CREWS_NS: Scheduling Train Crew in The Netherlands[1]

Ernesto M. Morgado **João P. Martins**

SISCOG - Sistemas Cognitivos Lda.
Campo Grande 30 - 6°B
1700 Lisboa
PORTUGAL

ernesto@gia.ist.utl.pt JMartins@interg.pt

Abstract

We present a system, CREWS_NS, that is used in the long-term scheduling of drivers and guards of the Dutch Railways. CREWS_NS schedules the work of about 5,000 people. CREWS_NS is built on top of CREWS, a scheduling tool for speeding the development of scheduling applications. CREWS heavily relies on the use of AI techniques and has been built in the perspective of a "white box" system, in the sense that the planner can perceive what is going on, can interact with the system by proposing alternatives or querying decisions, and can adapt the behaviour of the system to changing circumstances. Scheduling can be done in automatic, semi-automatic or manual mode.

CREWS has mechanisms for dealing with the constant changes that occur in input data, can identify the consequences of the change and guides the planner in accommodating the changes in the already built schedules (re-scheduling).

Problem Description

CREWS_NS is a system that addresses the long-term scheduling of train crew at NS, the Dutch Railways (NV Nederlandse Spoorwegen). Long-term scheduling of crew is typically done 6 to 12 months prior to the execution of the schedule and consists in arranging the tasks that have to be done by crew members into *duties* (sequences of tasks to be done by one crew member in one day -- Figure 1). A set of duties for the crew members of a certain personnel base with certain qualifications is called a *schedule*. Crew scheduling is known for its algorithmic complexity. The problem is even more complicated when the quality of a solution depends on subjective constraints that are hard to describe in quantitative terms. It is usually carried out manually by a small number of planners that acquire most of their knowledge through experience. In Dutch Railways long-term scheduling of crew involved 24 planners that, working full time, would take about 6 months to produce the duties for about 5,000 train drivers and guards.

Besides the human skill of efficiently arranging tasks into duties, crew scheduling has to deal with the ever changing information in data. Despite being done several months before the execution of the schedules, time constraints require this task to be started well before its inputs - the final timetable and rolling stock scheduling - have been completed. Moreover, after the schedules have been completed and execution has started, the inputs of the problem keep changing, requiring the outputs to be constantly updated. For this reason, the schedules produced have to be revised several times due to timetable changes as well as the physical resources associated with tasks. A human planner not only has to deal, within a short period of time, with huge amounts of data to produce schedules (*scheduling*), but has also to handle changes that pop up constantly (*re-scheduling*), most of them produced by different departments and thus often incomplete and inconsistent as a whole, which must be incorporated in the already produced schedules without producing much disturbance.

Another aspect that puts a high demand on planners is the increasing complexity of labour rules, required to comply with increasing social benefits given to workers. Because of this, scheduling crew is considered by railway companies to be much more difficult than scheduling rolling stock (equipment) or producing the train timetable (scheduling rail track resource). In short, crew scheduling requires human skills, knowledge, and hard work.

Being a typical resource allocation problem, we might be inclined to consider it as a job shop problem (Fox 1987) (Smith 1889), where the trains are activities and the resources are personnel individuals. Indeed, there are some similarities with this problem, but there are many differences, that prevent us from transposing techniques from one domain to the other. (1) Besides dealing with time constraints and other constraints of the job shop domain (e.g., equipment constraints), crew planners must also deal with space constraints to prevent space discontinuities in duties, having to position crew where they are needed, as passengers in trains or other transportation means. (2) They must also deal with complex train frequencies, such as week frequencies (e.g., a

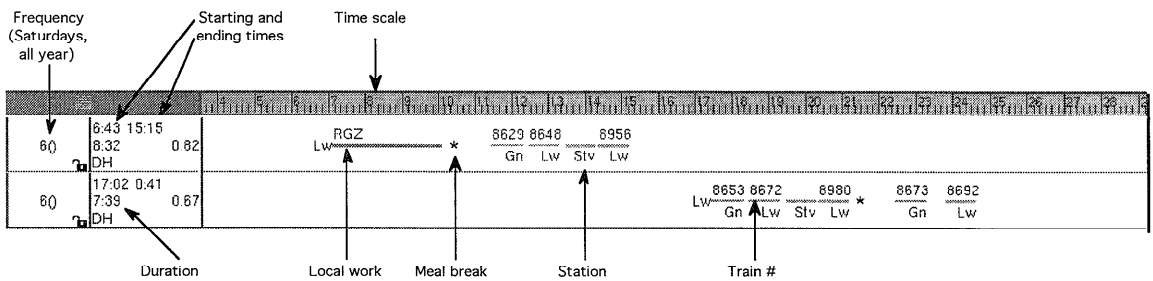

Figure 1: Example of duties.

train may run only on weekends), year periods (e.g., only during summer), and special days (e.g., on holidays and days before holidays), which puts additional constraints on the combinations of tasks. These two aspects are critical to the quality of the final schedules and to the efficiency of the scheduling process. (3) The work periods of crew do not have fixed times, as shifts in industry, but can slide during the day to accommodate the irregularity of the train operation, although subject to constraints. The resource sliding dynamics makes very difficult to analyse activity demand and resource contention as is usually done in the job shop domain (Sadeh and Fox 1991). (4) The labour rules are very complex and change every year due to unions pressure. Worse, to avoid personnel strikes planners must account for exceptions to the rules. This requires a high degree of flexibility in accommodating exceptions and in changing rules, without compromising the efficiency of the process. It requires also a representation model to be easily understood by the planners, different from a constraint representation model based only upon variables and values (Sadeh and Fox 1991).

Crew scheduling has also been approached by traditional programming (supported by operational research, aiming at an optimised solution), but the results obtained with full automatic "black box" optimisation algorithms had only limited success and have proven to be unsatisfactory in the following aspects: (1) when faced with a full size problem, these solutions tend to need computational resources that by far exceed what is available and they cannot cope with the combinatorial explosion; (2) they cannot provide explanations about the decisions placed in the solution; and (3) solutions cannot be manipulated by human planners to adapt them to changing circumstances or to ill-represented constraints.

In summary, crew scheduling presents several difficulties, some of them are external to the scheduling problem itself, while others are internal to the problem.

The *external difficulties* are related with the complexity and distribution of data (data is produced by different departments, at different times, and is often incomplete and inconsistent), with the fact that data is in constant change (data is not produced once for all; while the problem is being solved data keeps on changing and it is important to find out the effect that the changed data has on the work that was already planned), and with the large amounts of data involved in the problem (the problem is too big to be solved as a whole and by one single planner; it must be partitioned into manageable partitions, each of them being handled by one planner).

The *internal difficulties* are related to the combinatorial explosion of the problem and with the multiple and complex constrains that have to be satisfied by the duties. These constraints include *physical constraints* (e.g., temporal continuity in tasks, spatial continuity in tasks, and compatibility of frequencies and year periods), *labour and social constraints* (e.g., starting and ending place of a duty, starting and ending times of a duty, compatibility of rolling stock, transfer times between different rolling stock, maximum duty length, meal breaks -- associated with local resources, such as restaurants), and there are *economical constraints* (that may want to minimise the cost of the operation or the number of resources required). The constraints may either be *hard* (cannot be violated) or *soft* (may be violated, but violations should be avoided and recorded).

Application Description

Since human planners can build acceptable schedules where algorithmic solutions clearly fail, we took the challenge of using AI techniques as an alternative to traditional computer technology. One of our initial goals was to produce a "white box" system, in the sense that the planner could perceive what was going on, could interact with the system, proposing alternatives or querying decisions, and could adapt the behaviour of the system to changing circumstances. The result is a system that plays the role of a "digital colleague" interacting with planners to build schedules in a co-operative way.

Furthermore, since crew scheduling has idiosyncrasies for each company, we took the additional challenge of building a tool, CREWS, that contains the basic knowledge for crew scheduling, remains constant across companies, and only needs to be extended with the particularities of each one (domain, labour rules, scheduling strategies, and objectives).

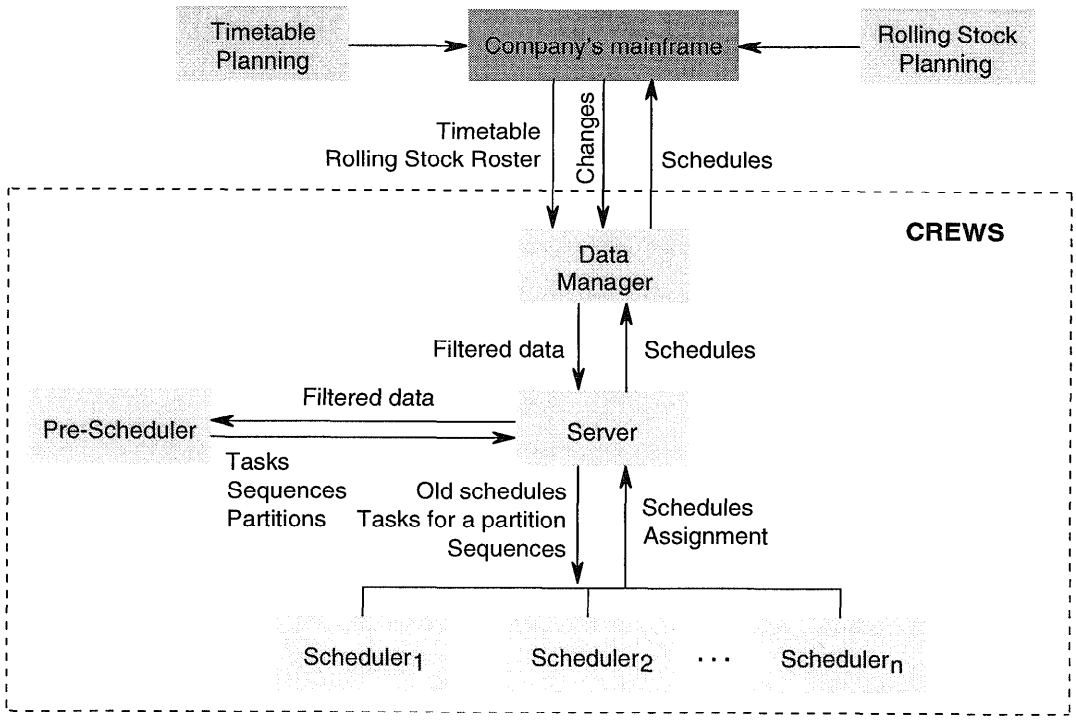

Figure 2: Architecture of CREWS.

The system reported here is built on top of CREWS and is called CREWS_NS. The goals set up for CREWS_NS were the following: to provide decision support in personnel scheduling; to speed up the scheduling process; to make scheduling more reliable; to take a considerable workload from the planners; and to keep the role of planners in taking the decisions.

The scheduling process is divided into three phases: (1) *Data management*, where an analysis of data is done, in order to find inconsistencies or incompleteness, both among input data and between input and output data (to identify which schedules have to be changed, due to the change of input data); (2) *Pre-scheduling*, where personnel tasks are *generated* according to rules that specify the number of personnel resources required for each type of activity, are *sequenced* according to criteria that will guide the scheduling phase, and are *distributed* among partitions that are scheduled quasi-independently; (3) *Scheduling*, where abstraction is used to reduce detail in data, heuristic search considers relevant alternatives to produce good solutions, and constraint satisfaction is used to reduce the size of the state space and to guide the search process.

The architecture of CREWS is based on components that address these phases, the *Data Manager*, the *Pre-Scheduler*, and the *Scheduler*, respectively (Figure 2). There are other components, such as the *Positioning Trip Generator* to position crew where they are needed, and the *CREWS Server* to support the client-server multi-user environment. The Data Manager is connected with outside systems that supply the timetable, the rolling stock roster and the corresponding updates. After generation of the schedules, these are sent to the mainframe of the company that distributes them to the relevant departments.

The Data Manager

The Data Manager supports the preparation of the input data, handles change in data, enables simulation of hypothetical data situations, and maintains the consistency and completeness of data (both before and during the scheduling process).

The Data Manager organises all data pertaining to the problem in a data set. A *data set* is a set of knowledge bases, each one corresponding to a certain concept relevant for the problem. The entities in the knowledge bases are linked together by *data dependencies*, that tell how each concept depends upon others. Data dependencies are used in the propagation of the effects of change in input data. These entities are color-coded to tell whether they are correct, inconsistent, or incomplete (Figure 3). The Data Manager guarantees that the other components receive their data in good condition and signals what must be changed in order to have the problem solved after the initial conditions

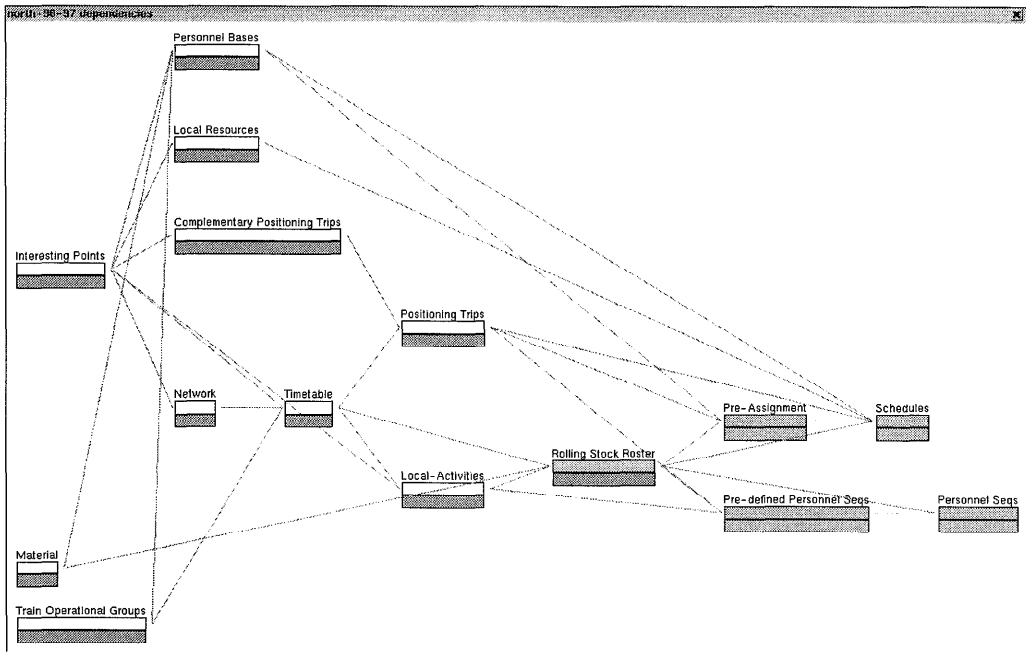

Figure 3: Data Set Dependencies.

have changed. The Data Manager has an explanation facility that tells which data dependencies were violated.

The Data Manager handles change by keeping a record of a sequence of situations. A *situation* corresponds to a state in time of a data set. The Data Manager enables simulations, by keeping alternative sequences of situations, organised in a tree of situations. Each branch of the tree represents a sequence of situations: one corresponds to the real problem, while the others correspond to different simulations.

The Pre-Scheduler

The Pre-Scheduler makes the preparations for scheduling, looking at the problem from a global perspective. The Pre-Scheduler is composed of the Task Generator, the Task Sequencer, and the Task Distributor.

In the description of the Pre-scheduler and of the Scheduler it is important to keep a distinction between activity and task. An *activity* is a specific action that provides a service, for example, a train, that allows passengers to travel from one place to another, or a shunting activity, that consists of moving cars and engines within a station to form a train. A characteristic of an activity is that it requires *resources*. An activity may require *rolling stock resources* (e.g., a train may require one engine and eight cars) and *personnel resources* (e.g., one driver and two guards), to provide the service associated with the activity or to operate the rolling stock resources. The rolling stock and personnel resources determine, respectively, *what* and *who* operates the activities. A *task* (in the sense presented in this paper) is the association of an activity with personnel resources. For example, a train activity may generate one task for a driver and one or more tasks for guards.

Task Generator. Computes the tasks that must be scheduled for a certain class of personnel. The generation of personnel tasks is important for computing the number of drivers and guards to allocate to each activity. Normally, in a train, only one driver is needed, but several guards may be needed, depending on the number and types of coaches. The number of rolling stock units are derived from the rolling stock rosters and other sources.

Task Sequencer. Uses abstraction to reduce the amount of detail the Scheduler has to work with. It groups the personnel tasks into *preferential sequences* to be performed by the crew, that suggest one out of several trains as the best one to follow. In many cases, these suggest that the personnel follows the rolling stock, avoiding problems resulting from train delays; however, there are cases where they may suggest something else. For example, if the operation is very regular and on time, it is not important to follow the rolling stock; it may be more convenient to follow a train series (a certain number of trains following an operation pattern). Sometimes it may be useful that the preferential sequences suggest the patterns of the duties produced the year before, so that changes are minimised. The preferential sequences are just suggestions, and may be changed later on in the Scheduler. The Task Sequencer produces these sequences, according to one of the criteria above. This phase heavily relies on heuristic knowledge about where and when it is reasonable to change the crew.

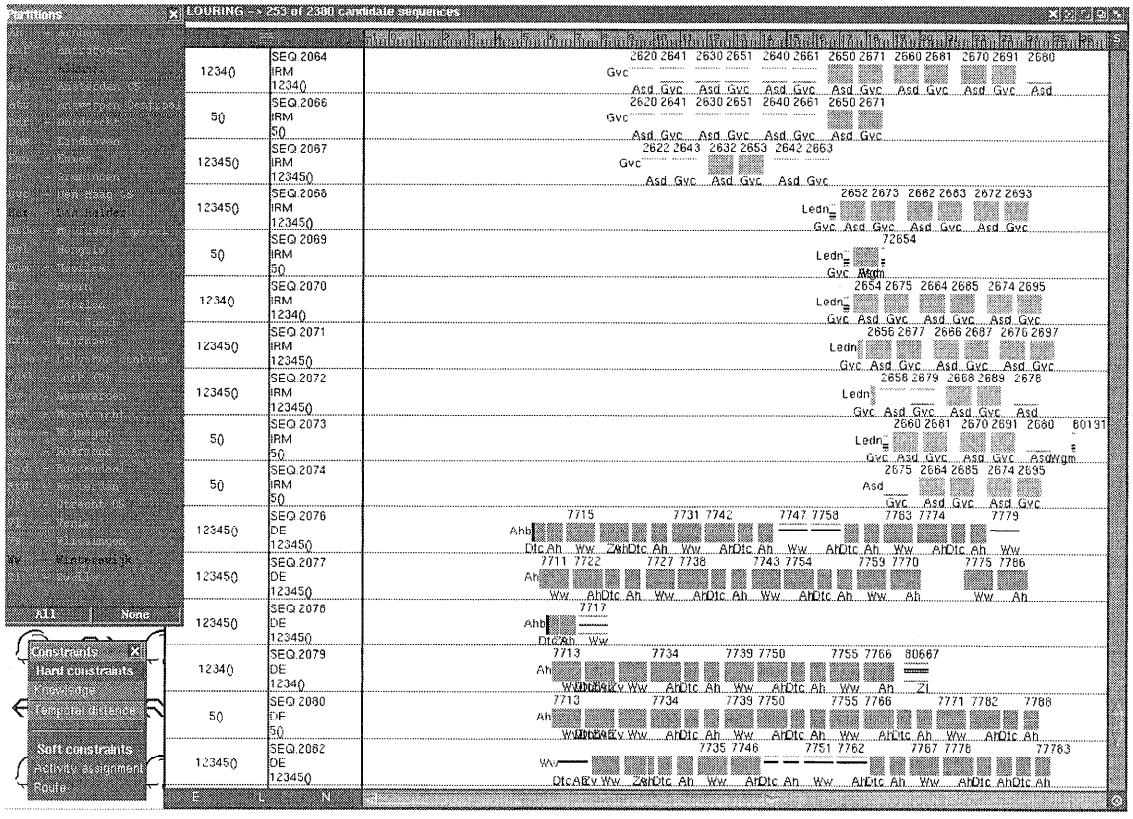

Figure 4: Pre-assigned and assigned tasks.

Task Distributor. Divides the problem into sub-problems (partitions), so that they can be managed effectively by the Scheduler. A problem partition may correspond, for example, to a base of a certain class of personnel. The Task Distributor determines to which partitions tasks can be *pre-assigned*, taking in account, among other aspects, the rolling stock and network knowledge of the personnel associated with each partition. A task can be pre-assigned to more than one partition, allowing sub-problems to overlap. In this way, several planners can try that task in their schedules. However, only one of them will be able to schedule it. When that happens, we say that the task has been *assigned* (as opposed to pre-assigned) to the partition.

In Figure 4, we show a screen used in the Task Distributor that illustrates some of these aspects. The available partitions are listed and each partition can be assigned a color; the constraints used in the task distribution are also shown; the sequences of tasks were computed by the Task Sequencer; tasks shown with a thin line are pre-assigned to the partitions with the same colors as the lines that correspond to the tasks; tasks with more than one thin line are assigned to more than one partition; and tasks shown with a thick line have been assigned to the partition with the color of the line corresponding to the task.

When scheduling a partition, the Scheduler only loads the tasks that have been pre-assigned to that partition and that have not yet been assigned (scheduled) in another partition. When saving a schedule, its corresponding tasks are assigned to the partition, if they have not yet been assigned to another partition in the meanwhile. Of course, when re-scheduling, the Scheduler also loads the tasks that have been already assigned (scheduled) in that partition.

Tasks can be pre-assigned to partitions either manually, by a planner, or automatically, following a strategy. These two methods are integrated, allowing the planner and the system to cooperate in partitioning the tasks. The strategies use heuristics and constraints that can be changed by the user. Tasks can also be swapped between partitions.

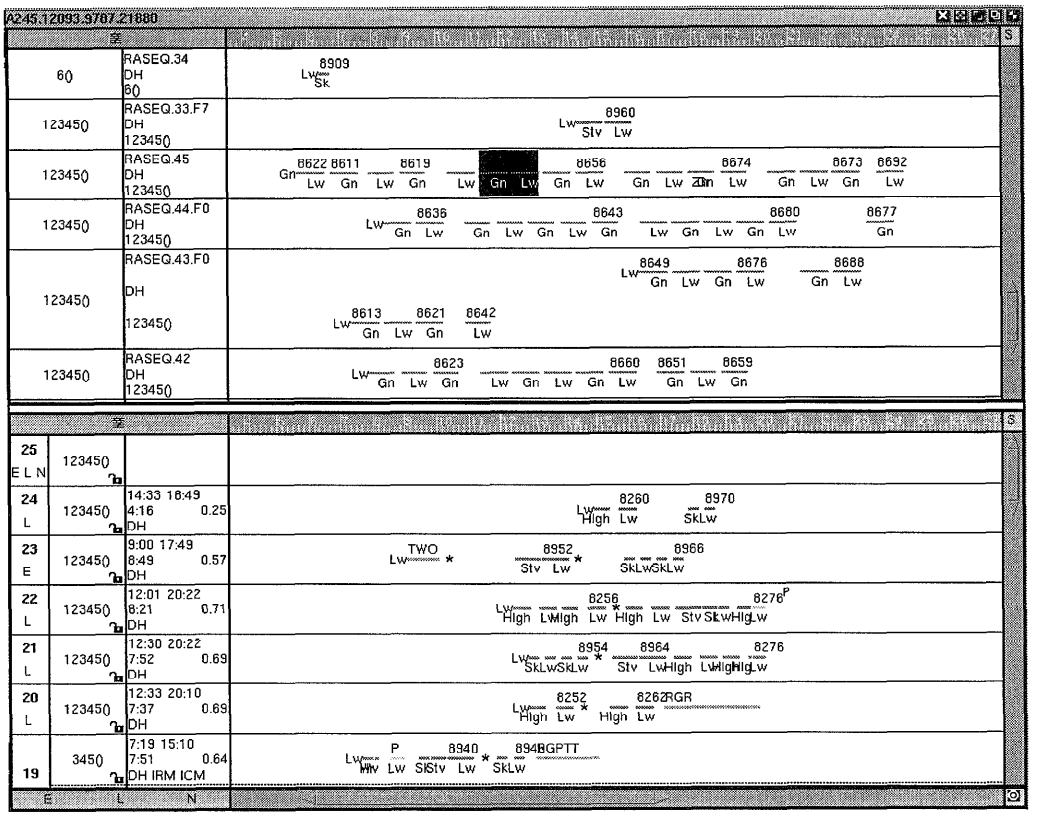

Figure 5: Moving tasks from the candidates to the duties (manual mode).

The Scheduler

The Scheduler works from the perspective of a partition, creating a schedule by grouping the tasks of the partition into duties according to rules and suggestions regarding the partition.

The creation of a schedule starts with a set of tasks produced by the Task Generator, pre-assigned to a partition by the Task Distributor, and grouped in sequences by the Task Sequencer. The Scheduler places those sequences, or parts of them, into duties. A *duty* is a sequence of tasks to be done by a crew member at a certain frequency.

Duties have to satisfy several constraints: the personnel cannot have duties that exceed a certain number of hours, cannot work continuously more that a certain number of hours without a meal break, must have line, rolling stock, and train knowledge in order to operate the trains, and so on.

The Scheduler resorts to *state-space search*. It uses a modified version of beam search (Bisiani 1987), with heuristics. A *state* is a pair containing the sequences of tasks that have to be scheduled (the *candidates*) and the duties that have been constructed so far (the *partial schedule*). States are generated by taking one (or part of one) candidate and placing it in a duty. A final state is a state in which there are no more candidates or when a predetermined number of duties has been reached. The search is guided by a *strategy* consisting of: (1) A process for selecting the initial state (e.g., what kinds of tasks should be considered, sequences or groups of sequences) and how it should be constructed (just with preferential sequences, or also with abstraction); (2) A set of operators to generate the successors of a state that resort to heuristic knowledge to limit the number of successors; (3) An evaluation function, composed of cost and heuristic functions; (4) A test for deciding whether a final state was reached. The Scheduler provides strategies, each one appropriate for a certain type of schedule or for a certain type of operation, scheduling or re-scheduling. The user can add new strategies to handle new types of schedules.

The generation of successors can be done in four different ways (Morgado and Martins 1989, 1992):

1. *Manual mode.* The user tells the system the (sub-) sequence of tasks that should be moved from the candidates to the duties (Figure 5). The system verifies all constraints imposed upon the resulting duty and tells the user the constrains that are violated by the operation (we consider both soft constraints and hard constraints). If the planner chooses to violate a soft

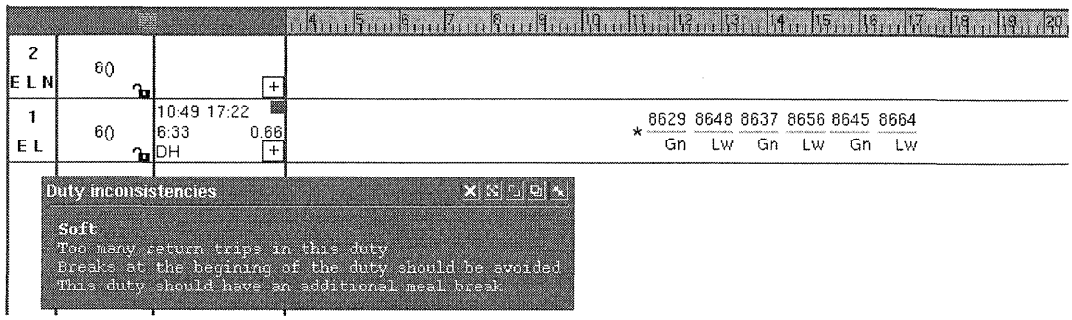

Figure 6: Showing violations.

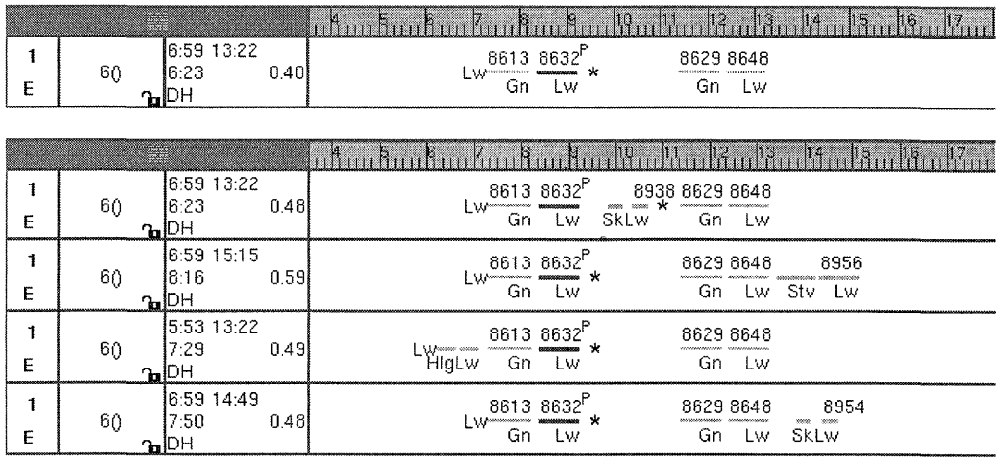

Figure 7: Duty #1 and alternatives for extending duty #1.

constraint, this violation is recorded and the duty is shown with a violation indication. Clicking in the violation indication icon generates a message that explains the violation (Figure 6).

In manual mode, the planner can also move tasks from the existing duties to the candidates, or from duties to other duties, removing the effect of any previously taken decision -- an operation called *forward backtracking* -- or can do traditional backtracking by moving into a previous state;

2. *Semi-automatic mode.* The system gives hints about how the planner should pursue the schedule. In this mode of operation, the system computes how the duties can be extended with sequences of tasks from the candidates and the role of the user is to select the proposal that he thinks is best. Figure 7 shows some of the alternatives for extending duty #1 (the number of alternatives presented is selected by the user);
3. *Automatic mode.* The system decides the sequence of tasks that should be present in each duty, following a strategy that is selected by the user;
4. *Mixed mode.* Combines the previous approaches. In mixed mode the planner constructs the schedule by resorting to an arbitrary combination of the other three modes of operation. Typically, the planner starts building some duties, according to criteria that he wants to impose to the final schedule, then uses the automatic mode to do the bulk of the work, and afterwards, manipulates (using manual mode) the resulting duties.

   The mixed mode of operation really shows the decision-support philosophy that was incorporated in CREWS since its inception. In fact, it provides a full cooperation between the planner and the system, showing what is going on, providing explanations about the decisions taken by the system (with an explanation facility provided by the Scheduler), enabling the interaction of the planner on the work being done by the system, and taking the bulk of work from the planner, when he selects to do so.

In summary, the main innovations offered by this system are: (1) to provide a decision support system for crew scheduling (in face of the traditional "black box"

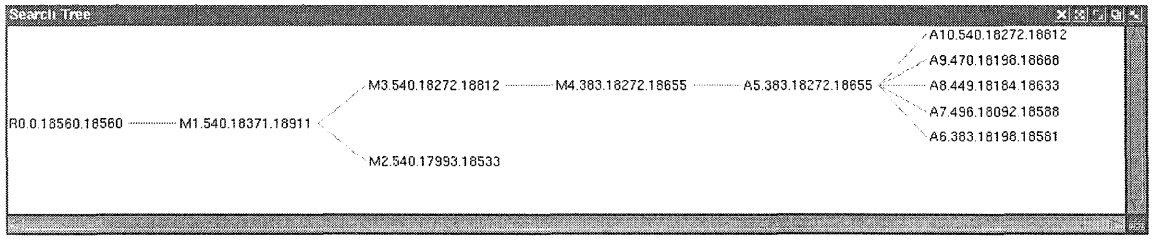

Figure 8: Simple search tree.

scheduling systems); (2) to trade the "optimal solution", for a solution that is "good enough"; (3) to give the possibility to the customer to change the behaviour of the system in order to adapt to new situations, labour rules, and strategies; (4) to integrate scheduling with the dynamics of data; (5) to partition the problem among several planners.

Uses of AI Technology

AI technology is the backbone of the operation of the system. The most visible part is state-space search (using a modified version of beam search). The search tree generated serves as the unifying media of all modes of operation. Whenever the planner uses manual mode, the system generates the selected successors in the search tree. If the planner decides to remove any tasks from the duties, the system generates successors of the current state that correspond to the removal actions (forward backtracking). If the planner decides to undo some action, he just has to move up in the tree to backtrack to a previous position. The search tree and its states can be inspected at any moment during the search process (Figure 8).

Abstraction is used in most phases of the scheduling process to reduce the amount of detail present in this domain. Without it, none of the other techniques used in the system would be capable of coping with the huge number of alternatives present, many of them having only minor differences.

The concept of strategy defined above is also central to the success of both the Automatic and Semi-automatic modes of the Scheduler. By combining clever heuristics (both in the generation and in the expansion of nodes) with adequate cost functions, the system can be fine tuned to optimise the relevant criteria chosen by the customer.

Another aspect of AI that is omnipresent is the use of constraints. These are used by the Automatic and Semi-automatic modes to select the most constrained tasks as preferential tasks to be used in node expansion (the most constrained tasks are those that are either in only one partition or that offer less possibility of combination with other tasks). Constraints are also used in the Pre-scheduler during the problem partitioning phase.

Data dependencies are used in the Data Manager to find out what concepts depend on a given concept. These dependencies are set up in a way that was influenced by TMS systems (Doyle 1979, Martins and Shapiro 88).

Application Use and Payoff

Before the deployment of the system, NS conducted extensive tests and the results obtained were better than our most optimistic expectations at the beginning of the project. Originally, at NS there were 24 planners (both for drivers and guards) that would take around 6 months to produce the duties for the company. The life-size test, conducted in early 1996, lead to conclude that the whole scheduling process can be completed by five planners in about two weeks (Notermans 1996). NS intends to use other existing planners for simulation studies and further improvement of the schedules. The comparison of the quality of the schedules produced by CREWS with the quality of the schedules produced by human planners points to a yearly reduction of personnel costs in the order of 6,000,000 Dutch Guilders (close to US$ 4,000,000) (Linssen 1995). This would mean that, just the savings of the drivers and guards, would pay the investment of NS in the system in less than one year after the start of operation.

The directly quantified gains were summarised by (van Aarle 1996) by a ten fold increase in speed of the scheduling process and a reduction of the required crew members in the order of 3%.

Besides the benefits of an automatic schedule generation, speeding up the process and making the generation criteria uniform, CREWS_NS, being an AI-based system, preserves the scheduling knowledge within the company easing the training of new planners (semi-automatic mode), brings a dramatic reduction of mechanical and tedious work done by planners, enables to adapt to new situations and to produce alternative solutions.

Another aspect that is currently starting to be explored by NS is the use of the system in "what-if" situations. These will be mainly used in three aspects: (1) NS wants to test the result of the introduction of changes in labour rules, in fact, this provides a valuable tool for supporting negotiation with the unions, since it can give an idea of

the overall impact of a rule change in the use and needs of personnel; (2) NS wants to perceive the effect that changes in the distribution of knowledge skills by different personnel bases will have on the existing schedules, based on the results of these studies, training actions may be taken in order to have qualified personnel in specific bases or bases may be closed down, being replaced by personnel in near-by bases; and (3) NS wants to find out further ways to improve the schedules, using different strategies and different duty patterns from what is being used nowadays.

Application Development and Deployment

The initial ideas for developing CREWS started in 1986 with a demonstration prototype for TAP/Air Portugal (in crew scheduling). Together with the development of this prototype, SISCOG started the development of a general crew scheduling tool, CREWS. Initially, CREWS was developed on Explorer LISP Machines using KEE as the underlying AI tool.

In 1988, SISCOG developed a crew scheduling prototype for the Portuguese Railways, CP, that was followed by the development of the ESCALAS system from March 1989 to January 1991. ESCALAS used a preliminary version of CREWS, that was a single user system, composed of one sub-system only (except for an early version of the Positioning Trip Generator), which included preliminary versions of the Scheduler, the Task Generator and the Task Sequencer.

After the completion of this system, we started to understand the strong constraints that the use of a LISP machine would place on the sale of CREWS and we ported CREWS (and ESCALAS) to a UNIX environment. At the same time, we also realised that KEE was placing an extra burden on the cost of the system (this was even more dramatic if we would take into account that only a small portion of KEE was being used by CREWS) and we developed a frame-based representation system, SIKE (SIscog Knowledge Environment), that would make available the knowledge representation characteristics needed by CREWS.

In September 1993, we started the development of CREWS_NS, the first product to include the CREWS system as described in this paper. This project, was completed in July 1996, and started operation in October 1996, after being thoroughly tested.

CREWS_NS was developed in three steps: (1) Single-user system that would work in manual mode. This included communication with external systems, manual-construction of duties, and manual assignment of trains to partitions; (2) Multi-user system. This included development of locking mechanisms, possibility of swapping of tasks among partitions, and automatic assignment of trains to partitions; and (3) Automatic and semi-automatic modes. This included development of special-purpose strategies and fine tuning of those strategies.

In each of these steps there were several partial versions delivered to NS that were tested by the members of the project team. At the end of each step, documentation about the completed version was delivered and a life-size test, with all data and a larger number of planners was performed.

The development of CREWS_NS and the new version of CREWS was carried out by a team composed of 10 programmers and knowledge engineers from SISCOG, two planners from NS, and one programmer from CVI (a software company daughter of NS and currently part of EDS). The role of SISCOG was to do knowledge elicitation and system development; the planners from NS supplied scheduling knowledge and tested the several versions of the system; and the programmer from CVI lead the team from NS side and implemented the man-machine interface. During the development we had 2-day meetings every other month during the first year of the project, that were mainly devoted to knowledge elicitation and versions testing. In the last two years of development these meeting were held every 3 months and were devoted to knowledge elicitation and evaluation of the system results.

Since SISCOG and NS are about 2,000 miles away, the communication and correction of bugs was chiefly done by e-mail which enabled us to correct small bugs within 24 hours of their detection.

Before deployment, NS had training sessions with one week duration, divided in two groups of planners. The first group started operation while the second group was being trained. The transition from traditional scheduling to automated scheduling was done with the emphasis on the manual mode and has gradually moved to the semi-automatic and automatic mode. The deployment was done without major problems and originated a lot of request of new functionality from the planners.

Maintenance

The maintenance of CREWS_NS should be considered under two different perspectives: the *planned changes* perspective corresponds to the changes in labour rules, in scheduling strategies, and the addition/replacement of types of tasks in the knowledge base. This is performed by NS itself, resorting to the technical person in charge of the system. The *unforeseen changes*, such as the needs for additional functionality are done by SISCOG (and may entail a change in CREWS itself) as part of a maintenance contract. The maintenance contract contemplates the delivery of new versions of CREWS, technical support, and a certain amount of manpower that is assigned to work on extra functionality for the system.

Future Directions

Although the development in LISP and Unix workstations has been quite satisfactory, we fell the market pressure to use a widespread language and PCs. For this reason, we started in January 1997 the porting of CREWS to C++ running under Windows NT. This development is expected to take over one year, after which we will be able to offer CREWS both in LISP and C++. The performance of this new version and the market demands will tell whether we should keep both versions or if we should only adopt one of them

Acknowledgements

We would like to acknowledge the important role that the teams from both CP and NS played, in the development of ESCALAS and CREWS_NS, respectively, and their dedication to the development of the projects. In particular, we would like to thank Ana Paula Cabeças, Claudia Freitas, Carlos Lopes, Manuel Silveira Lopes, and Arlindo Marques, from CP, and Robert Bijeman, Jos Broos, Jurjen Hooghiemstra, Hans Munk, and Math Notermans, from NS. Finally, we would like to thank the staff from SISCOG, Hugo Cadete, Patricia Fernandes, João Filipe, Antonio Frazão, Antonio Leitão, Luis Lisboa, Ricardo Saldanha, Sandra Sousa, João P. Varandas, and Antonio Vasconcelos, for their work and dedication.

References

Bisiani, R. 1987. Beam Search: *Encyclopedia of Artificial Intelligence, S.C.* Shapiro (ed.): 56-58, New York. John Wiley and Sons.

Doyle J 1979. A Truth Maintenance System. *Artificial Intelligence* 12(3): *231-272.*

Fox MS. 1987. *Constraint-Directed Search: A Case Study of Job-Shop Scheduling,* San -Francisco, CA: Morgan Kaufmann.

Linssen C. 1995. Written Communication.

Martins J.P.; and Shapiro S.C. 1988. A Model for Belief Revision, *Artificial Intelligence* 35(1): 25-79

Morgado E M.; and Martins J.P. 1989. CREWS A Crew Scheduling System. In Proc. of EXPERSYS-89, *Expert Systems Applications.*

Morgado E.M.; and Martins J.P. 1992. Scheduling and Managing Crew in the Portuguese Railways. *Expert Systems with Applications, An International Journal 5: 301-321.*

Morgado E.M.; and Martins J.P. 1993. An AI-Based Approach to Crew Scheduling. *Proc. IEEE Conference on AI Applications - CAIA* 93: 71-77.

Morgado E.M.; and Martins J P. 1996. Managing Change in Scheduling Data. *Computers in Railways V, Volume I: Railway Systems and Management: 479 - 4 8 9,* Southampton, UK: Computational Mechanics Publications.

Notermans M. 1996. Personal Communication.

van Aarle J. 1996. Scheduling Crew in the Dutch Railways, Communication presented in the Transportation Seminar, Lisbon, Portugal.

Sadeh, N.; and Fox, M.S. 1991. Variable and Value Ordering Heuristics for Hard Constraint Satisfaction Problems: An Application to Job Shop Scheduling, Technical Report CMU-RI, TR-9 l-23, The Robotics Institute, Carnegie Mellon University.

Smith S.F. 1989. The OPIS Framework for Modelling Manufacturing Systems, Technical Report TR-CMU-RI-89-10, The Robotics Institute, Carnegie Mellon University.