Intelligent Vehicle Scheduling: Experiences with a Constraint-based Approach.[†]

Tim Duncan
Artificial Intelligence Applications Institute,
University of Edinburgh,
Edinburgh, EH1 1HN

AIAI-TR-150
June 1994

Abstract
This paper describes the experiences and insights that AIAI have gained during the development of an intelligent vehicle scheduling system using a commercially available library for constraint-based programming. The application involves the assignment of drivers and vehicles for single-drop deliveries, under a wide range of regulations, physical compatibility restrictions, and operating preferences. The aims of the project were to produce good quality schedules within a limited time while minimizing operating costs. In addition to generating predictive schedules, the system provides some support for user interaction with the schedule generation process, and reactive scheduling.

1 INTRODUCTION

In this paper we describe the SCHEDULE–IT project, developed at AIAI, University of Edinburgh in collaboration with Intelligent Applications Ltd. In Section 2 we outline the architecture, the constraints and preferences involved and their representation, and the different modes of interacting with the system. Following this in Section 3, we discuss a number of the difficult design/implementation issues that were involved in the project.

SCHEDULE–IT is an intelligent vehicle scheduling system which was developed using ILOG SOLVER, a library that supports constraint-based programming. The application involves the assignment of drivers and vehicles for deliveries, under a wide range of regulations, physical compatibility restrictions, and operating preferences. For the most part drivers deliver one load per journey, so the problem is one of resource allocation rather than routing.

Currently the task is carried out by human schedulers, but the time available to produce schedules is limited, and so schedule quality tends to depend on the skill of the individual and the pressure of other work. It is also difficult to maintain schedule quality where late orders or delivery difficulties have to be accommodated at short notice.

[†]Submitted to Expert Systems'94

1.1 Aims of the project

Delivery is an important part of the company's business, accounting for a substantial part of their annual budget, and therefore a good area to look at with a view to improving efficiency. At the same time scheduling had been identified by the company as a key issue in improving customer service. These two forces have led to a number of potentially conflicting aims that we have borne in mind while developing SCHEDULE–IT. These include:

Limited time: Schedules have to be produced within a limited time (less than 30 minutes). As a result the schedulers currently consider only a fraction of the possible allocations. Automating, or partially automating the scheduling process opens up the possibility of considering a larger number of possible schedules.

Consistency: The quality of schedules produced by humans may depend on many factors. Individual skills, the scheduler's level of experience, the priorities which he or she takes into consideration, and pressure of work are only some of these. Automation can be expected to produce more consistency in the schedules produced.

Timeliness: Meeting the delivery times requested by customers is particularly important in this company's business, but very difficult to achieve. An important benefit which the company hopes to obtain through this project is the guarantee that schedules generated will allow orders to be delivered to customers within tolerances agreed with the customer.

Legal regulations: There are a large number of European Community regulations governing drivers' hours. These include weekly and daily driving limits, with the possibility of extension under specified circumstances, plus requirements on daily and weekly rest. The sheer number of regulations which need to be observed place a strain on human schedulers. In order to avoid the possibility of violating regulations by mistake, and at the same time to enable an auditable trace of adherence to the regulations, the system monitors drivers' hours regulations.

Reducing costs: One of the motivating factors in considering automation of the scheduling process is that by examining larger numbers of potential allocations more efficient schedules can be produced and operating costs reduced.

Meeting preferences: While reducing operating costs is a strong motivation, current working practices and agreements with the unions have led to preferences about certain aspects of the schedules that drivers work to. These include respecting drivers' preferences relating to early starts, trying to minimize the number of vehicle changeovers, and the fair distribution of overtime work. The company were also interested in investigating the effect that satisfying these preferences has on the schedules that can be generated.

Interactive: The system is viewed as a tool to assist rather than replace the human scheduler. In addition it was recognised that there are always likely to be situations which arise that have not been catered for in the program, therefore the human scheduler should have some control over the generation process and be able to specify aspects of the schedule generated.

Reactive: In addition to the primary aim of producing *predictive* schedules for the next day's work, a *reactive* capability was also required. That is the ability to deal with events, such as an extra order or vehicle breakdown, which render the predictive schedule unworkable.

1.2 Why a Constraint-based Approach?

The Planning and Scheduling group at AIAI have experience with a wide range of scheduling techniques [Beck 91] including constraint programming tools [Duncan 93]. There were a number of factors which influenced our decision to use ILOG SOLVER [Le Pape 93, Puget 94] for this project. First, the CSP (Constraint Satisfaction Problem) paradigm is well suited to solving scheduling problems [Prosser 92]. Second ILOG SOLVER offers good modelling, which is required in order to be able to represent the wide range of constraints in this application. Flexibility and ease of modification were important to us in order to allow experimentation with alternative representations for efficiency, and also to allow for changes in the problem specification (or clarifications) which inevitably occur during development in scheduling projects.

In contrast modelling in other methods such as Simulated Annealing and Genetic Algorithms tends to be more difficult, and once defined difficult to change. Such methods were also felt to be weaker because of their reliance on a single evaluation function for optimisation. In this project we have investigated an alternative approach to evaluating schedules which does not attempt to reduce preferences, cost, and other aspects of schedule quality to a single measure. Expert systems approaches to scheduling have tended to rely on user-derived heuristics to obtain good schedules. This makes it difficult for them to improve on the performance of human schedulers. Also, such heuristics may work well in some situations and not in others, making the system liable to "brittleness" when changes occur in the pattern of orders being dealt with. Further comparisons between the CSP and other approaches are discussed in [Van Hentenryck & Carillon 88].

2 APPLICATION OVERVIEW

In this section we give a high-level overview of SCHEDULE–IT's architecture, before briefly describing some of the constraints and preferences found in the application. Following this we will briefly discuss the way the problem is represented in ILOG SOLVER, and then look at some of the different modes of interaction with the system.

For simplicity SCHEDULE–IT is divided into two parts: the front-end and the scheduler itself. The front-end is responsible for maintaining data including orders, static data (*e.g.* information about vehicles, products, customers etc), dynamic data (*e.g.* records of drivers' hours, contents of trailers etc), and schedules that have been generated. The system allows alternative schedules to be generated and compared before committing to one. The scheduler has a one day time horizon, and generates a schedule for the data that is passed to it by the front end. Constraints involving longer timescales (*e.g.* weekly driving limits) are handled by the front end.

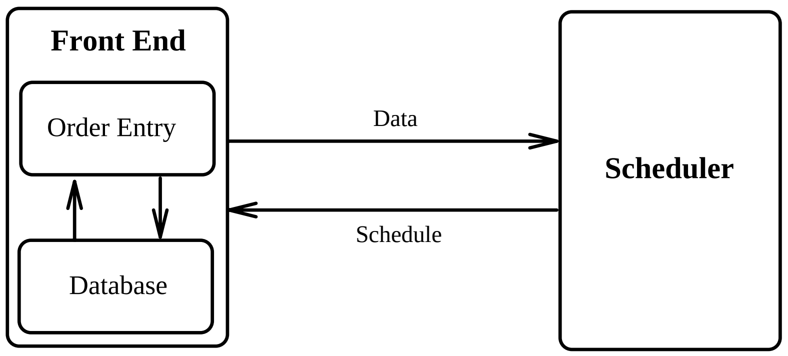

Figure 1: Overview of the SCHEDULE–IT system

Within a one day timeframe the basic task of the scheduler is to allocate a driver, vehicle (*i.e.* tractor and trailer), and a delivery time for each order. Where there are too many orders for the company drivers to be able to deliver, work can be given to a contractor. Thus the problem is never insoluble, but there is a cost involved. Minimizing the cost of work given to the contractor is one of the main aims of the project.

2.1 Constraints

The SCHEDULE–IT application involves a wide range of constraints both physical and legal. Some of the major constraints on allocations are noted here.

Capability: The trailer selected must be capable of carrying the class of product and the quantity involved. In addition there may be other requirements such as orders which require special equipment, or restrictions imposed by the delivery site. Such constraints mean that only some of the vehicles will be suitable for a particular order. As will be seen (in Section 2.3), these constraints generally act to reduce the initial *domains* (the set of possible values for allocations) of orders.

Compatibility: Issues of contamination lead to there being constraints on what trailers can carry. Specifically the product carried must be compatible with the previous contents of the trailer. While capability constraints can be handled initially before search begins, compatibility constraints need to be monitored dynamically.

Drivers' Hours: There are a large number of legal regulations regarding driving hours and rest periods. Fortunately for the scheduler most of these involve a longer perspective, and so the constraints reduce to the earliest time that each driver can start (given the time they finished the previous day and the legally required rest period) and the maximum number of hours driving that can be assigned for the day. The only regulation which does have to be monitored by the scheduler itself is the requirement for 45 minutes break for every $4\frac{1}{2}$ hours driving. This turned out to be much more difficult than we had anticipated, because of the possible interactions between two assignments. For example, a 3 hour job does not require a break, but the issue of rest time would need to be taken into consideration in any following journey. Breaks need to be known in order to determine the time required for a journey, however the scheduler might not make allocations in the order that they will be performed.

2.2 Preferences

In addition to constraints, current working practices provide preferences about certain aspects of the schedules. These include assigning drivers their usual vehicle; minimizing tractor-trailer changeovers; respecting drivers' preferences regarding early starts; and allocating overtime work fairly between drivers. Although stated as preferences, these are almost constraints since the company would only consider breaking them where there was a very strong incentive to do so. At the same time the company is interested in investigating the cost that observing these preferences has on the schedules that can be generated.

2.3 Representation and Constraint Propagation

In the CSP paradigm a set of variables, each of which has a *domain* of possible values, are linked by constraints which express a relationship between two or more variables, determining which values are compatible with which. In SCHEDULE–IT orders, drivers, tractors, and trailers are represented as structured objects within the system. Conceptually an order has slots for the driver, tractor, trailer, and delivery time that have to be allocated for it, and these are represented as *domain variables*. The domains of these variables contain (respectively) the drivers, tractors, or trailers which could be allocated to the order. The search mechanism in ILOG SOLVER provides an efficient method of searching for a set of values which respect the constraints placed on the variables.

Since a driver, tractor, or trailer cannot be used for two tasks at the same time a mechanism is required for representing their availability over time. In SCHEDULE–IT availability is modelled explicitly within the object representing a resource (*i.e.* driver, tractor, or trailer), and updated by a *demon* (implemented with ILOG SOLVER's facility for user-defined constraints). That is, when a resource is allocated to an order the demon will check the resource's availability and then mark it as unavailable to other orders for the duration of the journey (see Figure 2). When a resource is marked as unavailable for a period it can also be removed from the domain of any order which would require it to be used during that period. Demons are also used to check drivers' hours and compatibility between products and the previous contents of a trailer. Since a record is kept of the number of hours a driver has driven, and therefore the number of hours available can also be calculated, the system is also able to remove a driver from the domain of any order which would require a longer journey than the driver is allowed to make.

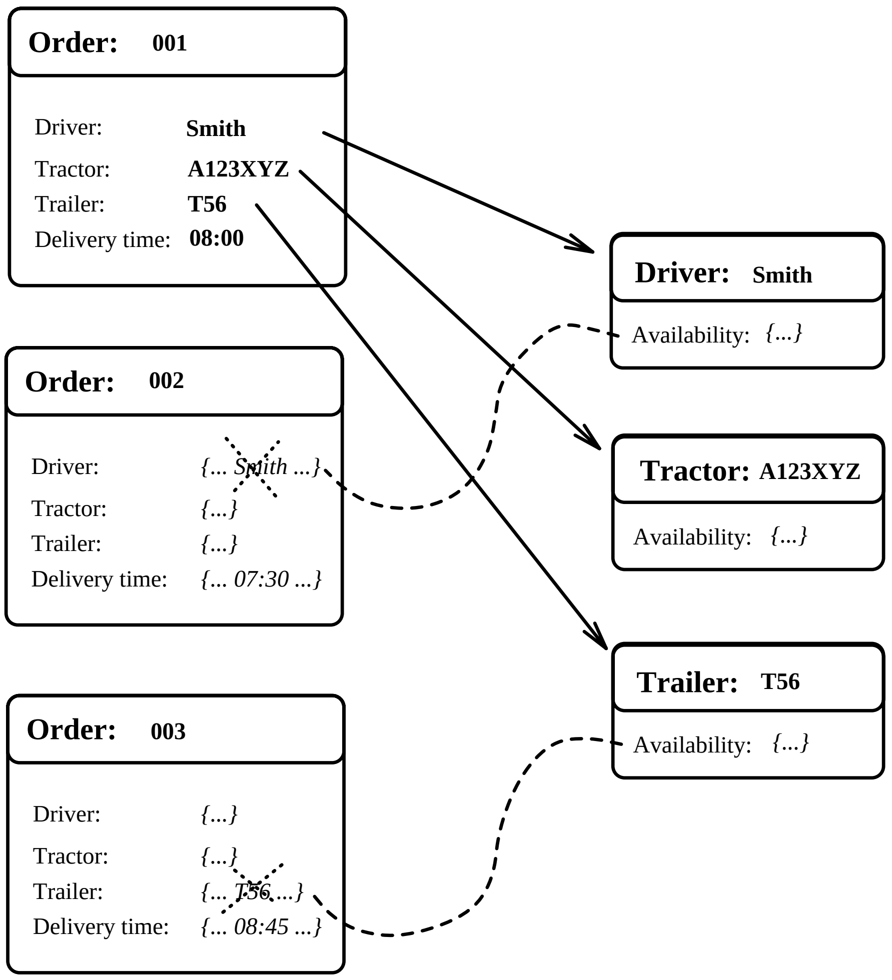

Figure 2: Propagation of availability information

2.4 The Schedule Generation Process

The usual way that the scheduler is used is for the front end to pass a set of orders, together with information about resources, drivers' hours etc to the scheduler, which then generates and returns a schedule.

Human Intervention: One shortcoming of the approach described so far is that the system takes responsibility/control away from the human scheduler. The data is fed in and the system produces the best schedule it can. However the notion of "best" involves a tradeoff between meeting preferences and reducing costs. Ultimately *quality* is something that only the human scheduler can judge, and so in SCHEDULE–IT it was felt that the user should be allowed to exert some influence on the schedule generation process. The mechanism for this is through *partial specification*, that is the user may restrict the resources (*i.e.* driver, tractor, or trailer) considered for a particular order. For example the user may specify that the driver for a particular order should be selected from a specified set of drivers, or that any driver can be selected *apart* from those specified. This is of course easy to implement since it is simply a restriction of the initial domain of the relevant variable. A second mechanism available is to modify the *availability* of a resource. This is used for example to handle planned maintenance of vehicles.

The user may choose to partially specify the resources for any order at the outset, however the more usual case would be where SCHEDULE–IT generates a schedule which the user examines and may choose to modify. Essentially modification amounts to re-specifying the problem and allowing the system to generate a new solution, however if there are parts of the schedule which the user likes, the generated schedule can be used as the basis of the resource restrictions.

Reactive Scheduling: Once an acceptable schedule has been created it is used to determine what to do the next day. For this reason it is said to be a *predictive* schedule. Unfortunately in the real world there are many things that can happen which cause the schedule to become unworkable. These include breakdowns, delays, cancelled orders, and the arrival of new orders. Although a cancelled order does not necessarily make a schedule unworkable, it provides an opportunity for re-optimisation. To deal with such situations a *reactive* capability is required.

In SCHEDULE–IT reactive scheduling is treated as re-scheduling, that is the problem is reformulated to take account of journeys that have already been performed and those that are underway, and the system generates a schedule for the new problem. A corollary of this is that the new schedule may vary greatly from the previous schedule. For this reason re-scheduling may not be a suitable approach in some types of application where decisions may have been made on the basis of the predictive schedule (*e.g.* manpower or materials planning). This is not generally a problem in the present application, however the partial specification of resources can be used to retain parts of the existing schedule where necessary.

3 IMPLEMENTATION ISSUES

In this section we discuss a number of design/implementation issues relating to the aims of the project. These concern the problem of optimising within a limited time; the relationship between preferences and constraints; and interactions between the aims of optimisation and attempting to satisfy preferences. These are interesting issues because standard solutions do not exist. There is also a degree of interaction between the decisions involved for each of the issues.

3.1 Solution within a Limited Time

One of the major constraints on the implementation is the need to produce a schedule within a fixed time. Given the size of the search space and the time available it is not feasible to employ complete search. Reducing operating costs is an important concern, but where exhaustive search is not possible, finding a near optimal solution becomes our goal.

Search Control: The strategy we have employed in SCHEDULE–IT tends to favour low cost solutions first, and continues to search for better solutions until the time limit expires. The primary mechanism for achieving lower cost solutions is a combination of *variable and value ordering*. Search in a CSP involves two decisions: which variable to select, and which value to choose for that variable. SCHEDULE–IT selects orders with the highest potential cost (*i.e.* cost if given to the contractor) first, and the *value ordering* strategy used gives preference to company drivers over the contractor. The result is that expensive orders tend to be given to company drivers. However, the system will continue to search and therefore may discover situations where, for example, it is more advantageous to give one expensive order to the contractor if two cheaper jobs (with a greater total cost) can then be allocated to a company driver.

Cost Function: The optimisation method used by ILOG SOLVER is based on branch-and-bound. Once a solution is found, the system adds a constraint that the solution should have a cost which is less than that for the solution which has already been found. In order to allow ILOG SOLVER to use this information to prune the search space, early propagation of cost is important. For this reason we moved from using the cost of jobs assigned to the contractor as our measure of cost, and used instead the sum of costs for jobs assigned to company drivers and to the contractor. Jobs assigned to company drivers were said to have a *negative cost* based on the charges contractors would have made, while jobs assigned to the contractor had a *positive cost*. Thus each allocation affects the bounds of the total cost, and when the lower bound becomes greater than the current best cost, ILOG SOLVER will backtrack.

3.2 Achieving Preferences

It has already been noted that value ordering is used to ensure that company drivers are allocated in preference to the contractor. Value ordering is also used to implement other preferences. Thus once a variable has been selected, its possible values are given scores for how well they meet the various preferences. The highest scoring value is then chosen first. Weightings are used to give the relative strength (*i.e.* importance) of the different preferences, and how much they contribute to the score. Of course backtracking may lead to other less preferred values being tried as part of the optimizing search, but only where this leads to a lower cost solution.

Some preferences relate more than one object, for example the preference that a driver is given his preferred vehicle. Thus decisions about one resource interact with the decision about another resource. If the driver, tractor, and trailer slots of an order are all represented as separate domain variables then decisions about their allocation are also separated, and poorer quality schedules may result. For example, when the system selects the driver for an order and then looks for a suitable vehicle, if the driver's preferred vehicle is not available, another will be chosen through backtracking. This will lead to a non-preferred driver-vehicle combination, whereas what should have happened is that an alternative *preferred driver-vehicle combination* should have been chosen. In order to avoid such problems, instead of having separate driver, tractor, and trailer variables, Schedule–IT deals with them as a unit, which we refer to as an *assignment*. Thus an order has two domain variables, one for the delivery time, and the other for the assignment. The domain of the order's assignment consists of combinations of driver, tractor, and trailer. Not only does the system ensure that only valid combinations are considered, but they can be ranked more easily according to their scores for meeting preferences.

3.3 Schedule Quality: The Tradeoff between Preferences and Cost

As noted earlier, preferences may end up being ignored where a cheaper solution results. In practice many of the preferences are so strong that they can be considered as constraints. For example the "cost" (in terms of good will) of ignoring a driver's preferences regarding early starts may be larger than the sum saved by the resulting schedule. Therefore in the Schedule–IT system a number of preferences are capable of being redefined as constraints, and this facility is placed under the user's control. Thus the company are able to compare the schedules generated with and without strict observance of the preferences.

In general promoting a preference to a constraint leads to a smaller search space and therefore better performance. The schedule produced is "better" in the sense that important preferences will have been observed, however a lower cost solution may have been missed. Recently we have been investigating an approach which gets away from the idea of the system producing a single "best" solution. First Schedule–IT is run for a preset time with the configurable preferences treated as constraints. This produces a schedule of a certain cost, which respects those preferences. Then the system is run for the remainder of the time available with the preferences relaxed again, but the additional constraint that the solution cost must be less than that already found. This additional constraint helps prune the search space, and any schedule that is found will be of lower cost. Thus the user is presented with two schedules: one which preserves the preferences, and another of lower cost which breaks some of them. The system also produces a schedule quality report which details, amongst other things, the preferences which have been broken. It is then up to the human scheduler to evaluate whether the savings in costs are worth the particular preferences which were broken.

4 CONCLUSIONS

In this paper we have described a reasonably complex application and some of the issues involved in the design and implementation of the system. The csp paradigm and the tool used for the project have proved to be flexible and easy to use. In particular this has been useful in allowing the project to investigate approaches beyond the standard examples of predictive schedule generation. The project took a pragmatic approach towards optimisation, since in this application as in many others in the real world, a near optimal solution was all that was required. The relationship between "preferences" and optimisation, however, is more complex, and in the end should perhaps be left to human judgement.

Acknowledgements

The author would like to thank the company who funded this project; the people who currently do the scheduling; also Howard Beck and Arthur Seaton for comments and discussion.

References

[Beck 91] Howard Beck. An Overview of AI Scheduling in the UK. In I.M. Graham and R.W. Milne, editors, *Research and Development in Expert Systems VIII*, British Computer Society Conference Series, pages 228–243. Cambridge University Press, 1991.

[Duncan 93] Tim Duncan. A Review of Commercially Available Constraint Programming Tools. Technical Report AIAI-TR-149, Artificial Intelligence Applications Institute, University of Edinburgh, 1993.

[Le Pape 93] Claude Le Pape. Using Object-Oriented Constraint Programming Tools to Implement Flexible "Easy-to-use" Scheduling Systems. In *Proceedings of the NSF Workshop on Intelligent, Dynamic Scheduling for Manufacturing*, Cocoa Beach, Florida, 1993.

[Prosser 92] Patrick Prosser. Scheduling as a Constraint Satisfaction Problem: Theory and Practice. In J. Dorn, editor, *ECAI-92 Workshop on Scheduling of Production Processes*, Vienna, 1992. Wiley.

[Puget 94] Jean-François Puget. A C++ Implementation of CLP. In *Ilog Solver Collected papers*. Ilog SA, 12 avenue Raspail, BP-7, 94251 Gentilly Cedex, France, 1994.

[Van Hentenryck & Carillon 88] Pascal Van Hentenryck and J-P. Carillon. Generality versus Specificity: An Experience with AI and OR Techniques. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-88)*, St. Paul, Minnesota, August 1988.