AI Magazine Volume 21 Number 4 (2000) (© AAAI)

# Ramp Activity Expert System for Scheduling and Coordination at an Airport

*Geun-Sik Jo, Jong-Jin Jung, Ji-Hoon Koo, and Sang-Ho Hyun*

■ In this project, we have developed the ramp activity coordination expert system (RACES) to solve aircraft-parking problems. RACES includes a knowledge-based scheduling system that assigns all daily arriving and departing flights to the gates and remote spots with domain-specific knowledge and heuristics acquired from human experts. RACES processes complex scheduling problems such as dynamic interrelations among the characteristics of remote spots-gates and aircraft with various other constraints, for example, customs and ground-handling factors, at an airport. By user-driven modeling for end users and near-optimal knowledge-driven scheduling acquired from human experts, RACES can produce parking schedules for about 400 daily flights in approximately 20 seconds; human experts normally take 4 to 5 hours to do the same. Scheduling results in the form of Gantt charts produced by RACES are also accepted by the domain experts. RACES is also designed to deal with the partial adjustment of the schedule when unexpected events occur. After daily scheduling is completed, the messages for aircraft change, and delay messages are reflected and updated into the schedule according to the knowledge of the domain experts. By analyzing the knowledge model of the domain expert, the reactive scheduling steps are effectively represented as the rules, and the scenarios of the graphic user interfaces are designed. Because the modification of the aircraft dispositions, such as aircraft changes and cancellations of flights, is reflected in the current schedule, the modification should be sent to RACES from the mainframe for the reactive scheduling. The adjustments of the schedule are made semiautomatically by RACES because there are many irregularities in dealing with the partial rescheduling.

The aircraft-parking problem is a scheduling problem that entails assigning every arriving and departing flight to in-terminal gates and remote gates, satisfying various demands. It is a kind of scheduling problem that also includes job-shop scheduling if we consider the (remote) gates as machines and incoming flights as jobs. In addition, this problem entails characteristics of temporal reasoning mechanisms. Theoretically, this problem belongs to the NP-class of problems in computational complexity. That is, if we try to assign *m* flights to *n* remote parking spots or gates (often referred to as bridges), then a nonpolynomial number of combinations $(m!)^n$ are possible. The scheduling becomes more dynamic and difficult if the frequency of arriving and departing flights increases, and unexpected events occur during the operational stage in a given time period. Increased air traffic and customs requirements also reflect difficulties in producing parking schedules. In practice, professional schedulers manually make the schedules once a day, which requires domain-specific knowledge, experience, heuristics, and a considerable amount of time and tedious paperwork to complete.

Traditionally, researchers have used mathematical programming techniques to solve these kinds of problem. However, it is difficult to model constraints and domain knowledge with only mathematical variables. In addition, there can be serious problems of remodeling and processing when unexpected events occur for large-scale practical problems. Recently, many researchers have proposed using AI techniques such as constraint-directed reasoning, expert systems, and the constraint-satisfaction problem (CSP) to solve these problems (Jo, Jung, and Yang 1997; Prosser 1993; Fox 1987). AI techniques provide more flexible and expressive power than mathematical programming in modeling a complex scheduling problem. Comparison between integer programming and AI is well described in another work (Dhar and Ranganathan 1990). In addition, modeling the reactive scheduling with integer programming is beyond our ability to formulate because of the dynamic addition of constraints and the necessity of the partial solutions.

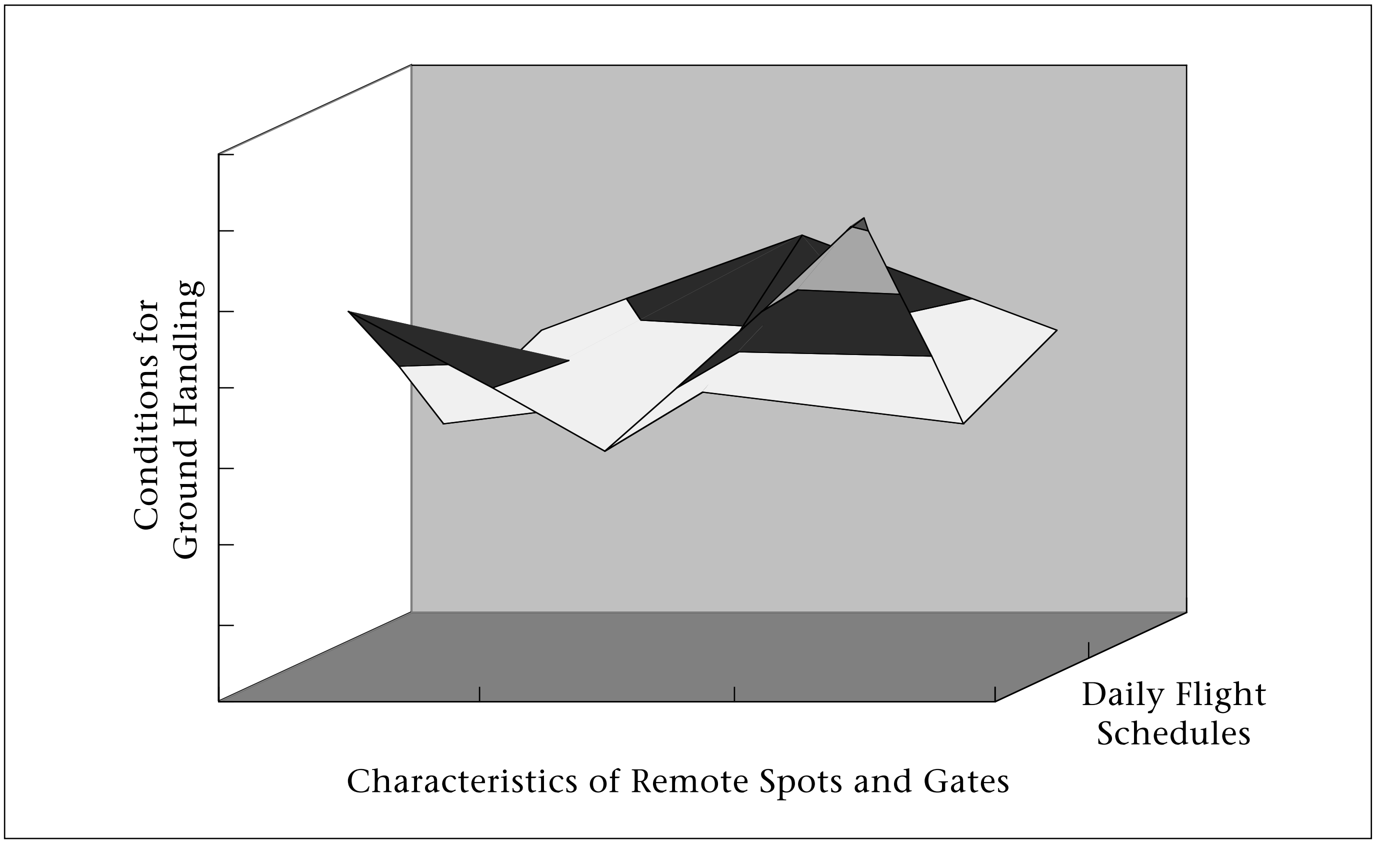
Conditions for Ground Handling
Daily Flight Schedules
Characteristics of Remote Spots and Gates

*Figure 1. The Feasibility Region in Gate Allocation.*

Many expert systems have been presented in these areas. Practical expert systems have developed in the airline industry. American Airlines (1993) developed GATEMANAGER to effectively manage busy air traffic and resources. The GATES system from Texas State University controls gate assignment and tracking at New York's John F. Kennedy airport (Brazile and Swigger 1988). Knowledge Engineering, a Singapore-based company, has successfully developed a constraint-based gate-allocation system using ILOG for Changi airport in Singapore (Berger 1995).

## Description of RACES

The ramp activity coordination expert system (RACES) assigns daily flights to the gates and remote spots with domain-specific knowledge and scheduling heuristics. Using domain-filtering techniques, we can remove the inconsistency in the domain for variables and confine the search space. To find a user-driven optimal solution, RACES utilizes an efficient heuristic scheduling method to satisfy constraints. RACES produces a near-optimal schedule with considerations for the flight schedules, aircraft type, characteristics of remote spots, and conditions for ground handling. Aircraft have to be assigned to adequate remote spots and gates with the satisfaction of given constraints. In addition, gates and remote spots are distinguished by size and hydrant facilities. RACES can be viewed as a three-dimensional (3-D) constraint solver, as in figure 1, and it also maps 3-D spaces into the 2-D spaces, which is represented in the form of a Gantt chart. RACES makes user-dependent, near-optimal schedules satisfying the given constraints with domain-specific knowledge and heuristics. RACES produces the Gantt chart, which represents the daily parking schedule a day beforehand.

In terms of constraint solving, there are three different types of constraint to satisfy: (1) the strong-hard constraint, (2) the weak-hard constraint, and (3) the soft constraint. The *strong-hard constraint* has to be satisfied during the scheduling process. If this constraint is violated, then the solution is no longer valid. The *weak-hard constraint* can be violated in specific environments. This constraint can be violated by interacting with users, but not by RACES. The *soft constraint* is applied to specific flights and specific times. During scheduling, a soft constraint is checked, and there is an attempt to satisfy the constraint if possible. If not, this type of constraint can be relaxed by RACES. RACES can be divided into two different knowledge-based systems: The first system is to generate a one-day schedule one day beforehand, which is described in the next section. The second system is to adjust a schedule during an operational day. More technical details with examples for the management of constraints and the representation of domain knowledge are presented in Jo, Jung, and Yang (1997).

## Initial Schedule Generation

After completing our documentation of knowledge acquired from domain experts, we had about 20 pages, which did not include the database description. Moreover, the knowledge itself was too domain specific for the average person to understand.

In this section, we describe the scheduling strategy being deployed in our system. In figure 2, RACES produces the Gantt chart, which represents the daily parking schedule for a day beforehand, with today's schedule for coordination.

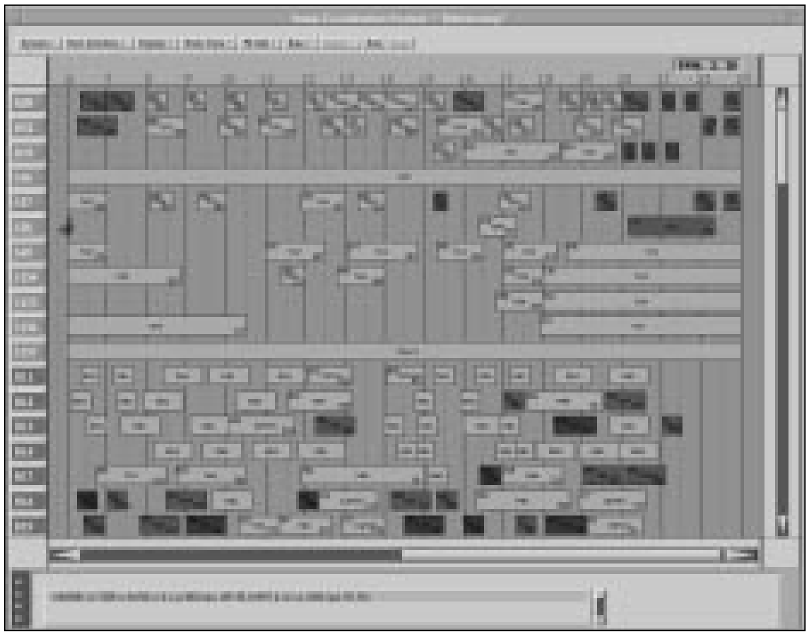

*Figure 2. Generation of the Initial Schedule in RACES.*

### Consistency by Domain Filtering

One scheduling procedure is responsible for binding continuous time values to the discrete time variables to satisfy constraints for the time restriction. To deal with the continuous time domain, we break down the continuous time values into discrete time elements. We also classify all the available remote spots and gates with time keys. When the system processes scheduling, it filters domains and removes elements violating constraints in three steps. An example of the domain filtering process for flight schedules is shown in figure 3.

In the first step, RACES filters domains with the knowledge and constraints of various aircraft. Second, the system filters domains with the knowledge and constraints of towing. Finally, the system filters domains with the knowledge and constraints of available parking spots. As a result of the process, RACES can prune the search space significantly.

### Knowledge-Driven Near-Optimal Scheduling

We consider the optimal solution in terms of the user's benefit. An important factor is to minimize the number of stand-by flights that are not yet assigned to the gates-spots because of congestion at the airport. During scheduling, RACES also tries to allow for the least number of towings at the airport. If the system fails to assign flights to the gates, the size of the aircraft must be taken into consideration. If stand-by flights involve relatively large aircraft, it is difficult for users to assign them manually after the automatic schedule is produced because the number of large spots is usually inadequate.

There are two heuristic scheduling methods in RACES: The first is the time-focused method using best-fit assignment in terms of the time span for parking. The second is the aircraft-size focused method. Each method has advantages and disadvantages in finding user-driven optimal solutions given that one method conflicts with the other method at certain points during the scheduling process. In our work, to avoid conflicts between the time-focused method and the size-focused method, we have empirically found and exploited a trade-off point between the two. A detailed description of these heuristic scheduling methods is presented in Jo, Jung, and Yang (1997).

## Knowledge-Based Reactive Scheduling

When the real operational day is reached after the scheduling has been completed, unexpected events can occur as the environment changes. If sudden changes in schedules occur, such as the delay of an aircraft or the change of aircraft for certain flights, then schedules must be adjusted. In adjusting the daily schedules, the domain-specific knowledge from the domain expert is encoded into RACES.

### Expert Model in Reactive Scheduling

The meta-agent in figure 4 knows which agent

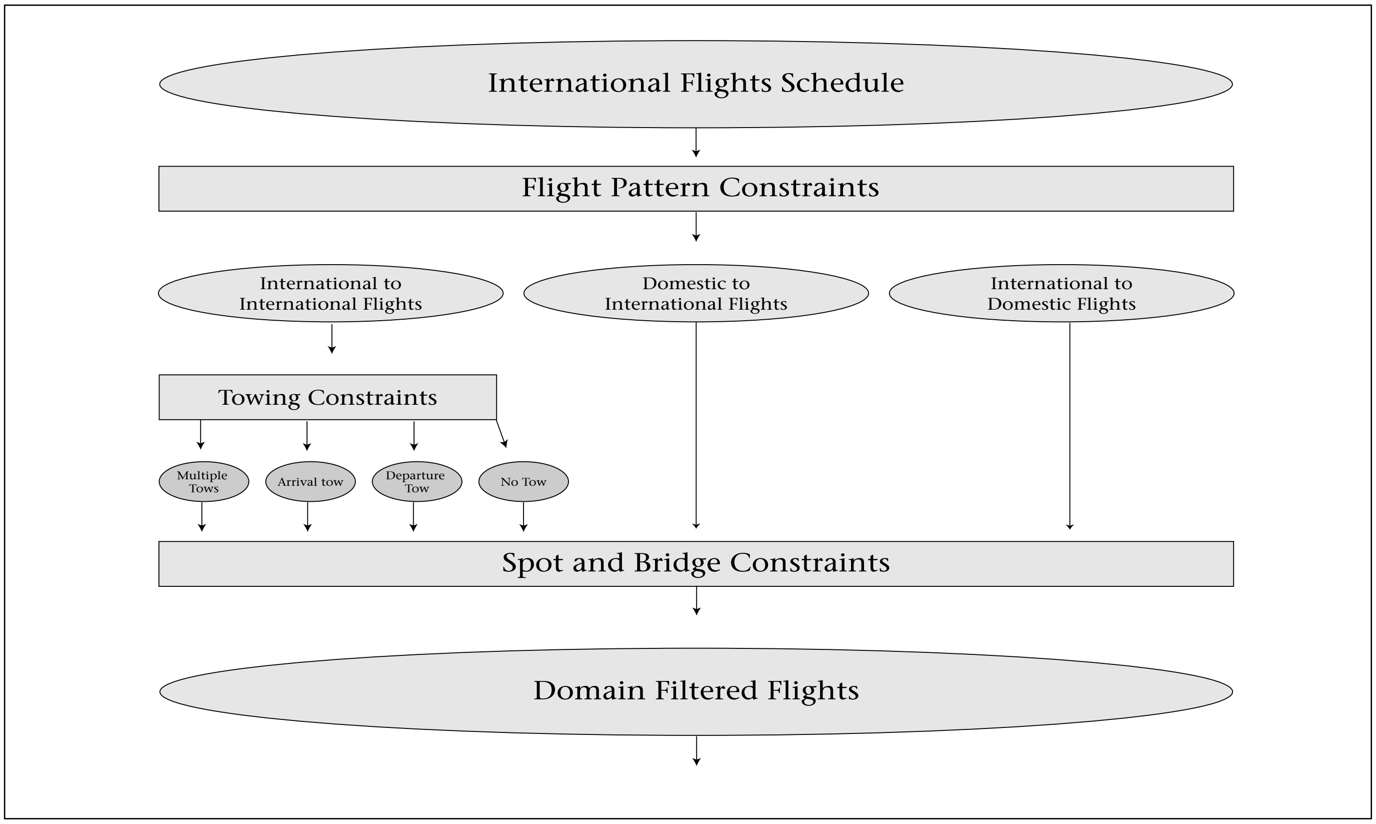
International Flights Schedule
Flight Pattern Constraints
International to International Flights
Domestic to International Flights
International to Domestic Flights
Towing Constraints
Multiple Tows
Arrival tow
Departure Tow
No Tow
Spot and Bridge Constraints
Domain Filtered Flights

*Figure 3. Domain Filtering before Scheduling.*

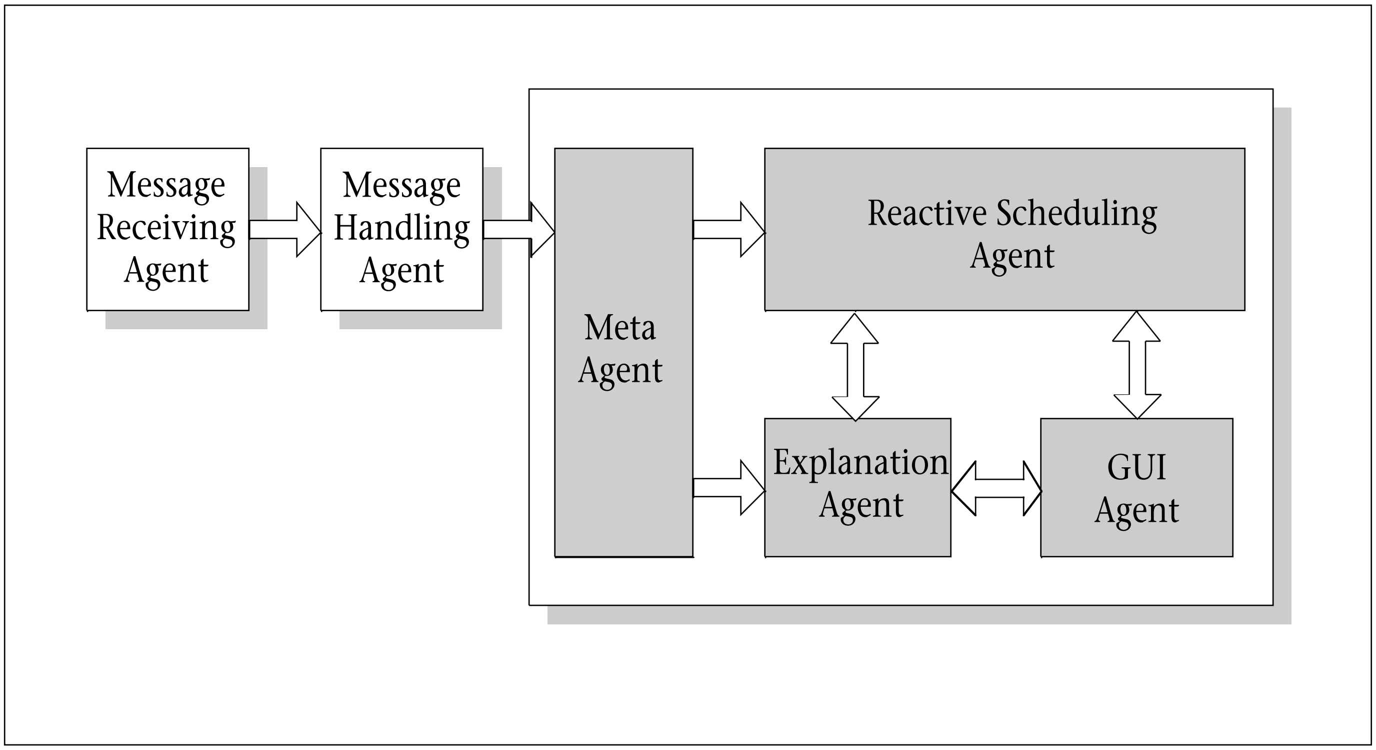

*Figure 4. Expert Model in Reactive Scheduling.*

is activated depending on the message from the message-handling agent. Although the role of each agent cannot be summarized because of page requirements, message-handling agents and reactive scheduling agents are explained in brief.

The task of the *message-handling agent* is to check *slings* (a sequence of flights that an aircraft should make), detect cycles, and group the related messages together. The messages for aircraft change come into RACES in a unit of flights that are not in an ordered form but, rather, in mixed forms. Therefore, the message-handling agent has to rearrange them into a unit with the registration number for an aircraft by their scheduled time. Then, the agent can check the fallacy of a sling order or an omitted sling. Generally, one aircraft makes a sequence of flights during a given day. A sling consists of a flight schedule that reflects a continuous time domain.

The task of the *reactive scheduling agent* is to reschedule according to messages received as the environment changes during the operation. This task can be divided into two different operations: (1) creating new standby bars and (2) assigning these bars. The most important task of the reactive scheduling agent is to assign new stand-by bars to the Gantt chart. This process requires complicated and delicate domain knowledge. A variety of adjusting rules are implemented for these processes.

### Interactive Graphic User Interfaces

The change of aircraft occurs in a case when an aircraft cannot operate the flight that it is scheduled to run because of a delay in operations, aircraft repair, or a breakdown. In this case, the flight will be operated by another aircraft; we call this situation an aircraft change (figure 5).

In figure 5, the capital letters stand for the HL number (the registration number for an aircraft), and the lowercase letters stand for flight numbers. In the original scheduling, aircraft *A* was supposed to arrive as flight *a*1 and depart as flight *a*2. Similarly, aircraft *B* was supposed to arrive as flight *b*1 and depart as flight *b*2 and, analogously, *C* as *c*1 and *c*2. If we rotate aircraft *A* with *B, B* with *C*, and *C* with *A* circularly, *A* will arrive as *a*1 and depart as *c*2 and, analogously, *B* as *b*1 and *a*2 and *C* as *c*1 and *b*2. As we showed in figure 5, each flight has a time slot for an arrival and a departure. If we change the schedule of flights for some aircraft, the length of the original solution bar can be changed, which means that we need to adjust the solution of the initial schedule. The aircraft rotation can occur not only between two aircraft but also among three, four, or more aircraft. They also occur in a cycle, as we see in figure 5. Because the fully automatic adjustments are not consistent from time to time and they are quite complex, these adjustments are too difficult to automate. The intelligent interactive graphic user interface (GUI) is used for manual adjustments.

## The Relationship between RACES and Other Systems

Figure 6 illustrates the relationship between RACES and other systems. RACES receives flight schedule data from the Computer Center to make an aircraft parking schedule. After RACES performs the scheduling, it transfers scheduling results to KATSCO (Korea Air Terminal Service Company) and SELMC (Seoul Maintenance Control Department). KATSCO is responsible for performing the ground tasks of an aircraft, for example hydrating, cleaning, and towing. With the scheduling result obtained from RACES, KATSCO can prepare their tasks efficiently. Also, SELMC is responsible for the maintenance of an aircraft. Maintenance personnel check device problems on an aircraft. They need the data from RACES to make their maintenance schedule. Actual towing of an aircraft needs the cooperation between KATSCO and SELMC. The aircraft-parking schedules from RACES allow them to prepare to tow an aircraft before they perform actual towing.

## Development History

Until 1995, the ramp activity coordination system for Korean Air Lines (KAL) at Kimpo Airport in Korea was managed by human experts using manual scheduling. To maximize the utilization of the resources of the airport, KAL and Expert Systems Lab at Inha University decided to develop the expert system with the experiences and heuristics of domain experts at KAL. Three domain experts and a system engineer from KAL and five graduate students at Inha University participated in this project between 1995 and 1997 under the supervision of Geun-Sik Jo. It was approximately a $125,000 project for Inha University, which excluded wages for three domain experts and a system engineer and expenses for an office and other utilities, which were paid by KAL. It took three months to make a prototype of RACES to convince KAL to pilot the project. Then, after about six months, we were able to complete the initial scheduling part of RACES. Most of the developing time was spent in reactive scheduling,

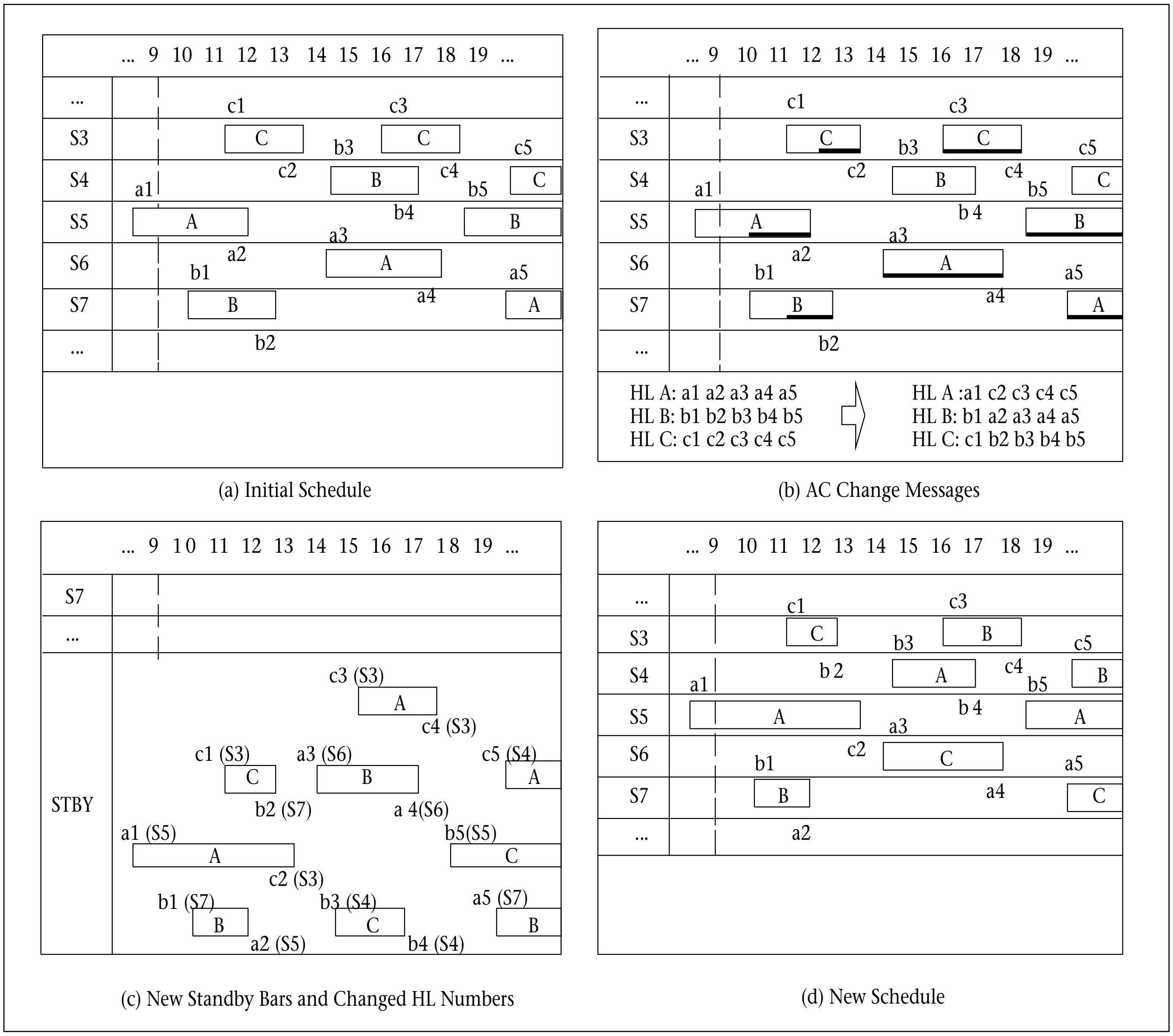

*Figure 5. The Steps for Adjusting an Initial Schedule.*

which took about nine months. For another four months, networking routines needed for the mainframe to integrate with RACES, which, in turn, interacts with other computers, were added to create a real environment. To evaluate the system, two end users tested the results against real flight schedule data for the previous 120 days. Their test results were accepted by domain experts so that RACES could be used for real environments. For the maintenance of RACES, we explained source codes and trained a system engineer to do the maintenance job himself. At the present time, a system engineer from KAL is responsible for maintaining RACES, and he is doing well. We think that the declarative features of Prolog have made it possible for only one person to do the maintenance job. In addition, at the developmental stage, we designed and implemented menus and submenus to prepare for the future extension and update of the knowledge base.

RACES was written in CHIP (constraint handling in Prolog) 5.0 under a UNIX operating system on an HP/712 machine. To retrieve and store the information on flights, bridges, and remote spots, the ORACLE database management systems was used, with an SQL interface from the Prolog code provided in the CHIP system.

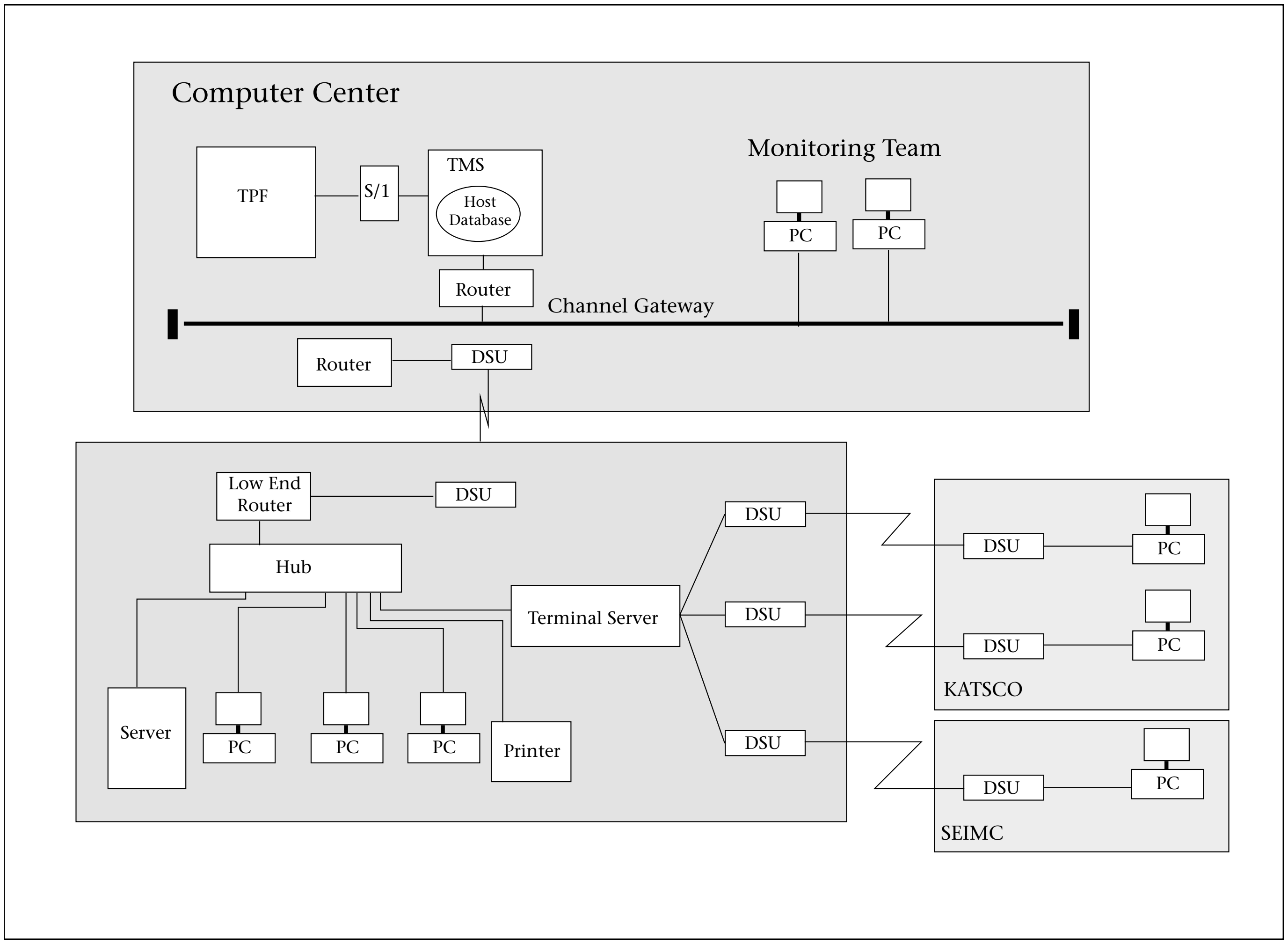

*Figure 6. The Relationship Diagram with Other Systems.*

## Deployment Process

We developed RACES in CHIP, which consists of about 50,000 lines of Prolog code with about 70 GUI menus. RACES solves the problem using methods similar to a human expert's problem-solving procedures. We represented and processed the domain-specific knowledge and experiences. When the system processes scheduling, it can prune the search space using a domain-filtering technique. RACES produces a user-driven optimal schedule using trade-off scheduling heuristics. To test the accuracy of the system, we implemented RACES with the daily operational data of an actual airline company for about 120 days, and the results were analyzed by domain experts. RACES has been approved by, and continues to receive, the approval of domain experts.

The system described in this article was successfully deployed at Kimpo Airport in Korea. RACES has been used at Kimpo airport by KAL since 1997. The controllers at the operational control center at KAL are now using this system for monitoring and controlling the assignment of remote spots and bridges. To date, the ground controllers using this system to actually tow aircraft, assign buses, and do other ground work are able to interact well with persons at the Operations Control Center. As long as the airport is in operation, this system should run almost 24 hours a day. When the flight information display system (FIDS) was connected to RACES, the time that the aircraft spent waiting to park after landing was greatly reduced.

RACES currently has the ability to reschedule in approximately 70 percent of the cases in a real environment situation. Reactive scheduling is one of the most important topics for researchers to use the scheduling systems in practice. However, when the system processes the rescheduling by itself, we often find that some adjustments are not adequate. Therefore, some of the adjustments should be done in cooperation with the users by adjusting the schedule manually through the GUI. Interactive GUI in RACES plays a supporting role by helping users make the right decision.

## Benefits of RACES

RACES made the work paradigm shift from manual to automatic scheduling in the ramp activity management that required domain-specific human knowledge and heuristics. It is clear that the initial investment is returned within a year after deployment. However, the following benefits are not currently measurable in terms of money: (l) the time and cost involved in scheduling are drastically reduced; (2) real-time adjustment for unexpected events and weather conditions is provided for; (3) interactive GUI in RACES plays a supporting role by helping users make the right decision at the right time; (4) aircraft waiting time for parking after landing is reduced; (5) high-quality passenger service is provided because RACES gives prior information about aircraft parking status, thereby, ensuring the quick movement of an aircraft after landing; (6) objective verification of ramp activity management is possible; and (7) RACES can perform the operations necessary to maximize the use of ramp activity.

### Acknowledgments

This project was successfully completed with the help of Korean Air. This work was also partially supported by the development program for the exemplary schools in information and communication from the Ministry of Information and Communication.

### Bibliography

American Airlines. 1993. GATEMANAGER. Sabre Inc., Fort Worth, Texas.

Berger, R. 1995. Constraint-Based Gate Allocation for Airports, ILOG Solver and Schedule. Paper presented at the First International User's Conference, 10–11 July, Abbaye des Vaux de Cernay, France.

Brazile, P., and Swigger, K. M. 1988. GATES: An Airline Gate Assignment and Tracking System. *IEEE Expert* 3(2): 33–39.

Croker, A. E., and Dhar, V. 1994. A Knowledge Representation for Constraint-Satisfaction Problems. *IEEE Transactions on Knowledge and Data Engineering* 5(5): 740–752.

Dhar, V., and Ranganathan, N. 1990. Integer Programming versus Expert Systems: Experimental Comparison. *Communications of the ACM* 33(3): 323–336.

Dincbas, M.; Van Hentenryck, P.; Simonis, H.; Aggoun, A.; and Graf, T. 1988. Applications of CHIP to Industrial and Engineering Problems. Paper presented at the First International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, 1–3 June, Tullahoma, Tennessee.

Fox, M. S. 1987. Constraint-Directed Search: A Case Study of Job-Shop Scheduling. In Research Notes in Artificial Intelligence, 1–184. London: Pitman.

Frost, D., and Dechter, R. 1994. In Search of the Best Constraint-Satisfaction Search. In Proceedings of the Twelfth National Conference on Artificial Intelligence, 301–306. Menlo Park, Calif.: American Association for Artificial Intelligence.

Jo, G.-S.; Jung, J.-J.; and Yang, C.-Y. 1997. Expert System for Scheduling in an Airline Gate Allocation. *Expert Systems with Applications* 13(4): 275–282.

Jung, J.-J.; Yang, J.-Y.; and Jo, G.-S. 1997. RACES: Ramp Activity Coordination Expert System. Paper presented at the Intelligent Processing and Manufacturing of Materials '97 Australia-Pacific Forum on Intelligent Processing and Manufacturing of Materials, 14–17 July, Gold Coast, Australia.

Liu, B. 1994. Problem Acquisition in Scheduling Domains. *Expert System with Applications* 6(3): 257–265.

Pierre, B.; Bruno, L.; Marie-Ange, M.; and Christiphe, V. 1994. A Scheduling Problem Optimization Solved with Constraint Logic Programming. *Artificial Intelligence* 42:200–231.

Prosser, P. 1993. Domain Filtering Can Degrade Intelligent Backtracking Search. In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, 262–267. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

**Geun-Sik Jo** is currently associate professor in the Department of Computer Science and Engineering, Inha University, Korea. He has been editor in chief for the Korean Intelligent Information Systems Society. Geun-Sik Jo received his B.S. from the Department of Computer Science, Inha University, in 1982; his M.S. in computer science from Queens College/CUNY in 1985; and his Ph.D. in computer science from the City University of New York in 1991. His research interests include intelligent scheduling, intelligent electronic commerce systems, intelligent agents, and constraint programming. His e-mail address is gsjo@inha.ac.kr.

**Jong-Jin Jung** received his B.S. from the Department of Computer Science, Inha University, in 1992; his M.S. from the Department of Computer Science, Inha University, in 1995; and his Ph.D. in computer science and engineering from Inha University in 2000. His research interests include intelligent scheduling, intelligent agents, and electronic commerce.

**Ji-Hoon Koo** is currently a development manager at Thinkfree Calc in Korea. He received his B.S. from the Department of Computer Science and Engineering, Inha University, in 1994 and his M.S. from the Department of Computer Science and Engineering in 1996. He is a Ph.D. student in computer science in the Engineering Department at Inha University. His research interests include intelligent agents, intelligent scheduling, and electronic commerce.

**Sang-Ho Hyun** is a manager in the Information Systems Department at Korean Air. He received his B.S. from the Department of Computer Science and Engineering, Inha University, in 1989. His research interests include airline flight operation systems and optimal aircraft rotation problems.