Transactions on Information and Communications Technologies vol 1, © 1993 WIT Press, www.witpress.com, ISSN 1743-3517

# Optimization with constraint logic programming: the hoist scheduling problem solved with various solvers

P. Baptiste, B. Legeard, M.-A. Manier & C. Varnier
*Laboratoire d'Automatique de Besançon, Impasse des Saint Martin, 25000 BESANCON, France*

## ABSTRACT

The Hoist Scheduling Problem in electroplating facilities is known as an NP-Hard and strongly constrained problem. Up to now, the Operational Research approach appeared to be limited to the resolution of specific kinds of lines. We have proposed a new approach using the Constraint Logic Programming. In this paper we give a model for simple lines, aiming at providing the optimal solution for the 1-periodic problem, and that can easily be extended. Then we compare the performances of two kinds of solvers: PROLOG III that uses rational variables, and CHIP that uses finite domains. We finally give results for a benchmark example.

## INTRODUCTION

The Hoist Scheduling Problem deals with scheduling the movements of hoists in most electroplating facilities. It is a strongly constrained problem, known as an NP-Hard one. This problem is detailed in the first part of this article. Several approaches use classical Operational Research tools, but they are generally restricted to simple line problems. Those elementary cases seem to be far from real industrial concerns.

We have proposed a new approach based on Constraint Logic Programming (CLP) in [1], [7]. The main characteristic of CLP languages is to allow declarative modelling both in terms of constraints over interpreted domains and in a logical framework. It is rather convenient for representing problems like the HSP. This

characteristic is interesting, and it insures an easy extension of the models and of their implementation.

We first obtained satisfying results by implementing our model with PROLOG III. We reached optimal solutions as fast as using previous methods, and we obtained very interesting extensions. The corresponding models are proposed in the second part of this article.

We wondered if such a "rational solver" is the most appropriate one for solving such a problem. The choice of the appropriate CLP solver for an industrial application is still an open issue in the field of using CLP Languages. Discrete problems with integer variables representation seem to need finite domains solvers, and problems with real variables require solvers using rational arithmetic (like in PROLOG III for example). But, this is too simple and we show that the nature of the techniques (partial or complete consistency) is more important than the variables' domains.

We also modelized our problem using CHIP, and we compared the two implementations. First, we can say that CHIP produces the same results than PROLOG III, but with much more erratic performances.

The lack of a clear relationship beetween the size of the problem and the performances of the program seem to be due to the partial consistency of the finite domains' solver. This problem is discussed in the last section.

## OPTIMIZATION WITH CONSTRAINT LOGIC PROGRAMMING (CLP)

### The Hoist Scheduling Problem (HSP)

An automated electroplating line (Fig. 1) is composed of :

- A "wetline" of several tanks (from 5 up to 50) generally organized in line, and used for applying a sequence of chemical and/or plating treatments.
- A set of hoists (or cranes) that can remove products (or carriers) from a tank, transport products from tank to tank, lower products into a tank, travel empty from tank to tank, and of course simply wait (as less as possible).

The control of the movements of such hoists is known as the Hoist Scheduling Problem (HSP), and it is the real problem that every industry meets as soon as it includes surfacing treatment.

The generic model found in literature is characterized by a line of tanks supplied with a single hoist, and a single product production. The treatment of each product consists of a given sequence of soak operations into processing tanks. Furthermore each processing time is bounded, and those limits must be strictly respected to ensure the quality of the products.

Some variations on this basic system could be considered:
- multi-functions tanks : the same kind of treatment appears twice or more in the sequence.
- duplicated tanks : tanks identified as bottlenecks of a line are duplicated, in order to increase the throughput of the line.
- multi-hoists : when the number of tanks reaches 10, the hoist becomes the critical resource. Therefore, it is necessary to introduce several hoists on the same line.

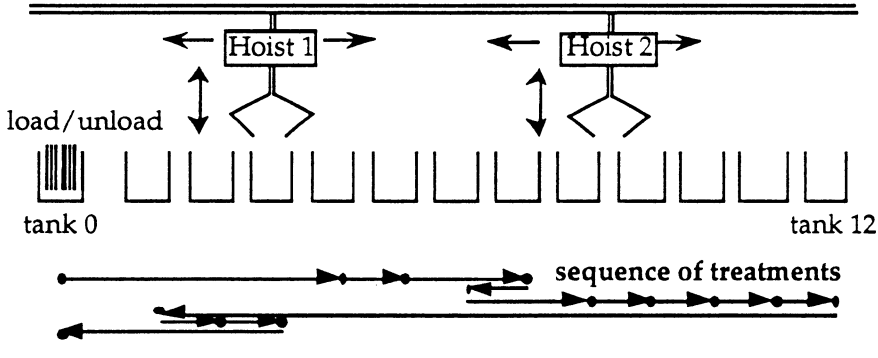


Figure 1. Example of line configuration.

The objective of the HSP is to find an optimal cyclic schedule for hoists that allows the programmation of hoists for always repeating the same sequence of movements. This sequence is called a **cycle**. The duration of a cycle is called a **period**.

Some results are already available for this problem. Most resolutions use classical well-known Operational Research tools. The first searchers, who became interested in this problem, are Phillips and Unger from Bell laboratories, in collaboration with the Western Electric Plant [12]. They have developed a mixed integer programming model to determine a 1-cycle time for the basic model with a single hoist and a single product production.

Shapiro and Nuttle have proposed another approach to solve the same problem, based on a branch and bound procedure, using linear programming to bound the search space [14], [15]. Shapiro and Nuttle affirmed that their model was able to find an n-cycle schedule, but "*the associated computation would probably be intolerable*".

Recently Lei and Wang proposed the first algorithm which was able to find a cyclic schedule for a two hoists system [11]. They developed an heuristic algorithm, that evaluates alternative problem partitions. For each partition they solved two single hoist sub-problems, by applying a procedure that they had first proposed for the single hoist scheduling problem. This procedure uses an interval cutting approach [9].

As you can see, most of the previous works use Operational Research tools. They have opened the way, giving interesting models, and finding the optimal solution for an academic system. Their main disadvantage is that they are strictly limited to the basic problem (which is not realistic). The problem is indeed NP-Hard [10] and each extension involves an increase of the complexity of both the model and its resolution. This is the reason why we have decided to work on a new approach, using a new model, and a new tool: the Constraint Logic Programming.

Basic model

Here we are presenting the basic model, which is capable of finding the optimal 1-periodic solution for a line composed with a single hoist and a single product production. These assumptions involve that in each tank the processing time will be the same for each product.

For the simplicity of exposition we first indexed the tanks by their order in the given process sequence. This can be done only if we assume that one tank is associated with only one process.

Notation We propose the following notation:

$c$ : number of tanks.
$d_{i,j}$ : time to move the hoist from tank i to tank j.
$r_i$ : time for the hoist to move a carrier from tank i to tank i+1. It includes the grasp and the raise of the carrier, the dripp above the tank, the transport from tank i to tank i+1, and the lowering into tank i+1.
the relation $r_i > d_{i,i+1}$ will always stand true.
$O_i$ : processing in tank i

$m(O_i)$ : minimum processing time in tank i.
$M(O_i)$ : maximum processing time in tank i.
$o_i$ : processing time in tank i.
$R_i$ : transport operation of a product from tank i to tank i+1
$t(R_i)$ : time when the hoist removes a carrier from tank i.
T : length of a cycle: time between the input of two successive products

Remarks:
- c, $d_{i,j}$, $r_i$, $m(O_i)$, $M(O_i)$ are given and fixed.
- the load and unload station are considered as fictive tanks, and respectively indexed 0 and c+1.

Variables The criterion that we aim at optimizing is the throughput. During a cycle exactly one carrier enters the system and one carrier leaves it. Hence maximizing the throughput means minimizing the period T.

The problem is completely defined by the (c+1) variables $t(R_i)$ and the period T.

We assume that a carrier enters the line when it starts being loaded, and leaves the line as soon as it finished to be unloaded. Moreover, no pre-emption is allowed for any operation, which involves the following relations for carrier *1*:

$$\begin{cases} t(R_0) = o_0 \\ t(R_i) = t(R_{i-1}) + r_{i-1} + o_i = t(R_0) + \sum_{k=1}^{i} (r_{k-1} + o_k) \end{cases} \quad (1)$$

We search for a 1-periodic solution, so that any carrier *k* is removed from a tank *i* at time $t(R_i) + (k-1)*T$.

Constraints The constraints involved by the basic problem are the following ones :

(C1) the processing time in each tank must respect a minimum and maximum limit,
(C2) only one product can be processed in each tank at the same time,
(C3) a hoist can perform only one transport operation at the same time,

(C4) a hoist must have enough time to move between two transport operations.

Each constraint Ci can be mathematically translated by linear inequalities.

The constraint (C1) is simply expressed by:

$$m(O_i) \leq o_i \leq M(O_i) \qquad (\text{for } i = 0 \text{ to } c+1) \tag{2}$$

The period T must be at least greater than each processing time, in order to respect the constraint C2:

$$T > o_i \qquad (\text{for } i = 0 \text{ to } c+1) \tag{3}$$

We can deduce from Equation (3) a lower bound for T. A higher bound for T can be the minimum treatment time of one product.

The constraint C3 allows to order each couple of hoist operations. Let us consider two carriers indexed $k_i$ and $k_j$ respectively being processed in tank $i$ and tank $j$. The hoist will remove carrier $k_i$ before carrier $k_j$ if:

$$\text{for } k_i, k_j \geq 1, \text{ with } k_i \neq k_j, \text{ for } i,j = 0 \text{ to } c,$$

$$[t(R_j) + (k_j-1)*T] - [t(R_i) + (k_i-1)*T + r_i] \geq d_{i+1,j} \tag{4}$$

this condition also ensures that the hoist has a sufficient time to move from tank $i+1$ to tank $j$, after lowering carrier $k_i$ and before removing carrier $k_j$ (constraint C4);
otherwise the hoist removes carrier $k_j$ before carrier $k_i$:

$$[t(R_i) + (k_i-1)*T] - [t(R_j) + (k_j-1)*T + r_j] \geq d_{j+1,i} \tag{5}$$

Since the solution that we are looking for has a 1-periodic structure, the order between the removal of carrier *1* from any tank $i$ and the one of any carrier $k$ from any tank $j$ will be the same as the

ordering between the removal of carrier *a* from tank *i* and the one of carrier *k+a-1* from tank *j*. Furthermore, for i ≤ j, we know that carrier *1* must be removed from tank *i* before the removal of carrier *k* (k>1) from tank *j*. The disjunctive Equations (4) and (5) become :

for k > 1, i = 1 to c, j = 0 to c-1,and i>j :

$$[t(R_j) + (k-1)*T] - [t(R_i) + r_i] \geq d_{i+1,j} \quad (6)$$

$$t(R_i) - [t(R_j) + (k-1)*T + r_j] \geq d_{j+1,i} \quad (7)$$

Strategy The study of all the disjunctive constraints follows a search tree (Fig. 2) . At each node a couple of transport operations for two products is chosen. This couple is ordered, arbitrarily or not. Then the associated constraint Equation (6) or Equation (7) is added to the current system. If the resulting set of constraints remains consistent, a new node is reached and another disjunction is studied. Otherwise, the constraint applied do not lead to a solution. Then it is not worth exploring the sub-tree generated by this branch since all the solutions resulting from this choice will include that wrong partial order. Therefore a backtrack permits to reconsider a preceding disjunction and make another choice.

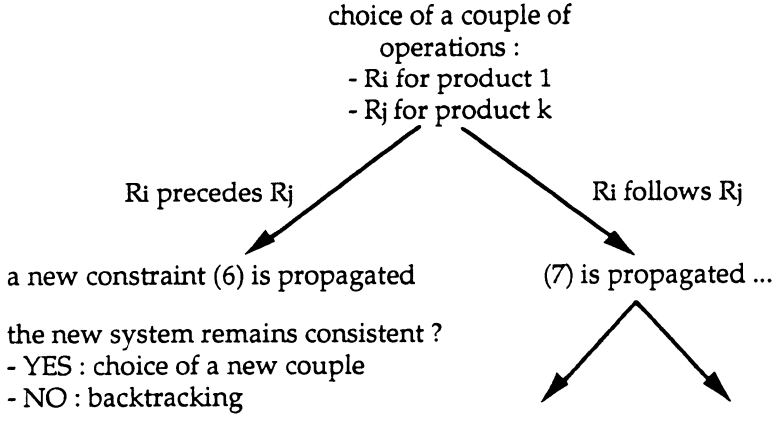


Figure 2. Search tree for feasible schedules.

A depth first procedure is used to explore the search tree, in order to reach a leaf as fast as possible. A leaf of the tree is reached when all the hoist movements $R_i$ are totally ordered, which corresponds to a **feasible schedule.** Then, the number of steps needed to find a feasible schedule equals at least the number of disjunctions. Moreover there are at the most $2^n$ leaves (n being the number of permutations of the hoist movements).

This led us to introduce strategies for the search of feasible solutions. For example, we can study first the couples for which the order is fixed by some constraints (depending on the bounds of the processing times : Equation (1) and (2).

For a given feasible schedule, finding one solution consists in assigning all the variables $t(R_i)$ and T. The optimal schedule is the one for which one solution minimizes the period T.

## IMPLEMENTATION WITH TWO DIFFERENT SOLVERS

Constraint Logic Programming languages [16],[17] are emerging techniques in Artificial Intelligence. The purpose of CLP is to integrate in a very sound way the resolution mechanism of the Prolog language with some specific constraint solving techniques [6],[8],[16],[2],[3]. In substance, unification is supplemented by constraint solving techniques on interpreted domains in order to enhance the efficiency of the resolution. Thus, CLP languages extend Prolog by adding solvers on several domains like rational numbers, booleans, trees, or finite domains. The main available products are : PROLOG III [3], CHIP [4],[16] and CLP(R) [8], [5]. The principal applications of CLP systems are problems with some combinatorial aspects, which need declarative modelling both by strategic knowledge representation and by constraints definition. It is the case for many industrial product automation problems like scheduling, planar cutting-stock or circuit design.

For the HSP the implementation using a CLP language is shared in two parts (Fig. 3). The first one, that we have presented in section 1.2.4., uses the constraints' propagation and the depth first procedure of CLP language, in order to obtain a feasible schedule. Then, for the resulting strongly constrained area, the second part uses the enumeration and optimization facilities of the CLP language to find the best solution of this feasible schedule. Finally, backtrack permits to repeat this procedure until all the feasible schedules are found. This ensures that the optimal solution will necessarily be found.

Moreover, we can introduce a new bounding criterion in the exploration of the search tree: when a solution is reached, we take the T value just found as the new higher bound for the period. This improves the efficiency of the resolution.

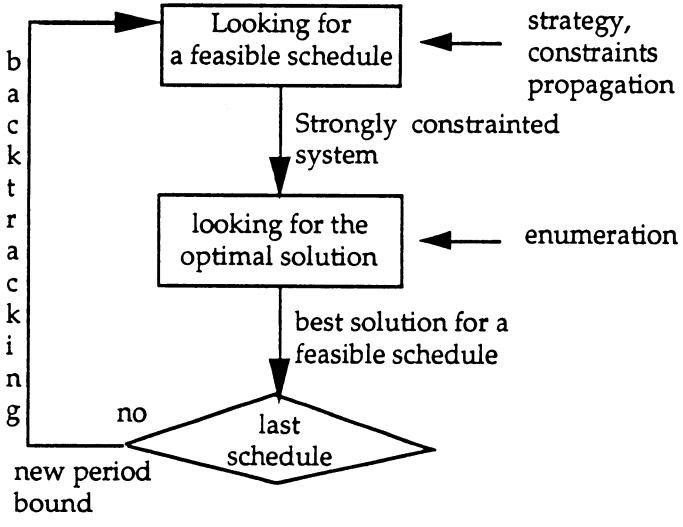


Figure 3. Strategy developed to find the optimal solution.

Prolog III

The first implementation uses the language PROLOG III developed at the G.I.A. in Marseille [3]. The PROLOG III solver uses an incremental simplex algorithm for linear inequalities on rational numbers and Gauss algorithm for equalities. It also proposes a booleans treatment based on a variation of SL-Resolution.

It seems to be an appropriate language for this kind of problem. Our problem is composed of constraints which can be expressed by linear equations and inequalities. Furthermore, we have previously shown that the strategy used to study the disjunctive constraints can easily be implemented in PROLOG III. The disjunction is simply expressed in logic programming by a non-deterministic clause.

Computational results with PROLOG III are available in [1] and [7], and are summarized in section 4..

CHIP

After some conferences where we have presented the results obtained with PROLOG III, a question remained without any answer: *"Why not use CLP language on finite domains, for example CHIP ?"*

CHIP (Constraints Handling In Prolog) was developed at the ECRC in Munich. This logic programming language embeds constraints solving techniques and contains three computation domains : finite domains, linear rational arithmetics and boolean algebra. For each of these domains CHIP uses specialized algorithms. While consistency checking techniques are used for finite domains, more mathematical tools (simplex algorithm and equation solving in boolean algebra) are used for rationals and booleans. Optimization primitives are also included in CHIP. For instance *minimize(Goal,Function)* can be used to find the solution of *Goal* which minimizes *Function*. It is implemented by using a branch and bound procedure. CHIP has been applied to various combinatorial problems, like Planing problems, Graph colouring or Disjunctive scheduling [4], [17]. CHIP appears to be a good approach for the HSP, as it allows to solve combinatorial problems using a combination of non-determinism and constraints resolution.

## CLP CONTRIBUTION IN OPTIMIZATION

Development facilities

The main advantage of CLP languages is the very **expressive power of constraints and logical representation.**

The various domains of constraints are very helpful to obtain a good matching between the model and the program. So, the design and the development of programs are quite simple once the system is understood. If the domains of the constraints match the domains of the application objects, a fine modelling of the problem can be completed in a very short time. The conciseness of the programs and the short development times encourage writing alternative versions. As shown by the test sets (see below), quite exhaustive experimentation can be performed in a convenient way. Hence, CLP is undoubtedly a key tool for a deep exploration of various models for an application. So, CLP Languages allow *evolutionary approach* , a basis for the prototyping life cycle. [13].

Although PROLOG III or CHIP enables one to keep up a high level of abstraction, writing a program in these languages has nothing to do with the simple translation of a formal specification

(comparing the formal specification of a critical set and the varied PROLOG III clauses computing such sets, may help to convince oneself ). It is a real programming task for which the main difficulties are to define an adequate representation of the objects and to find an efficient control strategy . Such a strategy is of course based on an active use of the constraints, but often one must also ***take advantage of the application object specific properties*** in order to obtain really good execution times.

Extension capabilities of the model

N-periodicity. The 1-periodic solution forces the processing time in each tank to be the same for all products. This involves strong constraints which are not set in the problem. We propose an extension of the first model, which provides a general N-periodic optimal solution and then reduces those constraints.

An N-periodic cycle is characterized by the input and the output of exactly N products. The entry of two successive products $k$ and $k+1$ (k=1 to N) is separated by the offset $t_k$. The cycle length that we have to optimize equals the sum of the N offsets.

The sequence remains the same for all products, but the processing time in each tank can differ. So we have to modify the preceding model by introducing an additional index $j$ to identify each product. Then the system is defined by the new variables $o_{i,j}$, $t(R_{i,j})$ and the N offsets $t_k$.

Multi-hoists extension It consists in partitioning the line in areas assigned to each hoist. A hoist will be allowed to remove a carrier only from a tank belonging to its area. A hoist can move a carrier to an area assigned to an adjacent hoist. We define movement limits for each hoist so that it can only interfere with adjacent hoists.

The P-hoists problem is then equivalent to a system composed of P single hoist problems and a set of disjunctive constraints dedicated to avoid collisions between each pair of adjacent hoists; to solve the P-hoists problem, we search for the optimal common solution of the P sub-problems which respects those constraints.

With such a partition, we identify only ten cases of possible collision. For each one, we have defined a rule that orders the operations of two hoists, so that a hoist has enough time to leave a place before the other one arrives.

Use of the results

One of the advantages of the utilization of CLP to solve such a problem is that we can obtain more than the optimal solution. We can enumerate all the solutions for all the feasible schedules. The particular structure of this set of solutions for a benchmark problem [12] led us to make various remarks : on the one hand, the number of feasible schedules is generally small. For example, for the benchmark problem, with bounds of the period fixed to 520 and 620 time units, we found only six feasible schedules. On the other hand, there are about 10.000 solutions! Even for the same feasible schedule and the same period, we can find different processing times!

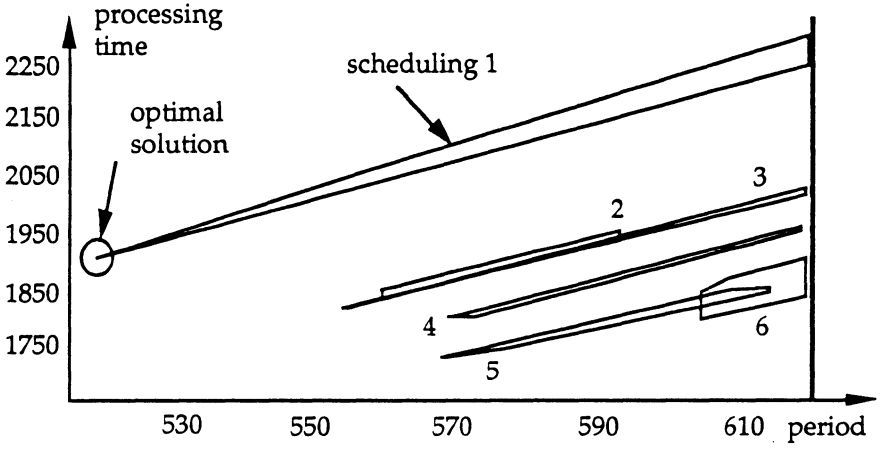


Figure 4. Set of all feasible solutions with T < 620.

The analysis of this set of solutions will probably help us to find best heuristics to improve the performances of the resolutions. On the benchmark solutions and some other examples, we have observed that for each feasible schedule we have a continuous domain of solutions T (Fig. 4). This could possibly be used to adapt the system in case of some perturbations by changing the solution applied (change of T or even only of some processing times). Moreover, for any value T (>T optimal), we can find at least one feasible schedule. We conclude for such structures that the solutions T form a continuous set. Since there are few feasible schedules, we can hope to solve the problem faster by using a dichotomous procedure on T, and by searching for the minimum value of the period for which a feasible schedule exists.

## SYNTHESIS ON THE SOLVERS USED

### Solvers performances

We implemented the model with the two CLP languages CHIP and PROLOG III. The first important result we can extract is that this new model provides the optimal solution. The computational time of the resolution is entirely comparable with the previous Operational Research approaches. Moreover, the N-periodic or/and multi-hoists extension does not increase much the complexity of the associated model and above all its implementation.

In order to compare our results with the ones of the Operational Research, we solved the benchmark example of Phillips [12] on a SUN 4 -SLC 8Mo RAM. We obtained the same optimal solution as Shapiro [15] (521 time units) for the single hoist configuration.

For the benchmark, with a 1-periodic and two hoists configuration, we obtained 240 time units for the optimal period. In the 2-periodic and single hoist case, the optimal solution is T=1004 time units (hence, an average cycle length of 502 time units).

### Cost of the partial consistency for CHIP

Let's remark that all the solvers used in CLP languages embed well-known Operational Research tools. In PROLOG III, a linear constraints system over rationals is solved with a classical simplex algorithm. For CHIP, the resolution of constraints on finite domains uses consistency checking techniques aiming at reducing the search space of variables. The CLP languages permit to easily program a mathematical model using a more declarative and natural expression of most problems. So, this approach allows us to avoid the heaviness of procedural languages. Consequently programs are much easier to extend.

As we have shown in the preceding section, the computational time for the resolution of the problem is obviously different for the two solvers we have used. So, we tried to understand what is causing such a difference. We have noted that all the solvers on finite domains use partial consistency techniques to reduce the search space of a solution. A complete consistency checking will appear to be computationally too expensive for a constraints programming language. The non completeness of such tools, leads our problem resolution to examine useless branches of the search tree. When a new constraint is added to the current system, the propagation mechanism of constraints does not eliminate all the impossible values of each variable. In some cases, when the

resolution reaches a leaf, the domain of some variables remains the same as the one of the beginning, and any values of this final domain is a possible solution. So, the associated schedule is not a feasible schedule. In fact, such branches of the search tree are failed when we minimize variables values. This is the reason why we consider CHIP as a good CLP language, but for large combinatorial problems like the HSP it is better to use exact resolution of constraints, as in PROLOG III. Of course, PROLOG III works with rational arithmetic, so it is necessary to enumerate on integers if we want to obtain integer solutions. Nevertheless, this is not the worst inconvenient, compared with the partial consistency in CHIP.

## CONCLUSION

As a conclusion, it is firstly very important to notice that any of the two CLP programs (one with PROLOG III, one with CHIP) give optimal results as fastly as the Operationnal Research methods dedicated to the HSP.

The advantages of CLP languages are: the capability to easily obtain a computable model with a high abstraction level, the possibility to include any kind of specific properties of the problem, the evolutive character of this model. The disadvantage is clearly that the solver is general and then, no particular solver is defined for a specific problem. For the HSP, it is clear that the advantages are heavier than the drawbacks. We can feel that it is mainly due to the strongly constrained character of our problem. As a matter of fact, we can also remark that our approach is not so different from the dedicated Operationnal Research method.

The comparison between two different languages shows that in this case, PROLOG III is more interesting than CHIP. Our approach is composed of two different phases: in the first one, we are looking for a feasible schedule, only known by a set of constraints, and in the second one, we enumerate all the values of "*T*" in order to find the smallest. With CHIP, the consistency is partial. So, the program cannot ensure that the schedule obtained is really feasible. Then, even with a powerful MinMax function, we cannot find any solution. With PROLOG III, the consistency is ensured, so we know that at least one feasible solution exists. In our case, the final domain of variables (after reduction) always includes an integer solution. As long as it is the case, it is more interesting to spend more time on the consistency than on an useless enumeration.

## REFERENCES

1. Baptiste, P., Legeard, B. and Varnier, C. 'Hoist Scheduling Problem : an approach based on Constraint Logic Programming', *Proceedings of IEEE International Conference on Robotics and Automation*, vol 2, pp.1139-1144, Nice, France, May 1992.

2. Cohen, J. 'Constraint logic programming languages',*CACM*, vol. 33, n°7, pp. 52-68, July 1990.

3. Colmerauer, A. 'An introduction to Prolog III', *CACM*, vol. 33, n°7, pp. 71-90, July 1990.

4. Dincbas, M., Van Hentenryck, P., Simonis, H., Aggoun, A., Graf, T. and Berthier, F. 'The Constraint Logic Programming Language CHIP'*Proceedings of the Int. Conference on Fifth Generation Computer System, ICOT*, pp. 693-702, Tokyo, Japan, November 1988.

5. Fattah, Y. E. 'Constraint Logic Programming: Applications and Implications' *Artificial Intelligence in Engineering,*, 7, pp. 175-182, 1992.

6. Jaffar, J. and Lassez, J.-L. 'Constraint Logic Programming' *Proceedings of Principle of Programming Languages*, pp. 111-119, Munich, Germany, 1987.

7. Lacoste, M.-A. and Baptiste, P. 'A Constraint Logic Programming Approach for the N-periodic Hoist Scheduling Problem' *Third International Workshop on Project Management on Scheduling*, Como, Italy, July 1992.

8. Lassez, C. 'Constraint logic programming' *Byte*, pp. 171-176, August 1987.

9. Lei, L. and Wang, T.-J. 'On the Optimal Cyclic Schedules of Single Hoist Electroplating Processes', *Working paper #89-0006*, Rugters University, U.S., april 1989.

10. Lei, L. and Wang, T.-J. 'A proof : The Cyclic Hoist Scheduling Problem is NP-Complete", *Working paper #89-0016*, Rutgers University, U.S., August 1989.

11. Lei, L. and Wang, T.-J. 'The Minimum Common Cycle Algorithm for Cycle Scheduling of two Material Handling Hoists with time window constraint', *Management Science*.

12. Phillips, L. W. and Unger, P. S. 'Mathematical Programming Solution of a Hoist Scheduling Program' *AIIE Transactions*, Vol 8, n°2, pp 219-225, june 1976.

13. Rueher, M. and Legeard, B. 'Which role for CLP in software Engineering? An investigation on the basis of first applications' *First international conference on the practical application of Prolog*, Research report 91-24, London, April 1991.

14. Shapiro, G.W. 'Hoist Scheduling for a PCB Electroplating Facility' *Thesis for the degree of Master Science (Program in Operations Research)*, Graduate Faculty of North Carolina State University, 1985.

15. Shapiro, G.W. and Nuttle, H. 'Hoist Scheduling for a PCB Electroplating Facility' *IIE Transactions*, vol 20 n° 2, pp 157-167, june 1988.

16. Van Hentenryck, P. *Constraint satisfaction in logic programming* MIT Press, Cambridge, Mass., 1989.

17. Van Hentenryck, P. 'The CLP language CHIP : constraint solving and applications' *Proceedings of the conference COMPON*, pp.382-387, San Francisco, 1991.