CN (Canadian National Railway) is the largest railway in Canada and a leader in the North American rail industry. In the bids and bulletins evaluation process, 7,500 CN employees submit their job preferences as bids and are assigned jobs in a manner to ensure that all positions are filled. Each week, CN posts bulletins describing the available jobs and their requirements. On the basis of the bulletins, employees submit a bid card that identifies and ranks the jobs in which they’re most interested. The bidding period runs for seven days. Forty-eight hours after the bidding period closes, job assignments are posted and become effective.

A legacy software system, coded in Cobol, assigns the jobs, but regional schedulers must manage exceptions and infeasibilities. Typically, to achieve a publishable schedule, the software must be run several times, with manual overrides. Because the legacy system is slow, schedulers sometimes resort to personalized spreadsheets to assist decision making.

CN is pursuing a phased implementation of SAP enterprise systems. As the company phases out legacy software, it’s looking for new ways to leverage information technology to improve operations and increase employee satisfaction.

To take a fresh look at the bids and bulletins evaluation process, partners SAP and Axon, on behalf of their client CN, jointly sponsored a Master of Engineering project at Cornell University’s School of Operations Research and Information Engineering in 2007–2008. The project had four major components: development of a bidding model and evaluations of a “buckets and buddies” strategy, a “meet and greet” strategy, and a job assignment algorithm.

The bidding model
CN provided a database of bulletins, bids, and employee information for a single bidding period in the Edmonton region (300 employees). From this, the student team discovered for themselves the skewed nature of the workforce distribution. Approximately 40 percent of the employees in the sample had worked in the company for five years or less. The most recently qualified engineer had been in that craft for 12 years. The approaching retirement of many of these engineers, coupled with a missing cohort behind them, represented a staffing challenge.

The seniority system’s effects were quite pronounced in the data. In the sample, more than 65 percent of senior employees received a job from their top three bids, compared to only 28 percent for junior employees.

The student team analyzed the bids to see what they suggested about employee preferences, recognizing that the likelihood of assignment as well as preference affects the actual bids. The results were intuitive: employees preferred high-wage-block jobs and road trips to low-wage-block jobs and outdoor yard jobs, respectively. Although employees preferred road trips, they much preferred jobs with fewer nights away (two or three nights) to jobs with more nights away (four or five nights).

The team constructed a model to simulate employees bidding on jobs. The model viewed jobs as bundles of attributes (craft, wage category, road or yard, nights away, and so on). It used a utility function based on these attributes to simulate how employees evaluate jobs. It viewed bids as arising from employees sorting the jobs by their utility value. The team set the model’s parameters in consultation with CN system managers. The team generated a set of simulated employees with demographics matching the sample data set. The model used these simulated employees to generate simulated bids for the known jobs.

The simulated bids appeared to replicate the general patterns observed in the actual bids. What was striking, however, was the plot of average utility values across each employee’s top 50 bids. The plot displayed several broad plateaus, suggesting that many jobs have very similar characteristics, thus leading to nearly equal utility values.
The analysis suggested that employees could group jobs into “buckets” such that they would be roughly indifferent to the particular job assigned, provided they were assigned some job in the bucket.

**Buckets and buddies**

These results led to a strategy that might improve employee satisfaction without violating seniority rules. Suppose that each employee designated buckets of jobs to which they were indifferent (relaxing the strict preference implied by bids) and that they listed fellow employees (“buddies”) with whom they would like to work. Then, a postprocessing phase of the assignment process could reassign employees whose buckets overlapped to work on teams with their buddies. The team implemented a simple two-opt swapping heuristic to optimize the number of buddy pairs created. A simulation revealed that, depending on the buckets’ sizes, this strategy could produce a large number of buddy pairs.

**Meet and greet**

The observation that employees prefer jobs with fewer nights away led to the idea of “meet and greet,” in which crews on routes traveling in opposite directions meet at a halfway point and swap trains, returning to their homes in half the time. Besides potentially improving job satisfaction, the swaps would substantially decrease costs by reducing meal and lodging expenses.

The team collected six weeks of actual train departures and arrivals for the Winnipeg region (17,000 train runs, after filtering). It created Gantt chart plots, such as in Figure 1, for every pair of complementary routes with possible meet-and-greet opportunities. In these plots, the horizontal axis indicates time and the vertical axis indicates the distance between stations. Dashed vertical lines mark 24-hour periods. Boxes indicate departures and arrivals; solid lines connect each departure with its matching arrival.

Figure 1a shows a route pairing for which almost no opportunities exist for swapping crews near a midpoint. Figure 1b shows a different route pair in which many crossings occur near the midpoint, but the timing of these crossings has high variability. The tentative conclusion was that opportunities exist for implementing the meet-and-greet strategy but that the implementation would require a very dynamic view of the evolving schedule.

**The job assignment algorithm**

The current algorithm is a complex process that matches employees to jobs on the basis of craft requirements and qualification, job and employee location, bid preferences, and employee availability (vacation or leave status). The most important factor is employee seniority, because all the jobs and employees are subject to collective-bargaining agreements. If two or more employees bid for the same job, the most senior eligible employee must be assigned the job. Because the jobs themselves are covered by different collective agreements, one employee can be more senior than another employee for one job but less senior than the same employee for a different job.

The algorithm essentially sorts the employee list by craft and seniority and assigns each employee to his or her highest-bid available job. If, later, a more senior employee bumps an employee from his or her assigned job, the assignment process
restarts from the bumped employee’s position in the list. The process also includes forcing rules to ensure that all jobs are filled if eligible employees are available.

To demonstrate that the student team understood the algorithm, they created an example (see Figure 2) that would fit on a single page but illustrate most of the algorithm’s key features. The CN systems managers reviewed the example for accuracy.

The team coded the algorithm in Java and ran it on a laptop for the Edmonton data set. The algorithm required only three seconds to run, including input and output processing. So, the legacy software’s long running times must be explained by something other than algorithmic inefficiency.

In spite of the algorithm’s apparent efficiency, the team found it complex and difficult to implement and modify. The problem is that the algorithm mixes rules from collective-bargaining agreements, management rules for efficient use of employee capabilities, and algorithmic rules for finding a feasible solution. The team recommended recasting the problem to modularize these different aspects. The complexity of seniority, preferences, and eligibility can be modularized into functions. Feasibility can then be expressed mathematically as a condition on a given schedule.

For example, let \( x(i,j) = 1 \) if employee \( i \) is assigned to job \( j \); otherwise, \( x(i,j) = 0 \). Let \( p(i,j_1,j_2) = 1 \) if employee \( i \) prefers job \( j_1 \) to job \( j_2 \); otherwise, \( p(i,j_1,j_2) = 0 \). Let \( s(i_1,i_2,j) = 1 \) if employee \( i_1 \) has seniority over employee \( i_2 \) for job \( j \). Then, a solution is feasible from a seniority perspective if and only if, for all combinations of \((i,j)\),

\[
\sum_{i} s(i,i,j)x(i,j) \geq \sum_{j} p(i,j,j)x(i,j)
\]

This modularity frees the software engineers to design different algorithms to solve the assignment problem. For example, assume a function \( \mathcal{I}(x) \) that can find the most senior individual, \( i^*, x(i^*, j) = 1 \), who prefers job \( j \) to the job to which he or she is currently assigned in the partial solution \( x \). There are data structures that can make implementation of this function quite efficient. Figure 3 shows an algorithm to find a seniority-feasible solution. The algorithm leaves the sorting sequence of open jobs unspecified, which allows further tuning of the algorithm.

Using this modular approach, the team coded two alternative algorithms. Instead of looping over employees, these algorithms loop over vacant jobs as in the pseudocode in Figure 3. They run no faster than the current algorithm (less than 10 seconds using MS Access Visual Basic for Applications), but the code’s structure is more modular. Changing seniority, preference, eligibility, management, or algorithmic rules is easier in the modular environment than in the current algorithm.

CN management learned several lessons from this study, including the value of reducing nights away, the potential of “buckets and buddies” to increase employee satisfaction, the opportunities and difficulties of implementing “meet and greet,” and the value of an architectural approach to constructing the job assignment algorithm. Plans are being made to extend this research to consider crew-scheduling problems.

**References**


Peter Jackson is a professor at Cornell University’s School of Operations Research and Information Engineering. Contact him at pj16@cornell.edu.

Yanbin Chen is a graduate of Cornell University’s School of Operations Research and Information Engineering. Contact him at yc465@cornell.edu.

Ramin Farhangi is an associate with the Boston Consulting Group. Contact him at rf75@cornell.edu.

Xiang Li is a software engineer at Microstrategy. Contact him at siliqis@gmail.com.

David A. Pilo Mansion is a program manager for Microsoft. Contact him at dap232@cornell.edu.

Evan Markel is a graduate of Cornell University’s School of Operations Research and Information Engineering. Contact him at em255@cornell.edu.

Ryan Morris is the founder and CEO of VideoNote LLC. Contact him at rjm59@cornell.edu.

Len Podgurny is system manager for Crew Management Solutions at CN, Montreal. Contact him at len.podgur02@cn.ca.

Al Randall is system manager for Crew Management Solutions at CN, Edmonton. Contact him at al.randall@cn.ca.

Paul Hofmann is vice president of research at the SAP Labs, Palo Alto. Contact him at paul.hofmann@sap.com.