Case Study: An Intelligent Decision-Support System

Zbigniew Michalewicz, *University of Adelaide*

Martin Schmidt, Matthew Michalewicz, and Constantin Chiriac, *SolveIT Software*

Distributing used cars to various automobile auctions is a complicated problem with multiple variables. A software system that combines prediction, optimization, and adaptation techniques has generated impressive profits for a large auto manufacturer.

Information technology applications that support decision-making processes and problem-solving activities have proliferated and evolved over the past few decades. In the 1970s, these applications were simple and based on spreadsheet software. During the 1980s, decision-support systems incorporated optimization models, which originated in the operations research and management science communities. In the 1990s, these systems were further enhanced with components from artificial intelligence and statistics.

This evolution led to many different types of decision-support systems with somewhat confusing names, including management information systems, intelligent information systems, expert systems, management-support systems, and knowledge-based systems. Because businesses realized that data was a precious asset, they often based these "intelligent" systems on data warehousing and online analytical processing technologies. They gathered and stored a lot of data, assuming valuable assets were implicitly coded in it. Raw data, however, is rarely beneficial. Its value depends on a user's ability to extract knowledge that is useful for decision support. Thousands of "business intelligence" companies thus emerged to provide such services. After analyzing a corporation's operational data, for example, these companies might return intelligence (in the form of tables, graphs, charts, and so on) stating that, say, 57 percent of the corporation's customers are between 40 and 50, or product *Q* sells much better in Florida than in Georgia.

Many businesses have realized, however, that the return on investment for pure "business intelligence" is much smaller than initially thought. The "discovery" that 57 percent of your customers are between 40 and 50 doesn't directly lead to decisions that increase profit or market share. Moreover, we live in a dynamic environment where everything is in flux. Interest rates change, new fraud patterns emerge, weather conditions vary, the stock markets rise and fall, new regulations and policies surface, and so on. These economic and environmental changes render some data obsolete and make other data—which might have been useless just six weeks ago—suddenly meaningful.

We developed a software system to address these complexities and implemented it on a real distribution problem for a large car manufacturer. The system detects data trends in a dynamic environment, incorporates optimization modules to recommend a near-optimum decision, and includes self-learning modules to improve future recommendations. As figure 1 shows, such a system lets enterprises monitor business trends, evolve and adapt quickly as situations change, and make intelligent decisions based on uncertain and incomplete information.

Problem overview

We developed the system for a US-based car manufacturer that has more than 1 million cars returned from leases or rentals each year. The manufacturer owns the cars, and the problem is how to best distribute these cars among hundreds of auction sites around the United States. The cars vary by make and model, mileage, options, wear and tear, and so on. These characteristics, along with others, influence the car's sale price at each particular auction. Our central challenge was to achieve the "best" possible distribution among these auction sites—that is, the distribution that maximizes the net proceeds from all sales.

The process of making optimal recommendations involves many considerations, ranging from price pre-

1541-1672/05/$20.00 © 2005 IEEE
Published by the IEEE Computer Society

diction for various car types at different locations, to price depreciation and volume effects, to transportation issues. One million cars per year corresponds to approximately 4,000 cars per working day. So, each day, a remarketing team must make 4,000 decisions regarding which auction site will maximize the sales price of each car. Further, due to volume effects, assigning cars to auctions is highly interrelated, and therefore it's not possible to process these cars sequentially.

Say, for example, that a company uses 50 auction sites and processes a mere 1,000 cars per day. This results in a mind-boggling $50^{1,000}$ distribution choices! No computer can check out all these possible combinations in a human lifetime. Nevertheless, the manufacturer requires decisions on all of the cars *today*.

Problem complexity

To illustrate the task, we'll use a silver, four-door 2002 Toyota Corolla with 34,983 miles, a sunroof, automatic transmission, power windows, power seats, and many other options. At the moment, the car sits at a dealership in Virginia, and we must decide where to send it. At first glace, this looks easy. We might be tempted to simply look up the car's average sales price at each auction using one of many guides, such as the *Black Book*, *Kelley Blue Book*, or *Manheim Auction Report*. After adjusting the price based on the car's mileage, options, and so on, and estimating transportation costs—both manageable calculations—we might simply decide to target the auction with the highest current average sales price.

So, what's the problem? In a word: volume. Per car, it's cheaper to ship a truckload of cars from one place to another than it is to ship one or a few cars at a time. Assuming fixed "from" and "to" locations, a typical transportation cost structure would look something like this:

- 1–6 cars cost $120 per car.
- 7–10 cars cost $95 per car.
- 11–14 cars cost $85 per car.

For more than 14 cars, we usually calculate transportation costs by determining the fee for transporting 14 cars (or multiples of 14 cars), then calculate the remainder at the applicable rate. For example, transporting 20 cars would cost us $85 per car for the first 14 cars, and then $120 per car for the remaining 6 cars, for a total cost of $1,910.

Beyond transportation costs, we must also address other considerations, such as the volume effect, auction and transportation scheduling, depreciation, and insurance and risk factors (cars might be damaged or stolen in transit). The *volume effect* kicks in as the number of similar cars for sale grows. If we send many similar cars to a single auction site—which is reasonable, assuming it offers the best net price—the volume will result in less money per car (see figure 2).

Say, for example, the current average sale price for a 2002 Toyota Corolla on a particular auction site is $7,200. We would likely get this price if we ship up to seven cars to that location. However, transporting 30 similar cars to that site would drop the average price to $6,900. To complicate this further, "similar" doesn't mean the same make, model, and color. Even though the makes and models might differ, shipping 30 silver sedans to the same auction gives buyers more options, thereby depressing the average sales price per car. Also, the volume effect's curve is different for different types of cars. With Toyota Corollas, for example, the volume effect is significant, whereas for Porsche 911s, it is moderate.

Scheduling is also a major issue. Every auction has a typical sales day, such as at 11 a.m. every second Friday. So, let's say we have 20 cars that we'd like to ship to an auction site, the transport time is 10 days, and the next auction is 11 days away. If there is even a slight delay in the delivery of these 20 cars, then we might miss the auction. The cars would then have to sit in the auction's parking lot for almost two weeks. This is bad—not only because the company wants them sold as soon as possible, but because the cars would lose value each day. The *price*

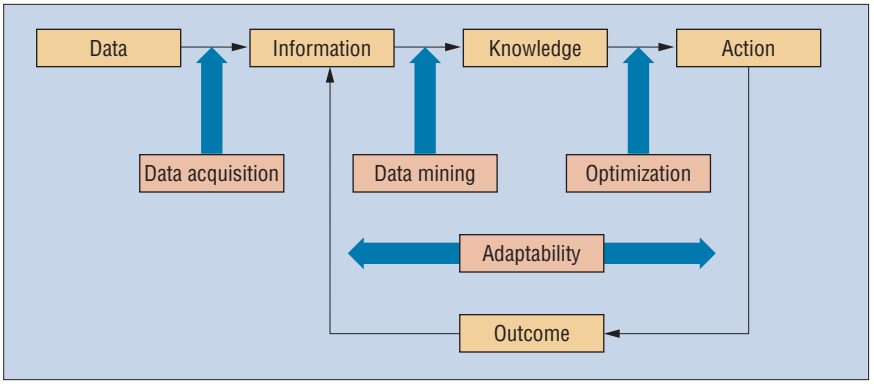

Figure 1. Adaptive business intelligence. This diagram shows the flow from data acquisition to recommended action, including an adaptive feedback loop.

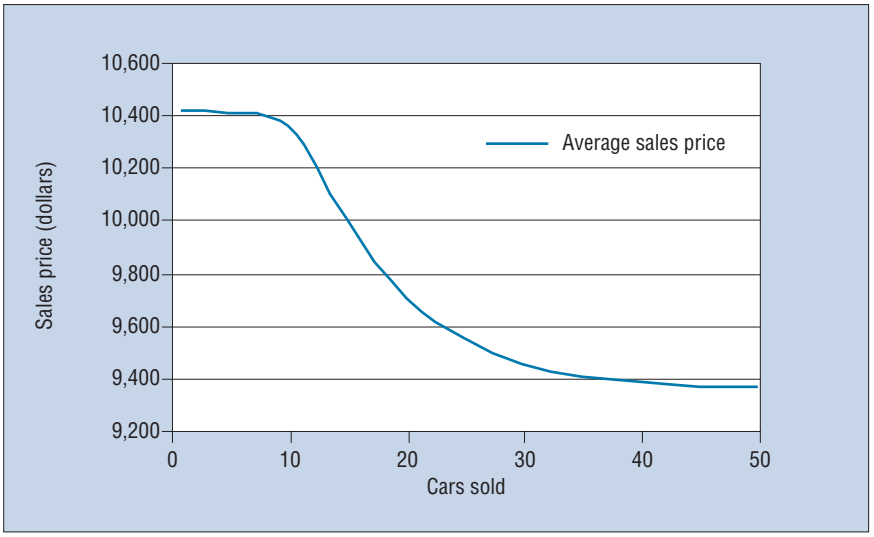

Figure 2. The volume effect. Past a certain point, selling many similar cars at the same auction reduces the price per car.

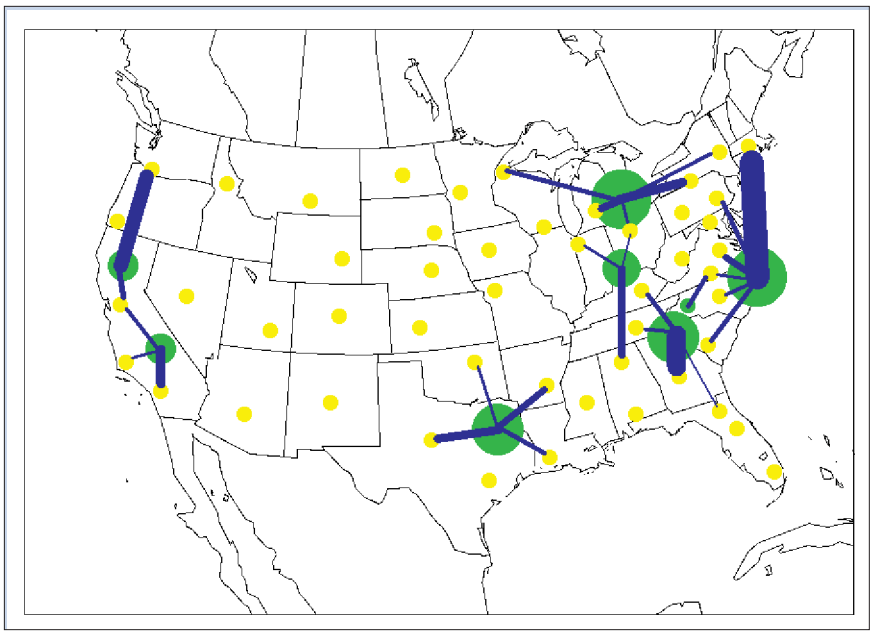

Figure 3. Example distribution. The green circles indicate leased car returns, with the circle size indicating the return volume. The yellow circles indicate the fixed auction sites, and the blue lines illustrate the transportation between car locations and auction sites, with line thickness indicating the shipping volume.

depreciation for the average car is around $10 per day.

Another thing that complicates our job is that the market price for cars changes frequently—sometimes slightly and sometimes dramatically (following the terrorist attack on 11 September 2001, for example, car prices dropped significantly; major hurricanes in Florida have a similar effect on car prices). So, on top of all the other issues, we must stay on top of the changing car prices. Finally, car prices often reflect a *seasonality effect* depending on the geographic region. It's not easy to sell a convertible Corvette in Boston in October, for example, but Florida's temperature at that time of year is perfect for such vehicles.

Endgame

This problem requires a distribution decision at the end of each workday, regardless of whether the decision is "optimal" and maximizes company profits, or if another distribution exists that would increase net sales by $150 per car (which, with 4,000 cars per day, would increase total net sales by $600,000).

Figure 3 shows a distribution example. The green circles indicate the return of leased cars: the bigger the circle, the larger the return volume (clearly, most cars were returned in the eastern US on this particular day, though the next day might look entirely different). Yellow circles indicate 50 possible auction sites. The auction site locations are fixed, regardless of the car-return volume (however, companies occasionally change the auction sites they do business with—by dropping some and adding others—which raises important issues that we discuss later). The blue lines illustrate the transportation connections between the cars' current locations and auction sites. The thicker the line, the more cars are shipped.

Clearly, this is not a trivial problem. Any decision must consider the details of each car; many different potential auctions, their timetables, and inventories; complex transportation costs; volume effects; nationwide vehicle inventories en route to auction; price depreciations; dynamic price changes at each auction; sale price predictions up to two months ahead of time; and so on. Every leasing company therefore has an entire remarketing team dedicated to distributing returned cars. These teams face the formidable task of recommending the best possible distribution for each day's carload, day after day, week after week, month after month. As we noted earlier, a small mistake or an inferior recommendation might result in a net loss of "only" $150 per car, but this can translate into hundreds of thousands of dollars in a single day, depending on distribution volume. In contrast, if a smart decision-support system improves the daily car distribution and thereby lifts net sales by, say, $200 per car (which is only a 1.33 percent increase in the price of an average $15,000 off-lease car), the leasing company would increase its annual profits by hundreds of million of dollars. Exploring this possibility is clearly worthwhile.

The solution

To address this problem, we developed an intelligent system comprising several building blocks, including prediction, optimization, and adaptation modules. All three modules involve research challenges, which we briefly review in the "Research Issues in Dynamic Optimization" sidebar.

Prediction module

The prediction module consists of several components. After the default base price is set, the remaining modules adjust this price to create a final predicted price.

- *Default base pricing.* Using regularly updated *Black Book* data, this component specifies the price based on region, make, model, year, and mileage.
- *Zip-code-based make/model adjustment.* This component uses historic data, such as sales records, to adjust the price of the specific make and model.
- *Zip-code-based color and group adjustment*. This component specifies adjustments based on the car's color and group (luxury, midsized, compact, and so on).
- *Mileage adjustment*. This adjustment is based on the vehicle's make, model, mileage category, and model-year age—that is, when the 2006 model comes out (say, in September 2005) the 2005 model's year age is one. Mileage adjustments are broken down according to the following categories: 0–10,000, 10,001–30,000, 30,001–60,000, 60,001–90,000, 90,001–120,000, 120,001–150,000, and over 150,000.
- *Seasonality adjustment*. This component adjusts the price based on the car's model year, make, model, and region. It is not zip-code specific. The module calculates the seasonality adjustment using a daily depreciation rate, which is defined each month. The module calculates the total adjustment by combining the daily adjustments from the date of the default base pricing through the predicted sales date.

Research Issues in Dynamic Optimization

Most data-mining and optimization algorithms assume static data and a static objective. Typically, they search for a snapshot of "knowledge" and a near-optimum solution with respect to some fixed measure (or set of measures), such as profit maximization or minimization of task-completion time. However, real-world applications operate in dynamic environments, where it's often necessary to modify the current solution due to changes in the problem setting, such as machine breakdown or employee illness; or the environment, such as consumer trends or changes in weather patterns or economic indicators.

It's therefore important to investigate adaptive algorithms that don't require restart every time a change is recorded. In many commercial situations, such restarts are not an option.

Evolutionary techniques

An obvious starting point here is evolutionary computation techniques,[1] which are optimization algorithms inspired by the continuously changing natural environment. However, it's important to investigate which evolutionary algorithm extensions are actually useful in business scenarios. Unfortunately, most current approaches ignore dynamics and assume that re-optimization should occur at regular intervals. However, significant benefits can be realized when researchers explicitly address *dynamism*.

Many researchers have proposed various benchmarks for studying optimization in dynamic environments.[2] Among the proposals are the moving peaks benchmark, the dynamic knapsack problem, dynamic bit-matching, scheduling with new jobs arriving over time, and the greenhouse control problem. Researchers have also proposed various measures, including offline error, percentage of covered peaks, and diversity.[2] Among the partial conclusions reached in this research:

- standard evolutionary algorithms get stuck on a single peak;
- diversity preservation slows down the convergence;
- random immigrants introduce high diversity from the beginning, but offer limited benefits;
- memory without diversity preservation is counterproductive; and
- nonadaptive memory suffers significantly if peaks move.

What's missing?

However, several essential points are seemingly missing in the key research on optimization in dynamic environments. Most researchers emphasize an ultimate goal of approximating real-world environments, but they fail to address several key issues for successful adaptive-system development. The following issues, which constitute the conceptual research framework, are essential for creating a methodology for building intelligent systems.

Nonstationary constraints

Here, the task is to optimize a non-stationary objective function $f(\mathbf{x}, t)$, subject to nonstationary constraints, $c_i(\mathbf{x}, t)$ 0 (i = 1, 2 , … , k). We have applied this approach successfully in the context of a collision situation at sea.[3] By accounting for particular maneuvering-region boundaries, along with information on navigation obstacles and other moving ships, we reduced the collision-avoidance problem to a dynamic optimization task with static and dynamic constraints. The proposed algorithm computed a safe and optimum ship path in both static and dynamic environments.

Prediction component

Environmental changes are seldom random. In a typical real-world scenario—where constraints change over time—it's possible to calculate some failure probabilities by analyzing past data, and thus predict a possible environmental change. Our work on collisions at sea offers a good example here as well.[3] We based a ship's safe trajectory in a collision situation on *predicted* speeds and the other ships' directions. Studying dynamic environments where change is somewhat predictable is important, but so far, little work exists along these lines.

Parameter adaptation

In nonstationary environments, researchers must study parameter control, particularly when the adaptive system includes predictive methods.[4]

Solution robustness

Research into robustness concentrates on questions such as, What constitutes flexibility in the specific context? How can we integrate a flexibility goal into the algorithm? To answer these questions, we must take into account a predictive model (for environmental changes) and the prediction's estimated error. This has yet to occur. Many researchers have recognized the importance of solution robustness.[1,2] Existing approaches vary, from techniques to "disturb" individuals in the population to those using search history. Some researchers have considered an aspect of robustness—sometimes called *flexibility*—in which the problem requires sequential decision-making under an uncertain future, and the decision influences the system's future state. In such situations, the decision-making process should anticipate future needs. That is, rather than focusing exclusively on the primary objective function, it should try to move the system into a flexible state.

References

1. Z. Michalewicz and D. Fogel, *How to Solve It: Modern Heuristics*, 2nd ed., Springer-Verlag, 2004.
2. J. Branke, *Evolutionary Optimization in Dynamic Environments*, Kluwer, 2001.
3. R. Smierzchalski and Z. Michalewicz, "Modeling of Ship Trajectory in Collision Situations by an Evolutionary Algorithm," *IEEE Trans. Evolutionary Computation*, vol. 4, no. 3, 2000, pp. 227–241.
4. A.E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter Control in Evolutionary Algorithms," *IEEE Trans. Evolutionary Computation*, vol. 3, no. 2, 1999, pp. 124–141.

- *Universal Vehicle Code adjustment*. The UVC provides a more detailed vehicle specification than the Vehicle Identification Number (VIN); when the UVC is available, adjustments are made accordingly.
- *Other pricing adjustments*. This module supports additional adjustments if more information is available, such as auction-specific sales data from various auction sites.

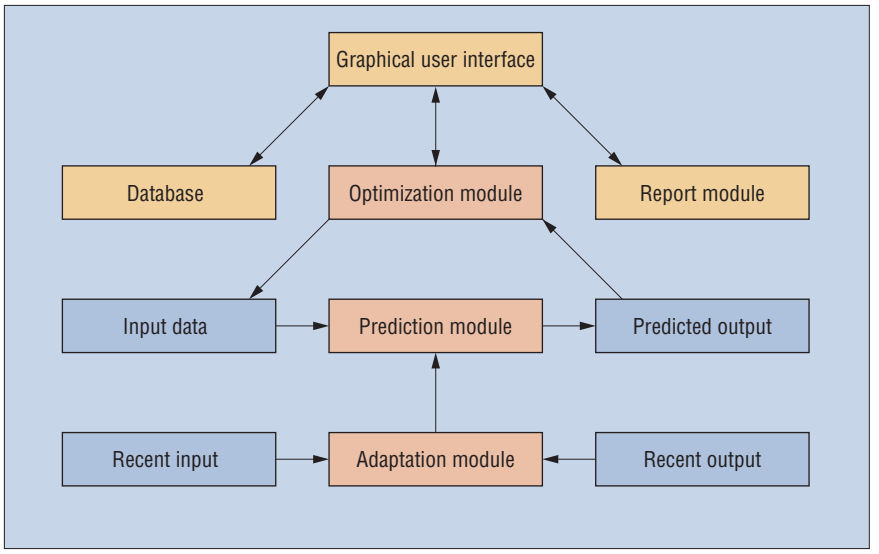

Figure 4. The intelligent decision-support system. The prediction, optimization, and adaptation modules feed information into the user interface, the information database, and the report module.

All these components include parameters, which we adjust at regular intervals to capture the changing trends in the used car market.

Optimization module

The optimization module's job is to recommend the best distribution of cars to auction sites. Because this recommendation is based on prediction, there is a strong relationship between the prediction and optimization modules. Generally, the optimizer produces a possible solution, which becomes the input to the prediction module. The optimizer then uses the prediction module's resulting output to evaluate the solution (in terms of the total value of the proposed distribution).

For the car distribution system, our optimization module implementation uses an evolutionary algorithm to represent each solution as a vector of 4,000 values that provide—for each car separately—auction site indices from 1 to 60. The optimizer's initial distributions are produced by combining randomly generated solutions, built from random numbers (from 1 to 60) for each entry, and heuristic-based solutions, based on calculating the distance between cars and various auction sites.

A vector of all 1s would indicate that each car is scheduled for transport to the closest auction site. The optimizer's evaluation function provides the proposed distribution's total value because a solution vector specifies each car's exact destination. For each distribution, the optimization module

- calculates the expected total of all sales (at specified locations and at specified time intervals, including seasonal effects);
- modifies the total to include price depreciation and volume effect; and
- subtracts costs related to the given distribution (transportation costs, auction fees, and so on).

Finally, we designed the module to output the given distribution's expected net proceeds. For this representation, we apply standard operators of various mutations and crossovers, which we proposed and experimented with for integer vectors.[1] In this case, however, we extend the algorithm with an additional local search routine. We also designed a problem-specific decoder to deal with problem-specific constraints (such as excluding all red cars from an auction site in Austin, Texas, or excluding all cars more than four years old from all California auctions). We also use the so-called *elitist* strategy, which forces the best solution from one generation to the next.

The population of solutions in the evolutionary algorithm consists of approximately 2,000 solutions (this number is a function of the number of cars processed on a particular day). We process the solutions in the evolutionary loop. The system evaluates 2,000 individuals, selects parents, applies crossover and mutation operators to create offspring, creates the next generation, and so on. The system repeats this simulated evolutionary process until the generation counter t hits a threshold (again, this number—around 150,000—is a function of the number of cars processed that day). At this stage, the system applies a local search method. It thereby locates many quality solutions within the final population and reports back the best individual solution.

Adaptation module

Although intelligent software systems require a prediction module and optimizer, by themselves they're insufficient for today's rapidly changing environment. The prediction module must be adaptive and learn from environmental changes, as today's accurate prediction might not be accurate tomorrow.

We accomplish adaptability by slightly altering the learned relationship between input and output as needed. This alteration might be required every second, minute, hour, day, week, or month, for example, depending on how quickly the environment changes. Some classic forecasting methods—such as exponential smoothing methods[2]—approach this problem by emphasizing recent data. An ideal adaptive solution can decide the update frequency for itself by continuously measuring its own prediction errors and adjusting its parameters accordingly. Hence, the adaptive system adapts its own speed of adaptation.

Our adaptation module takes recent input and output from very recent history, using historic data to construct and train the prediction module. The whole system then makes regular (daily) recommendations on where to send cars. The system also provides a sales price prediction for each car, adjusted for volume effect. All this is recorded in the company's database. Once the cars are actually sold a few weeks later, it can compare the predicted and actual sale prices and, for example, detect new price trends. In any case, the actual prices constitute recent output, and the auction sites, dates, and car information constitute the recent input. The adaptation module uses both, and, if necessary, adapts the prediction module's parameters to decrease the prediction error based on environmental changes.

Solution context

The prediction, optimization, and adaptation modules are the foundation of our adaptive solution. To address basic user needs, the system also features an intuitive graphical user interface, a database to store information, and a report module for easy information access.

Figure 4 shows a diagram of the system.

Comparing it with figure 1 shows that the adaptation module in figure 4 is responsible for the loop from "data" to "information," while the optimization module converts "knowledge" into "action," and so on.

Our system can support intelligent decisions in many domains, from fraud detection to routing to portfolio management and beyond. In most cases, specific problem characteristics boil down to prediction, optimization, and adaptation. Because we can design and build the first two modules independently—then tune them to the problem at hand—the system is widely applicable in many domains.

System implementation

Our system runs on a Pentium 4 1-Ghz PC, and the car manufacturer has used the system on a daily basis since 2002. The total running time for optimization is around 1.5 hours, though additional time is required to load all the necessary input files.

The client's inventory management system generates the inventory of cars to be distributed, which our system automatically downloads from the FTP server and processes early in the morning. The files include data about the cars, current, up-to-date inventory of cars at the auctions, and information about cars that have been sold (to tune the price prediction module).

The optimization process starts automatically once the input files have been loaded and processed. Once this process is complete, a proposed solution is ready by the start of the business day. However, the system implements the solution only after a remarketing officer checks the results and (possibly) makes small adjustments. Less than 1 percent of the recommended cars are changed manually; these changes typically correspond to last-minute actions based on new information. Once the checking process is finalized, the system sends the final output files to the transportation-management system.

Results and future work

We measured the system's performance in two ways. First, we divided a daily sample of 4,000 cars into two sets of 2,000 cars each, with an almost identical distribution of make and models. We then distributed one set of cars using the "old" system and compared results with the cars that were distributed through our system. Second, we used the old system on selected days (Mondays, Wednesdays, and Fridays) and our new system on the remaining days (Tuesdays and Thursdays), and again compared results after the cars were sold. Based on these comparisons, we estimate that, since its implementation in 2002, our intelligent decision-support system has generated a multimillion dollar sales lift per year.

In future versions of our system, we might include several additional features, including the ability to

- make adjustments based on current weather conditions,
- change the destination of cars en route to a specific auction site, and
- account for new body styles on particular makes and models.

More information on this system and other applications of adaptive software for business problems is available at www.solveitsoftware.com. ■

The Authors

Zbigniew Michalewicz is a professor at the School of Computer Science, University of Adelaide. He also holds professorships at the Institute of Computer Science, Polish Academy of Sciences, and the Polish-Japanese Institute of Information Technology. He chairs the board of SolveIT Software, a technology company he co-founded, and holds an honorary professorship at the State Key Laboratory of Software Engineering of Wuhan University, China. His current research interests are in evolutionary computation. Michalewicz received a PhD in applied mathematics from the Institute of Computer Science, Polish Academy of Sciences. Contact him at the School of Computer Science, University of Adelaide, Adelaide, SA 5005, Australia; zbyszek@cs.adelaide.edu.au; www.cs.adelaide.edu.au/~zbyszek.

Martin Schmidt is cofounder and chief technology officer of SolveIT Software, where he works on the commercialization of software solutions based on evolutionary algorithms, simulated annealing, large-scale simulations, neural networks, fuzzy logic, and hybrid systems. He is also an adjunct research fellow at the University of Adelaide. His research interests include applying computational intelligence in real-life constraint optimization and simulations, as well as in classification and prediction systems. He received a PhD in computer science from the Polish Academy of Sciences in Warsaw. Contact him at SolveIT Software, PO Box 3161, Adelaide, SA 5000, Australia; ms@solveitsoftware.com.

Matthew Michalewicz is chief executive officer at SolveIT Software, where he sets the company's strategic mission and is involved in the business aspects of commercializing software applications based on computational intelligence. He is also a visiting fellow at the University of Adelaide. He received a BS in financial management from the University of North Carolina, Charlotte. Contact him at SolveIT Software, PO Box 3161, Adelaide, SA 5000, Australia; mm@solveitsoftware.com.

Constantin Chiriac is cofounder and chief software architect of SolveIT Software, where he works in the development and commercialization of software systems. He is also an adjunct lecturer at the University of Adelaide. He was previously a software project leader for the National Bank of Moldova. He received an MSc degree in computer science from the University of North Carolina at Charlotte, and an MS in economics from the International Independent University of Moldova. Contact him at SolveIT Software, PO Box 3161, Adelaide, SA 5000, Australia; cc@solveitsoftware.com.

Acknowledgments

We thank all the remarketing team members for sharing their valuable domain knowledge.

References

1. Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed., Springer-Verlag, 1996.
2. S. Makridakis, S.C. Wheelwright, and R.J. Hyndman, *Forecasting, Methods, and Applications,* John Wiley & Sons, 1998.