Dynamic Workforce Scheduling for British Telecommunications plc

David Lesaint
Intelligent Systems Research
BT France
11, place des Vosges
92061 Paris La Defense, France

Christos Voudouris
Intelligent Systems Research
BT Adastral Park
MLB1-PP12
Ipswich IP5 3RE, United Kingdom

Nader Azarmi
Intelligent Systems Research
BT Adastral Park

British Telecommunications plc (BT) employs thousands of field engineers across the UK to maintain networks, repair faults, and provide service to customers. To allocate work efficiently, BT developed Work Manager, an information system that automates work management and field communications. In 1996, the Intelligent Systems Research group of BT enhanced Work Manager with a dynamic scheduler (DS) based on a combination of heuristic search and constraint-based reasoning. Rolled out in 1997 and reaching 20,000 engineers in 1998, DS with Work Manager is saving BT $150 million a year on operational costs. When deployed over the targeted workforce of 40,000 people, the system will save an estimated $250 million a year.

Simply stated, field-workforce scheduling is about sending the right engineer to the right customer at the right place at the right time with the right equipment—at any time and in any operational environment. For BT, which employs over 50,000 field engineers, workforce scheduling is critical, and its ability to provide high quality service while achieving maximum productivity and low operational costs is vital to the company's success and competitiveness.

Many factors contribute to the complexity of the problem. First, skill requirements vary immensely: putting in a switch on business premises may take several engi-

0092-2102/00/3001/0045/$05.00
1526–551X electronic ISSN

INFORMATION SYSTEMS—DECISION SUPPORT SYSTEMS
TRANSPORTATION—SCHEDULING, PERSONNEL

neers several days while recovering redundant equipment may take an unskilled person a few minutes. Three basic operational areas have distinct skill demands: business and residential customer access, national business communications, and core network. Each area is the responsibility of a separate BT division.

The second factor contributing to the complexity of scheduling is the geographical distribution of the workforce. For example, the 15,000-person workforce in the access division consists of 175 separate groups that operate within nonoverlapping geographical domains. Each group manages its domain independently and controls from a dozen to a few hundred engineers and each group may perform a few hundred to a few thousand tasks per day. This geographical and operational decomposition allows the management of smaller-sized workforces than would be the case if the entire division's workforce were managed centrally.

Complexity also lies in the multiple requirements and objectives of workforce scheduling. These requirements reflect standard allocation constraints and corporate rules, and they determine the feasibility and acceptability of work assignments. Some constraints recur across domains and divisions:

—each engineer operates primarily within a predefined area;
—each engineer has off-hours and predefined breaks during the day (for example, lunch, time for personal appointments) that must be inserted into his or her schedule;
—engineers can perform only one activity at a time;
—the last task of the day may require overtime, if allowed;
—some tasks must be performed within an agreed-upon time window;
—some tasks must be started or completed by an agreed-upon time;
—engineers must be matched to tasks;
—access to customer premises may be granted only to certain individuals at certain times;
—task execution may be broken according to rules that take into account task and break details;
—the duration of a task depends on the engineer's experience and skills, and the duration of a journey depends on its start time;
—some tasks must be sequenced in time, for example, collecting equipment before making a visit; and
—some tasks must be performed in parallel by different engineers in different places at the same time.

(Geographical) domain-dependent requirements are rare. In fact, domains differ only on nonessential features, such as their road networks, their distribution of customers, the size and skills of their workforces, or their task-notification patterns.

Scheduling decisions within domains are driven by the same set of corporate objectives and customer preferences:

—maximize the productivity of the workforce;
—improve service quality, measured according to the type of customer (residential, business), the priority of the work (provision, installation, maintenance, repair), and any service-level agreement (on appointment windows and due dates);
—make best utilization of skills, for exam-

ple, assign the most suitable engineer to a task;
—minimize the operational costs resulting from travel time, waiting time, and work in overtime; and
—satisfy work controllers' and engineers' preferences as to location and types of assignments, for example, avoid assigning low-level work to specialized individuals.

Constructing feasible and good-quality work schedules under these conditions is hard! In fact, the problem may be viewed as a complex variant of the vehicle-routing problem featuring multiple vehicles, multiple depots, various compatibility constraints, time constraints, operational constraints, synchronisation constraints, and conflicting quality objectives.

The problem is further compounded by the inherent instability of the environment. Indeed, much schedule information is uncertain, imprecise, and incomplete:
—BT and its customers may request, cancel, or amend jobs unpredictably;
—engineer availability is subject to last-minute changes, and estimates of task duration and travel time cannot be totally accurate;
—work controllers may modify provisional work assignments and review business objectives at any time; and
—the environment itself (weather, traffic conditions) is unpredictable.

Therefore, any scheduling process must continuously incorporate new data to generate valid work assignments but it must also minimize the impact of these data on the current work schedule. Indeed, anyone interacting with the work-allocation system must be provided with schedule information that is as stable as possible over time. For instance, a resource manager who elaborates short-term plans based on the schedule information provided must be confident that this information will resist future changes.

Towards a Unified Work-Management Process

To improve and unify its work-allocation operations, BT decided in 1989 to automate its work-management and field-communication processes. In 1993, it introduced an information system called Work Manager [Garwood 1997]. Work Manager provides (Figure 1):
—an interface to other BT systems that feeds it with work;
—task-dispatch and task-progress-and-closure functions;
—a task-jeopardy management system to ensure that work controllers are alerted to potential failure;
—feedback services to keep the system that originated a task informed of its progress;
—management of sequences and lineups;
—an interface to inventory management systems; and
—a real-time and historic management information system.

In developing Work Manager, BT wanted to enable and benefit from automated scheduling. It initially equipped Work Manager with a real-time allocation algorithm (RTA), and each geographical domain ran an instance of this system. The RTA is an extension of the Hungarian algorithm that operates on an assignment-cost matrix to determine the minimal-cost assignment set [Laithwaite 1995]. Within Work Manager, the RTA ran every five minutes to identify and assign the next

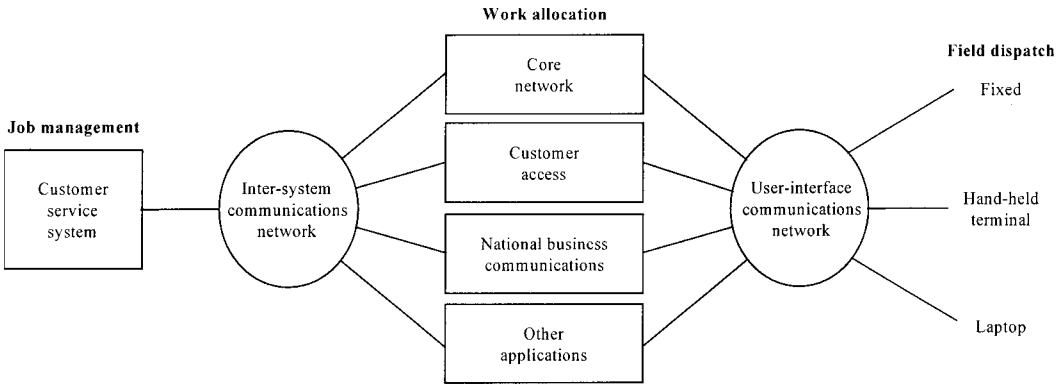

Figure 1: Work Manager is the system that manages the allocation of customer orders and their execution by field engineers in different operational environments (core network, customer access, and national business communications). Field engineers communicate with the system through various interfaces (laptops, hand-held terminals, and so forth).

task to each engineer. Rolled out in 1994, Work Manager with the RTA made tangible improvements in productivity and quality of service.

RTA originally complemented an overnight batch-scheduling process called TourBuild which scheduled the more predictable and stable provision work. However, business needs were changing rapidly, and TourBuild could not support a more integrated workforce with dynamic repair and provision. BT then successfully enhanced RTA to automatically allocate engineers both types of work, separately and as a mixed workload. However, it was clearly not optimal for situations that called for many scarce skills, for long tasks (a day or more), or for tasks with interdependencies—all of which required greater look-ahead to schedule and allocate efficiently. In these situations, work controllers found it difficult to minimize ineffective time and balance work across the workforce because of RTA's short-term view, and they had to intervene manually to get optimal results. Indeed, work controllers can still schedule tasks manually if they are not satisfied with the assignments the RTA proposes.

BT needed a scheduler that would be as good as the RTA for very dynamic workloads with short response times but much better for more complex work and mixed workloads requiring greater look-ahead. In April 1995, BT decided to explore alternative scheduling algorithms that could replace the RTA within Work Manager and retain the benefits and savings already achieved. BT UK customer service division set up a cross-divisional team technically led by the intelligent systems research group of BT advanced communications engineering division to design and develop a new dynamic scheduling system called Dynamic Scheduler (DS).

Dynamic Scheduling

BT had the following objectives for Dynamic Scheduler: to extend the scope of Work Manager to all work and people, to maintain or improve quality of service, to reduce travel time, to improve field and control productivity, to reduce control in-

terventions, to improve management control through data visualization, and to improve resource planning.

The computational problem DS faces is to generate feasible and high-quality schedules and to adapt them over time to accommodate the dynamics of reality. Based on our experience with the RTA and our analysis of the problem, we built DS based on two principles:
—coupling loosely an on-line allocator and a predictive scheduler to preserve responsiveness while benefiting from global optimization (Figure 2); and
—using a uniform constraint optimization approach to provide a generic and efficient underlying computational model.

Because of the complexity of the feasibility problem, we decided to exploit the most appropriate technology for handling constraints, constraint programming [Tsang 1993]. This could ensure generating realistic schedules using optimal constraint-propagation techniques and encountering no limitations in scope if problem specifications evolve (as they did for the RTA).

However, a fully constraint-based scheduler is not appropriate for the optimization dimension of the problem. The ideal approach would combine proven optimization techniques with constraint-based reasoning to tackle all aspects of the problem comprehensively and efficiently. We chose heuristic search because of its track record in vehicle routing [Laporte and Osman 1995]. After running comparative tests on simulated annealing, tabu search, and genetic algorithms over large collections of simplified workforce scheduling problem instances [Brind, Muller, and Prosser 1995], we elected simulated annealing [Kirkpatrick, Gelatt, and Vecchi 1983] as the best algorithm based on efficiency, simplicity, and robustness criteria.

The challenge in integrating simulated annealing and constraint-based reasoning is to preserve the former's performance without losing the exactness and extensibility of the schedule model as implemented by the constraint-based representation. For this reason, we developed a systematic algorithm to preschedule the most-constrained tasks: the long-duration tasks (typically, over eight hours) and the interdependent tasks (tasks that must be synchronised to satisfy a complex work request). This algorithm performs a tree search and is guided by domain-dependent heuristics defined over the choice of tasks, engineers, and tour positions for tasks.

After the prescheduling stage, we use simulated annealing to schedule the other tasks (the independent or short-duration tasks) without any performance deterioration. Both algorithms operate with the same schedule horizon, which is typically two days in advance. Simulated annealing does most of the work since most tasks are independent (more than 90 percent in the access domain, more than 80 percent in the core domain). The output of the two-stage scheduling is a provisional schedule defining a sequence of tasks for each engineer. This schedule complies with all the constraints specified and minimizes a cost function reflecting the various objectives of the problem (we describe the algorithms in detail in the Appendix and in Lesaint et al. [1998]).

The tree-search and simulated-annealing

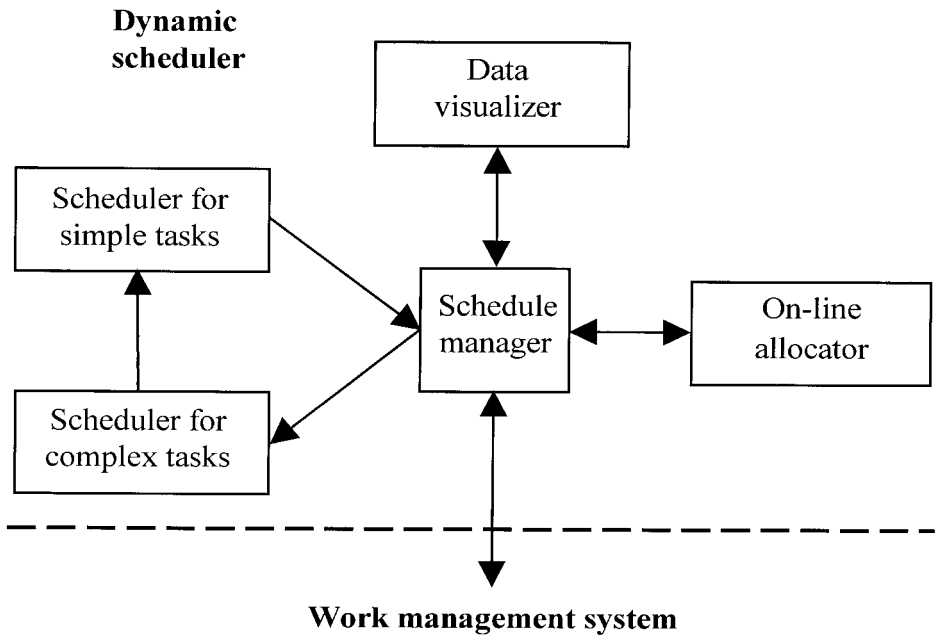

Figure 2: The dynamic scheduler consists of a schedule manager that stores and communicates the provisional schedule to the work-management system, a scheduler for complex tasks and a scheduler for simple tasks that periodically produce the long-term schedule, an on-line allocator that adjusts the schedule just before dispatch and a schedule and data visualizer (a desktop computer) that field-resource controllers and managers use to access problem and schedule information.

algorithms form the predictive component of DS. They are run every five to 15 minutes to generate the provisional schedule, which is then fed to the on-line allocator. We adopted this activation strategy to match the dynamic nature of the problem.

The on-line allocator dispatches work assignments to engineers: it is the ultimate decision maker (apart from work controllers, who can always interfere). It is triggered by external events, for example, tasks requested by on-line engineers and tasks that require immediate work before they become failed business targets. By default, the allocator simply forwards the assignments prescribed in the provisional schedule to the engineers, but it may modify these assignments if unforeseen events have caused infeasibility or suboptimality, for example, late notification of a high-priority task or early arrival of an engineer. The allocator's decision making is rule based, and its scope is limited to a subset of the resources and tasks so it can deliver answers within a few seconds.

Provisional assignments often prove inappropriate at dispatch time and the allocator must interfere. This is unavoidable given the uncertainty and the pace of change in the problem. Ideally, one would tackle instability by capturing and reasoning about elements of change using such techniques as stochastic programming. However, it is very difficult to forecast the evolution of BT's workforce-scheduling environments accurately. Although we

have identified general patterns (for example, on customer calls), no forecasting model is accurate enough to be exploited during scheduling [Simpson et al. 1995].

DS is equipped with an interface that allows one to view provisional schedules and any information generated by the predictive scheduler. Both temporal and spatial dimensions of the schedules are displayed through a Gantt chart and a geographical map. Pie charts, bar charts, and graphs provide additional statistics and measures on algorithms and solutions. This interface can also be used to implement what-if scenarios by running the predictive algorithms off-line on selected scenarios and with different settings.

Broad Impact

BT first deployed DS in November 1996 in South London. After two successful months, it decided on a national rollout starting in 1997. In May 1998, the coverage reached 20,000 of the 28,000 engineers under Work Manager's control. DS has helped BT in its aim of repairing "today's faults today" and achieving high customer satisfaction.

After two years of exploitation, the results have confirmed the superiority of the approach over the short-term scheduling strategy of the RTA. First, the use of DS in Work Manager annually saves $150 million on engineers' and controllers' pay costs and other workforce related costs, such as vehicles, equipment, tools, training, and administration. This is a 10-percent increase in savings over Work Manager used with the RTA. The savings realized with the DS are based on a four-percent productivity increase nationwide, which comes from reductions in travel time and waiting time and better allocation of work.

In addition, DS has paved the way for the automated management of another 12,000 engineers now managed outside Work Manager because their work was not suitable for RTA's allocation. BT expects a 15-percent productivity increase for this workforce, saving BT an additional $100 million a year. The savings for the full coverage of 40,000 engineers should reach $250 million a year. The expected 15-percent increase is made up of an 11-percent productivity increase achieved by the introduction of the Work Manager and an additional four-percent increase resulting from DS deployment.

DS has also improved field-control operations. Because it provides a forward view (of up to five days ahead), work controllers can be more proactive in managing task jeopardy and in planning short-term resources. Using the visualizer and its what-if capability, they can identify and resolve resource bottlenecks and potential task failures in advance. The accuracy and quality of the schedule information the system provides has given work controllers confidence in its performance. As a result, they have reduced their manual interventions dramatically: for example, the percentage of manual allocation dropped by 40 percent during the first field trial.

DS has helped BT to meet due dates, appointment windows, and skill requirements more consistently than it used to. Furthermore, field engineers' work schedules have improved. They travel less; their absences (for lunch, appointments, sick-

ness, and so forth) are smoothly inserted within their work schedules; work interruptions tend to be less frequent; and legal rules agreed between BT and unions on the management of overtime or travel in own time are rigorously implemented.

DS has also enhanced work controllers' interactions with Work Manager. The interface for configuring the predictive scheduler is much simpler than the RTA's. The user has to select a few heuristics to "drive" the prescheduler, for example, to order jobs based on priority, urgency, or skill levels. Regarding the simulated-annealing algorithm, the user can configure the cost function through intuitive parameters, for example, a penalty for minutes spent on travel or a weight to balance the importance of one task category over another. To operate the visualizer itself, users need basic knowledge of the window-environment. They take national training courses to familiarize themselves with the system and to learn how to drive it. However, because each domain is unique, DS needs extensive tuning before going live, and work controllers also have to get accustomed to the peculiarities of these domain-dependent adjustments.

Because of the generic nature of its constraint-based schedule model, DS requires little reengineering for use in scheduling within BT. In fact, DS can facilitate the removal of geographical and operational boundaries (domains and divisions) that divide BT's work-management operations. BT is investigating this simplification of its work-management organization. DS (as any other optimization software) is limited only by the size of problems in relation to the expected response-time levels. Overall, DS has:

—reduced BT's operational costs;
—improved the integrity, quality, and consistency of work allocation for the whole workforce;
—improved customer satisfaction;
—enhanced resource control;
—improved field engineers' and work controllers' perception of Work Manager; and
—offered BT the potential of simplifying its work-management organization.

From a purely technical viewpoint, DS demonstrated that integrating heuristic-search and constraint programming is a successful approach for tackling large-scale industrial vehicle routing problems. In fact, we believe that such hybridization is the key to solving a wide range of hard combinatorial problems.

Acknowledgments

We acknowledge our colleagues in BT UK Customer Service and Advanced Communications Engineering Divisions: Ian Alleston, Bob Laithwaite, Andy Noble, John O'Donoghue, and Paul Walker.

APPENDIX

Constraint Optimization Model

We designed the constraint-optimization model employed in the tree-search and simulated-annealing algorithms to solve large problems every five to 15 minutes. In the context of BT's work management, a typical (static) problem consists of 50 engineers and 300 tasks that have to be scheduled over a horizon of two days. The problem size rarely exceeds 100 engineers and 500 tasks. However, the schedule horizon is often set to five days in stable operational environments. The algorithms usually return a good schedule in less than a minute.

Both algorithms operate with the same

schedule definition. In this definition, a schedule is a set of tours and each tour is a sequence of tasks assigned to an engineer over the schedule horizon. To allow for the representation of incomplete schedules, we introduce a virtual engineer who is "assigned" the nonallocated tasks. Both algorithms modify schedules iteratively with an operator called *relocate*. Given a task, an engineer, and a valid position in the tour of this engineer, *relocate* relocates the task in the position of the engineer's tour.

Tree-Search

The tree-search algorithm applies the operator *relocate* in a systematic fashion by selecting tasks, engineers, and positions in the order the heuristics prescribe. Precisely, the tasks to preschedule are ranked using six ordering heuristics: earliest possible start time (earliest first), business importance score (maximum first), number of engineers required (greatest first), due date (earliest first), duration (longest first), skill-level required (highest first). The heuristics are applied one after the other until two tasks can be differentiated. The user prioritizes the heuristics.

Once the tasks are ranked, the algorithm attempts to allocate each task in turn. For a given task, different engineer tours must be considered as well as different positions in each tour. The algorithm ranks some of the possible pairs (engineer-position) using four heuristics: due dates' satisfaction or failure (satisfaction first), slack time before due date (least first), travel time required (minimum first), skill-level offered (lowest first). Once it has ranked the position-engineer pairs, it attempts each in turn. It backtracks whenever an insertion makes the schedule infeasible. It stops when it has scheduled all the tasks or when it has exceeded its cpu time budget. Because of the backtracking strategy, the algorithm may leave some tasks unallocated but usually it obtains a complete schedule easily since the number of tasks to preschedule is small and it uses long schedule horizons (at least a couple of days).

Simulated Annealing

The simulated-annealing algorithm applies the *relocate* operator by selecting tasks, engineers, and positions in a random fashion except for appointed tasks. For these tasks, it selects the engineer randomly but attempts the positions in the tour one after the other until the relocation makes the schedule feasible or all the positions have been exhausted. This increases the frequency of successful moves given the tight time constraints imposed by the time windows of appointments. In any case, the algorithm accepts a task relocation if the schedule is feasible and either the cost of the schedule has decreased or it has increased and the increase is accepted using simulated annealing's acceptance probability.

The simulated-annealing algorithm tries to reuse the provisional schedule it generated in the previous run so as to maintain schedule stability between successive runs. It simply replicates the previous schedule and combines it with the preschedule before starting the search. This also saves processing time since the search does not start from an empty schedule and, if few changes have affected the problem since the last algorithm's run, this starting point should already be of good quality.

Cost Function

The cost of a schedule is defined by a weighted sum of measures: quality of service, travel time, work in overtime, preferences on assignments, and so forth. Each measure is itself a weighted sum of measures over tasks or engineers. Each individual measure, say the quality of service for a task, is obtained by evaluating a normalized function that uses the attributes of the entity, for example, customer type, work category, due dates for a task, overtime budget, or skill preferences for an engineer. The user sets the weights of the

cost function so as to achieve the balance he or she wants between the different criteria.

Constraint-Based Model

A constraint solver determines the feasibility of a schedule by checking the compatibility constraints that restrict the matching of tasks and engineers and the temporal constraints that restrict the positioning of tasks. We model these constraints by defining a constraint-satisfaction problem for each schedule generated (a constraint-satisfaction problem (CSP) is defined by a set of variables, a finite domain for each variable, and a set of constraints that restrict the values the variables may take either individually or collectively. A solution to a CSP is an assignment of values to all the variables that satisfy all the constraints). With this approach, checking the feasibility of a schedule amounts to proving that the associated CSP has a solution or not.

The CSP representing a schedule consists of two distinct subproblems: a CSP defining the compatibility constraints and a CSP defining the temporal constraints. In the compatibility CSP, we define one variable for each task to represent the engineer assigned the task. The initial domain of the variable is the whole set of engineers including the virtual engineer. The compatibility constraints reflect the skill-matching requirements and the access and time restrictions imposed by the schedule horizon, any time windows, the working hours of the engineer, the travel time between breaks, the business rules governing travel outside of paid time and work in overtime, and the rules on the modalities of task breaking.

The constraint solver solves these constraints before the search starts by removing the incompatible engineers from each domain. Skill-matching constraints and most time restrictions are easily solved. However, the solver has to determine the impact of engineer breaks on the windows of opportunity of the activities (allocations task-engineer). For each activity, it intersects the task and engineer time windows and computes the impact of each break on the resulting windows of opportunity. A break may prevent task execution, it may force the breaking of the execution, or it may have no effect at all. Consequently, the modalities of the execution vary in time: the activity may not be feasible at certain times, it may be carried out as normal at others, or it may have to be split over a break or a series of breaks. Ultimately, if no start time is feasible over the schedule horizon, the constraint solver considers the activity infeasible and removes the engineer from the domain of the task.

The remaining compatibility constraints are binary and reflect requirements that two tasks be assigned to the same individual but the individual is unspecified. Such constraints must be checked whenever one of the two tasks is relocated.

To sum up, most of the compatibility constraints are preprocessed, which reduces the search space (the search algorithms do not have to consider all the engineers when selecting a candidate for a task) and speeds up the search (the constraint solver does not need to check compatibility constraints during task relocations). The preprocessing procedure has a linear worst-case complexity $O(e \times t \times d \times w)$, where e is the number of engineers in the problem, t is the number of tasks, d is the number of days covered by the schedule horizon, and w is the largest of the maximum number of time windows per day for a task and the maximum number of breaks per day for an engineer.

Assuming the compatibility CSP is satisfied, we define the temporal CSP by associating four variables to each allocated task i:

—two variables s_i and e_i, representing respectively the start and end times of i; and

—two variables a_i and d_i, representing re-

spectively the arrival time on the site of i and the departure time from the site of i of the engineer assigned to i.

These variables are assigned the same initial domain, which is the discrete time interval defined by the schedule horizon. The grain of this interval is the one-minute grain used in Work Manager processes. The preprocessing stage reduces each domain by removing the dates that are inconsistent with the time constraints. This reduction results in a fragmentation of the initial interval into disjoint intervals.

The other temporal constraints are binary. First, we state a constraint between the start and end time of each activity to reflect the impact of breaks. Indeed, the overall duration of an activity increases if it is split over breaks and the extent of the increase depends upon the duration of the breaks and of any required travel time. The constraint solver computes this relationship during preprocessing and stores it using a sequential timetable. The timetable records the evolution of the activity's duration and other cost-related elements based on the activities' start and end times.

The other temporal constraints are:
—$a_i \leq s_i$ to express that the engineer must be onsite to start the activity i;
—$e_i \leq d_i$ to express that the engineer cannot leave the site of the activity i before completion;
—a travel constraint between d_i and a_j for every pair of successive activities i,j to express that the engineer has to travel between the sites (the travel time, that is the difference a_j-d_i, is variable because the journey may be disrupted by breaks);
—$s_i = s_j$ to express that the activities i and j are parallel parts of a single job; and
—$e_i \leq s_j$ to express that the activities i and j are sequential parts of a single job.

To sum up, the temporal CSP is a binary CSP with fragmented integer domains that is implemented using intervals, timetables and procedural definitions. The fundamental feature of this problem is its tractability. Indeed, all the constraints fall in the class of binary min-closed and max-closed constraints, and min-max-closed binary constraint networks can be solved in polynomial time [Jeavons and Cooper 1995] using arc-b-consistency. In other words, we can determine the feasibility of a schedule whenever a task relocation is attempted in polynomial time by enforcing arc-b-consistency on the temporal constraint network.

Arc-b-consistency is enforced by propagating constraints on domain bounds only. Applied to a min-max-closed binary CSP, it either proves that the problem has no solution or it reduces the domains of the variables such that the domain minima are guaranteed to be a solution to the problem as well as the domain maxima. As for the temporal CSP, the minima of the start-time variables (s_i) provide the earliest feasible start times for the tasks in the schedule while the maxima of the end-time variables (e_i) provide the latest feasible completion times for the tasks.

In our case, the worst-case time complexity of arc-b-consistency enforcement is $O(h \times t)$, where h is the size of the schedule horizon and t is the number of tasks allocated. Given that h is the duration in minutes of the schedule horizon (time is represented with a one-minute grain) and that the schedule horizon typically spans several days, we clearly cannot afford to hit this worst case during the search.

The worst-case scenario occurs if and only if the constraint graph contains a temporal cycle: for example, i before j, j before k, and k before i. In our problem, the interdependent tasks (involved in line-ups or sequences) are the only ones responsible for the occurrence of cycles. This is why we have decomposed the search into a two-stage process, in which the first stage is prescheduling these tasks. Once prescheduled consistently, the interdependent tasks are made "transparent" to

the simulated-annealing algorithm by preventing their relocation. In this way, we create no cycle during the simulated annealing's run, and the feasibility computations remain acceptable. In fact, the worst-case time of arc-b-consistency enforcement in this configuration is $O(t)$, while its average time complexity is on the order of t/e, the average tour size. Also the tree-search algorithm rarely creates cycles when scheduling the interdependent tasks thanks to the task-ordering heuristics.

Travel-Time Estimations

The travel time between two points is estimated by dividing the euclidian distance between the points with a travel factor. Distance measures are based on exact geographical coordinates. Travel factors are based on group codes (equivalent to postal codes) and reflect the current weather, traffic, and geographical conditions. Work Manager generates them using historical data and they can be adjusted manually to reflect current travel conditions. Although travel-time-computation methods using real-time transportation information systems are more accurate, this method guarantees fast computation, which is vital for the scheduling processes.

References

Brind, C.; Muller, C.; and Prosser, P. 1995, "Stochastic techniques for resource management," *BT Technology Journal*, Vol. 13, No. 1, pp. 55–63.

Garwood, G. J. 1997, "Work Manager," *BT Technology Journal*, Vol. 15, No. 1, pp. 58–68.

Jeavons, P. G. and Cooper, M. C. 1995, "Tractable constraints on ordered domains," *Artificial Intelligence*, Vol. 79, No. 2, pp. 327–339.

Kirkpatrick, S.; Gelatt, C. D.; and Vecchi, M. P. 1983, "Optimisation by simulated annealing," *Science*, Vol. 220, No. 4598, pp. 671–680.

Laithwaite, R. H. 1995, "Work allocation challenges and solutions in a large-scale work management environment," *BT Technology Journal*, Vol. 13, No. 1, pp. 46–54.

Laporte, G. and Osman, I. H. 1995, "Routing problems: A bibliography," *Annals of Operations Research*, Vol. 61, pp. 227–262.

Lesaint, D.; Azarmi, N.; Laithwaite, B.; and Walker, P. 1998, "Engineering Dynamic Scheduler for Work Manager," *BT Technology Journal*, Vol. 16, No. 3, pp. 16–29.

Simpson, J. A.; Noble, B. F.; Egan, D.; Morton, T.; Richards, T.; and Burstin, M. 1995, "Experience in applying OR techniques to the solution of practical resource management problems," *BT Technology Journal*, Vol. 13, No. 1, pp. 38–45.

Tsang, E. 1993, *Foundations of Constraint Satisfaction*, Academic Press, London and San Diego.

Kevin Bradley, BT UK Operations Manager, said, "Dynamic Scheduling provides a much more stable work allocation mechanism. It has the ability to look several days into the future which gives us much more benefit . . . The ability of this system to allocate high priority work is frankly awesome."

Ian Smith, Managing Director, BT UK Customer Service, said, "Business customers demand that we repair today's fault today. We set ourselves a task of achieving that more than 90% of the time. Dynamic Scheduling has enabled us to do that. I am grateful for the fact that it's done it."