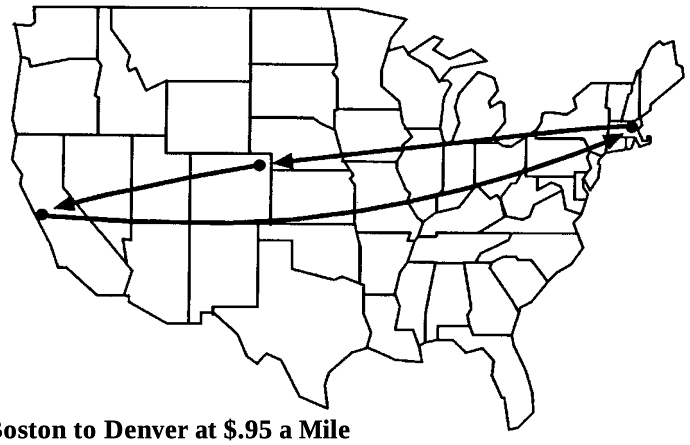
National Dispatch Router: A Multiparadigm-Based Scheduling Adviser

From Prototype to Production System

Janet Rothstein

The national dispatch router (NDR) system is an application of AI technologies to a continuous mileage scheduling program. This system is the only such system we are aware of. It is written in Knowledge Craft and has been in production use since May 1988. NDR saves Digital Equipment Corporation 10 percent of its Continuous Mileage Program shipping costs in addition to providing several other benefits. The system also represents two successful technology transfers. The first was from a university, where the initial proof of concept was performed to Digital's Applied Intelligent Systems Group (AISG), and the second was from AISG to the user's Information Systems (IS) group, which has maintained the system for the past two years.

Send Trucks as Far and Fully Loaded as Possible



1. Boston to Denver at $.95 a Mile
Denver to San Jose at $.95 a Mile
2. Boston to Denver to San Jose at $.65 a mile

Figure 1. Route Map.

Problem Definition

One of the functions of Digital's U.S. Area Distribution (USAD) organization is to schedule ground transportation of shipments within the continental United States. In this capacity, Digital participates in a Continuous Mileage Program with external shipping companies. The main principle of this program is that the per-mile cost of transporting goods decreases as the distance a truck travels in a single trip increases, providing various constraints are met (figure 1). Digital's primary goal in participating in a Continuous Mileage Program—and the goal of NDR—is to reduce transportation costs without reducing customer service.

Under the Continuous Mileage Program, requests to move shipments are processed by a central dispatcher, who arranges with shipping companies to transport goods to the appropriate destinations. The shipping must be done as economically as possible, and the dispatchers must be ever mindful of the customer's service requirements. This task is complicated by the various conventions used by the trucking companies when determining the cost per mile to be charged. For example, no truck can be idle for more than 24 hours, or the trip is considered two trips ,and the mileage is not accrued. Similarly, if a truck returns to its point of origin and completely unloads, the trip is considered ended. In any case, the maximum mileage that can be counted for any one trip is 7500 miles. These requests must be addressed in the context of a truck being able to travel only 450 miles a day unless there are two drivers; however, the extra driver involves an additional cost to Digital. In addition to these constraints, the dispatcher must also consider the service requirements of the customers—when a shipment is ready to be picked up and when it must be delivered.

The list of constraints continues. The main point is that this scheduling task can become complex, particularly when one considers the number of facilities to be serviced and, therefore, the number of trucks on the road at any given time. The dispatcher must also be able to quickly respond to uncontrollable factors (such as mechanical breakdowns and inclement weather), which necessitate the redesign of existing schedules.

Business Prior to the National Dispatch Router

In the past, Digital had one dispatcher, who along with his backup, performed all the dispatch functions by hand. The process consisted of logging all the shipments and the way they were scheduled onto dispatch cards. As the calls to move goods came in from around the country, the dispatcher would look at his hard-copy records with all the existing contracts (trucks on the roads) and try to determine if any of these trucks could handle the new shipment; if not, a new truck would be ordered. The resulting schedules would then be updated as needed.

From the start, it was obvious many benefits were to be gained by automating this task: (1) assisting the dispatcher in determining the best ways to schedule shipments, thereby resulting in more cost-effective schedules; (2) enabling a novice dispatcher to use the system to schedule trucks when the experienced dispatcher was unavailable; and (3) decreasing new hire training time by using the system as an aid when teaching the complex task of scheduling under the Continuous Mileage Program.

Early Project History

With these factors in mind, the NDR project began in late 1985 as a research effort to determine if there were any available software packages well suited to this problem. Because none could be found, a research effort was started with Carnegie-Mellon University (CMU) to develop a prototype scheduling assistant for the Continuous Mileage Program.

The prototype developed by CMU was a knowledge-based system, serving mainly as proof of concept. It consisted of a knowledge representation covering the basic entities, a limited scheduling rule set, and a specific search method to determine and evaluate different scheduling alternatives (Husain & Reddy 1987). Although by no means complete, the prototype served an extremely valuable role in the overall development process.

Technology transfer from CMU to AISG took place over a four-week period. During this time, the researcher worked directly with the Digital developer on enhancing and debugging the prototype as well as demonstrating its capabilities. We find we are far more successful if the person or group receiving the new technology works with the individuals transferring the technology prior to completing the work. This apprenticeship approach results in the technology recipients being far better prepared to take charge of the new work than if we simply dropped the prototype in their laps, expecting the recipients to continue development or just support the product.

Paving the Path to Successful Deployment

While the prototype was being transferred to AISG, it was necessary to reaffirm the business organization's commitment to the project. This reaffirmation involved running tests with the prototype to demonstrate its potential to succeed with the task at hand. In spite of the prototype's shortcomings (such as having a rough interface that could not directly be run by the users, being extremely slow, and overlooking several aspects of the problem), it was still of the utmost importance to present the system's output to the program manager and the users. Making these presentations at such an early stage in the project helped us overcome the users' initial skepticism about whether the system would be able to help with a task this complicated.

Once the prototype was transferred to AISG, it was necessary to work through many issues to move from prototype to production system. These tasks involved making enhancements to the system's speed, interface, and intelligence as well as working closely with USAD to ensure the system's smooth integration into its business process (in other words, to change the way the users conducted their business). It is not enough to address the technical issues and then assume the system will be used. By the time an application is completed technically, it should also be well on the way to being a part of the business process it is designed to address.

Three factors enabled us to begin this integration task early in the project. First, we had a working prototype capable of demonstrating the potential to eventually succeed with the task at hand. Second, we involved the users in the system design as soon as possible. These users are the ones who know their task the best; thus, their input is invaluable in defining and then fine tuning the system. Making this fact clear to the users gives them control over the system and ensures that the system will be one they are comfortable using. Along the same lines, it proved useful to implement their requests as quickly as possible, reinforcing the fact that their input was indeed valued. Finally, it was essential to have a dedicated program manager from within the user's organization, who had the influence to say that use of the system should be incorporated into the users' job responsibilities and be considered part of their job reviews. Coming from an outside organization, this task is something the system developer was incapable of doing on her own (Intelligent Systems Technologies Group, 1985).

During AISG's development of the system, additional knowledge-acquisition sessions (or interviews) were held with the users, resulting in an increase in the number of scheduling heuristics (rules of thumb used to derive schedules) and the generation of more economical schedules. The knowledge representation was extended to incorporate new aspects of the problem. The entire search method was redesigned to improve the quality of the schedules as well as greatly reduce the processing time (the original prototype had to run overnight to process a batch of shipments). Throughout this process, the prototype was used as a tool in the knowledge-acquisition sessions. It allowed the users to clearly identify aspects of the problem that were completely missing or incorrectly implemented or in need of fine tuning. Because the intricacies of the problem were not well understood a priori, this stepwise development was essential to the successful completion of NDR.

The final step in developing the production system was a three-month transition period, during which a member from USAD IS worked closely with the AISG developer to come up to speed on the project as well as the technology (which was also new to USAD IS). This technology transfer was accomplished through course work along with a joint effort to complete the remaining development work. By actually working with the developer on the remaining enhancements, the IS member was able to gain a first-hand knowledge of the workings of NDR prior to becoming responsible for its support. No amount of course work or system analysis alone could have achieved the same level of confidence or competence in this amount of time as the hands-on experience with the system developer. Support of the system and further enhancements have been handled by USAD IS for the past two years.

```
{{ carrier1
   instance: carrier
   name: carrier1
   print-name: "Frugal Freight, Inc."
   leg-id-increments: '01 '02 '03 '04 '05 '06 '07 '08 '09 10 '11 '12 '13
   distance-1dr:  500
   distance-2dr:  1000
   rate-structure: '(0 300 1.81) '(301 500 1.62) '(501 700 1.34)
   '(701 900 1.33) '(901 1100 1.30) '(1101 1300 1.29) …
   expedite: .31
   highest-run-number: 10
   print-order: 2}}
```

Figure 2. Carrier Schema.

Technical Implementation

NDR represents the application of AI (for knowledge representation and problem solving) to a complex vehicle scheduling and routing problem. Because of the complexities of the problem (not all of which were well understood a priori), it was necessary to use an extremely flexible development methodology. The problem was further suited to expert system technology in that it was well bounded, and an acknowledged expert existed (the dispatcher). At the same time, this expertise was not common (only a few people in the company were skilled at doing this function), so there was ample motivation for capturing the knowledge.

NDR was designed using a hybrid architecture, combining constraint-based reasoning, production rules, object-oriented programming techniques, and search techniques. The distinct advantage to such an approach is that specific techniques can be applied to the particular parts of the task for which they are best suited.

The concepts in the system (that is, contracts, shipments, facilities, carriers, and so on) are represented using a frame-based approach. This approach facilitates the inheritance of information and the use of object-oriented programming techniques. A frame representing a carrier (or shipping company) would include information such as the expedite charge (or the mileage charge for adding an additional driver) and the rate structure (each list in the rate structure consists of a range of mileages followed by the per-mile charge for any trip falling in that range.) (figure 2). A *shipment frame* contains information such as size, location and time it is to be picked up and dropped off, and the latest possible time it can leave without arriving late. Once the shipment is scheduled, its frame will also contain information, such as the time it will be picked up, the estimated time of arrival, a lateness factor (in other words, if it is going to arrive late and how late), the truck trip (contract) it was scheduled on, and the specific legs of the contract it was scheduled on. A *leg* of a contract is a part of the trip between any two stops. For example, if a truck is picking up two loads in New York, dropping one off in Colorado and continuing with the other to California, then this trip would consist of two legs. The first would be from New York to Colorado and the second from Colorado to California. The *contract frame* contains information such as the shipments scheduled on the truck, the legs that make up this trip, the carrier whose rate structure should be used in computing the charge for this trip, the method to be used in rating the contract, and any constraints that must be considered in evaluating the contract.

The contract frame and the frames of the legs making up the trip and the shipments scheduled on them constitute a complete description of a truck's trip. A *schedule* is a collection of trips (including all the frames associated with these trips). One can think of a schedule as a snapshot, representing one way the shipments can be scheduled in the context of all the trucks currently on the road. The schedules are generated using a beam search (see Winston p.96 for an explanation of beam search) in combination with a constraint-directed strategy for rating the alternate schedules at any given ply of the search. This process involves repeating two steps for every shipment that must be scheduled. In the first step, the scheduling heuristics generate alternative possible means of scheduling a shipment. In the second step, the beam search uses a constraint-directed strategy to rate the alternative schedules and determine which ones to further explore (Fox 1986).

The generation of alternatives is achieved by using heuristics to match a shipment to existing contracts. The simplest scheduling heuristic is to order a new truck to deliver the shipment, thereby starting a new trip (this heuristic does not require any matching). An example of a scheduling heuristic that is only slightly more complex is to add a shipment to an existing trip. In other words, if there is enough room on a truck already traveling between the shipment's origin and destination, and the truck is leaving and arriving at times that satisfy the requirements of the new shipment, then arrange for this truck to pick up the additional shipment.

Another relatively simple heuristic would involve extending an existing trip. Some of the conditions for this heuristic include (1) the trip's destination is the same as the shipment's origin; (2) the shipment will be ready to go within 24 hours of the truck's arrival, and it won't be delayed too long by waiting for the truck to arrive; (3) the trip's current destination is not the same as its origin; and (4) continuing to the shipment's destination will not cause the trip's total mileage to exceed the maximum mileage allowed.

The scheduling heuristics become more complex when we start to deal with diverting trucks from their original routes, adding a driver to allow due dates to be adhered to, and using various combinations of these types of scheduling strategies. These heuristics are more complex, in part because they can cause shipments scheduled later in the trip to be affected, and one must be careful to ensure that these shipments are not being negatively affected. The scheduling mechanism also contains information that allows the system to avoid creating contracts that would violate any of the conventions of the Continuous Mileage Program. Some examples of these are (1) a truck cannot remain idle for more than 24 hours and (2) if a truck returns to its point of origin and completely unloads, the trip is viewed as completed (in other words, the trip cannot be extended).

Once the scheduling alternatives have been generated for a particular shipment, the beam search evaluates the various constraints to determine which schedules to further explore. The number of schedules selected depends on the width of the beam search. The two constraints evaluated when rating contracts and schedules are to keep the cost as low as possible and minimize lateness. Each constraint has a function that is evaluated to determine its value, a relative weight, and a method for normalizing the value along a scale of -1.0 to 1.0 (where -1.0 is the worst, and 1.0 is the best). The cost constraint must account for the decrease in the cost for each mile as the length of a trip increases, the different rate structures for the different carriers, and any additional charges for expediting the shipment (by adding an additional driver). The lateness constraint deals with the total lateness of all the shipments on the contract or schedule in question. Once the values associated with the cost and lateness are normalized, the overall rating becomes the weighted average of these normalized values. The use of weights makes it easy to adjust the balance between the goals of keeping cost down and satisfying service requirements.

During the prototyping stage, the scheduling heuristics were implemented using CRL-OPS. This approach proved valuable in that not all the scheduling heuristics were well understood or well articulated at the outset. Therefore, the flexibility afforded by the rule-based programming style was invaluable in working with the experts to identify, implement, and fine tune the overall scheduling mechanism. Once the scheduling mechanism was fairly stable, it became apparent that the heuristics could be reimplemented using Lisp functions instead of CRL-OPS. This change was then made to achieve a significant improvement in run-time performance. Along the same lines, all the constraints were initially implemented using object-oriented programming techniques, and they were all evaluated after the scheduling alternatives were generated. These alternatives included constraints whose violation would result in an invalid contract (because of the violation of a convention of the Continuous Mileage Program). Once the scheduling mechanism was stable (and all the constraints were identified), those constraints whose violation would result in an invalid contract were integrated into the Lisp functions constituting the scheduling mechanism. This integration was done in such a way as to prevent the system from ever generating schedules containing invalid contracts. Although there is a clear trade-off here between modifiability and efficiency, the significant improvement in run-time performance, coupled with the relative stability of the scheduling mechanism, made the reimplementation an appropriate choice for this application. As previously mentioned, the flexible rule-based approach was crucial to the development effort because we were identifying and fine tuning the scheduling mechanism; however, the basic premises of the Continuous Mileage Program have not really changed over the past two years, so the maintenance issue has not been a problem. In fact, this piece of the scheduling mechanism has not required any further modification since it was put into production use. A final technique was to use CRL-Prolog for consistency checking as well as for querying the database about the “current state of the world” as viewed by the system.

Although NDR could not be considered a generic dispatcher in the realm of all possible types of scheduling problems, it could easily be adapted to other Continuous Mileage Program–type scheduling problems. As discussed earlier, the use of object-oriented programming techniques facilitates the modification or addition of factors that go into the evaluation (or rating) of the trips and schedules, and it is easy to adjust the balance between the emphasis on these factors. Additionally, the use of the object-oriented programming techniques makes it easy to modify, add, or delete instances of the various entities that are dealt with in NDR (that is, facilities, routes, carriers, and so on).

Scheduling Example

Suppose NDR is given three shipments. For the purposes of this example, CA, CO, and MA are facilities located somewhere in California, Colorado, and Massachusetts, respectively. The first shipment is 50 percent of a truck load and is going from CA to MA. It will be ready to go at 8:00 on 3/7/88, and is due by 8:00 on 3/11. The second shipment is also 50 percent of a truck and is going from CO to MA. It will be ready

Scheduling Heuristics... Extend An Existing Trip:

Example

- **Shipment: Ready at 8:00 on 3/11 in Boston
  Due at 13:00 on 3/13 in Colorado Springs**

- **Truck: Originating from Los Angeles
  Arriving in Boston at 13:00 hours on 3/10**

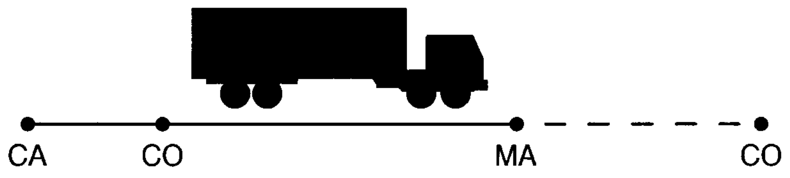
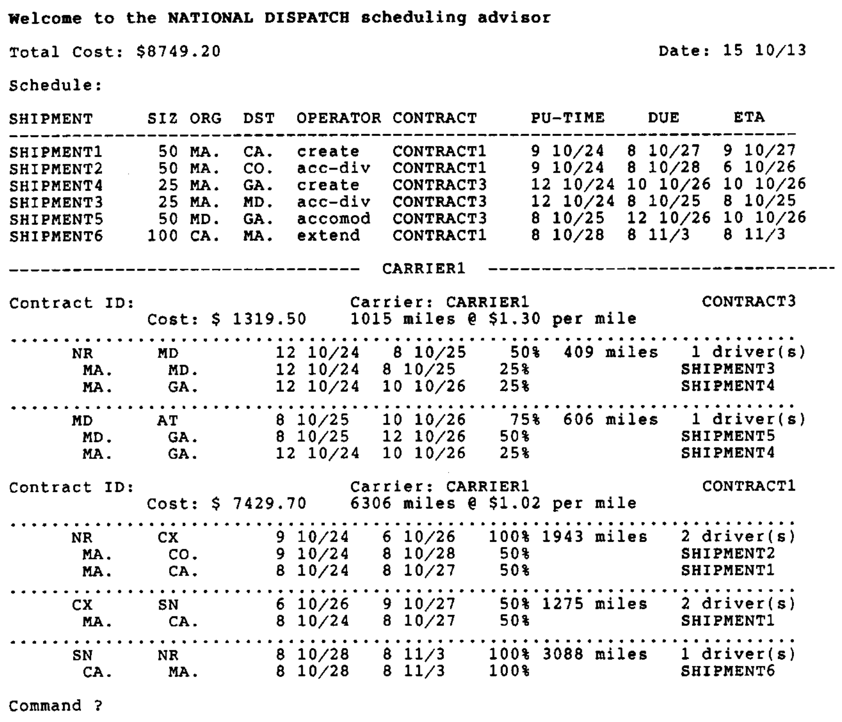


Figure 3. Scheduling Heuristics.

to go at 17:00 on 3/8 and is due by 8:00 on 3/14. The final shipment is 100 percent of a truck and is going from MA back to CO. It will be ready at 8:00 on 3/11 and is due by 8:00 on 3/15. In this example, NDR will start by creating a trip because no current trips can match any of these shipments. This first trip will go from CA to MA and will have a single load taking up 50 percent of the truck. Depending on the CA facility's operating hours, this trip might not begin until 9:00 on 3/7, and its arrival time at MA would be based on the distance between CA and MA. NDR would then have two options for scheduling the shipment from CO to MA. Because there is still room on the truck, NDR could schedule this shipment by diverting the truck to CO and picking up the shipment prior to continuing on to MA where it would drop off both shipments. It could also schedule it by creating a new trip to handle this shipment (creating a new trip is always an option when scheduling any shipment). As long as the beam width is greater than one, the system would continue exploring both of these possible schedules.

The final shipment (from MA back to CO) can then be scheduled in one of several ways. The simplest way would be to create a new trip. If NDR were exploring the schedule in which it created a separate trip for each of the first two shipments, the system could schedule the final shipment by extending either of these two trips (that is, because both of these trips end at MA, and either one could be there in time to pick up the third shipment, providing there are two drivers, extending either trip is a possibility). Alternatively, if the system is exploring the



```
Welcome to the NATIONAL DISPATCH scheduling advisor

Total Cost: $8749.20                                      Date: 15 10/13

Schedule:

SHIPMENT     SIZ ORG  DST  OPERATOR CONTRACT     PU-TIME    DUE     ETA
-----------------------------------------------------------------------
SHIPMENT1     50 MA.  CA.  create   CONTRACT1    9 10/24  8 10/27  9 10/27
SHIPMENT2     50 MA.  CO.  acc-div  CONTRACT1    9 10/24  8 10/28  6 10/26
SHIPMENT4     25 MA.  GA.  create   CONTRACT3    12 10/24 10 10/26 10 10/26
SHIPMENT3     25 MA.  MD.  acc-div  CONTRACT3    12 10/24 8 10/25  8 10/25
SHIPMENT5     50 MD.  GA.  accomod  CONTRACT3    8 10/25  12 10/26 10 10/26
SHIPMENT6    100 CA.  MA.  extend   CONTRACT1    8 10/28  8 11/3   8 11/3

-------------------------------- CARRIER1 --------------------------------

Contract ID:                Carrier: CARRIER1                CONTRACT3
             Cost: $ 1319.50    1015 miles @ $1.30 per mile
........................................................................
      NR      MD          12 10/24   8 10/25    50%  409 miles   1 driver(s)
       MA.     MD.        12 10/24  8 10/25    25%               SHIPMENT3
       MA.     GA.        12 10/24  10 10/26   25%               SHIPMENT4
........................................................................
      MD      AT          8 10/25   10 10/26    75%  606 miles   1 driver(s)
       MD.     GA.        8 10/25   12 10/26   50%               SHIPMENT5
       MA.     GA.        12 10/24  10 10/26   25%               SHIPMENT4

Contract ID:                Carrier: CARRIER1                CONTRACT1
             Cost: $ 7429.70    6306 miles @ $1.02 per mile
........................................................................
      NR      CX          9 10/24   6 10/26   100% 1943 miles   2 driver(s)
       MA.     CO.        9 10/24   8 10/28    50%              SHIPMENT2
       MA.     CA.        8 10/24   8 10/27    50%              SHIPMENT1
........................................................................
      CX      SN          6 10/26   9 10/27    50% 1275 miles   2 driver(s)
       MA.     CA.        8 10/24   8 10/27    50%              SHIPMENT1
........................................................................
      SN      NR          8 10/28   8 11/3    100% 3088 miles   1 driver(s)
       CA.     MA.        8 10/28   8 11/3    100%              SHIPMENT6

Command ?
```

Figure 4. National Dispatch Router Screen.
(All rates have been fabricated for purposes of publication.)

schedule in which the second shipment was scheduled by diverting the first trip, this trip could be extended to return to CO after dropping the first two shipments at MA (figure 3).

In this example, NDR would select the final alternative because it would be the most cost effective, and it would still get all the shipments in on time. Although figure 4 does not refer to this same example, it is an actual screen shot of NDR and shows what it looks like as NDR displays a schedule it just derived. In NDR's output, the line following the contract ID summarizes the contract's total cost, number of miles, and rate per mile. Note that the rate per mile excludes expedite charges, whereas these are included in the total contract cost.

Project Life Cycle

Digital's development of NDR took approximately two years. It is hard to break out the amount of time it took to successfully transfer the system into the field because of the stepwise, iterative development cycle that was used. Throughout the development cycle, efforts were made to ensure that once the system was completed technically, the transition to the field would be smooth and would simply look like a continuation of the process that had been taking place all along. This strategy consisted of frequent meetings with the users to ensure their input into the system's definition and to establish their control over it. Also, the rapid prototyping method, combined with an incremental development style, allowed some version of the system to operate through most of the development period. At first, the prototype was not particularly useful to the users except as a tool to help them define their requirements. However, as later versions of the prototype (with additional function) were installed, the users could see more and more value in it. By the time the production version was installed, the users were already used to running the system.

Deployment and Benefits

There were several criteria for successfully deploying the application. First, the system had to generate economical and correct schedules. Second, the users had to want to use it. Finally, the system had to be owned and maintained by USAD and its IS organization. Having met these criteria, the system has been in full production use since May 1988. Perhaps the greatest evidence of NDR's successful deployment is the enthusiastic praise the system continues to receive from its users. NDR is currently installed at the user site on a MicroVAX II running VMS, VAX Lisp, and Knowledge Craft. The dispatcher (or his backup) uses NDR on a daily basis. For the past two years, maintenance and additional enhancements have been handled by USAD IS. To date, the required maintenance to the system has been minimal, consisting primarily of updates to the system's database.

NDR is currently used as a stand-alone system and is judged to save Digital 10 percent of its scheduling costs within the Continuous Mileage Program. Several qualitative benefits are not included in this 10-percent figure: First, the advanced capacity planning of trucks for end-of-fiscal-quarter skews allows the dispatcher to order the trucks in advance, thereby increasing Digital's ability to obtain sufficient trucks to cover all their loads during these critical, high-volume times. Second, a modeling capability for establishing regularly scheduled runs helps the network planners negotiate more favorable contracts for Digital with the external carriers. Third, NDR minimizes prior critical dependency on a single dispatcher. Fourth, it increases new hire productivity and decreases training time. Fifth, the automation of a previously manual function results in greater efficiency, cost effectiveness, and consistency of information. Sixth, NDR assists in quick schedule redesign.

Conclusions

When discussing the deployment of an application, it is the business innovations that are of the utmost importance. The bottom line for any application is the degree of impact it has on the business it is supporting. As indicated throughout this chapter, NDR is innovative in several ways. First, it is the only system we are aware of that addresses the issues of a Continuous Mileage Program. Second, it is a prime example of successful technology transfer both from a university to a corporation and from an AI technology group to an operational organization. Finally, the system is written in Knowledge Craft using a multiparadigm approach to a continuous mileage scheduling problem.

Acknowledgments

I want to acknowledge Mitch Tseng whose vision kept NDR alive; Mike Kiskiel and Leila Habib for their guidance throughout the project; Dave Van Cleave for his editing assistance; Nizwer Husain for his work on the prototype at CMU; Paul Michaud for his efforts during the transfer of NDR from AISG to USAD IS; and, lastly (but definitely not least), our business partners, Greg Sullivan, Jim Salvi, Marianne Wentworth, and Al Millett, without whom there would have been no project.

References

Fox, M. S. 1986. "Observations on the Role of Constraints in Problem Solving." In Proceedings of the Annual Conference of the Canadian Society for Computational Studies of Intelligence, 172–186. Toronto, Canada: Canadian Society for Computational Studies.

Grant, T. J. 1986. Lessons for OR from AI: A Scheduling Case Study. *Journal of the Operational Research Society* 37(1): 41–57.

Hedberg, S. 1989. "AI in Scheduling. Where We Are in Scheduling and Where Should We Be?" *Spang Robinson Report on Artificial Intelligence* 5(3): 2–4.

Husain, N., and Reddy, Y. V. 1987. Continuous Mileage Vehicle Scheduling as Rule-Based Search. In *Artificial Intelligence, Expert Systems, and Languages in Modeling and Simulation,* eds. C. A. Kulikowski, R. M. Huber, and G. A. Ferraté, 129–137. Amsterdam: North-Holland, 1988.

Intelligent Systems Technologies Group. 1985. Guide to Expert Systems Program Management, Artificial Intelligence Guide Series, Digital Equipment Corporation.

Winston, P. H. 1984. *Artificial Intelligence.* Reading, Mass.: Addison-Wesley.