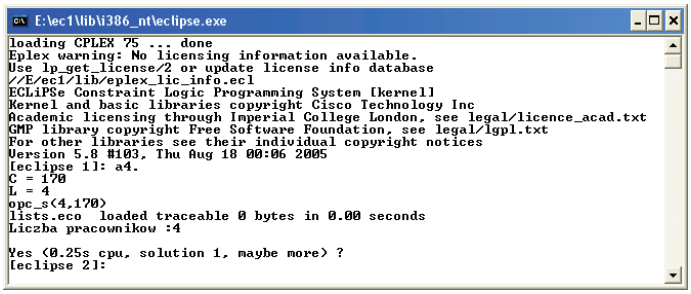

Fig. 8. Answer to the question implemented in predicate *opc_s(4,170)* – ***Yes*** (Example_1a).

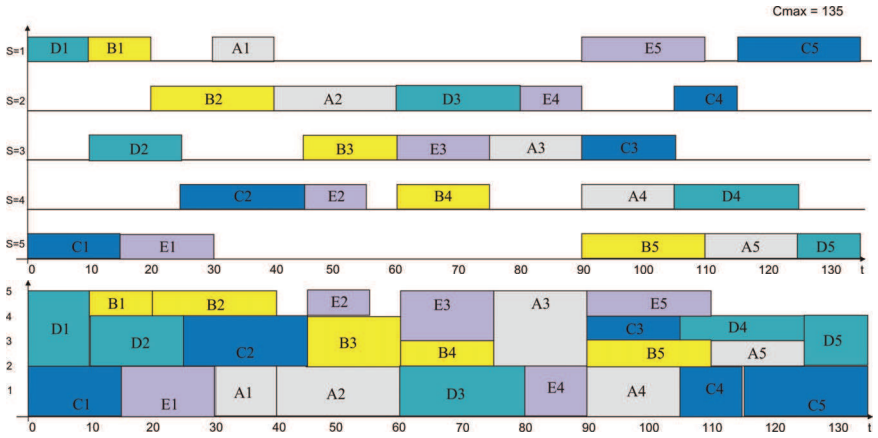

Fig. 9. Gantt's charts for the answer to the question implemented in predicate *opc_g(5,_), C_{max}*=135, L=5* (Example_1a).

	Predicate	L	C_{max}	Yes	No	Time (s)
1	*Opc_g(_,_)*	7	120	---	----	0,13
2	*Opc_d(_,120)*	6	120	---	---	0,35
3	*Opc_s(4,155)*	4	155	NO		0,02
4	*Opc_s(4,170)*	4	170	YES		0,25
5	*Opc_g(5,_)*	5	135	---	---	1,35

Table 3. (Example_1a**) –** Results of asked predicates (Fig.5-9).

The second version of computational example (Example_1b) was carried out with processing times of operation/activity dependent on allocated additional resource (workers). The parameters of computational Example_1b are presented in table 2 without processing times and number of allocated people. The processing time is a function of allocated workers $f(p_j,a_j,u_j)$ Fig. 10.

$f(p_j,a_j,u_j) = p_j - a_j*(u_j - x_j)$
and
$f(p_j,a_j,u_j) > 0$,
$a_j = 5$, $x_j \leq u_j \leq 2*x_j$
where :
p_j - processing time from Example_1
u_j - number of allocated workers
x_j - number of allocated workers from Examle_1a
a_j - acceleration factor

Fig. 10. Processing time for Example_1b.

There is a simple linear function in this example. It can be any function in general case or relationship (relational database table). For the computational example (Example_1b) the following questions (write following predicates) were asked:

- *opc_g(_,_)* (see Fig. 11.).
- *opc_s(8,80)* (see Fig. 12.).
- *opc_d(_,60)* (see Fig. 13.).
- *opc_s(6,60)* (see Fig. 14.).

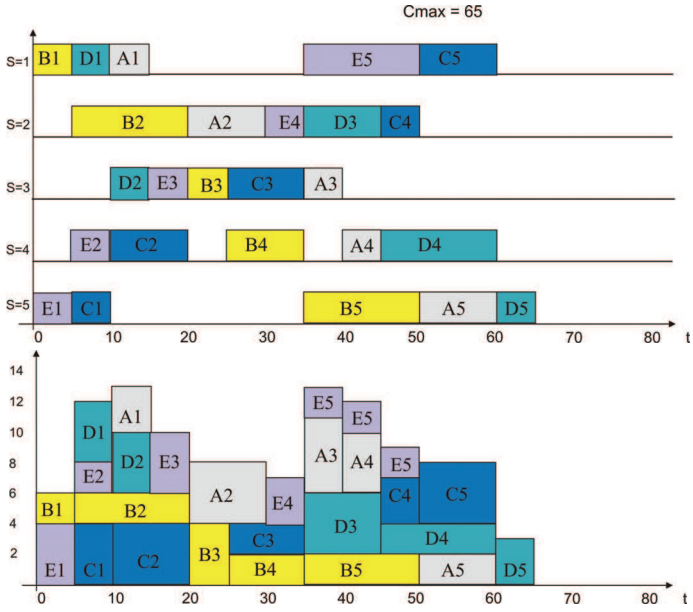

Fig. 11. Gantt's charts for answer to the question implemented in predicate *opc_g(_,_)*, C_{max}*=65, L=14 (Example_1b).

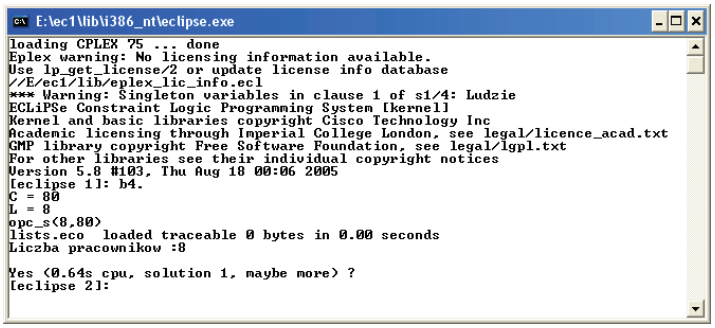

Fig. 12. Answer to the question implemented in predicate *opc_s(8,80)* – ***Yes*** (Example_1b).

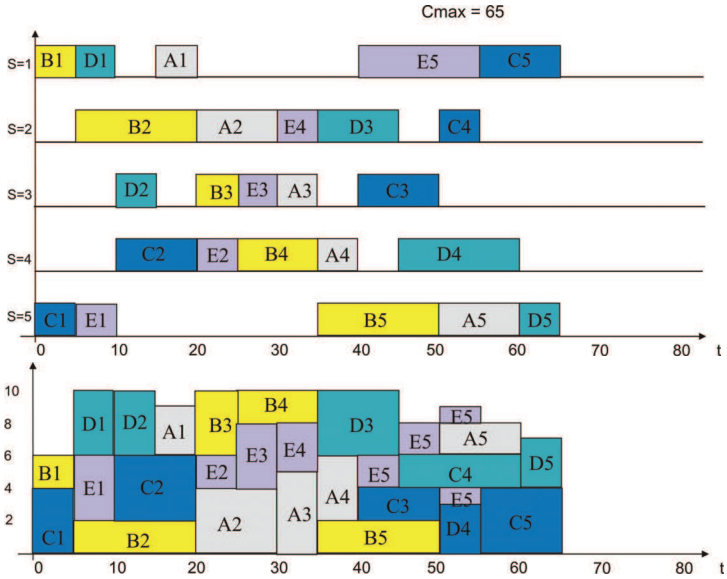

Fig. 13. Gantt's charts for answer to the question implemented in predicate *opc_d(_,65)*, $L_{min}=10$, $C_{max}= C_{max} * =65$ (Example_1b).

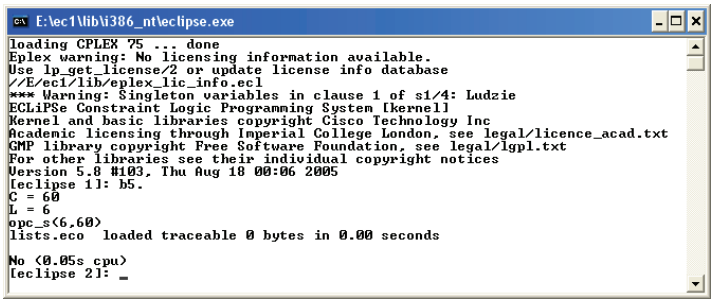

Fig. 14. Answer to the question implemented in predicate *opc_s(6,60)* – ***No*** (Example_1b).

	Predicate	L	C_{max}	Yes	No	Time (s)
1	*Opc_g(_,_)*	13	65	---	---	0,14
2	*Opc_d(_,65)*	10	65	---	---	17,84
3	*Opc_s(8,80)*	8	80	YES		0,64
4	*Opc_s(6,60)*	6	60	NO		0,05

Table 4. (Example_1b**) –** Results of asked predicates (Fig.11-14).

6.1 Example 2- Project scheduling – building a house

A typical modern-day project has a variety of complications not considered in the original PERT/CPM methodology. There are three particular situations:

- You may be able to accelerate the completion of a project by speeding up or "crashing" some of the activities in the project.
- Your ability to finish a project quickly is hindered by limited resources (e.g., two activities that might otherwise be done simultaneously, in fact have to be done sequentially because they both require a crane and you have only one crane on site).
- How long it takes to do each activity is a random variable.

In table 5, we list the activities involved in a simple, but nontrivial, project of building a house. An activity cannot be started until all of its predecessors are finished. The network activity for this project has been shown in Fig. 15. To solve this example the DSS with declarative programming (section 4) was used. In this example the processing times of activities depend on allocated manpower resource.

On.	Activity Time	Min_MAN	Max_MAX	Name of activity
1	10	2	2	Dig Basement
2	12	4	6	Pour Foundation
3	6	1	3	Pour Basement
4	6	2	3	Install Floor Joists
5	6	1	3	Install Walls (ext)
6	4	2	8	Install Rafters
7	4	2	4	Install Walls (int)
8	4	2	2	Install Roof
9	16	4	8	Install Windows, Doors (ext)
10	12	4	8	Install Networks
11	12	6	8	Interior Plastering
12	4	2	4	Painting (int)
13	6	2	3	Finish Interior
14	18	6	9	Finish Terrace
15	4	2	4	Garden Arrangement
16	18	6	12	Exterior Plastering

MIN_MAN – minimum manpower (workers) for activity.
MAX_MAN – maximum manpower (workers) for activity.

Table 5. Parameters of Example_2

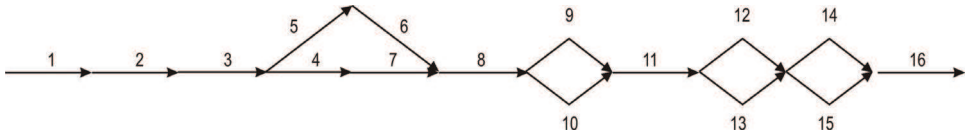

Fig. 15. Activity network for Example_2.

For the computational example the following questions (write following predicates) were asked (see section 4):

- *opc_g(150,200* (see Fig. 16., Table 6,7).
- *opc_g(5,400)* (see Fig. 17., Table 6,7).
- *opc_g(7,200)* (see Fig. 18., Table 6,7).
- *opc_g(12,200)* - processing times of jobs dependent on the allocated additional resource (workers) (see Fig. 19., Table 6,7).

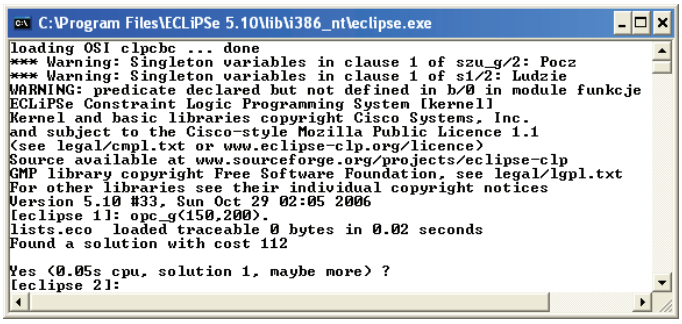

Fig. 16. Answer to the question implemented in predicate *opc_g(150,200)* – ***Yes*** (Example_2).

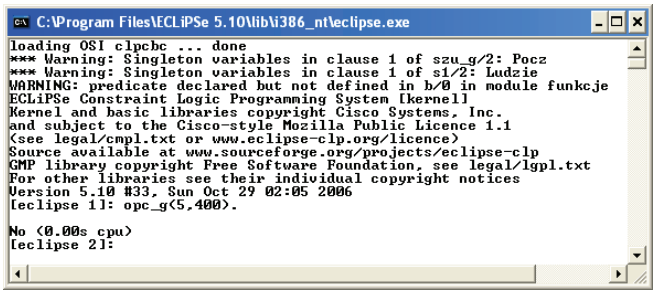

Fig. 17. Answer to the question implemented in predicate *opc_g(5,400)* – ***No*** (Example_2).

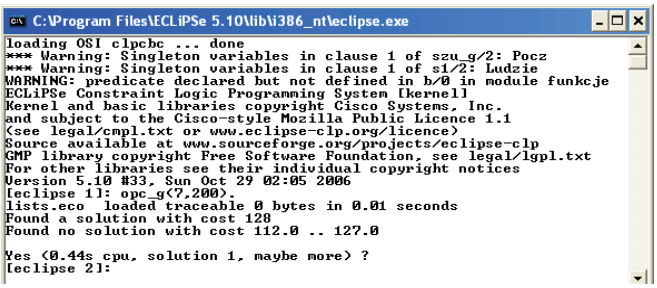

Fig. 18. Answer to the question implemented in predicate *opc_g(7,200)* – ***Yes*** (Example_2).

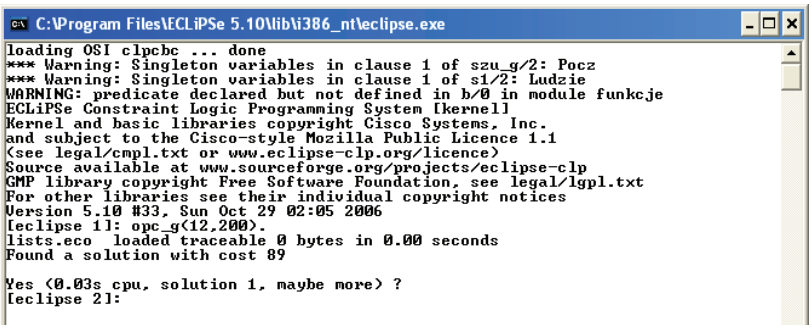

Fig.19. Answer to the question implemented in predicate *opc_g(12,200)*– ***Yes*** (Example_2).

On.	Predicate	L	C	Yes	No	Time (s)
1	*Opc_g(150,200)*	8	112	YES		0,05
2	*Opc_g(5,400)*	5	400	NO		0,00
3	*Opc_g(7,200)*	6	128	YES		0,44
4	*Opc_g(12,200)*	12	89	YES		0,03

Table 6 (Example_2) Results of asked predicates (Fig.16-19).

On.	Opc_g(150,200)	Opc_g(5,400)	Opc_g(7,200)	Opc_g(7,200)
1	0	----	0	0 (2)
2	10	----	10	10 (6)
3	22	----	22	20 (3)
4	28	----	2	24 (3)
5	28	----	28	24 (3)
6	34	----	34	28 (3)
7	34	----	34	29 (4)
8	38	----	38	31 (2)
9	42	----	42	35 (8)
10	42	----	58	35 (4)
11	58	----	70	47 (8)
12	70	----	82	57 (2)
13	70	----	82	57 (3)
14	76	----	88	62 (9)
15	76	----	106	62 (2)
16	94	----	110	77 (12)

Table 7 (Example_2) Result of asked predicates-start times of activities (additional number of allocated workers- only for predicate *Opc_g(7,200)*).

The results obtained for illustrative examples confirm suitability of the proposed framework for building decision support systems in constrained search problems. In scheduling problems the decision maker is provided with support related to possibilities of task accomplishment in the set time, necessary resources and their exploitation in time, possibilities of the realization of other tasks, decision optimization, etc.

7. Conclusions

The proposed approach can be considered to be a contribution to scheduling and especially to scheduling problems with additional/external resources. In many enterprises this kind of resources can have an influence on production and delivery schedules. That is especially important in the context of cheap, fast and user friendly decision support in SMEs. Great flexibility of the presented approach and practically unlimited possibilities of asking questions through creating predicates cannot be overestimated. What is more, the whole decision system can be built in one modeling and programming declarative environment, which lowers costs and adds to the solution effectiveness. The CLP-tools fulfill the need of intelligent production management structures and can be based successfully in cases of scheduling problems with external resources. The proposed approach seems to be a viable alternative option for supporting quite a number of decision making processes. The originality of our approach, which achieves the transition from custom imperative programming to declarative programming in a field of scheduling problems, consists in the data structure and CLP implementation. The presented framework can be implemented in many other constrained search problems apart from scheduling such as planning, routing, placement etc.

8. References

Liao S.Y., Wang H.Q., Liao L.J. "An extended formalism to constraint logic programming for decision analysis, Knowledge-based Systems" 15, 2002, pp 189-202.

Pape C.Le "Three Mechanisms for Managing Resource Constraints in a Library for Constraint-Based Scheduling Proceedings" INRIA/IEEE Conference on Emerging Technologies and Factory Automation, 1995.

Ryu U. Young ."Constraint logic programming framework for integrated decision supports" Decision Support Systems 22, 1998, pp 155-170.

Bisdorff R., Laurent S. "Industrial linear optimization problem solved by constraint logic programming", European Journal of Operational Research 84 (1), 1995, pp 82-95.

Lamma E., Mello P., Milano M. "A distributed constrained-based scheduler", Artificial Intelligence in Engineering 11,1997, pp 91-105.

Lee H.G., Lee G. Yu., "Constraint logic programming for qualitative and quantitative constraint satisfaction problems", Decision Support Systems 16 (1), 1996, pp 67-83. http://www.cs.kuleuven.ac.be/

Apt K.R., Wallace M.G. "Constraint Logic Programming using ECL^{i}PS^e ", Cambridge, 2007

Automation and Robotics

Edited by Juan Manuel Ramos Arreguin

ISBN 978-3-902613-41-7
Hard cover, 388 pages
Publisher I-Tech Education and Publishing
Published online 01, May, 2008
Published in print edition May, 2008

In this book, a set of relevant, updated and selected papers in the field of automation and robotics are presented. These papers describe projects where topics of artificial intelligence, modeling and simulation process, target tracking algorithms, kinematic constraints of the closed loops, non-linear control, are used in advanced and recent research.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Sitek Pawek and Wikarek Jaroslaw (2008). A Declarative Framework for Constrained Search Problems in Manufacturing, Automation and Robotics, Juan Manuel Ramos Arreguin (Ed.), ISBN: 978-3-902613-41-7, InTech, Available from: http://www.intechopen.com/books/automation_and_robotics/a_declarative_framework_for_constrained_search_problems_in_manufacturing

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2008 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.