AI Magazine Volume 19 Number 1 (1998) (© AAAI)

CREWS_NS

Scheduling Train Crews in The Netherlands

Ernesto M. Morgado and João P. Martins

■ We present a system, CREWS_NS, that is used in the long-term scheduling of drivers and guards for the Dutch Railways. CREWS_NS schedules the work of about 5000 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 as 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 behavior of the system to changing circumstances. Scheduling can be done in automatic, semiautomatic, 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 (rescheduling).

CREWS_NS is a system that addresses the long-term scheduling of train crews at NS, the Dutch Railways (NV Nederlandse Spoorwegen). Long-term crew scheduling is typically done 6 to 12 months prior to the execution of the schedule and consists of 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. Scheduling is usually carried out manually by a small number of planners who acquire most of their knowledge through experience. At the Dutch Railways, long-term scheduling of crew involved 24 planners, who, working full time, would take about 6 months to produce the duties for about 5000 train drivers and guards.

Besides the human skill of efficiently arranging tasks into duties, crew scheduling deals with ever-changing data. Despite being done several months before the execution of the schedules, time constraints require crew scheduling to be started well before its input—the final timetable and rolling stock scheduling—have been completed. Moreover, after the schedules have been completed and execution has started, the problem keeps changing, requiring the data to constantly be updated. Thus, the schedules have to be revised several times because of timetable changes as well as changes in the physical resources associated with tasks. A human planner not only has to deal with huge amounts of data within a short period of time to produce schedules (*scheduling*), but he/she also has to handle changes that pop up constantly (*rescheduling*), most of which are produced by different departments and, thus, are often incomplete and inconsistent as a whole, and that must be incorporated into existing schedules with little disturbance.

Another aspect that puts a high demand on planners is the increasing complexity of labor rules that must comply with the increasing social benefits given to workers. Scheduling crew is thus 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.

As a typical resource-allocation problem, we might be inclined to consider it a job shop problem (Fox 1987) (Smith 1989), where the trains are activities, and the resources are individual personnel. Indeed, there are some simi-

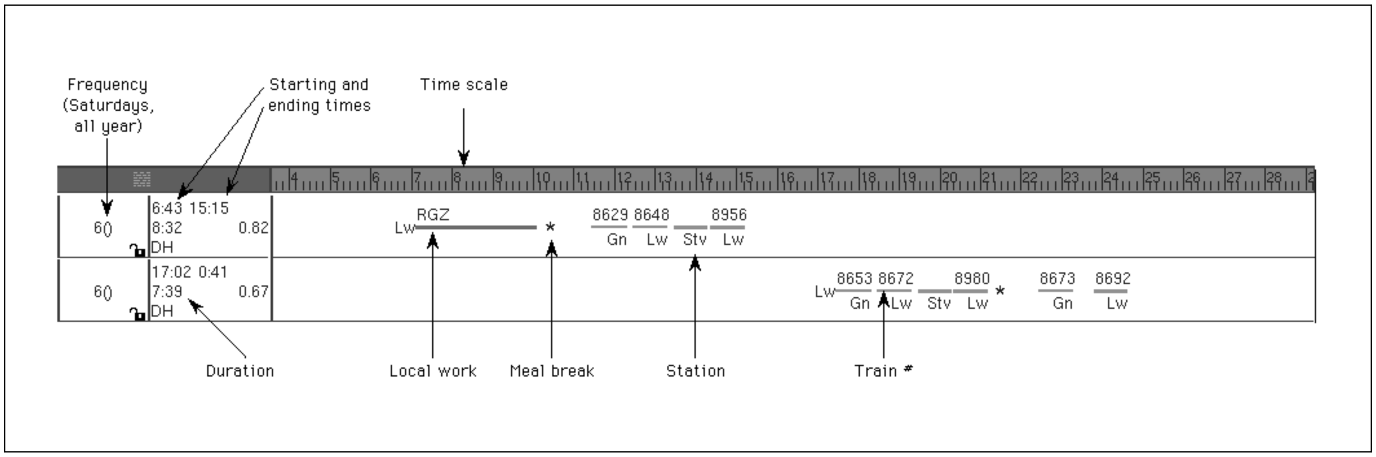


Figure 1. Example of Duties.

larities to this problem, but many differences prevent us from transposing techniques from one domain to the other. First, besides dealing with time constraints and other constraints of the job shop domain (for example, equipment constraints), crew planners must also deal with space constraints to prevent space discontinuities in duties, positioning crew where they are needed, whether as passengers in trains or other transportation means. Second, crew planners must also deal with complex train frequencies, such as week frequencies (for example, a train might only run on weekends), year periods (for example, only during summer), and special days (for example, on holidays and days before holidays), which put additional constraints on the combinations of tasks. These two aspects are critical to the quality of the final schedules and the efficiency of the scheduling process, requiring abstraction techniques to be used extensively. Third, 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 make it difficult to analyze activity demand and resource contention as is usually done in the job shop domain (Sadeh and Fox 1991). Fourth, the labor rules are complex and change every year because of union pressure. Worse, to avoid personnel strikes, planners must account for exceptions to the rules, which requires a high degree of flexibility in accommodating exceptions and changing rules without compromising the efficiency of the process. It also requires a representation model to be easily understood by the planners, different from a constraint representation model based only on variables and values (Sadeh and Fox 1991).

Crew scheduling has also been approached by traditional programming (supported by operational research, aiming at an optimized solution), but the results obtained with full automatic black-box optimization algorithms had only limited success and proved unsatisfactory in the following areas: (1) when faced with a full-size problem, these solutions tend to need computational resources that 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 which are external to the scheduling problem itself, others that are internal.

The *external difficulties* are related to the complexity and distribution of data (data are produced by different departments, at different times and are often incomplete and inconsistent), to the constantly changing data (data are not produced once for all; data continue to change, and it is important to find out the effect that the changed data have on the work that was already planned), and to 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 which is handled by one planner).

The *internal difficulties* are related to the combinatorial explosion of the problem and to the multiple and complex constraints that have to be satisfied by the duties. These constraints include *physical constraints* (for example, temporal continuity in tasks, spatial continuity in tasks, and compatibility of frequencies and year periods), *labor and social*

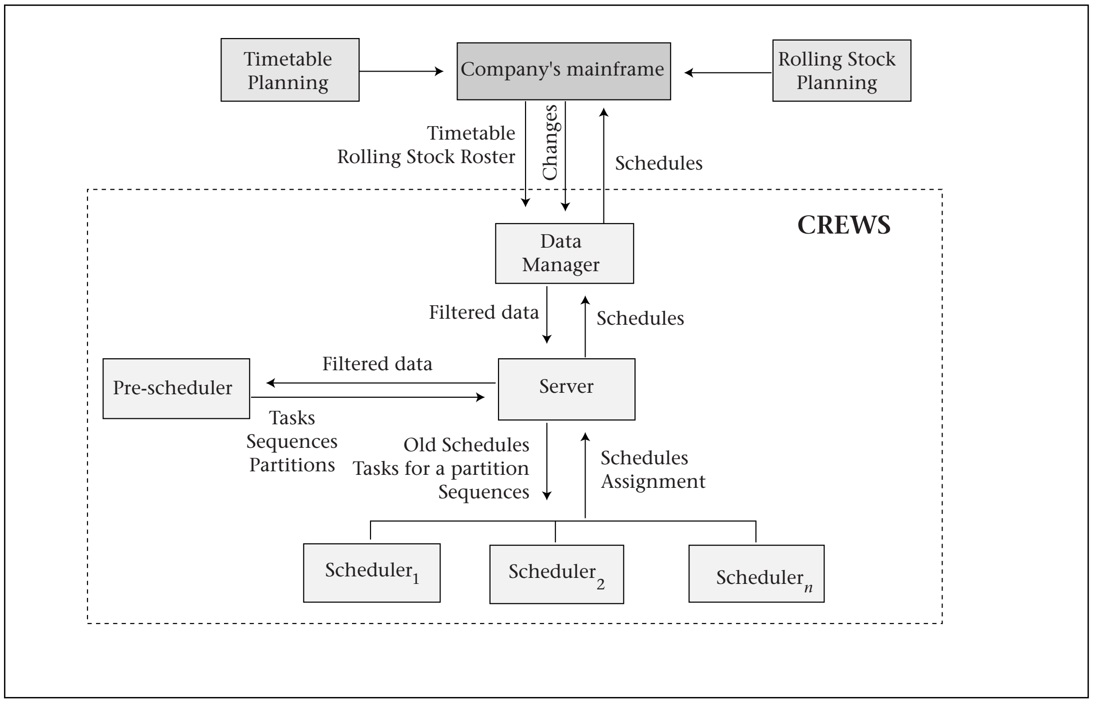


Figure 2. Architecture of CREWS.

constraints (for example, starting and ending places 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), and *economical constraints* (for example, those that might minimize the cost of the operation or the number of resources required). The constraints might be *hard* (cannot be violated) or *soft* (can be violated, but violations should be avoided and recorded).

Application Description

Because 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 can perceive what is going on, can interact with the system by proposing alternatives or querying decisions, and can adapt the behavior 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 cooperative way.

Furthermore, because 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, labor rules, scheduling strategies, and objectives).

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

The scheduling process is divided into three phases: (1) *data management*, where an analysis of data is done to find inconsistencies or incompleteness, both among input data and between input and output data (to identify which schedules have to be changed because of the change in input data); (2) *prescheduling*, where personnel tasks are generated according to rules that specify the number of personnel resources required for each type of activity, sequenced according to criteria that will guide the scheduling phase, and distributed among partitions that are scheduled quasi-independently; and (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 guide the search process.

The architecture of CREWS is based on components that address these phases: the data manager, the prescheduler, and the scheduler, respectively (figure 2). There are other compo-

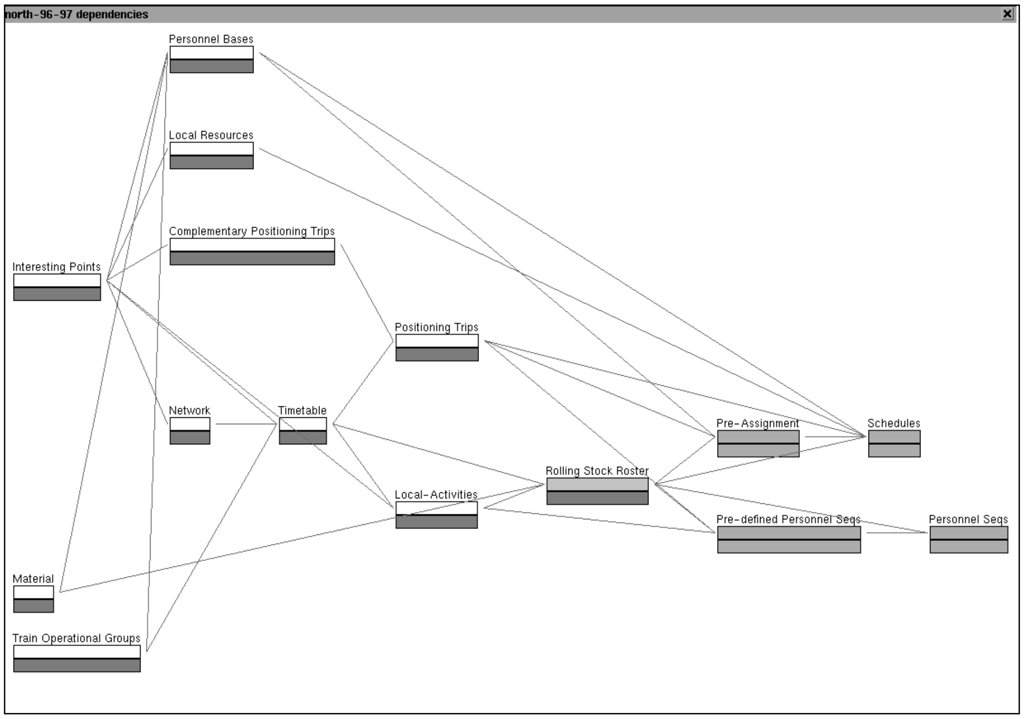


Figure 3. Data Set Dependencies.

nents, such as the positioning trip generator to position crew where they are needed and the CREWS server to support the client-server multiuser 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* (Cadete and Sousa 1994) 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 organizes 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 to the problem. The entities in the knowledge bases are linked together by *data dependencies* that tell how each concept depends on others and that can be specified and changed by the transportation company. 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 to solve the problem once the initial conditions change. 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, which are organized in a tree of situations. Each branch of the tree represents a sequence of situations: One corresponds to the real problem, but the others correspond to different simulations.

The data manager can work with data pertaining to all classes of personnel, such as the timetable and rolling stock roster (*generic data manager*), or with data that are specific to a certain class of personnel, for example, the drivers' local activities and schedules (specific data manager). This organization of classes of personnel forms a hierarchy of data managers, each one operated by users working at different levels in the company and validating and installing data for the next lower level. Data managers at the bottom level install data to be used by the corresponding preschedulers and schedulers.

The Prescheduler

The *prescheduler* makes the preparations for scheduling, looking at the problem of a specific class of personnel from a global perspective. The prescheduler is composed of the task generator, the task sequencer, and the task distributor.

In the description of the prescheduler and the scheduler, it is important to make the 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 can require *rolling stock resources* (for example, a train might require one engine and eight cars) and *personnel resources* (for example, 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 article) is the association of an activity with personnel resources. For example, a train activity might generate one task for a driver and one or more tasks for guards.

Task Generator The *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 might be needed, depending on the number and types of coach. The number of rolling stock units is derived from the rolling stock rosters and other sources, using task-generation rules that can be specified and changed by the transportation company.

Task Sequencer The task sequencer uses clustering and 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 sequences suggest that the personnel follow the rolling stock, avoiding problems resulting from train delays; however, there are cases where they might suggest something else. For example, if the operation is regular and on time, it is not important to follow the rolling stock; it might be more convenient to follow a train series (a certain number of trains following an operation pattern). Sometimes it can be useful for the preferential sequences to suggest the patterns of duties produced the year before, so that changes are minimized. The preferential sequences are just suggestions and can be changed later by the scheduler.

The task sequencer produces these sequences according to one of the previous criteria. This phase relies heavily on heuristic knowledge about where and when it is reasonable to change the crew. This knowledge is important in guiding the scheduling phase, in both automatic and manual modes. This phase also uses abstraction to group sequences at distinct frequencies into one single sequence at the union of the frequencies and without regard for irrelevant differences to reduce the combinatorial explosion of the automatic mode and the number of scheduling steps that the user must perform in constructing the duties in manual mode. Clustering and abstraction are guided by *sequencing rules*, which can be specified and changed by the transportation company.

Task Distributor The *task distributor* divides the problem into subproblems (partitions) so that they can be managed effectively by the scheduler. A problem partition can correspond, for example, to a base of a certain class of personnel. The task distributor determines which partitions tasks can be preassigned to, taking in account, among other aspects, the rolling stock and network knowledge of the personnel associated with each partition. A task can be preassigned to more than one partition, allowing subproblems to overlap. In this way, several planners can try the task in their schedules. However, only one of them will be able to schedule it. Thus, we say that the task has been *assigned* (as opposed to preassigned) 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 in the upper left corner, and each partition can be assigned a color. The constraints used in the task distribution are shown in the lower left corner. The sequences of tasks were computed by the task sequencer; tasks shown with a thin line are preassigned 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 are 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 preassigned to this partition and that have not yet been assigned (scheduled) to another partition. When saving a schedule, its corresponding tasks are assigned to the partition if they have not yet been assigned to another partition. Of course, when rescheduling, the scheduler also loads the tasks that have already been assigned (scheduled) to this partition.

Tasks can be preassigned to partitions (colored) either manually by a planner or automatically, following a task-distribution strategy that can be specified and changed by the transportation company. 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. The system provides a set of statistics and graphics to enable the user to evaluate the quality of the distribution (figure 5).

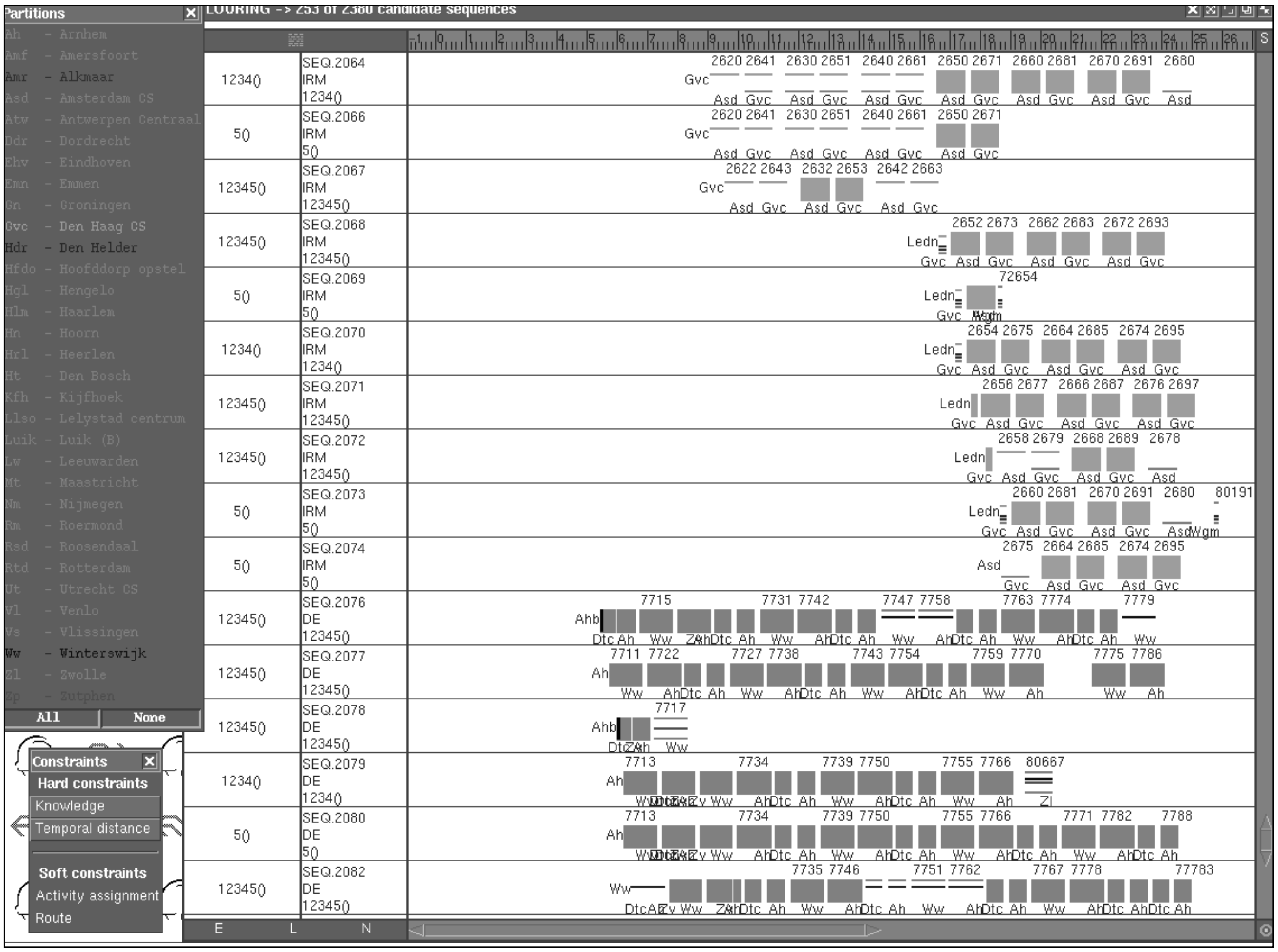


Figure 4. Preassigned and Assigned Tasks.

The Scheduler

The *scheduler* works from the perspective of a partition of a given class of personnel, 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, preassigned to a partition by the task distributor, and grouped in sequences by the task sequencer. The scheduler places these 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 for more than a certain number of hours without a meal break; must have line, rolling stock, and train knowledge to operate the trains; and so on. These constraints are specified in scheduling rules that can be changed by the transportation company.

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 to this point (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 where there are no more candidates, or a predetermined number of duties have been reached. The search is guided by a scheduling strategy consisting of (1) a process for selecting the initial state (for example, what kinds of task should be considered, sequences or groups of sequences) and its construction (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

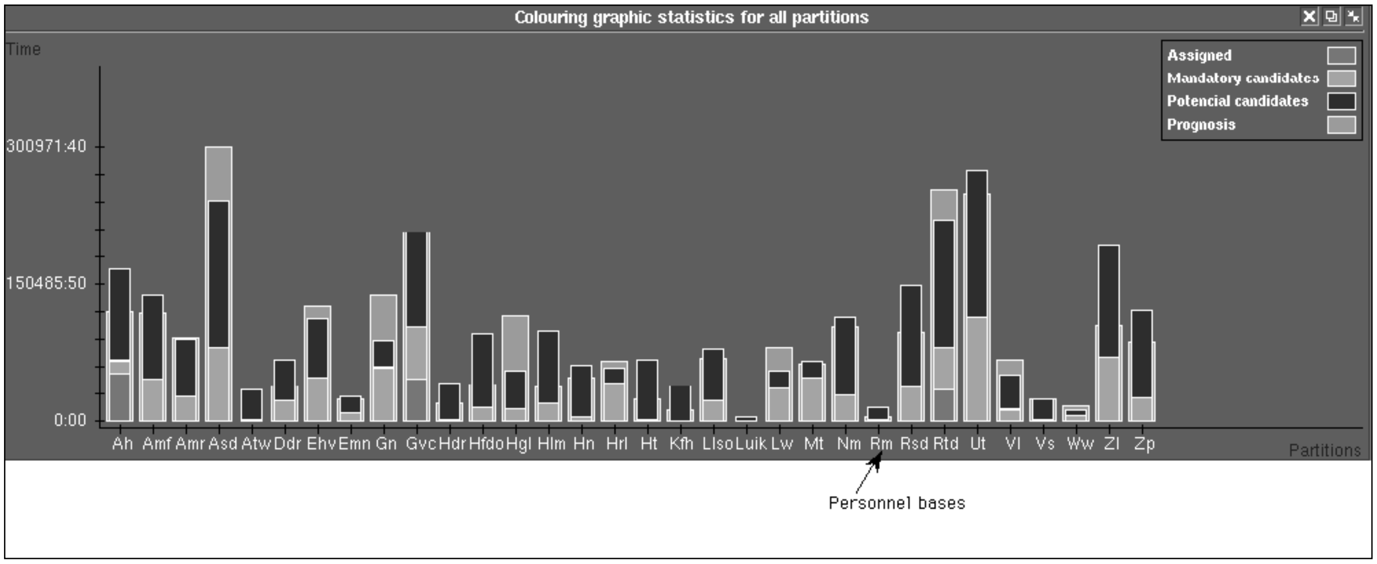


Figure 5. Task-Distribution Statistics.

of cost and heuristic functions, and (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 rescheduling. The user can add new strategies to handle new types of schedule.

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

First is the *manual mode*. The user tells the system the (sub-) sequence of tasks that should be moved from the candidates to the duties (figure 6). The system verifies all constraints imposed on the resulting duty and tells the user the constraints that are violated by the operation (we consider both soft constraints and hard constraints). If the planner chooses to violate a soft constraint, this violation is recorded, and the duty is shown with a violation indication. Clicking on the violation-indication icon generates a message that explains the violation (figure 7). 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 previous decision—an operation called *forward backtracking*—or can do traditional backtracking by moving into a previous state.

Second is the *semiautomatic mode*. The system gives hints about how the planner should pursue the schedule, following a strategy that is selected by the user. 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/she thinks is best. Figure 8 shows some of the alternatives for extending duty 1 (the number of alternatives presented is selected by the user).

Third is the *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.

Fourth is the *mixed mode,* which 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/she wants to impose on the final schedule; then uses the automatic mode to do the bulk of the work; and, finally, manipulates (using the manual mode) the resulting duties.

The mixed mode of operation really shows the decision-support philosophy that was incorporated into CREWS. In fact, it provides full cooperation between the planner and the system, showing what is going on, providing explanations about the decisions made 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/she elects to do so.

In summary, the main innovations offered by this system are (1) to provide a decision support system for crew scheduling (as opposed to the traditional black-box scheduling systems); (2) to trade the optimal solution for a solution that is good enough; (3) to give the ability to the customer to change the behavior of the

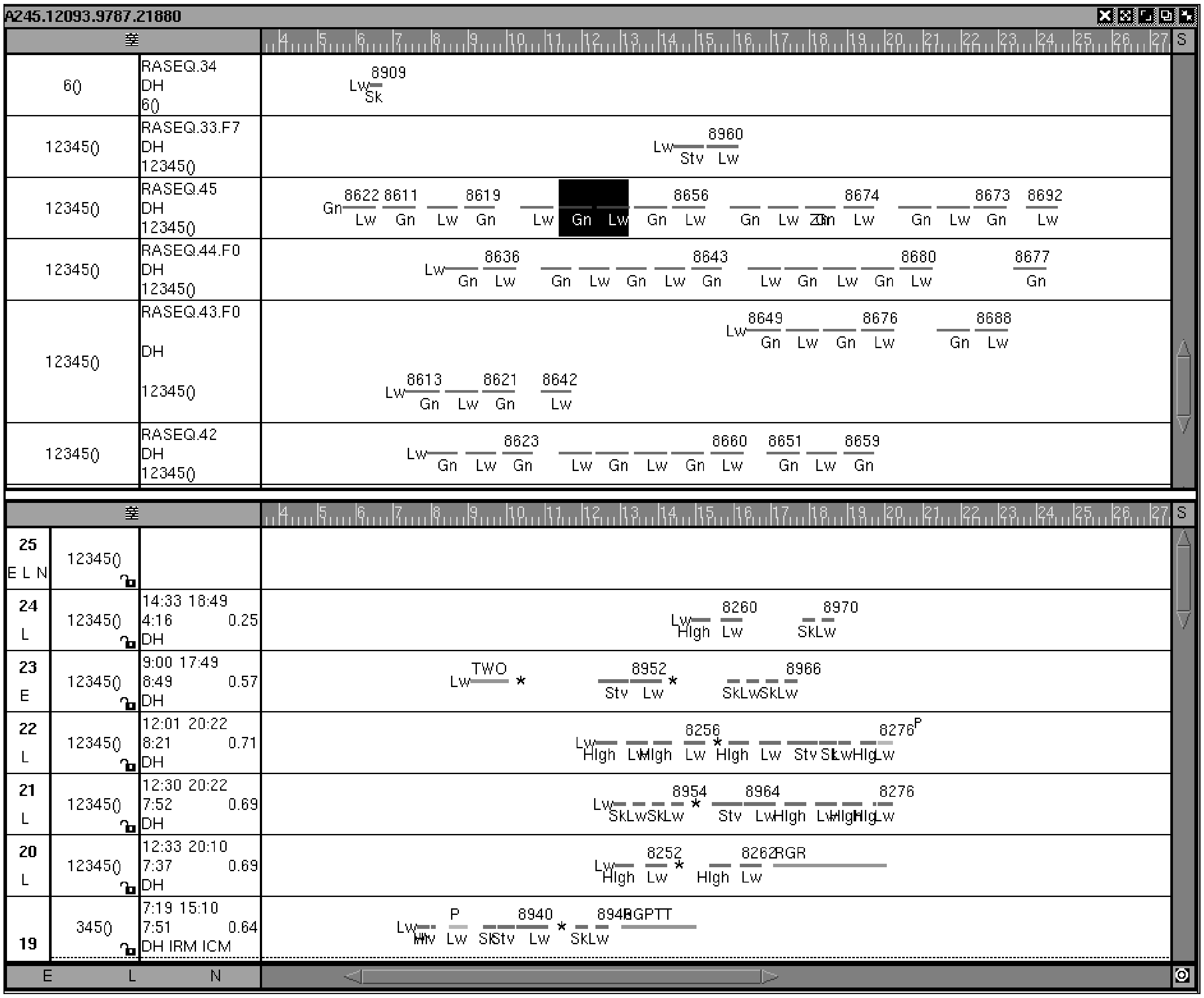
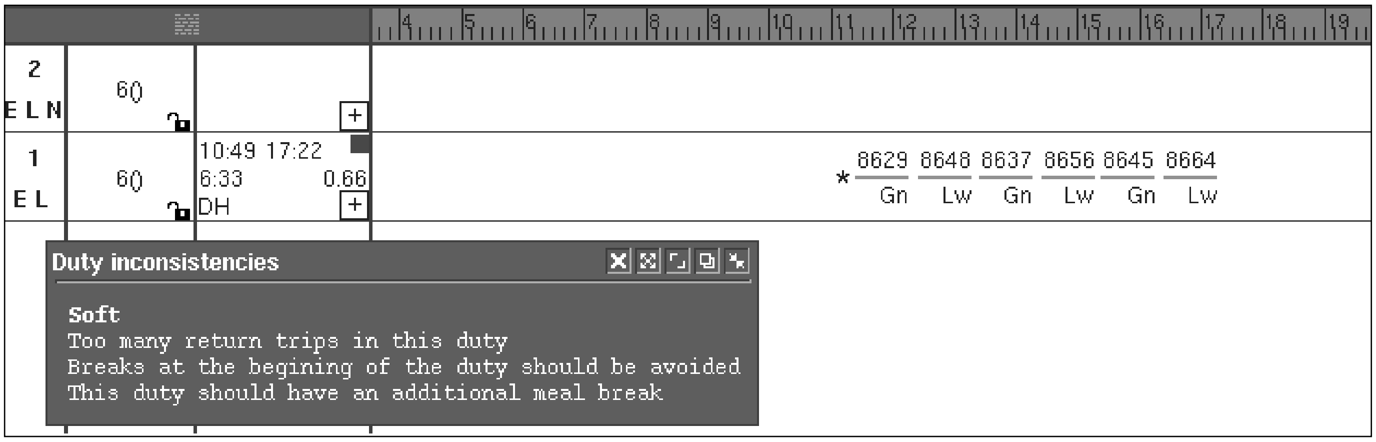

A245.12093.9787.21880
RASEQ.34 DH 6()
RASEQ.33.F7 DH 12345()
RASEQ.45 DH 12345()
RASEQ.44.F0 DH 12345()
RASEQ.43.F0 DH 12345()
RASEQ.42 DH 12345()
25 E L N 12345()
24 L 12345() 14:33 18:49 4:16 0.25 DH
23 E 12345() 9:00 17:49 8:49 0.57 DH
22 L 12345() 12:01 20:22 8:21 0.71 DH
21 L 12345() 12:30 20:22 7:52 0.69 DH
20 L 12345() 12:33 20:10 7:37 0.69 DH
19 345() 7:19 15:10 7:51 0.64 DH IRM ICM
E L N

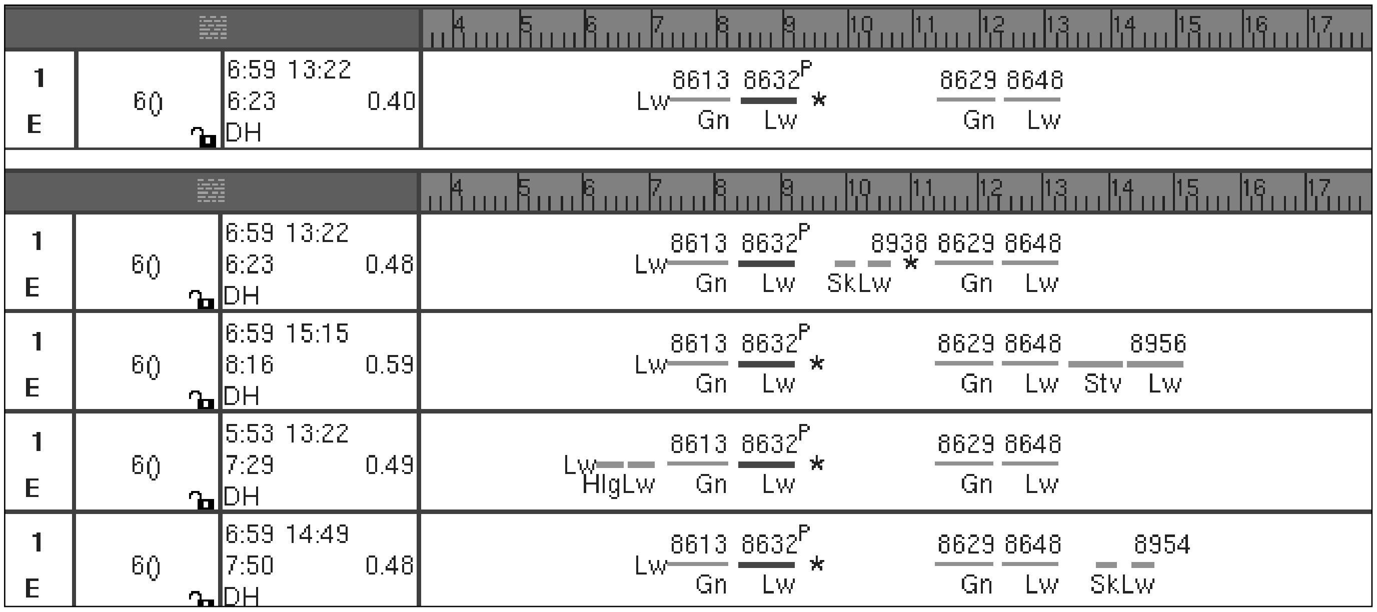
Figure 6. Moving Tasks from the Candidates to the Duties (Manual Mode).



Duty inconsistencies

Soft
Too many return trips in this duty
Breaks at the begining of the duty should be avoided
This duty should have an additional meal break

Figure 7. Showing Violations.



1 E	6()	6:59 13:22 6:23 0.40 DH	Lw 8613 Gn 8632P Lw * 8629 Gn 8648 Lw
1 E	6()	6:59 13:22 6:23 0.48 DH	Lw 8613 Gn 8632P Lw SkLw 8938 * 8629 Gn 8648 Lw
1 E	6()	6:59 15:15 8:16 0.59 DH	Lw 8613 Gn 8632P Lw * 8629 Gn 8648 Lw Stv 8956 Lw
1 E	6()	5:53 13:22 7:29 0.49 DH	Lw HlgLw 8613 Gn 8632P Lw * 8629 Gn 8648 Lw
1 E	6()	6:59 14:49 7:50 0.48 DH	Lw 8613 Gn 8632P Lw * 8629 Gn 8648 Lw SkLw 8954

Figure 8. Duty 1 and Alternatives for Extending Duty 1.

system to adapt to new situations, labor rules, and strategies; (4) to integrate scheduling with the dynamics of data; and (5) to partition the problem among several planners.

Working with the Application

For generating the schedules of a company, several steps must be followed with an application built with CREWS:

First, data are received from the outside systems that supply the timetable, the rolling stock roster, and other entities, following CREWS data-interface protocols.

Second, data can also be introduced using the data-manager interface, but in this case, it is not considered official and must be validated later by data coming from the outside systems.

Third, all these data are validated for overall consistency and completeness. Data that are consistent and complete are installed for use by other components of CREWS.

Fourth, this process with the data manager is hierarchically done in several steps, starting with the generic data manager and pursuing in parallel with several specific data managers, each one for each class of personnel.

Fifth, once data for a specific personnel class are installed, the planners can start with the prescheduler to distribute the tasks for the respective data set among the predefined partitions (personnel bases). The *task-distribution process*, also called the coloring process, defines a static partitioning of the problem that conditions the tasks that can be loaded later in each scheduler session.

Sixth, once tasks have been preassigned (colored) to partitions, several schedulers can start in a multiuser environment, each one loading a different partition and building a different schedule.

Seventh, because partitions can overlap, planners compete for tasks to build their schedules. The system assigns a task to the schedule that first saves the task. When conflicts arise, they must be resolved cooperatively. If the owner of the task agrees to release it for use in another schedule, the planner in charge of this schedule can schedule it immediately if the task was already preassigned to his/her partition or must go first to the prescheduler.

Eighth, scheduling all tasks can involve several cycles between the prescheduler and the scheduler, trying to get the partitioning of the problem that produces the best composition of schedules.

Ninth, during this process, planners can load both the prescheduler and the specific data manager to check the overall state of the scheduling process for the corresponding class of personnel. The planners can check which tasks have not been scheduled yet and which schedules are still incomplete.

Tenth, because data keep changing, this process is even more complex, involving a wider cycle than the one described previously that also includes the data manager. Whenever some data change, the planners should start

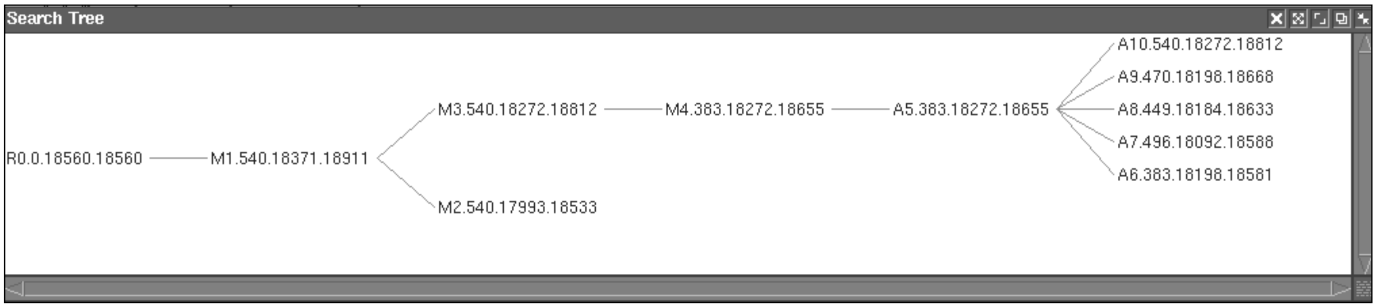


Figure 9. Simple Search Tree.

the whole process, validating and installing the data changes; generating, sequencing, and coloring the corresponding tasks; and, finally, incorporating the changes in the schedules using rescheduling.

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 for all modes of operation. Whenever the planner uses the manual mode, the system generates the selected successor in the search tree. If the planner decides to remove any tasks from the duties, the system generates a successor of the current state that corresponds to the removal action (forward backtracking). If the planner decides to undo some action, he/she just has to move up in the tree to backtrack to a previous position from which he/she can pursue with a different successor. The search tree and its states can be inspected at any moment during the search process (figure 9).

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 previously is also central to the success of both the automatic and the semiautomatic modes of the scheduler. By combining clever heuristics (both in the generation and the expansion of nodes) with adequate cost functions, the system can be fine tuned to optimize the relevant criteria chosen by the customer.

Another aspect of AI that is omnipresent is the use of constraints. These constraints are used by the automatic and semiautomatic modes to select the most constrained tasks as preferential tasks to be used in node expansion (the *most constrained tasks* are those that are in only one partition or that offer less possibility of combination with other tasks). Constraints are also used in the prescheduler 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 truth-maintenance systems (Martins and Shapiro 1988; Doyle 1979).

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) who would take about 6 months to produce the duties for the company. The life-size test, conducted in early 1996, led us 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 in personnel costs on the order of 6,000,000 Dutch Guilders (about US$4,000,000) (Linssen 1995). Thus, in less than one year, just the savings for the drivers and the guards would pay for the investment by NS in the system. The directly quantified gains were realized (van Aarle 1996) with a 10-fold increase in the speed of the scheduling process and a reduction in the required crew members on the order of 3 percent.

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

Another aspect that is currently being explored by NS is the use of the system in what-if situations. Such situations will primarily be used for three reasons: First, NS wants to test the result of the introduction of changes in labor rules; this tool is valuable in supporting negotiation with the unions because it can represent the overall impact of a rule change in the use and needs of personnel. Second, NS wants to understand the effect that changes in the distribution of knowledge skills by different personnel bases can have on the existing schedules. Based on the results of these studies, training actions can be taken to have qualified personnel in specific bases or bases can be closed down and the work transferred to nearby bases. Third, NS wants to find out other ways to improve the schedules, using different strategies and different duty patterns than are used today.

Application Development and Deployment

The initial ideas for developing CREWS started in 1986 with a crew-scheduling demonstration prototype for TAP/Air Portugal. Together with the development of this prototype, Sistemas Cognitivos, Lda. (SISCOG), began developing 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, which was followed by the ESCALAS system in January 1991. ESCALAS used a preliminary version of CREWS that was a single-user system composed of only one subsystem (except for an early version of the positioning trip generator); it included preliminary versions of the scheduler, the task generator, and the task sequencer.

After the completion of this system, we started to understand the restrictions 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 realized the extra cost of KEE (which was even more dramatic when we considered 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 article. This project was completed in July 1996 and started operation in October 1996 after thorough testing.

CREWS_NS was developed in three steps: The first step was a single-user system that would work in manual mode. This step included communication with external systems, manual construction of duties, and the manual assignment of trains to partitions. The second step was a multiuser system. This step included the development of locking mechanisms, the possibility of swapping tasks among partitions, and the automatic assignment of trains to partitions. The third step was the automatic and semiautomatic modes. This step included the development of special-purpose strategies and the fine tuning of these strategies.

In each of these steps, several partial versions were 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 the 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 of 10 programmers and knowledge engineers from SISCOG, 2 planners from NS, and 1 programmer from CVI (a software company daughter of NS and currently part of Electronic Data Systems). 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 led the team from the NS side and implemented the man-machine interface. During the first year of project development, we had two-day meetings every other month that were mainly devoted to knowledge elicitation and version testing. In the last two years of development, these meeting were held every three months and were devoted to knowledge elicitation and evaluation of system results.

Because SISCOG and NS are about 2000 miles apart, the communication and correction of bugs (through software patches) was chiefly done by e-mail, which enabled us to correct small bugs within 24 hours of their detection. The use of Lisp clearly helped this process.

Before deployment, NS had one-week training sessions divided into 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 gradually moved to the semiautomatic and automatic modes. The deployment was done without major problems and generated a lot of requests from the planners for new functions.

Maintenance

The maintenance of CREWS_NS should be considered from two different perspectives: (1) planned changes and (2) unforeseen changes. *Planned changes* correspond to the changes in labor rules and scheduling strategies and the addition-replacement of types of task in the knowledge base. This maintenance is performed by a technical person at NS who received training from SISCOG. The *unforeseen changes*, such as the needs for additional functions are handled by SISCOG (possibly entailing a change in CREWS itself) as part of a maintenance contract. The maintenance contract considers the delivery of new versions of CREWS, technical support, and the personnel who are assigned to work on extra functions for the system.

Future Directions

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

Also, CREWS is being constantly improved to provide new functions as demanded by the Dutch Railways or required by other company realities. We are currently working on a new version of CREWS that presents a revised architecture and additional functions, providing much more flexibility to users and even more gains in the overall efficiency of the scheduling process and the quality of the resulting schedules.

The main changes in the architecture concern the following:

First, the data manager can handle all levels of data (from the most generic to the most specific) instead of just one (see the previous description of the data manager), enabling the installation of all data in one single session and, thus, speeding the overall process.

Second, partitioning of the problem is much more flexible using a server of tasks and duties that can serve just those that are requested at any specific moment by the user in a distributed fashion. The planner is no longer constrained to work with just the tasks that have been preassigned to a base and the duties that are part of a schedule.

Third, the prescheduler function was incorporated into the scheduler, providing only one module rather than two. This function, together with the previous one, prevents users from jumping between two modules when they cannot find the tasks they need for a schedule or when they want to release tasks to be used by some other user (a result of a bad partitioning of the problem).

Fourth, the scheduler can work with more than one personnel base simultaneously, letting the system choose dynamically the task to which it should go rather than use a trial-and-error process between the task distributor and the scheduler. In this way, the partitions of the problem are no longer the bases but, rather, some loose concepts.

Fifth, the sequences of tasks can be generated manually by the planners and saved as patterns to be used later in subsequent sessions, improving the quality of the sequences and, consequently, the quality of the resulting schedules.

Sixth, scheduling can start from sequences of tasks organized in a time-space diagram (figure 10), instead of just a Ghantt chart. Thus, planners can visualize not only the time relationships among tasks but also their spatial relationships, resulting in more efficient ways of combining tasks into duties.

All these improvements have already been incorporated into CREWS, together with the addition of new strategies and heuristics in CREWS_NS. This version is currently under testing and evaluation, at both SISCOG and the Dutch Railways, and the results seem promising, bringing even more benefits to the scheduling operation of the Dutch Railways.

New functions will be introduced in the near future to answer the needs of West Anglia Great Northern Railway, the Norwegian Railways (both of which have signed contracts with SISCOG), and other companies in the railway as well as other transportation domains. The amount of interest raised by the success of CREWS_NS opens up new opportunities for SISCOG in this domain and is clear proof of the benefits of using AI techniques in the real world.

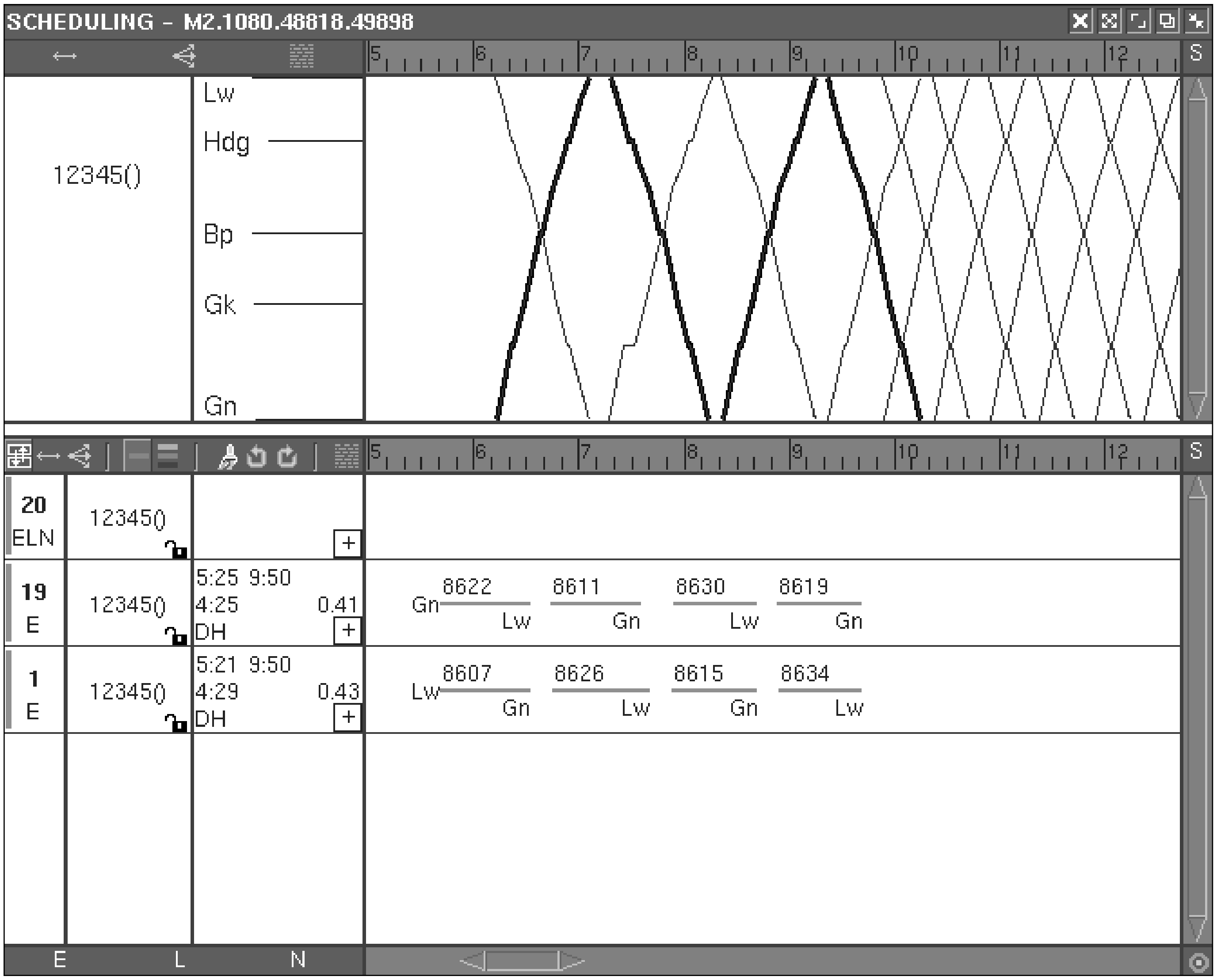


Figure 10. Scheduling with Time-Space Diagrams.

Acknowledgments

We would like to acknowledge the important role that the teams from both Caminhos de Ferro Portugueses (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, Cláudia 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, Patrícia Fernandes, João Filipe, António Frazão, António Leitão, Luís Lisboa, Ricardo Saldanha, Sandra Sousa, João P. Varandas, and António Vasconcelos—for their work and dedication.

References

Bisiani, R. 1987. Beam Search. In *Encyclopedia of Artificial Intelligence*, ed. S. C. Shapiro, 56–58. New York: Wiley.

Cadete H., and Sousa S. 1994. Tratamento de Mudança em Sistemas de Escalamento (Handling Change in Scheduling Systems). SISCOG, Lisbon, Portugal.

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

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

Linssen, C. 1995. Written communication. September, Utrecht, The Netherlands.

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. Paper presented at EXPERSYS-89, Expert Systems Applications, 23–24 October, Paris.

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. In Proceedings of the IEEE Conference on AI Applications, CAIA 93, 71–77. Washington, D.C.: Institute of Electrical and Electronics Engineers.

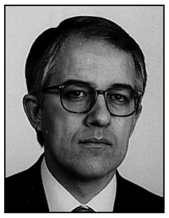
Morgado, E. M., and Martins, J. P. 1996. Managing Change in Scheduling Data. *Computers in Railways V, Volume 1: Railway Systems and Management,* 479–489. Southampton, U.K.: Computational Mechanics.

Notermans, M. 1996. Oral communication. October, Utrecht, The Netherlands.

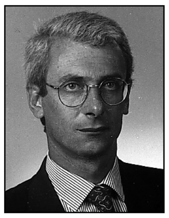
van Aarle, J. 1996. Scheduling Crew in the Dutch Railways. Paper presented at the Transportation Seminar, October, Lisbon, Portugal.

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-91-23, The Robotics Institute, Carnegie Mellon University.

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



Ernesto M. Morgado is associate professor of computer science and AI at IST, the engineering faculty of the Technical University of Lisbon, and is cofounder and president of Sistemas Cognitivos Lda., a Portuguese AI company. He holds a Ph.D. in AI from State University of New York at Buffalo. His research interests include scheduling, search-planning, knowledge representation, and programming methodology. His e-mail address is emorgado@ siscog.pt.



João P. Martins received an M.S. in computer science in 1979 and a Ph.D. in AI from the State University of New York at Buffalo in 1983. He shares his time between Sistemas Cognitivos Lda., a company he cofounded in 1983 that is devoted to applications using AI, and the Technical University of Lisbon, where he is full professor of computer science. His interests include knowledge representation, reasoning, and scheduling. His e-mail address is jpmartins@siscog.pt.