6 Building Industrial Applications with Constraint Programming

Helmut Simonis*

COSYTEC SA, Orsay, France

Current Address: Parc Technologies Ltd, London, United Kingdom

6.1 Introduction

In this chapter[1] we will give an overview of real-life applications developed with constraint logic programming. Constraint logic programming (CLP) combines declarative logic based programming with specialised constraint solving methods from artificial intelligence, Operations Research (OR) and mathematics. It allows the clear and concise expression of a wide class of combinatorial problems together with their efficient solution. In parallel with ongoing research in this field, CLP is now increasingly used to tackle real world decision making problems. In a first part of the chapter, we will briefly present the methods and tools used for CLP and describe typical application areas. We introduce the concepts of global constraints, meta-heuristics and constraint visualisation, which are central to large scale constraint solving.

A second part of the presentation will give a classification of problem domains suitable for the constraint programming approach and discusses various large scale applications which have been developed using the CHIP CLP system. These applications were co-developed by COSYTEC with specialists in the respective application domain.

- The ATLAS application is used at the Monsanto plant in Antwerp for the scheduling of a herbicide production and bottling plant. It is a typical example of a semi-process industry scheduling problem with constraints due to limited storage capacity at all levels.
- The FORWARD-C system is used in 5 oil refineries world-wide for short- and medium term planning of the production sequence. A graphical user interface is used to describe the refinery and generate the constraint model. The constraint program is coupled with a non-linear simulation tool written in FORTRAN.
- The TACT system is an integrated transport planning and scheduling system developed for a large food-industry company in the UK. The system plans the delivery of raw materials to different processing factories with a fleet of lorries and a variety of other resources.

* Supported by the ESPRIT working group CCL-II, ref. WG # 22457.

[1] This paper is an extended version of reference [Sim95a].

H. Comon, C. Marché, and R. Treinen (Eds.): CCL'99, LNCS 2002, pp. 271−309, 2001.

The last part of the presentation will concentrate on lessons learned from this application development, showing interesting new research fields and practical problems in developing state-of-the-art decision support systems.

6.2 Constraint Programming

In this section we give some background on constraint programming, the type of problems that are handled and the techniques that are used.

6.2.1 Characteristics of Problems

We now briefly introduce constraint satisfaction problems and discuss the motivation for the use of constraint logic programming (CLP). We explain which type of problem is best suited for the CLP approach and where these problems occur in practice. They all share a set of characteristics, which make them very hard to tackle with conventional problem solving methods. A more general introduction to constraint logic programming can be found in [JM93] [FHK92] [MS98].

Combinatorial problems occur in many different application domains. We encounter them in Operations Research (for example scheduling and assignment), in hardware design (verification and testing, placement and layout), financial decision making (option trading or portfolio management) or even biology (DNA sequencing). In all these problems we have to choose among many possible alternatives to find solutions respecting a large number of constraints.

We may be asked to find an admissible, feasible solution to a problem or to find a good or even optimal solution according to some evaluation criteria. From a computational point of view, we know that most of these problems are difficult, since they belong to the class of NP-hard problems. This means that no efficient and general algorithms are known to solve them.

At the same time, the problems themselves are rapidly evolving. For example, in a factory, new machines with unforeseen characteristics may be added, which can completely change the form of the production scheduling problem. New products with new requirements will be introduced on a regular basis. If a scheduling system can not be adapted to these changes, it will rapidly become outdated and useless.

Another aspect is that the complexity of the problems is steadily increasing. This may be due to increased size or complexity of the problem, or may be caused by higher standards on quality or cost effectiveness. Many problems which have previously been handled manually now exceed the capacity of a human problem solver.

At the same time, humans are typically very good at solving these complex decision making problems. They have a good understanding of the underlying problem and often know effective heuristics and strategies to solve them. In

many situation they also know which constraints can be relaxed or ignored when a complete solution is not easily found. For this reason, it is necessary to build systems which co-operate with the user via friendly graphical interfaces and which can incorporate different rules and strategies as part of the problem solving mechanism.

Developing such applications with conventional tools has a number of important drawbacks:

- The development time will be quite long. This will increase the cost, making bespoke application development very expensive.
- The programs are hard to maintain. Due to their size and complexity, a change in one place may well require other changes in different parts of the program.
- The programs are difficult to adapt and to extend. As new requirements arise during the life cycle of the program, changes become more and more difficult.

The use of constraint logic programming should lead to a number of improvements:

- The development time is decreased, as the constraint solving methods are reused.
- A declarative problem statement makes it easier to change and modify programs.
- As constraints can be added incrementally to a program, new functionality can be added without changing the overall architecture of the system.
- Strategies and heuristics can be easily added or modified. This allows to include problem specific information inside the problem solver.

6.2.2 Main Concepts

Constraint programming is based on the idea that many interesting and difficult problems can be expressed declaratively in terms of *variables* and *constraints*. The variables range over a (finite) set of values and typically denote alternative decisions to be taken. The constraints are expressed as relations over subsets over variables and restrict feasible value combinations for the variables. Constraints can be given explicitly, by listing all possible tuples or implicitly, by describing a relation in some (say mathematical) form. A *solution* is an assignment of variables to values which satisfies all constraints.

Constraint propagation is the mechanism which controls the interaction of the constraints. Each constraint can deduce necessary conditions on the variable domains of its variables. The methods used for this constraint reasoning depend on the constraints, in the finite domain case they range from general, rather syntactic inference rules to complex combinations of algorithms used in global constraints. Whenever a constraint updates a variable,

the constraint propagation will wake all relevant constraints to detect further consequences. This process is repeated until a fixpoint is reached, i.e. no further inference is possible.

For many interesting constraint systems, the constraint reasoning and constraint propagation is incomplete, i.e. it can not detect inconsistency at all times. Propagation must be combined with *search*, which is used to assign values to variables. After each assignment step constraint propagation restricts the possible values for the remaining variables, removing inconsistent values or detecting failure. If a failure is detected, the search returns to a previous decision and chooses an alternative. Search in constraint programming is usually under user control, heuristics and strategies can be easily programmed as search routines.

Constraint programming can be expressed over many different domains like linear terms over rational numbers, Boolean algebra, finite/infinite sets or intervals over floating point numbers. Very interesting development is possible for most of these domains or even more general, domain independent constraint solvers. But in this paper we only discuss finite domain constraint programming, currently the most developed constraint domain. At the moment, probably more than 95% of all industrial constraint applications use finite domains. Most examples in this paper are using the CHIP [DVS88] [VH89] system and its finite domain solver.

6.2.3 A Combination of Ideas

We will now review some of the background techniques which are used inside CLP. Figure 6.1 shows the basic influences on constraint logic programming. In the next paragraphs, we will discuss these different influences.

CLP can be seen as a natural extension of logic programming. Problems are expressed declaratively in Horn clause form, and the built-in search mechanism provides a simple way to generate solutions. Most of the theoretical results about correctness and completeness can be extended to CLP [JL87] [JM93] [MS98].

Constraint propagation and consistency checking are techniques originating in artificial intelligence. These methods provide the underlying framework for the constraint solvers in CLP tools.

Many of the actual methods to solve particular constraints come from the areas of finite mathematics and of Operations Research. Often classical algorithms can be changed to work within the constraint framework. They usually must be made *incremental*, i.e. avoiding repeated work when more information becomes available during the search and *backtrackable*, i.e. remembering the state of the propagation so that we can return to this state later during our search.

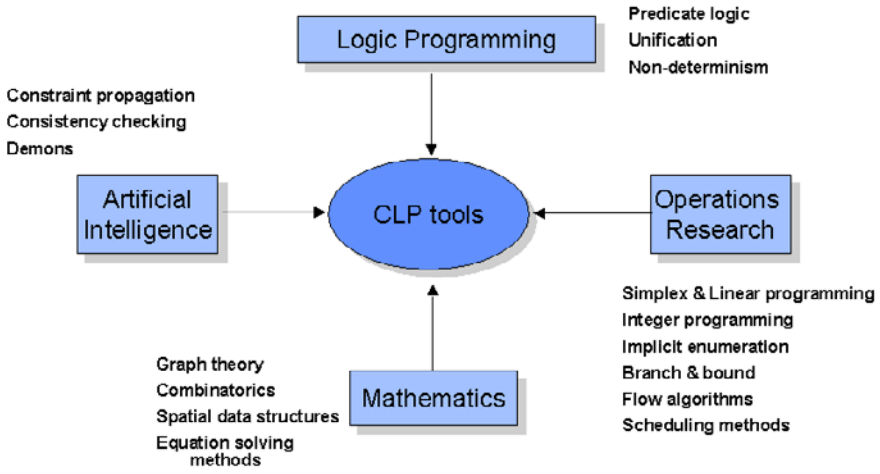

Fig. 6.1. Techniques behind constraints

6.2.4 The Emergence of Global Constraints

Much work on finite domain constraints in recent years was centred on the introduction of *global constraints* [AB93] [BC94] and *partial search techniques* [BBC97] [HG95]. We will briefly describe these topics before giving some examples of their use in a number of constraint applications.

The first versions of finite domain constraint systems were using rather simple propagation techniques. Consider the example shown in figure 6.2. Four variables are constrained by pair-wise disequality constraints. The constraint propagation for disequality is using *forward-checking* [VH89], which will solve the constraint as soon as one variable is assigned. With the given domains, no propagation is possible. Starting assignment with variable X, the values 1, 2, 3 are tested and lead to a failure, before the consistent value 4 is found for the variable X. We can avoid this useless search by applying better reasoning. The four variables must be pair-wise different, they must be alldifferent. Considering this one constraint, we can reformulate the problem as a *bi-partite matching problem* between variables and values and apply classical graph algorithms to solve it. We can deduce that the value 4 must be used by the variable X and the value 1 by the variable Z. The values 2 and 3 can be used for the variables Y and U.

This reformulation is the key to the understanding of global constraints. Instead of expressing a problem by many primitive, binary constraints we describe it with a few global constraint which work on sets of variables. Using this higher abstraction has two main advantages:

- Problems can be expressed more easily and with fewer constraints.

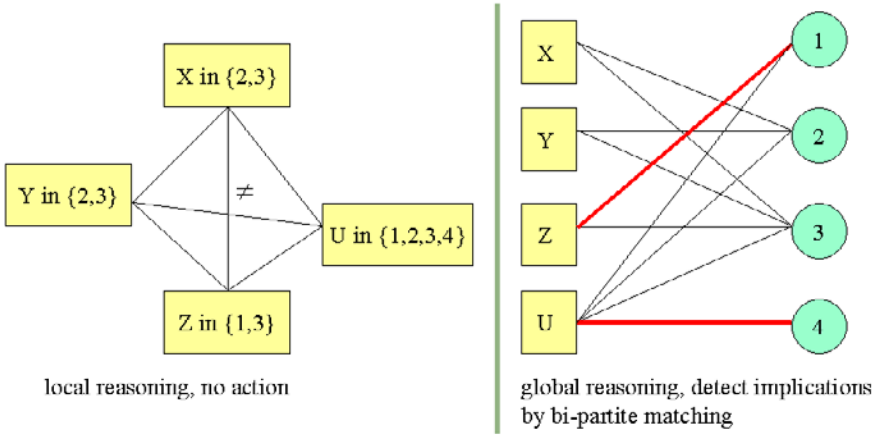

Fig. 6.2. The need for global reasoning

- The abstraction allows us to use more complex algorithms inside the constraints.

Obviously, it is not advisable to create a new global constraint for each application that we solve. They should satisfy the following criteria:

- The constraint can be used for many different applications.
- The abstraction is as general as possible.
- It can be used together with other constraints.
- Efficient methods for constraint reasoning are known or can be developed for this constraint.
- The algorithmic complexity of the methods is acceptable.

In CHIP, the development of global constraints is based on the constraint morphology shown in figure 6.3. There are five basic concepts (shown at the bottom) for constraints. The *different* concept states that items are different from each other, the *order* concept states that items are (partially) ordered. The *resource* concept expresses that items use limited resources, The *tour* concept states that items in different locations must be visited in some sequence and the *dependency* concept states that some item depends on some other items.

In the first versions of finite domain solvers, primitive constraints for each of these concepts were introduced (shown in the second line from the bottom). A binary *inequality* constraint could be used to express a partial order between variables, for example. The propagation methods for all these constraints are quite simple and can be described with only a few syntactic inference rules [VH89].

A number of constraints can be introduced to solve particular problems with global reasoning (shown in the center), but which are not sufficiently

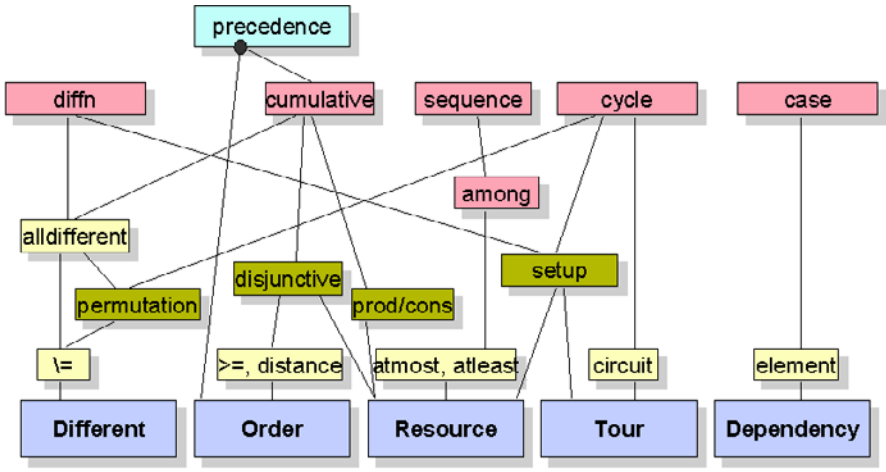

Fig. 6.3. Global Constraint Morphology

general to be considered as global constraints. A *disjunctive* constraint for example can be used to express resource constraints where each resource has capacity one and each task uses one resource.

But we can generalise the constraint to the case where the resource capacity can be arbitrary as well as the resource requirements for each task. This leads us to the *cumulative* [AB93] constraint, which is one of the CHIP global constraints (shown at the top). Connections between constraints show that the one below can be expressed by the one above.

The *precedence* [BBR96] constraint (at the top) is a further abstraction. It combines several cumulative constraints with a precedence graph between tasks. Considering these constraints together allows us to deduce even more information.

A number of other constraint systems include variants of global constraints. Variants of the cumulative constraint can be now be found for example in Sicstus Prolog [COC97], Eclipse [WNS97], IF Prolog and Oz [Smo95]. They differ not only in the parameters and side constraints that can be expressed, but also in methods and the mix of these methods inside the constraint reasoning.

6.2.5 Understanding Search

An important advantage of constraint programming over for example integer programming is the ease with which the programmer can control the search process. Most constraint systems have pre-defined search primitives, which can be easily extended for a particular program. The use of logic programming with its embedded backtracking search provides a particularly simple

framework. Two complementary developments have significantly improved search methods in the last years.

One is the development of partial search techniques as meta-heuristics. Chronological depth-first search is often limited in exploring only a small part of the search tree. Decisions which are made early in the tree are not reconsidered since too many alternatives for variables below must be explored. Partial search methods allow to overcome this problem by limiting the number of alternatives explored in different nodes. Schemes like *limited discrepancy search* (LDS) [HG95] or *credit-based search* [BBC97] can consider many different paths in the search tree, without excessive implementation overhead. These meta-heuristics can be combined with the usual heuristics and strategies based on variable and value selection, and so provide problem independent tools to improve search.

The visualisation of the search tree with tools like the Oz Explorer [Sch97] or the CHIP search tree tool allow a much deeper understanding of the problem solving process. Developed within the DiSCiPl Esprit project (22532), the CHIP search tree tool [SA00] (shown in figure 6.4) adds functionality to analyse propagation and the state of the domain variables in each node. This can be combined with the visualisation of global constraints [SAB00], which show the reasoning inside the constraint for each propagation step.

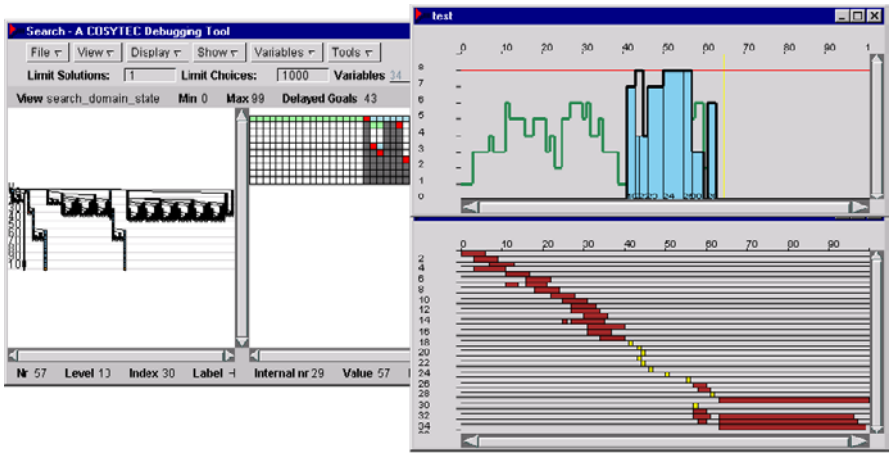

Fig. 6.4. Search tree tool and constraint visualisation

6.3 The CHIP System

While constraints are the centrepiece of the CHIP application environment, a number of other components play an important role in developing applications. We now give an overview of this environment.

6.3.1 System Components

The CHIP system consists of a number of different components which together greatly simplify the development of applications. The tool extends the functionality of a host language with constraints and other sub systems. Two host languages are supported at the moment.

- The Prolog based version of CHIP uses intrinsic language features like unification and backtracking search to achieve a deep integration of the constraints with the host language. Constraints and search procedures can be easily defined and extended in the base language.
- The C/C++ version of the CHIP system takes the form of a constraint library which can be used inside application programs. Since search and variable binding are not part of the host language, the integration is not as seamless as in the Prolog version.

Other modules of the CHIP environment include XGIP, a graphical library based on Xwindows, QUIC, an SQL interface to relational databases and foreign language interfaces CLIC and EMC. The different modules can be used together with an object modelling system based on the object layer CHIP++ to build large scale end-user systems [KS95].

6.3.2 History

CHIP [DVS88] was one of the first CLP systems together with PROLOG III [Col90] and CLP(R) [JL87]. It was developed at the European Computer-Industry Research Centre (ECRC) in Munich during the period 1985-1990 [DVS88] [DVS87] [DSV90]. Different constraint solvers were developed and integrated in a Prolog environment. At the same time the technology was tested on a number of example applications. The CHIP system is now being marketed and developed by COSYTEC. The early versions of CHIP were also the basis for a number of derivatives by ECRC shareholder companies. BULL developed CHARME, a constraint based system with a C like syntax, ICL produced DecisionPower based on the ECRC CHIP/SEPIA system and Siemens is using constraints in SNI-Prolog (later IF-Prolog). ECRC has also continued research on constraints, which led to the Eclipse system, now maintained by IC-PARC. COSYTEC's current CHIP version V5.2 differs from the earlier constraint system by the inclusion of powerful global constraints [AB93] [BC94], partial search methods [BBC97] and visualisation tools [SA00] [SAB00].

6.3.3 Behind the Scene

The CHIP system consists of around 220000 lines of C code. Table 6.1 gives an overview of the size of the different modules. The Prolog kernel accounts for less than 10%, the different interfaces and the graphical system use 20%.

The constraint kernel uses another 10% of the code. The remaining 60% of the system are used in the code of the global constraints, which combine multiple (up to 40) algorithms and constraint methods for each primitive.

Table 6.1. CHIP system lines of code

Module	Size (lines of code)
CHIP kernel	175000
Prolog	15000
Core Constraints	20000
Global Constraints	140000
Interfaces (CLIC,EMC,QUIC)	5000
XGIP graphics and libraries	37000

This distribution shows the importance of both the graphical interface XGIP and in particular the global constraint handling methods inside CHIP.

6.4 Application Studies

From the beginning, the development of CHIP was linked to application requirements. A large number of application studies showed the practical interest of constraint technology. Many of these studies were benchmark comparisons with either conventional, special purpose programs or Operations Research (OR) tools. Three initial areas of interest were covered, combinatorial problems from OR [DVS87] [DVS88] [VC88] [DSV90] [VH89] [DSV92] [DSV88] [DS91], circuit design applications [SND88] [SD93] [GVP89] and financial decision support [Ber90].

An overview of early application studies can be found in [DVS88a] [VH89]. Other application studies with CHIP (outside COSYTEC) are for example [BDP94] [BLP95]. Results on benchmark problems can also be found in [VSD92] [AB93] [BC94] [BBR96]. A general classification of problems suitable for the constraint approach is given in [Sim96], while different types of constraint applications are listed in [Wal96].

6.5 Industrial Applications

In this section we present some large scale, industrial systems developed with CHIP. These examples show that constraint logic programming based on Prolog can be used efficiently to develop end-user applications. The constraint solver is a crucial part of these applications. Other key aspects are graphical user interfaces and interfaces to data bases or other programs. Some background on how such applications can be developed with logic programming is given in [KS95].

We have grouped these applications in different categories (following the classification from [Sim96]) according to the types of constraints which are included.

6.5.1 Assignment Problems

Assignment problems were the first type of industrial application that were solved with the emerging constraint tools. A typical example is the stand allocation for airports, were aircraft must be parked on the available stands during their stay at the airport. Different from scheduling problems, the (provisional) arrival and departure times are fixed in this type of problem. Additional constraints further restrict which resources can be used for which activity.

HIT (ICL). Probably the first industrial CLP application was developed for the HIT container harbour in Hong Kong [Per91]. The program was written in an early version of ECRC's CHIP, later adapted to DecisionPower from ICL. The objective was to allocate berths to container ship in the harbour, in such a way that resources and stacking space for the containers are available.

APACHE (Air France). The APACHE system [DS91] was a demonstrator for stand allocation at Roissy airport in Paris. Here the objective was to replan the allocation whenever a change of arrival/departure times occurred. A detailed model of this program is described in [Sim97]. An operational version of such an application is running in Korea.

Refinery Berth Allocation (ISAB). Another instance of this application type is refinery berth allocation. Arriving crude oil tankers and barges for the transport of finished products must be loaded/unloaded at different jetties. Due to the high operating cost of crude oil tankers, ships must be processed immediately after arrival.

6.5.2 Network Problems

Applications in this group deal with generalised versions of the warehouse assignment problem already studied as a constraint problem in [VC88]. Limited capacity of warehouses or multiple product lines add new types of constraints to this otherwise well-understood problem from OR.

Locarim (France Telecom, Telesystemes, COSYTEC). This application lays out the computer/phone network in large buildings. Telephone connections in different rooms must be linked together via concentrators, which have limited capacity. Cables must be run according to the building

layout, which is scanned in from architectural plans. The system is used by France Telecom to propose a cabling and to estimate cabling costs for new buildings. This application was developed by the software house Telesystemes together with COSYTEC.

Distribution Planning (EBI). The Turkish company EBI has build a distribution planning system which determines the placement of different products in selected warehouses. Customers for multiple products are served from these warehouses. The objective is to minimise storage cost while achieving customer satisfaction with small delivery delays.

PlaNets (Enher, UPC/CSIC). This application controls the reconfiguration of an electrical power network in order to perform repairs and maintenance on some segments of the network. The system re-routes energy in order to minimise end-user problems respecting multiple electrical and other constraints. The program was developed at the university of Catalonia in Barcelona for the power company Enher [CGR95].

Banking Network (ICON). ICON, an Italian software house, has developed a system for facilities and load distribution in the Italian inter-banking network [CF95]. The program distributes applications over a network of mainframes in order to provide services to customers while balancing and minimising network traffic.

CLOCWiSe. The Esprit project CLOCWiSe (IN 10316I) is using CHIP for the management and operational control of water systems. A first implementation is under way at Bordeaux in France. The constraint model is being developed by the university of Catalonia in Barcelona.

6.5.3 Production Planning/Scheduling

Production planning and detailed scheduling are major application areas of the CHIP constraint system. Typical constraints are temporal sequences and limits, resource capacity limits and resource allocation problems. For a more detailed view on scheduling with constraints see for example [Sim95] [Sim97] [Sim98].

Plane (Dassault Aviation). Dassault Aviation has developed several applications with CHIP. PLANE [BCP92] is a medium-long term scheduling system for aircraft assembly line scheduling. The objective of the program is to decide on changes of the speed of production for the assembly line, called the cadencing of production. By changing the speed often, production

can follow varying demand more closely, decreasing inventory costs. On the other hand, each change in speed incurs a large cost for assigning/reassigning personnel and material.

Made (Dassault Aviation). Another application developed by Dassault, MADE [CDF94] is a detailed scheduling system for a complex work-cell in the factory. The program controls a cutting press and related machines in a work-shop. As part of the scheduling, the system has to find a 2D placement of different products on sheets of metal, corresponding to different orders which are processed at the same time on the press.

Coca (Dassault Aviation). The COCA system of Dassault is used for production step planning. The program decides which operations on one module of the aircraft can be performed together, depending on the rotational orientation of the aircraft and the place on the module where the operations are performed. Precedence constraints and safety rules must also be taken into account.

Production Schedule (Amylum/OM Partners). This program schedules production lines for a glucose producing factory in Belgium. Main constraints are the set-up restrictions, which exclude certain product sequences, and machine choices, which change production rates. The objective is to find a compromise between achieving due-dates for orders and using long production runs to minimise operating cost.

SAVEPLAN (ATOS). The French software house ATOS has redeveloped an existing, FORTRAN based scheduling and inventory control system with the help of the CHIP constraint solver inside an object based development environment.

MOSES (COSYTEC). The MOSES application has been developed by COSYTEC for an animal feed producer in the UK. It schedules the production of compound feed for different animal species, eliminating contamination risks and satisfying customer demand with minimal cost. This system was developed in 1997 and is now operational in 5 factories, parts of its constraint model are described in [Sim98].

6.5.4 Transport

The transport area contains many different types of interesting constraint problems. A major concept is tour planning, finding for example the optimal sequence of deliveries for a set of lorries which transport goods from factories to customers. The CHIP *cycle* constraint [BC94] was especially developed to model this type of problem.

EVA (EDF/GIST). EVA is used by EDF (the French electricity company) to plan and schedule the transport of nuclear waste between reactors and the reprocessing plant in La Hague. This problem is highly constrained by the limited availability of transport containers and the required level of availability for the reactors. The program was co-developed by EDF and the software house GIST.

Transport planning (EBI). As a second stage of the project described in section 6.5.2, EBI has developed a tour planning system for multiple warehouses/customers with capacity constraints on the lorry fleet. Each lorry can transport a limited amount of goods from some warehouses to customers. The sequence of deliveries is controlled by the urgency of the order and the objective to minimise travel costs.

DYNALOG. The DYNALOG Esprit project (26387) studies the use of constraints and high-performance computing for general logistics and distribution problems. Research there is focused on large-scale problems and the use of parallelism to speed up constraint reasoning.

COBRA (NWT/PA Consulting/COSYTEC). Another type of transport problem is solved with COBRA [SC98]. This program generates diagrams (work plans) for train drivers and conductors of North Western Trains in the UK. For each week, around 25000 activities must be scheduled in nearly 3000 diagrams, taking a complex route network into account. On benchmark problems, improvements of 2-3 % were obtained compared to the previous solution.

LCCR (Eurocontrol/COSYTEC). Yet another transport planning problem is handled in the LCCR system. Eurocontrol is responsible for allocation air-traffic control sectors to all flights in the European airspace. Due to severe capacity limitations, flights must be allocated start and landing times such that the capacity of the sectors is not exceeded. This demonstrator is still under development, using the C++ version of CHIP.

6.5.5 Personnel Allocation

Another important application group are personnel assignment problems. Applications in this domain are characterised by the variety of rules and restrictions which must be taken into account. These constraints arise from government regulations, union contracts or technical limits, and can become very complex.

EPPER (Servair). The EPPER [DD96] system developed for the catering company SERVAIR by the software house GSI performs personnel planning and assignment for the French TGV train bar/restaurant personnel. It creates a four week planning, assigning agents of the correct qualification to different catering jobs in the trains. The assignment respects travel times, rest periods and other hard constraints and tries to balance the overall workload for all agents.

TAP-AI (SAS Data/COSYTEC). TAP-AI [BKC94] is a planning system for crew assignment for the airline SAS. It schedules and reassigns pilots and cabin crews to flights and aircraft respecting physical constraints, government regulations and union agreements. The program is intended for day-to-day management of operations with rapidly changing constraints.

DAYSY (Lufthansa, SEMA, COSYTEC, U. Patras). Another Esprit project, DAYSY (8402), aims at developing a personnel re-assignment system for daily operations of an airline. It reacts to delays, cancellation or addition of flights or changes in the availability of personnel by changing the existing crew assignment to satisfy the new constraints. The project was undertaken by Lufthansa, Sema Group, COSYTEC and the university of Patras, Greece.

OPTISERVICE (RFO/GIST/COSYTEC). The system [Col96] generates a personnel assignment for each of the TV stations of RFO in the French overseas departments. It allocates qualified personnel to each of the tasks in the station, from reporting events to providing 24h service for technicians. The constraint system is combined with a rule-based module to identify working-time limits based on the different types of working contracts.

Gymnaste (UJF/PRAXIM/COSYTEC). The Gymnaste [CHW98] system produces rosters for nurses in hospitals. An important aspect in this application is the possibility to modify rules and regulations interactively, as well as taking multiple social factors into account. The program is operational in multiple hospitals in France.

MOSAR (Ministère de la JUSTICE). The MOSAR application generates the roster of prison guards in France in 10 regional sites for 200 prisons. It generates provisional plans for a one year period and helps to determine personnel needs. It was developed by Cisi and COSYTEC. The objective of the system is a reduction of overtime cost and an increase of personnel satisfaction by a more balanced, impartial rostering process.

6.5.6 Others

There are quite a number of CHIP applications for other application areas, ranging from database query optimisation to military command and control systems. One of the first large scale CHIP systems in daily use was the SEVE application described in the next paragraph.

SEVE (CDC). SEVE is a portfolio management system for CDC, a major bank in Paris, developed in-house. It uses non-linear constraints over rationals and finite domain constraints to choose among different investment options. The system can be used for "what-if" and "goal-seeking" scenarios, based on different assumptions on the development of the economy.

DSP Design. Another area where finite domain constraint can be used to advantage is the design of digital signal processing (DSP) hardware. The problem is to implement some filtering algorithm with the right mix of hardware modules. The trade-off between speed/size/cost leads to multiple scheduling and resource allocation problems which have to be solved in order to find a good design compromise. Very impressive results with CHIP are described in [SGK99].

6.6 Case Studies

In this section we present several case studies in more detail. The first application is a production scheduling system in a chemical factory in Belgium, the second a scheduling and simulation tool for oil refineries, the third an integrated transport planning and scheduling system for a food company in the UK.

6.6.1 ATLAS

The ATLAS system is a scheduling application for one part of the MONSANTO plant in Antwerp. The program schedules the production of different types of herbicides. The production process is shown in figure 6.5. Production basically consists of two steps, a batch formulation part and a continuous packaging (canning) operation. In the formulation part, batches of several ten thousand litres of chemical products are produced. Each formulation batch takes several hours. Several processors with different characteristics are available. After the formulation, the product is kept in buffer tanks with a very limited capacity. Exceptionally, it can be stored in movable containers for a limited time period. From the buffer tanks, products are fed to the different canning lines where the products are filled in bottles and cans of different sizes. The canning lines differ in speed, manpower requirements and the type

of products that they can handle. The bottles and cans are packed in cartons and palettes and stored in a holding area before delivery. Both the storage area for packaging material and the holding area are limited in size, which imposes constraints on the production process.

Besides many constraints associated with machine choice, set-up times and cumulative manpower constraints, producer/consumer constraints [SC95] play a major role in the problem solver. Intermediate products must be produced in the right quantity before they can be used on the packaging lines, and packaging material must be available in the warehouse before a task can be started.

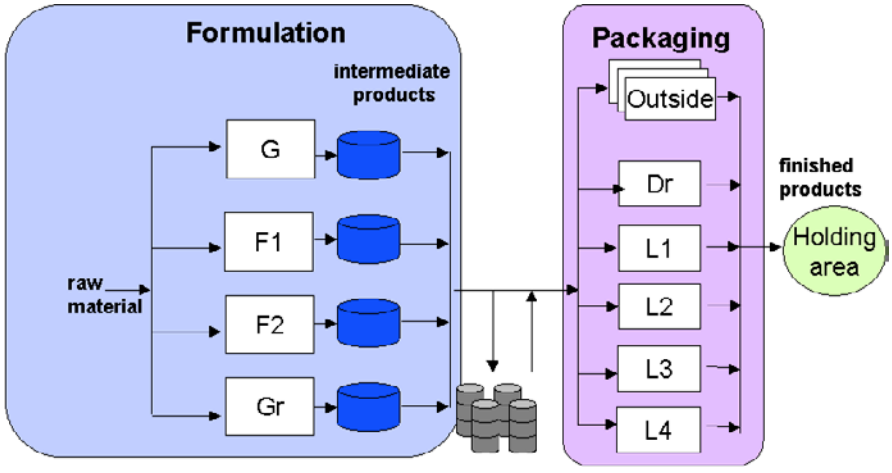

Fig. 6.5. ATLAS production process

The system works with data sets of several dozen intermediate products and more than one thousand packaging materials. A typical schedule may contain several hundred batches and canning tasks split into many sub-tasks to handle the stock level constraints with an eight hour resolution. In addition, several thousand constraints are needed to express other parts of the scheduling problem.

There is a production planning problem for the batch formulation part, where the system determines the correct number of batches for intermediate products and their optimal sizes. Since the storage tanks for intermediate products are limited in number and size, the correct amount of each product must be formulated to avoid left-overs which would block the storage tanks.

In figure 6.6, we show an example screen dump of the ATLAS application. The schedule is shown both in the form of a Gantt chart and as a textual list. The user can interact with the program via direct manipulation of items on

the screen. A number of different views is provided to manipulate schedules, resource availability or stock levels.

The system is used by several users with different responsibilities. The scheduler for the factory works together with people responsible for scheduling the formulation and the inside and outside packaging lines. Other persons control the inventory and are responsible for ordering packaging materials. These users continuously exchange information via the system. An overview of the dynamic scheduling environment is given in figure 6.7. Orders and sales forecasts are added to the data base from external sources. Information about stocks of packaging material and deliveries are also available, shop floor data is entered to update the current situation on the manufacturing side. The current production schedule is updated on demand and is then used to determine new requirements of raw materials and packaging materials. The program can be used on a "what if" basis in order to see the impact of accepting new orders or changing delivery due dates.

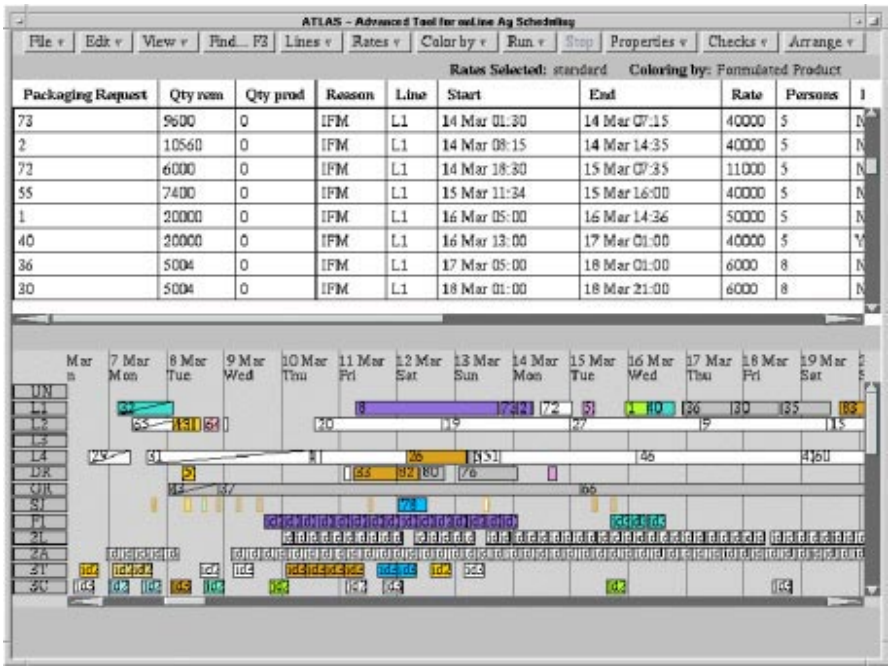

Fig. 6.6. The ATLAS application

The system architecture of the ATLAS system is shown in figure 6.8. The application runs on a UNIX workstation together with an Oracle data base. The different users connect to this application via their personal computers running an Xwindows emulation. The workstation is also connected to diffe-

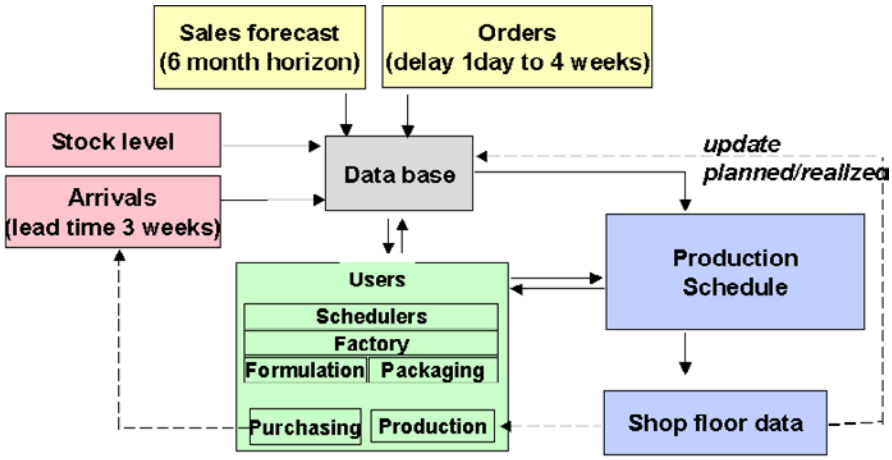

Fig. 6.7. The scheduling process

rent mainframe computers which keep information about orders, stock levels and deliveries. The master schedule is kept inside the data base, but each user can create and store local scenarios in file form. These scenarios can be "what-if" type evaluations, or alternative production plans, which can be exchanged between users.

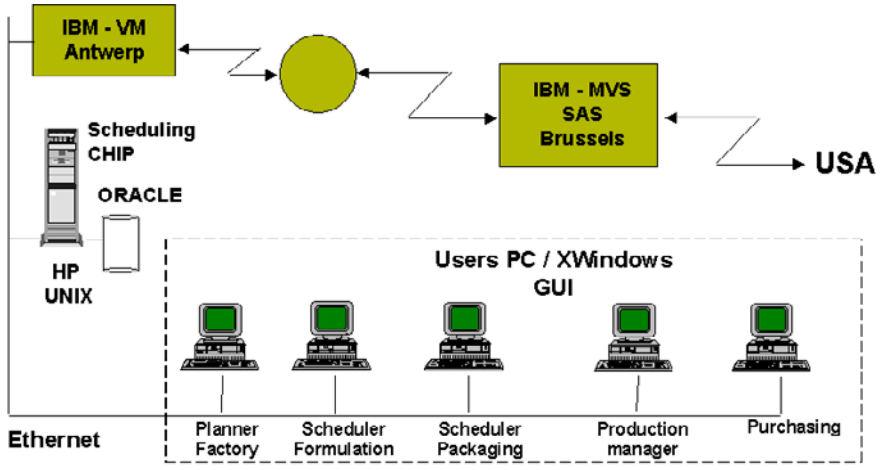

Fig. 6.8. ATLAS system architecture

The ATLAS system is operational since July 93, the complete project ended in spring 94. As an end-user system, the constraints are not visible to the user, all interaction is via the Gantt chart and stock level diagrams.

The user can move tasks manually and override the automated scheduler or change its strategy.

The system shows the type of complex scheduling problem which is well suited to the constraint approach. It is a real-life, not a pure problem, and does not fit easily into existing OR problem classifications. On the other hand, solutions which satisfy most constraints are hard to find. There are many forms of interaction between constraints and conflicts between different constraint types which can not be resolved automatically. The system is used as a decision support system, which helps the human scheduler, but does not replace him.

In important aspect during development was the possibility to incrementally express the constraints of the program. Rather than starting with a complete set of conditions, constraints were added by groups to understand the effect and requirements of each constraint type.

We discuss some implementation aspects below in section 6.8. A significant development effort was required to model and solve the problem and to generate the integration and graphical interface parts. A major revision of the application was performed in 1998 to provide an interface with SAP/R3.

6.6.2 FORWARD

The second case study is the FORWARD [GR97] system, a scheduling and simulation tool for oil refineries. It is used to plan refinery operations for production changes on a rather detailed level for next three days and at an overview level for the next three weeks. The system is currently in use at three sites in Europe and two refineries in Asia. Each refinery has its own particular features and operation procedures. This means that the program must be extensively customised for each site. The first installation is operational since September 94. FORWARD is a joint development of the engineering company TECHNIP and COSYTEC. It combines a non-linear simulation tool written in FORTRAN with a scheduling and optimisation part, which were developed in CHIP together with the graphical user interface.

To explain the function of FORWARD, we present an (idealised) overview diagram of a refinery in figure 6.9. Crude oils arrive by tanker or pipeline and are stored in large tanks in the tank farm. Different types of crude may be mixed to feed the process part of the refinery, starting with the crude distillation unit (CDU). A large number of other units are connected to different branches of the CDU output. They are used to maximise the percentage of light (valuable) products, which are obtained from the crude oil. The mixture of crude oils determines the throughput of the processing units and thus the economic efficiency of the refinery.

At the end of the process part semi-finished products are stored and then blended together to obtain commercial products like gasoline of defined qualities. Up to 12 different components can be used to generate gasoline for example. The ratio of the different products determines the cost of the

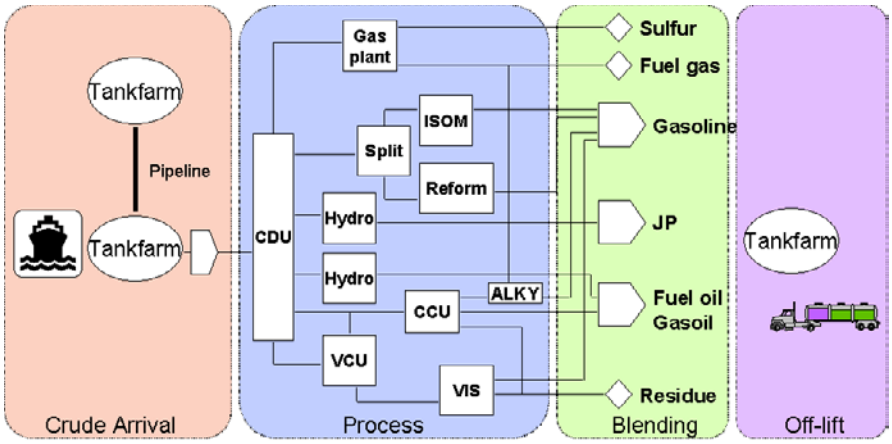

Fig. 6.9. Refinery schematic diagram

finished product and therefore the margin of the refinery against the market price. At the output side of the refinery, the off-lift of the finished products must be scheduled to satisfy customer demand and optimise resource use (pumps, berths, etc).

A refinery will have perhaps 30 different types of processing units, 250 tanks of different sizes in different parts of the refinery, connected by thousands of pipes and pumps which allow many, but not all product movements.

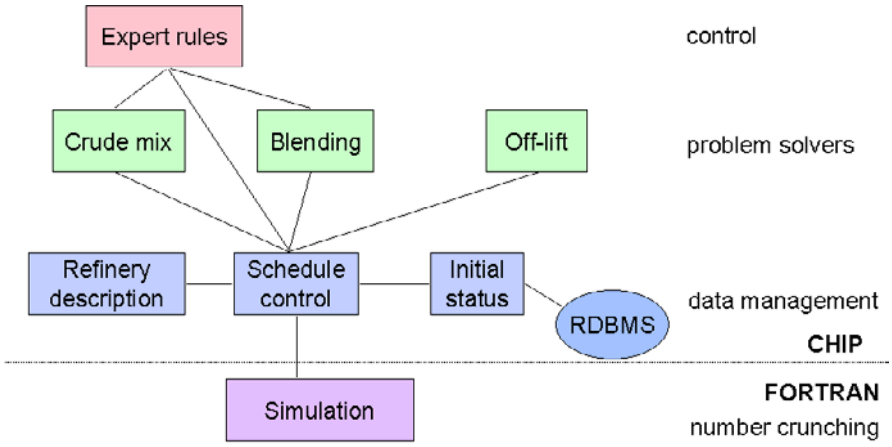

Fig. 6.10. FORWARD system modules

Oil mixes with non-linear blending laws for many properties. The FORWARD system tracks around 70 measured values and uses up to 1000 data points for each oil mix in the system. There is a library of several hundred

crude oils, defining their properties in detail. The FORTRAN part of the FORWARD system uses these data and complex non-linear models of processing units to calculate flow-rates, yields and properties for each unit in the refinery. Different operating modes determine the type of behaviour expected from the unit.

The system modules in the FORWARD system are shown in figure 6.10. The *simulation part* in FORTRAN handles the calculation of unit yields and flow rates. It exchanges information about the refinery structure with the *scheduling control system*. This event processing system written in CHIP is also linked to the other parts of the system. The *refinery description module* is a graphical editor to enter and modify refinery descriptions interactively. All units, tanks and pipes in the refinery are entered using this module. In the *initial status module*, data from the refinery data base are loaded to initialise the simulation. An automatic comparison with the results of previous simulation runs detects inconsistencies and unforeseen events. On top of this data management part, there are several problem solvers, written in CHIP. One module controls the *mix of crude oils* in order to maximise the throughput of the CDU. A large non-linear optimisation module is responsible for *blending optimisation*. It is implemented on top of the linear rational solver of CHIP. For each blend, it calculates the optimal recipe to produce end products with very strict quality limits at minimal cost. As this optimiser is used inside the simulation tool, we often encounter overconstrained problems, which allow no solution. A special explanation tool, written as a meta program on top of the problem solver, applies constraint relaxation in a variable, four-level hierarchy. It either finds the maximal achievable quantity or determines the missing component.

Another problem solver is responsible for the scheduling of *ship arrivals* at the refinery. This has to take the limited berthing capacity at the refinery into account as well as the storage limits for finished products.

On top of these modules, an *expert rule system* allows to recognise and react to standard situations in the refinery. These rules explain for example which tank to use next when a crude oil tank becomes empty or full. The rules are entered graphically using a special rule editor, which guarantees syntactic correctness of the rules.

The FORWARD system clearly shows the flexibility obtained by the CLP approach. The five current users belong to different companies, with different management styles and slightly different objectives in running their refinery. Each site requires custom modifications to take new units or special physical properties into account. At some sites, pipeline operations must be scheduled concurrently with the refinery operations. Different users use different blending laws which directly affects the blending optimiser. This problem solver is customised for each user to reflect their different view point. The interfaces to other tools and to the data base are customised as well. As a large part of

the system is developed in CHIP using CHIP++ objects, it is easy to extend or modify the function of different modules.

6.6.3 TACT

The TACT application was developed for a food-industry company in the UK [SCK00]. The problem can be described in a simplified form as follows (see figure 6.11): In the food industry, birds (chicken and turkeys) are grown on a large number of farms. When the birds reach a certain age, they are transported to the factories, where they are killed and processed for different types of products. For each day, an order book describes the different farms to be visited, the number of birds of different types to be collected and the order in which the birds must be delivered at the factories. The arrival of the lorries at the factories must be scheduled carefully, in order to avoid either running the factory out of stock or to deliver the birds too early, which would be unacceptable for quality control reasons.

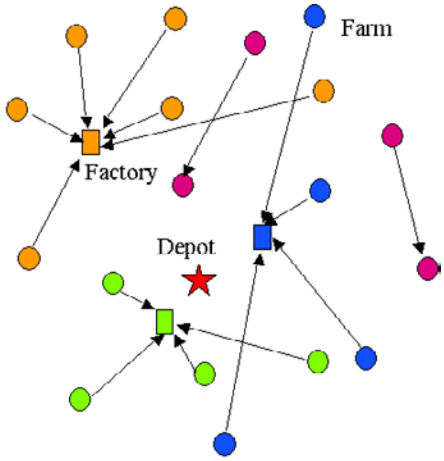

Fig. 6.11. TACT problem

In addition, other transport must be scheduled between breed and grow-out farms to distribute the next generation of birds at the correct time.

The order book is usually known well in advance, but often changes at the last minute in order to react quickly to some new event. Reaction time is crucial, since the affected personnel must be notified in time of any changes in the plan.

The company uses its own fleet of vehicles for the transport, but sometimes must hire in extra drivers or vehicles to handle the work load. The farms are at a distance of between 5 minutes and 3 hours driving time from the

factories, so that geographical locations play an important role in the scheduling. Some farms can only be reached by small lorries, since they are at the end of difficult roads. For a number of farms, environmental constraints limit the time when lorries can drive to the farms.

The actual transportation process requires a number of interdependent resources:

- bird catching teams
- drivers
- lorries
- trailers
- fork lifts
- fork lift trailers

We will now describe some of the constraints attached to the different types of resources.

Catching teams. The catching team is a group of persons which on the farms catches the birds and collects them into crates. The rate of work, their working time and their preferred work (catching turkeys or chicken, male or female) varies for each team. The teams start work at the depot, drive to a farm with a mini-bus, collect a number of birds there and either continue to another farm or return to the depot. For sanitary reasons, it may be necessary to return to the depot to change before going to another farm. The total working time and the number of crates collected by day are limited, and between two working days the teams must have a certain rest time. This means that a team which works late on Monday can not be used very early on Tuesday. If the work load can not be handled by the given number of teams, it is possible to hire in outside help at a significantly higher cost.

Drivers. The drivers will always clock on at the depot, collect a lorry and make some trips to farms and from there to the processing sites. At the end of the day, the driver must drop the lorry at the depot and clock off there. In a normal day, they will make between two and five trips (depending on distances, etc). After delivery at a factory, the lorry is cleaned and can be used for another trip. The driver may change lorries several times during a day, he is not assigned full-time to one particular vehicle. They also do not have fixed clock-on and clock-off times, but follow a roster, which for each week gives the sequence in which drivers must clock-on. The usual working rules apply, i.e. maximal number of hours working, maximal number of hours consecutive driving, maximal mileage per day are all limited. While a lorry is loaded on a farm, the driver can take the legally required rest period. At the end of a trip, the driver may be required to change the lorry configuration, i.e. drop a trailer or change the large crates for turkey with smaller crates for chicken. This may require an empty trip to another factory in order to pick up these new modules.

Lorries. The factory fleet consists of different types of vehicles, some single lorries of 24 ton weight, lorries with trailers with a total weight of 32, 35, 38 or 42 ton total weight plus some articulated trailers which can be used with outside haulage. Not all lorries are available at all times, there are times reserved for maintenance or inspections. The birds are transported on the lorries in crates which are placed in modules. These modules are loaded and unloaded with fork lifts.

The number of lorries available is given. If for some reason more lorries are required, they can be rented, but at a significant extra cost.

Previous solution. Before the introduction of the system, the scheduling of the transport problem was done by hand. The scheduler started in the morning using the current order book for the next day. Usually, it took about 6-8 hours to find a solution. If the order book was changed in a significant way, much of the work had to be redone, delaying the publishing of the schedule. The two experts in the company could produce a very good schedule in most situations, but it was nearly impossible for an outsider to understand the process. Often, some rule would be bent in order to find a solution more quickly or to patch a solution after some change in the input data. As a result, it would be often unclear which constraints were actually handled correctly in a given schedule.

Solution Approach. Given the complexity of the transportation problem, it is understandable that only a decomposition can lead to manageable sub-problems of feasible size. In the manual solution, the scheduler were cutting the problem into three sub parts. In TACT, we follow the same approach (see figure 6.12).

Trip Cut. In a first step, called trip cut, the number of trips and the different lorry types for each entry in the order book is determined. This creates a set of trips, each transporting a given number of birds on a particular type of vehicle. The solver of this step is a small integer programming module written in CHIP. The constraints consist in a number of equations and inequalities and we are interested in a solution which utilises an integral number of lorries of each type while generating the smallest number of trips possible.

Line Balancing. The second step, called line balancing, is to schedule the trips generated in the first steps in such a way that the supply for the factories is guaranteed and that operational constraints of the loading and transport are taken into account. The main constraints in this problem are producer/consumer constraints for the different bird types. The birds should arrive just in time at each factory, too early and the limited storage space in the factory is exceeded, too late and a factory with several hundred employees grinds to a halt. Figure 6.13 below shows a typical constraint curve for one bird type over the period of a day.

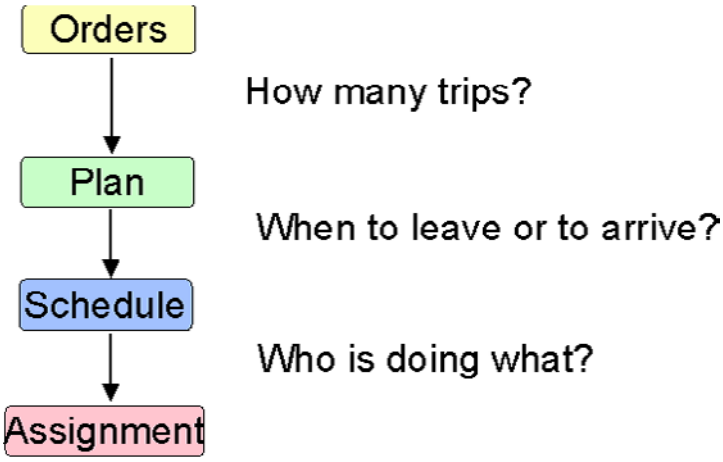

Fig. 6.12. Solution Approach

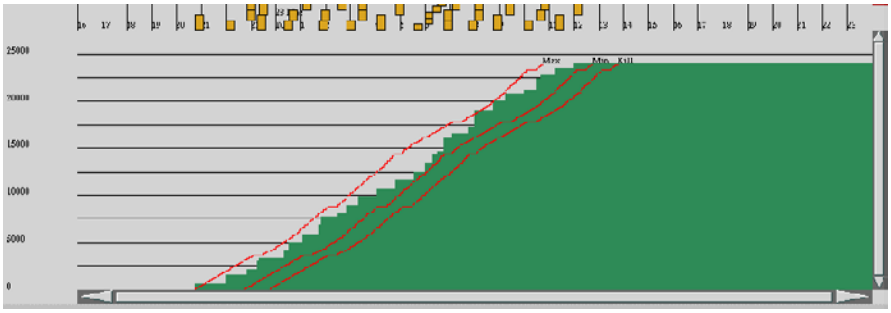

Fig. 6.13. Producer/consumer constraint for factory intake

Assignment. The third step consists in assigning individual resources to all activities. Depending on the type of resource, the timing may have to be re-evaluated at the same time. The constraint model is a typical multi-resource scheduling and assignment problem. The main resource constraint is handled by *diffn* constraints, variable transport (set-up) times are handled with the *cycle* constraint and overall resource limits are expressed with (redundant) *cumulative* constraints.

After each step, the results are presented to the user in the graphical user interface, where he can react to the system. Quite often, the user may decide to re-run the trip cut (first step) with some changes in the parameters before continuing with the second phase.

Each of these phases are run for the orders of a single day, although the generated schedules (30 hours) overlap in time. For each day, the schedule of the previous day is considered as fixed work in progress. Only the fork lift

planning is made for one week at a time, in order to minimise the transport required.

Handling over-constrained problems. An important aspect in the resource assignment is the handling of over-constrained problems. If the resource assignment doesn't find a solution in a reasonable time this can either mean that there is no solution or that the heuristic is not good enough to find it. If there is no solution, this can be because there are not enough drivers, or not enough lorries of some type or not enough catching teams. It is important for the user to find the real cause of the problem, in order to avoid costly mistakes like renting another lorry when more drivers for the existing lorries are required.

In order to provide some form of automatic explanation, the system automatically relaxes different types of constraints if no solution can be found.

Another specialised method is the calculation of lower bounds for some type of resource in independent solvers. This method covers the given amount of work with work profiles of a given duration, ignoring all other constraints. A detailed description together with the constraint model of this lower bound calculation is given in [Sim96a]. Note that the latest version of CHIP includes these lower bound calculation as options for the global constraints cumulative, diffn and cycle.

The user can modify the schedule interactively and then check if the modifications satisfy the constraints. In some situations, the user may wish to violate a constraint in the system in order to solve a particular problem. The automated solver, for example, does not allocate over-time to catching teams. The scheduler, on the other hand, can decide to use over-time for some team and to discuss this with the affected persons, rather than calling in outside contractors for this work. This negotiation process can not be automated, but is crucial to finding a good compromise between different objectives.

Evaluation. The system has been operational since beginning of 1995 and has completely replaced the manual scheduling process. While a manual solution required up to 8 hours of work, it is now possible to find solutions in as little as 15 minutes (about 5 minutes of constraint solving). This gives more flexibility to the scheduler who can devote more time reacting to sudden changes in the order book or to compare different scenarios balancing contradictory objectives.

The TACT application is not used as a black-box problem solver, but rather as a decision support tool for the scheduler. It speeds up the generation of schedules and ensures that constraints and regulations are taken into account. The scheduler can control the different solvers by enabling/disabling constraints or selecting different heuristics.

It is difficult to compare resource utilisation results before and after the introduction of the system as business in general has changed significantly. But it is clear that the tool allows to analyse a problem earlier and in much more detail so that expensive last minute changes are avoided.

A return on investment in 6 month has been quoted by the customer, in particular due to reduced capital expenditure on new lorries.

In five years of operation, a number of changes were introduced into the system to handle new constraints or restrictions. The company has updated its lorry fleet, so that constraints on capacity and farm access had to be changed. The contractual status of the catching teams changed, so that work-rules, time limits and overtime pay had to be revised. The loading constraint balancing full and partially empty vehicles was completely revised. All these changes could be implemented quite rapidly without affecting other parts of the constraint model.

6.7 Application Framework

We now briefly discuss some aspects of the application framework used for many of the applications shown above. A typical decision support application consists of different parts. A central data model is used to drive the graphical user interface, the problem solver and reporting modules. The data are stored in a data base which exchanges data with the internal data model. Libraries are used for many of these functions in order to minimise application specific code.

The application framework for CHIP provides such building blocks for the data management and graphical user interface. Based on a central data model as a data description file, it automatically creates CHIP++ class descriptions, generates data base tables and code to upload and download data from the data base. In the same way, it is used to manipulate graphical interface modules like lists or dialog panels. All these components share the data description which is a single point of reference for the data model of the application.

The framework also contains libraries for often required graphical tools like Gantt charts or diagrams.

The system is general enough for different application domains, it has been used for problems of manufacturing, transport, chemical industry production scheduling, or human resource management. There are obvious benefits in the code size and development speed, but also improved quality and a decrease in maintenance requirements.

6.8 Analysis

In this section we want to evaluate some aspects of the application development with constraints. We first look at an analysis of the different parts

of the ATLAS application and then compare a number of large scale CHIP applications.

6.8.1 Needs of an Industrial System

Table 6.2 shows the percentage of the overall size of different parts of the ATLAS application. The largest amount is taken up by the data base and integration parts, which are developed using SQL*Forms and Reports from Oracle. Of the CHIP part, the application specific graphics and graphical libraries on top of XGIP take a third of the overall program size. The problem solver only accounts for 9%, while the remainder is split between fixed parts for I/O, data management and dialogs.

Table 6.2. Percentage of application size

Part of application	Percentage of overall size
Data base/integration	46%
Graphics	19%
Graphics libraries	14%
Problem solver	9%
Static program parts (I/O, dialog description)	6%
Others	6%

The table only gives the percentages of code size of the finished application. In terms of development effort, a larger percentage must be attributed to the constraint solving part. Prototyping and experimenting with different strategies leads to re-writes or modifications of major parts of the constraint module. In terms of overall effort on the application, the solver accounts for perhaps 20% of the overall development time.

The problem solver is also the part with the high development risk. Very often, it is not clear at the beginning of the project which aspects of the system can be integrated in the solver. Yet without a problem solver, the overall application does not perform its job. This means that prototyping and risk reduction are key aspects to a successful CLP project.

But even taking this into account, the table clearly shows the importance of the integration and data manipulation parts of the finished system. With the application framework, we are working on reducing the amount of special purpose code required for this part of the system in the same way as the constraint system has reduced the amount of work on the problem solver.

6.8.2 Application Comparison

The following table 6.3 compares a number of large scale CHIP applications co-developed by COSYTEC. The entries are given in temporal sequence of

their project start. The table shows the name of the system and the constraint solver used inside the program (F means the finite domain solver, R the linear rational solver). The next two columns state whether the program contains a graphical user interface and data management parts. The number of lines of the complete application is given next. An asterisk means that the complete system contains parts written in other languages, which are not counted here. The last two columns give the number of lines in the constraint solving part of the application and the elapsed project time. This number is not the total size of the project in man months.

Table 6.3. Application comparison

System	Solver	GUI	Data Management	Total Lines	Lines solver	Duration
Locarim	F	yes	yes	70000	3000	24 month
Forward	R,F	yes	yes	60000(*)	5000	24 month
ATLAS	F,R	yes	no	15000(*)	4000	18 month
TAP-AI	F	no	no	7000(*)	6000	6 month
TACT	F	yes	yes	36000	7000	10 month

It is interesting to see that graphical user interfaces and data management modules account for a large percentage of the overall code size. Building a complete end-user system also requires a significant time period for specification and changes of the business process to use the new system to advantage. The TACT application shows how the use of the application framework increasingly reduces both project time and code size.

The constraint solver parts are of a rather similar size. As new global constraints are added to the system, we can handle more and more complex problems which were unmanageable before. We therefore see an increase of complexity in more recent applications which has no significant impact on the project duration.

6.8.3 Why Use Prolog for CLP?

All applications mentioned above use the PROLOG version of the CHIP system. While the declarative structure of logic programming is often advocated as a main advantage of Prolog, we find other aspects of at least equal importance.

For application development, the interpreting environment of Prolog is very important. Changes and revisions can be loaded without recompiling and linking the complete system by reconsulting small parts of the system. This speeds up the development of the problem solver as well as the coding of the graphical user interface.

The built-in relational form of Prolog is also very convenient to store data and parameters inside the application.

The backtracking mechanism of Prolog is most important for developing complex assignment strategies in the problem solver. While predefined search procedures can give satisfactory results on small problems, they are not sufficient to solve large, complex problems. Developing such search strategies in a conventional language like C or C++ dramatically increases complexity.

Another useful aspect of logic programming are meta-programming facilities which can analyse programs and generate new code dynamically. Several of the presented applications use model generator programs or diagnosis engines as explanation facilities. Expert rule systems written in Prolog (with suitable graphical interfaces) allow end-user programming for example in the FORWARD or OPTISERVICE systems.

On the other hand, there are several problems which must be overcome when using logic programming for large scale application programming. The largest problem clearly is the lack of acceptance of PROLOG in the industry, which creates both commercial and technical problems. Writing efficient logic programs is clearly possible, but requires training and experience. This problem is aggravated for constraint programming where there is no strict separation between modelling and implementing tasks.

Other problems are of a more technical nature. Prolog does not immediately support a graphics model based on callbacks. This requires exchange of global data between queries, which is normally not well supported in Prolog. It also lacks proper data abstraction tools. Symbolic terms are rather weak data structures, which cause problems for re-use and specialisation of existing code. These problems are overcome in CHIP with the help of the CHIP++ object layer, which adds feature-terms, objects and class structure to PROLOG.

6.9 Does CLP Deliver?

In section 2.1, we mentioned several key advantages which were claimed for constraint logic programming. *Short development time* was the first of these advantages. This is clearly true for many of the prototypes which have been developed very rapidly in CHIP in order to understand and define the problem to be solved. For large scale applications, time limits are usually given by the project framework and possible changes to the business environment. Basically constraint systems enable us to remove the problem solver from the critical path in the project planning. Compared to a standard approach, constraint applications also use much *less code* than conventional programs. The use of the application framework increases this tendency by also shrinking the amount of custom programming for other parts of a system like graphics or data management. The programs are also quite *easy to modify* and to extend. A good example is the FORWARD system, which has been quite easily adapted to four different users and problems as described above. The *performance* of the problem solver is typically not the limiting factor in the application.

Answers are achieved from within seconds to within a few minutes for the complex constraint problems. This is usually quite acceptable compared to the time required to enter data or to make manual modifications.

6.10 Limitations

The list of applications shown above indicates that constraint programming is already used for a large variety of application domains. But this experience with application development also indicates a number of limitations and shortcomings of the current tools.

6.10.1 Stability

The most common problem stated by users of the constraint systems is the sometimes unpredictable behaviour of constraint models. Even small changes in a program or in the data can lead to a dramatic change in the performance. The process of performance debugging, designing and improving constraint programs for a stable execution over a variety of input data, is currently not well understood.

6.10.2 Learning Curve

Related to this problem is the long learning curve that novices have experienced. While simple applications can be build almost immediately, it can take much longer to become familiar with the full power of the constraint system.

6.10.3 Problem Size/Complexity

Another problem aspect is the conflict between problem size and complexity. For many small, but very hard problems the constraint systems must perform as much propagation as possible in order to avoid expensive, blind search. For larger, but simpler problems this propagation is too expensive. Much time is spent in the constraint reasoning when a simple enumeration would be more efficient. If a large problem contains a number of smaller, but difficult sub-problems, the difficulties in problem solving increase.

6.10.4 Cost Optimization

A particular problem in many constraint models is the cost optimisation. Depending on the cost function, in particular for additive costs with many contributing factors, constraints may find rather weak lower bounds of the cost. In optimisation problems this often leads to difficulties to improve an initial solution, which is found with a heuristic. Systematic exploration of the whole search space is not possible due to the large number of choices to be explored.

6.11 Future Trends

In this section we will present some ideas for the further development of finite domain constraint techniques. These ideas are intended to be non-exclusive, there are obviously many other interesting approaches (for example the combination of modelling with global constraints and stochastic search methods, or other hybrid methods). We group the ideas in four areas, starting with modelling issues, followed by constraint reasoning and a section on search and optimisation. The important aspect of usability and ease of use is discussed in the last section.

6.11.1 Modeling

Current constraint languages like CHIP already use a rather powerful set of constraint primitives and global constraints. Further development will be driven by two main factors:

- Requirements for particular applications often lead to new constraints, which are then extended to satisfy the rules for global constraint expressed above.
- A general analysis of possible constraint types may lead to a novel and more systematic classification of constraint pattern. Some early research in that direction can be found in [SAB00] [Bel00].

A more general question concerns the modelling language used to express constraint problems. At the moment, most constraint systems are either extensions of a programming language (often Prolog) or libraries which are used together with conventional programming languages (C, C++). Currently there is work on introducing constraint modelling languages [VH98] similar to algebraic modelling languages known from Operations Research [FGK93].

Another approach to modelling is the use of visual tools to express and generate constraint programs. A wide range of methods is possible, starting from application specific visual tools to dedicated visual constraint programming languages. In the FORWARD refinery scheduling [GR97] system for example, the model of the refinery is entered in a graphical drawing tool and the constraint model is generated from the layout. In [Lab98], a graphical tool to express search methods has been presented. A first version of a graphical specification tool for CHIP was presented in [Sim97a].

6.11.2 Constraint Reasoning

As mentioned above, for many large scale problems we must find a compromise between the use of expensive, but powerful methods and the need to find a solution rapidly for simple problems. In a typical global constraint, up to 40 propagation methods are working together. The study of the interaction

of these methods and their possible control is one of the most challenging problems in constraint development.

Another current topic is the integration of constraint programming with integer programming techniques [BK98]. This integration can take very different forms. One possibility is the introduction of a global constraint concept in integer programming and its use in a branch-and-cut scheme [Kas98]. A second approach is a redundant modelling with both global constraints and integer programming methods. A difficult question in this approach is the integration of the search techniques of both methods which are quite different.

6.11.3 Search and Optimization

Controlling search is probably the least developed part of the constraint programming paradigm. For many problems the choice of a search routine is crucial to performance, but it is generally developed on a ad-hoc basis. The possibilities of partial search and local search methods as alternatives to depth first chronological backtracking are also not full understood.

The difference to integer programming is quite striking. In integer programming, cost and cost estimation are driving search to the exclusion of most user control. In constraint programming, the cost estimation often is very weak and does not guide the search, so that user defined heuristics are needed to find solutions. A combination of both techniques, improved cost reasoning and user control over the search should provide dramatic improvements in solution quality for many hard problems.

6.11.4 Ease of Use

The most limiting factor for a more wide-spread use of constraint programming at the moment is probably the difficulty of understanding and mastering the technology. Part of this difficulty is the inherent complexity of the problems tackled, but much further development is also needed in methodology [Ger98] and ease of use. Visualisation and debugging tools like the ones developed in DiSCiPl [SA00] [SAB00] [SCD00] can help understand the crucial problem of performance debugging.

6.12 Conclusions

We have shown in this paper a number of large scale applications developed with CHIP, a constraint logic programming system and have seen that complex end-user systems can be built with this technology. On the other hand, constraint logic programming does not offer a "silver bullet" for problem solving. There is a significant learning period before the technology can be mastered. This also requires application domain knowledge to model problems and to express solution strategies. Other aspects, like integration and

graphical user interface, form a significant part of the overall development requirements. Clearly, constraint logic programming is now an industrial reality which can be applied to many domains where high quality decision support systems are required.

Acknowledgments

This overview would not have been possible without the work of the team at COSYTEC together withtheir customers and partners in different projects.

References

[AB93] A. Aggoun, N. Beldiceanu. Extending CHIP in Order to Solve Complex Scheduling Problems. *Journal of Mathematical and Computer Modelling*, Vol. 17, No. 7, pages 57-73 Pergamon Press, 1993.

[BKC94] G. Baues, P. Kay, P. Charlier. Constraint Based Resource Allocation for Airline Crew Management. In *Proceedings ATTIS 94*, Paris, April 1994.

[BBC97] N. Beldiceanu, E. Bourreau, P. Chan, D. Rivreau. Partial Search Strategy in CHIP. In *Proceedings 2nd Int. Conf on Meta-heuristics*, Sophia-Antipolis, France, July 1997.

[BBR96] N. Beldiceanu, E. Bourreau, D. Rivreau, H. Simonis. Solving Resource-Constrained Project Scheduling Problems with CHIP. *Fifth International Workshop on Project Management and Scheduling*, Poznan, Poland, April 1996.

[BC94] N. Beldiceanu, E. Contejean. Introducing Global Constraints in CHIP. *Journal of Mathematical and Computer Modelling*, Vol. 20, No. 12, pp 97-123, 1994.

[Bel00] Nicolas Beldiceanu. Global constraints as graph properties on a structured network of elementary constraints of the same type. In Rina Dechter, editor, *Principles and Practice of Constraint Programming - CP 2000*, volume 1894 of *Lecture Notes in Computer Science*, pages 52–66, Singapore, September 2000. Springer-Verlag.

[BCP92] J. Bellone, A. Chamard, C. Pradelles. PLANE -An Evolutive Planning System for Aircraft Production. In *Proceedings First International Conference on the Practical Application of Prolog*. 1-3 April 1992, London.

[Ber90] F. Berthier. Solving Financial Decision Problems with CHIP. In *Proceedings 2nd Conf Economics and AI*, Paris 223-238, June 1990.

[BLP95] R. Bisdorff, S. Laurent, E. Pichon. Knowledge Engineering with CHIP - Application to a Production Scheduling Problem in the Wire-Drawing Industry. In *Proceedings 3rd Conf Practical Applications of Prolog (PAP95)*, Paris, April 1995.

[BK98] A. Bockmayr, T. Kasper. Branch and Infer - A Unifying Framework for Integer and Finite Domain Constraint Programming. *INFORMS J. Computing* 10(3) 287-300, 1998.

[BDP94] P. Bouzimault, Y. Delon, L. Peridy. Planning Exams Using Constraint Logic Programming. In *Proceedings 2nd Conf Practical Applications of Prolog*, London, April 1994.

[COC97] Mats Carlsson, Greger Ottosson, and Björn Carlson. An open-ended finite domain constraint solver. In H. Glaser, P. Hartel, and H. Kucken, editors, *Programming Languages: Implementations, Logics, and Programming*, volume 1292 of *Lecture Notes in Computer Science*, pages 191–206, Southampton, September 1997. Springer-Verlag.

[CDF94] A. Chamard, F. Deces, A. Fischler. A Workshop Scheduler System written in CHIP. In *Proceedings 2nd Conf Practical Applications of Prolog*, London, April 1994.

[CHW98] P. Chan, K. Heus, G. Weil. Nurse Scheduling with Global Constraints in CHIP: Gymnaste. In *Proceedings Practical Applications of Constraint Technology (PACT 1998)*, London, March 1998.

[CF95] C. Chiopris, M. Fabris. Optimal Management of a Large Computer Network with CHIP. In *Proceedings 2nd Conf Practical Applications of Prolog*, London, April 1994.

[Col96] C. Collignon. Gestion Optimisee de Ressources Humaines pour l'Audiovisuel. In *Proceedings CHIP Users' Club*, Massy, France, November, 1996.

[Col90] A. Colmerauer. An Introduction to Prolog III. *CACM* 33(7), 52-68, July 1990.

[CGR95] T. Creemers, L. R. Giralt, J. Riera, C. Ferrarons, J. Rocca, X. Corbella. Constrained-Based Maintenance Scheduling on an Electric Power-Distribution Network. In *Proceedings Conf Practical Applications of Prolog (PAP95)*, Paris, April 1995.

[DVS87] M. Dincbas, H. Simonis, P. Van Hentenryck. Extending Equation Solving and Constraint Handling in Logic Programming. In *Colloquium on Resolution of Equations in Algebraic Structures (CREAS)*, Texas, May 1987.

[DVS88] M. Dincbas, P. Van Hentenryck, H. Simonis, A. Aggoun, T. Graf and F. Berthier. The Constraint Logic Programming Language CHIP. In *Proceedings of the International Conference on Fifth Generation Computer Systems (FGCS'88)*, pages 693-702, Tokyo, 1988.

[DVS88a] M. Dincbas, P. Van Hentenryck, H. Simonis, A. Aggoun and T. Graf. Applications of CHIP to Industrial and Engineering Problems. In *Proceedings First International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Tullahoma, Tennessee, USA, June 1988.

[DSV90] M. Dincbas, H. Simonis and P. Van Hentenryck. Solving Large Combinatorial Problems in Logic Programming. *Journal of Logic Programming* 8, pages 75-93, 1990.

[DS91] M. Dincbas, H. Simonis. APACHE - A Constraint Based, Automated Stand Allocation System. In *Proceedings of Advanced Software Technology in Air Transport (ASTAIR'91)*. Royal Aeronautical Society, London, UK, 23-24 October 1991, pages 267-282.

[DSV88] M. Dincbas, H. Simonis, P. Van Hentenryck. Solving the Car Sequencing Problem in Constraint Logic Programming. In *Proceedings European Conference on Artificial Intelligence (ECAI-88)*, Munich, W. Germany, August 1988.

[DSV92] M. Dincbas, H. Simonis, P. Van Hentenryck. Solving a Cutting-Stock Problem with the Constraint Logic Programming Language CHIP. *Journal of Mathematical and Computer Modelling*, Vol. 16, No. 1, pp. 95-105, Pergamon Press, 1992.

[DD96] A. Dubos, A. Du Jeu. Application EPPER Planification des Agents Roulants. In *Proceedings CHIP Users' Club*, Massy, France, November 1996.

[FGK93] R. Fourer, D. Gay, B.W. Kernigham. AMPL - A Modelling Language for Mathematical Programming. The Scientific Press, San Francisco, CA, 1993.

[FHK92] T. Fruewirth, A. Herold, V. Kuchenhoff, T. Le Provost, P. Lim, M. Wallace. Constraint Logic Programming - An Informal Introduction. In *Logic Programming in Action*. Springer Verlag LNCS 636, 3-35, 1992.

[Ger98] C. Gervet. LSCO Methodology- The CHIC2 Experience. In *DIMACS workshop of constraint programming and large scale discrete optimization*, Rutgers University, September 1998.

[GR97] F. Glaisner, L.M. Richard. FORWARD-C: A Refinery Scheduling System. In *Proceedings Practical Applications of Constraint Technology (PACT97)*, London, March 1997.

[GVP89] T. Graf, P. Van Hentenryck, C. Pradelles, L. Zimmer. Simulation of Hybrid Circuits in Constraint Logic Programming. In *Proceedings IJCAI89*, Detroit, August 1989.

[HG95] W.D. Harvey, M.L. Ginsberg. Limited Discrepancy Search. In *Proceedings IJCAI95*, 1995.

[JL87] J. Jaffar, J.L. Lassez. Constraint Logic Programming. In *Proceedings 14th POPL*, Munich ,1987.

[JM93] J. Jaffar M. Maher. Constraint Logic Programming: A Survey. *Journal of Logic Programming*, 19/20:503-581, 1994.

[Kas98] T. Kasper. A Unifying Logical Framework for Integer Programming and Finite Domain Constraint Programming. PhD Thesis, Universitat des Saarlandes, 1998.

[KS95] P. Kay, H. Simonis. Building Industrial CHIP Applications from Reusable Software Components. In *Proceedings Practical Applications of Prolog (PAP95)*, Paris, April 1995.

[Lab98] F. Laburthe. Contraintes et Algorithmes en Optimisation Combinatoire. PhD Thesis, University Paris VII, 1998.

[MS98] K. Mariott, P. Stuckey. Programming with Constraints - An Introduction. MIT Press, Cambridge MA, 1998.

[Per91] M. Perrett. Using Constraint Logic Programming Techniques in Container Port Planning. *ICL Technical Journal*, May, 1991, pp 537-545.

[Sch97] C. Schulte. Oz Explorer: A Visual Constraint Programming Tool. In *Proceedings of the Fourteenth International Conference On Logic Programming*, Leuven, Belgium, pages 286-300. The MIT Press, July 1997.

[SND88] H. Simonis, N. Nguyen, M. Dincbas. Verification of Digital Circuits using CHIP. In G. Milne (Ed.), *The Fusion of Hardware Design and Verification*, pages 421-442, North Holland, Amsterdam, 1988.

[SD93] H. Simonis, M. Dincbas. Propositional Calculus Problems in CHIP. In A. Colmerauer and F. Benhamou, Editors, *Constraint Logic Programming - Selected Research*, pages 269-285, MIT Press, 1993.

[Sim95] H. Simonis. Scheduling and Planning with Constraint Logic Programming. *Tutorial Practical Applications of Prolog (PAP95)*, London, UK, April 1995.

[SC95] H. Simonis, T. Cornelissens. Modelling Producer/Consumer Constraints. In *Proceedings Principles and Practice of Constraint Programming (CP95)*, Cassis, France, September 1995.

[Sim95a] H. Simonis. Application Development with the CHIP System. In *Proceedings Contessa Workshop*, Friedrichshafen, Germany, September 1995, Springer LNCS.

[Sim96] H. Simonis. A Problem Classification Scheme for Finite Domain Constraint Solving. In *Workshop on Constraint Applications, CP96*, Boston, August 1996.

[Sim96a] H. Simonis. Calculating Lower Bounds on a Resource Scheduling Problem. In *Workshop on Constraint Programming, ASIAN96*, Singapore, December 1996

[Sim97] H. Simonis. Standard Models for Finite Domain Constraint Solving. *Tutorial at Practical Applications of Constraint Technology (PACT97)*, London, UK, April 1997.

[Sim97a] H. Simonis. Visual CHIP - A Visual Language for Defining Constraint Programs. *CCL II workshop*, September 1997.

[Sim98] H. Simonis. More Standard Constraint Models. *Tutorial at Practical Applications of Constraint Technology (PACT98)*, London, UK, March 1998.

[SC98] H. Simonis, P. Charlier. COBRA - A System for Train Crew Scheduling. In *DIMACS Workshop on Constraint Programming and Large Scale Combinatorial Optimization*, Rutgers University, New Brunswick, NJ, September, 1998.

[SA00] H. Simonis and A. Aggoun. Search-tree visualization. In P. Deransart, J. Maluszynski, and M. Hermenegildo, editors, *Analysis and Visualisation Tools for Constraint Programming*. Springer LNCS 1870, 2000.

[SAB00] H. Simonis, A. Aggoun, N. Beldiceanu, and E. Bourreau. Global constraint visualization. In P. Deransart, J. Maluszynski, and M. Hermenegildo, editors, *Analysis and Visualisation Tools for Constraint Programming*. Springer LNCS 1870, 2000.

[SCK00] H. Simonis, P. Charlier, and P. Kay. Constraint handling in an integrated transportation problem. *IEEE Intelligent Systems and their applications*, 15(1):26–32, 2000.

[SCD00] H. Simonis, T. Cornelissens, V. Dumortier, G. Fabris, F. Nanni, and A. Tirabosco. Using constraint visualization tools. In P. Deransart, J. Maluszynski, and M. Hermenegildo, editors, *Analysis and Visualisation Tools for Constraint Programming*. Springer LNCS 1870, 2000.

[Smo95] Gert Smolka. The Oz programming model. In Jan van Leeuwen, editor, *Computer Science Today*, Lecture Notes in Computer Science, vol. 1000, pages 324–343. Springer-Verlag, Berlin, 1995.

[SGK99] R. Szymanek, F. Gruian, K. Kuchcinski. Application of Constraint Programming to Digital Systems Design. In *Workshop on Constraint Programming for Decision and Control*, Institute for Automation, Silesian University of Technology, Gliwice, Poland, June 1999.

[VH89] P. Van Hentenryck. Constraint Satisfaction in Logic Programming. MIT Press, Boston, MA, 1989.

[VH98] P. Van Hentenryck. The OPL Optimization Programming Language. MIT Press, Cambridge MA, 1998.

[VSD92] P. Van Hentenryck, H. Simonis, M. Dincbas. Constraint Satisfaction using Constraint Logic Programming. *Journal of Artificial Intelligence*, Vol.58, No.1-3, pp.113-161, USA, 1992.

[VC88] P. Van Hentenryck, J-P. Carillon. Generality versus Specificity: an Experience with AI and OR Techniques. In *Proceedings American Association for Artificial Intelligence (AAAI-88)*, St. Paul, MI, August 1988.

[Wal96] M. Wallace. Practical Applications of Constraint Programming. *Constraints Journal*, Vol 1, Nr1, 2, Sept 1996, pp 139-168.

[WNS97] M. Wallace, S. Novello, and J. Schimpf. ECLiPSe - a platform for constraint programming. *ICL Systems Journal*, 12(1):159–200, 1997.