Global planning of several plants

Sylvie Bescos

BIM sa/nv
Kwikstraat,4
B-3078 Everberg (Belgium)
e-mail: sb@sunbim.be

Abstract

This paper discusses an attempt to solve the problem of planning several pharmaceutical plants at a global level. The interest in planning at this level is to increase the global control over the production process, to improve its overall efficiency and to reduce the need for interaction between production plants. In order to reduce the complexity of this problem and to make it tractable, some abstractions have been made. Based on these abstractions, a prototype is being developed within the framework of the EUREKA project PROTOS, using Constraint Logic Programming techniques.

Introduction

This paper describes the development of a prototype "global planning tool" within the framework of the EUREKA project PROTOS [PROTOS90]. PROTOS aims at the application of Prolog-based techniques to real-life planning and scheduling problems. The problem addressed by this prototype was proposed by one of the PROTOS partners, which is a large swiss pharmaceutical company.

The whole production of this company is split over several plants. The aim is to compute a global production plan for all these plants. Up to now, there is no such global plan, and all the coordination and adjustments of the production process between the different plants is achieved through phone calls between plant managers; there is no global control. This scheme works because of the experience and know-how of the plant managers, but the result is far from optimal.

If a good global plan could be provided, ensuring that no major coordination problem should occur, then each plant could make local optimisations as long as the constraints imposed by the global plan are respected; also the resulting production process would become much closer to optimality. As a side effect, this global plan would also reduce the need for the phone call based coordination, although it is not expected to suppress it totally.

As it is far too complex to take into account all details of the local data of each individual plant, the considered global planning tool is based on an approximation of the local reality. Thus, the output of this tool is only a "rough" global plan, that will then be further refined at each plant, by the local scheduling tool (in this case a job-shop scheduling tool).

The implementation tool chosen was the Prolog III system [Col90], in order to take advantage of the recent advances in the Constraint Logic Programming field [Coh90, VH89].

1 Problem description

Scheduling problems are known to become quickly intractable, because of combinatorial explosion. This gets even worse when trying to compute a global plan for several plants, as it is practically impossible to consider all details of each plant. This problem has to be simplified somehow.

The work described here is based on one approximation of the local reality, which is the abstraction of individual machines in *machine groups*.

In order to define a machine group, some terms have to be introduced:

- the word "product" designates both intermediate and finished products.
- several production steps are needed to go from one or several intermediates to the product of the next upper level; all these steps are grouped in a single "production order".

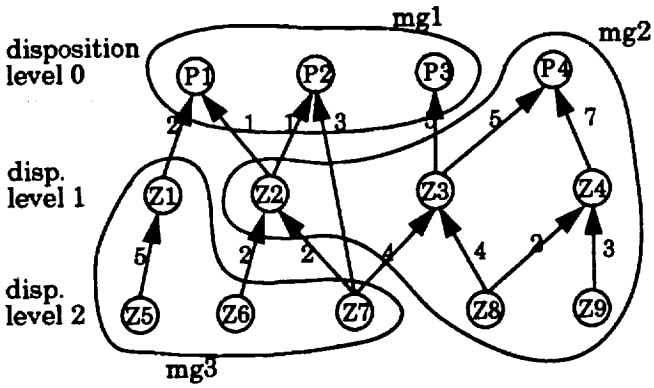
A machine group is a set of machines located physically close to each other, and each order can be completely executed using only machines within one machine group.

Also, at the global planning level, the different production steps of one order are abstracted in only one production task. Thus, one order is considered as being one task using one resource.

The global planning tool takes as input:

1. demands for finished products, a demand[1] being a pair (amount, due date),
2. the allocation of machine groups to products (each product is considered as being always produced on the same machine group),
3. *the dispositive bill of materials*:

   E.g. dispositive bill of materials with machine group allocation to products:



   Legend:

   * circles are products,
   * an arrow from A to B means that product A is an input to the production of B,
   * a number *n* near an arrow between A and B means that *n* units of product A are needed to produce 1 unit of product B,
   * shapes round products represent machine group allocation: e.g., products P1,P2 and P3 will be produced on machine group mg1.

4. stock data,
5. eventually, existing machine group allocations to some orders.

The requirement is to generate time windows for all finished and intermediate products appearing in the dispositive bill of materials, from the demands of finished products. For this, a convenient sequence for the production of the required finished and intermediate products has to be found.

The prototype has to perform *backward scheduling* where planning starts from the finished products and the allocations are made as late as possible. Backward scheduling in this way tends to minimize stocks.

While it is hoped that a conflict-free solution can be found in most cases, this might not always be the case because of the abstractions/approximations made. When no conflict-free solution exists, the global planning tool has to generate the best imperfect solution (i.e. featuring some conflicts on resource allocations). This best imperfect solution can then be used at the local scheduling level, which still has some flexibility that does not appear at the global planning level, and which could possibly solve conflicts.

The complexity of the problem not only comes from the number of demands to plan, but also from the handling of stocks and residuals:

- stocks may be available at the beginning of the planning period;
- additional stocks are likely to be generated during the production process because of some production constraints: it is not possible to produce less than a minimum quantity of a product at once (minimal lot size);
- residuals can be regenerated during the production process: e.g. the production of Z3 regenerates a certain amount of Z7, that could be used as input for the next demand of Z3 (not shown on the dispositive bill of materials drawn above).

This results in a "chicken and egg" problem:

- to find a sequence between the production tasks, it is needed to know the amounts to be produced, as the duration of a production task depends on the amount to be produced;
- the amounts to be produced depend on stocks, and the stocks evolve with time during the planning period depending on the chosen sequence of production.

2 Cutting the complexity

2.1 Decomposition of the planning horizon into sub-periods

To solve this "chicken and egg" problem, a further approximation was introduced in the planning process model. This approximation divides the planning horizon into several "sub-periods". This means that stocks are taken into account as if they were available only at the frontiers between these sub-periods.

1. In the following, "order", "production task", "demand" will be used indiscriminately.

In this way, it is still possible that more is produced during a sub-period than is strictly necessary: some stocks created during this sub-period (because of minimal lot size constraints) could have been used to reduce some demands for the same products occurring later in the same sub-period. However these stocks are likely to be used during the next sub-period, as a particular product is often produced again several times in the year[1]. Thus stock levels over the whole planning horizon should remain relatively stable.

It is not necessary to actually perform the planning of a sub-period in order to know how much stocks will be available at the end: all the demands in this sub-period will be produced, so it is not needed to know the exact sequence to compute the global result in terms of stocks available at the end.

It is then sufficient to:

- group the demands into sub-periods, according to their due dates;
- rearrange the demands, within each sub-period, taking into account stocks available at the end of the preceding sub-period, and compute the new stock levels at the end of the current sub-period.

This process is repeated for each disposition level in turn, starting with level 0 (i.e. finished products). The reason for starting with disposition level 0 is simply that initially, there are only demands for finished products, from which demands for intermediate products have to be successively derived.

2.2 Decomposition of the problem according to machine groups

The planning problem consists in making choices about a sequence and precise dates for all the demands to be produced. This search space is too wide to expect reasonable computation times. It is then needed to decompose the search space into several sub-spaces that can be treated independently.

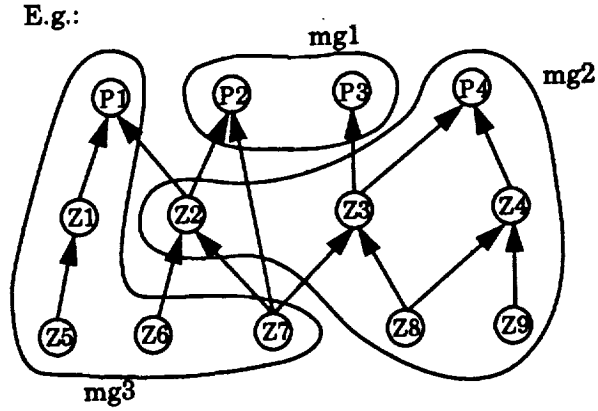
The machine groups serves as a basis for this decomposition:

- the list of machine groups is ordered according to dependency links, to obtain a so-called *machine group graph* (this more or less reflects the fact that a machine group is allocated to lower or higher levels of the dispositive bill of materials),
- for each machine group in turn:
  - * a sequence and particular dates for the tasks are chosen;
  - * these choices are committed;
  - * due dates and earliest beginning dates[2] for tasks allocated to the remaining machine groups are propagated.

2.3 Cycles in the machine group graph

There is a cycle in the machine group graph when, between two production tasks that are allocated to the same machine group, there exists one or several intermediate production tasks to be performed on other machine groups. According to the experts of the pharmaceutical company, this is unusual, and it is acceptable in such cases if the result is not as good.

E.g.:



In this example, there is a cycle between mg2 and mg3, because of the links between P1 and Z2, Z2 and Z6, and Z2 and Z7. If mg2 is treated before mg3, the demand on Z2 coming from that on P1 will be planned as late as possible with respect to the precedence constraints, which will eventually result in no freedom being left for the demand of P1. If mg3 is treated before mg2, then the demands on Z2 and even Z3 will eventually be too constrained.

Such cycles must be cut. The minimum number of links in the dispositive bill of materials that have to be cut in order to eliminate the cycle are marked (in the above example, the link Z2 → P1 is cut rather than the links Z6 → Z2 and Z7 → Z2). When there is a dependency between two production tasks along one of these links, these tasks are

1. Regulations require a pharmaceutical company to have several years of stocks, so external demands are not customer-driven. For most finished products, the yearly demand is split into several ones with due dates distributed over the year.

2. The earliest beginning date of a demand is the date when all input products are available.

further constrained so that the planning freedom is equally shared out among these tasks.

3 The program

The program has been implemented using Prolog III, a prolog interpreter with integrated constraints over rationals, booleans, and lists.

The basic algorithm is:

- first the machine group graph is computed from the dispositive bill of materials and the machine group allocation to products;
- then the data structure, which is a network of demands linked by constraints, is constructed;
- a schedule is computed;
- finally, the resulting plan is shown in a graphical form.

The construction of the data structure and the planning process will now be described:

Data structure

The data structure is a list of demands/orders represented each by a term:

```
[ id, product, machine group, due date,
  duration, end date, dependency info ]
```

It is constructed starting from the highest level of the dispositive bill of materials (i.e. finished products) going to the lower levels. At each level, for each product:

1. demands are grouped into sub-periods according to their due dates;
2. for each of these sub-periods in turn:
   a. demands are rearranged according to minimal lot sizes constraints, residuals and stocks available at the beginning;
   b. stocks that will be available at the end are computed;
3. from all these rearranged demands (over the whole planning horizon), demands for intermediate products are derived, and data about residuals is updated.

During the construction of this data structure, several kinds of constraints are enforced:

- precedence constraints,
- stocks availability constraints:
  * stocks of a product are considered to be available only after the end of the last allocation for this product during the preceding sub-period.
  * a demand that will take some amount of an input product from stocks is constrained to begin later than this date.
- residuals availability constraints: the use of residuals is allowed only if the demand is already constrained to begin later than the end date of the residuals production.

Planning, making choices

Even after decomposing the problem according to machine groups, the search space still needs to be reduced in order to make the program reasonably efficient. As it seems sensible to treat together demands that are close in time, sub-periods will be introduced again here. Choices will be committed after planning each sub-period.

However this decomposition into sub-periods implies some additional constraints. In order to express these constraints, it is needed to define the "planning limit" for a machine group as the latest end date of all allocations of this machine group for the demands of the previous sub-period. The additional constraints are that no allocation for the current sub-period can be made before this planning limit (to reduce the complexity, otherwise it would be needed to check disjunction with allocations of preceding sub-periods).

For each machine group in turn (starting from the machine groups allocated to the higher levels of the dispositive bill of materials):

- the demands are grouped into sub-periods, according to their due dates;
- for each sub-period:
  * if it is possible to find a conflict-free sequence, a maximisation of the minimum of all end dates is performed (so that the whole set of production tasks is planned the latest as possible);
  * if no conflict-free solution exists, conflicts are progressively allowed but minimised. This minimisation has to be based on a conflict evaluation. However, finding a convenient cost function of conflicts is a problem in itself, and one of the objectives of this prototype is to experiment with different ones. Up to now, the implemented measure is simply a count of the number of days in overlaps.

These optimisations are local to one machine group during one sub-period because a global opti-

misation would be too expensive in computation time.

The resulting plan contains precise dates for each production task instead of just time windows, as was requested at the beginning. In fact, this result can be viewed as a particular "fully instantiated" solution of the problem. In order to leave some freedom to the local plants, a more general solution could be retrieved, by deducing time windows from these precise dates and from the dependency information which was kept in the data structure.

4 Computational results

Two versions of the program exist:

- a coarse one for getting a rough idea of the resulting plan quality allowed by a given machine group allocation,
- a finer (but slower) one for getting the best possible plan for a given machine group allocation.

There is currently a dearth of representative examples (the extraction of the machine group information from the detailed description of each plant is still an open problem being tackled by people from the pharmaceutical company), and so no figures are yet available.

However, what has been learned from the development of the current prototype is the adequacy of the CLP approach for prototyping. The CLP approach allowed a switch from one version of the algorithm to alternative ones in a very short time, because of the declarativity and expressiveness of CLP languages.

Conclusion

The validation of the approach described in this paper can only come from the experimental use of this prototype together with several instances of a local plant scheduling tool, in order to check whether feasible plans are obtained. Such experiments have not yet been possible because of the difficulty in extracting the machine group information from the detailed data.

Up to now, the main interest in this work has been the refinement of the approach during discussions with experts from the pharmaceutical company. These discussions were based on hypothetical examples and on the successive versions of the program which have lead to the one presented here.

When representative examples become available, this research will go on by using this prototype to experiment with different evaluation functions of conflicts, and to investigate about the validation of the resulting plan.

Acknowledgements

I would like to thank all PROTOS partners for many fruitful discussions. I would also like to thank Pierre-Joseph Gailly and Paul Massey for their helpful comments on earlier drafts of this paper. Partial funding for this work was provided by the ESPRIT II project 5246 PRINCE (PRolog INtegrated with Constraints and Environment for industrial and financial applications).

Bibliography

[PROTOS90] The EUREKA Project PROTOS. H.-J. Appelrath, A. B. Cremers, and O. Herzog Ed., Zurich, Switzerland, April 9, 1990.

[Coh90] Jacques Cohen. *Constraint Logic Programming Languages*. Communications of the ACM, Vol.33, No.7, July 1990.

[Col90] Alain Colmerauer. *An Introduction to Prolog III*. Communications of the ACM, Vol.33, No.7, July 1990.

[VH89] P. Van Hentenryck. *Constraint Satisfaction in Logic Programming*. Logic Programming Series, The MIT Press, Cambridge, MA, 1989.