Workforce optimization: Identification and assignment of professional workers using constraint programming

Y. Naveh
Y. Richter
Y. Altshuler
D. L. Gresh
D. P. Connors

Matching highly skilled people to available positions is a high-stakes task that requires careful consideration by experienced resource managers. A wrong decision may result in significant loss of value due to understaffing, underqualification or overqualification of assigned personnel, and high turnover of poorly matched workers. While the importance of quality matching is clear, dealing with pools of hundreds of jobs and resources in a dynamic market generates a significant amount of pressure to make decisions rapidly. We present a novel solution designed to bridge the gap between the need for high-quality matches and the need for timeliness. By applying constraint programming, a subfield of artificial intelligence, we are able to deal successfully with the complex constraints encountered in the field and reach near-optimal assignments that take into account all resources and positions in the pool. The considerations include constraints on job role, skill level, geographical location, language, potential retraining, and many more. Constraints are applied at both the individual and team levels. This paper introduces the technology and then describes its use by IBM Global Services, where large numbers of service and consulting employees are considered when forming teams assigned to customer projects.

Introduction

Employees are the most important asset of any technology-based company. This statement is not a mere slogan, but a genuine business reality that requires careful consideration at all management levels in the company. While this reality has been recognized for a long time, only recently have rigorous processes, backed by automation, become central in reaching workforce-related decisions. One of the main reasons for this is the fact that professional workers, being humans, are complex entities. They each have individual skills, interests, expectations, and limitations. They may live in a particular area, have family-related constraints, prefer working solo, or function best as team players. They may be more or less susceptible to pressure, easy or difficult to retrain, and motivated by completely diverse factors. Most significantly, it is perceived that human professionals cannot possibly be described as a mere set of attributes, no matter how large the set. For example, most resumes—formal documents designed to best describe the aspects of people relevant to their hiring—contain lengthy textual descriptions rather than a structured list of attributes and values. Summarized eloquently, it is often maintained that "people are not parts."

While it is true that people are not parts, the situation still exists in which a large number of professionals must be matched and assigned to a similarly large number of demanding jobs. In fact, this problem lies at the heart of the execution phase of the workforce management (WM) cycle [1]. The problem applies to many different business cases in the technology industry, including assigning service professionals to short-term maintenance tasks [2], team-building for contracted projects [3], maintaining staff with multiple skills [4], and more.

0018-8646/07/$5.00 © 2007 IBM

In all of these cases, the consequences of failing to find the best assignments for the jobs are extremely severe. Less-than-optimal assignments can be manifested in three general forms: underqualified professionals assigned to highly demanding jobs, overqualified professionals assigned to less-demanding jobs, and a total number of assignments smaller than the maximum achievable. An underqualified assignment may result in the need to reperform the job without compensation, costly onsite training, customer dissatisfaction with the job, eventual loss of this customer, and loss of referrals from the customer. In addition, *qualification* may refer to various attributes, not necessarily the professional level of the worker. For example, an underqualification may be a travel distance that is too long, with direct travel costs being incurred by the provider. The costs of an overqualified assignment may relate directly to the unrecovered high salary of the professional or indirectly to the loss of a more profitable job assignment for the employee, employee dissatisfaction, and eventual employee attrition. A less-than-optimal number of assignments may result in loss of revenue from unassigned jobs, increased costs from subcontracting external providers for the unassigned jobs, and the general dissatisfaction of the customers ordering the unassigned jobs.

The usual way of solving the general assignment problem presented above is to examine the full list of jobs in some predefined order and for each job find a corresponding shortlist of best-fitting candidates, then assign one of those candidates to the job. (An equivalent option is to look at the full list of professionals in a predefined order, find a shortlist of best-fitting jobs for each professional, and then assign the professional to one of those jobs.) This procedure is simple and can be accomplished by a human resource deployment professional (RDP), because at any one time the actual fitting procedure looks only at a single job and a shortlist of professionals. As part of this procedure, the RDP may use search tools to search for an employee with characteristics required by the job, provided that relevant data for all professionals is stored in some database. However, the procedure has the following significant drawbacks:

- It is tedious, repetitive, and time-consuming.
- Since the shortlist of matches is not prioritized within itself, it requires further manual work to rank-order the individuals in the shortlist and is thus likely to result in a suboptimal choice, even for the single job currently considered.
- The first job considered will likely be assigned the best-found professional for the job (a *greedy* policy), even though that professional may be better suited to other jobs that have not yet been considered. This may lead to fewer assignments to jobs because the other jobs may not find another match, while alternative professionals may exist for the current job. It can also lead to possible overqualification of the professional for the assigned job.
- Competition among jobs considered (or *owned*) by different RDPs is even less likely to be resolved fairly, because each RDP sees and applies only his or her own criteria, and there is no mechanism for finding a fair and optimal assignment among all RDPs.
- When the number of available jobs and professionals is large, say a few dozen or more, it becomes impossible to find the best matches manually. This is true even when the matching criteria are stated correctly and the RDPs are motivated to seek a global best solution. The reason for this is that the optimization problem is known mathematically to be *NP-hard*, which means that beyond a certain number of jobs, an exponentially large number of comparisons between different candidates must be done in order to reach the optimal assignment.
- Only the most simplistic types of matching rules, or *constraints*, can be considered by human RDPs. One example of a simple rule may be exact matches on several searched attributes, such as skills, availability, and pay rates. However, even a simple matching rule that requires, e.g., a short travel distance between work and the person's location is difficult to enforce manually, because this distance must be recalculated for any job–candidate pair. Finding a good solution that complies with rules that are inherently complex (for example, team-building rules) is far beyond the capacity of a human RDP.
- In searching for candidates who possess a number of desired attributes, all attributes are viewed as having the same importance. When some attributes are of higher importance than others, finding the best matches must be achieved manually by first searching for candidates with the most important attribute, then reducing the list to those also having the next important attribute, and so on. In addition to the slowness of this procedure, it will likely miss a professional with many of the less-important required attributes who lacks only one of the more-important attributes.

Given the above drawbacks, the potential for large amounts of data, and the need for a short response time, an automatic procedure to optimize the set of assignments could offer a significant benefit. However, there are two major obstacles to using ordinary

operations research (OR) optimization techniques, such as those used in supply chain management. First, we must accept that people are not parts. One cannot assume that a person can be modeled as a mathematical set of attributes. This implies that the engagement and business rules related to people are bound to be much more complex than ordinary rules observed in mainstream OR. Second, there is no meaningful way to define a rigorous mathematical cost function for the WM optimization problem. For example, one cannot seriously quantify the cost associated with a dissatisfied overqualified employee, nor the cost of a person with imperfect foreign-language skills working at an offshore location. Without a good cost function, much of the substance of OR techniques—primarily linear programming and its derivatives, and metaheuristics—is lost.

In this paper we present a completely different optimization approach to the identification and assignment (ID&Assign) WM problem described above. Our approach is based on constraint programming (CP) [5], a subdiscipline of artificial intelligence. We suggest that using CP can bridge the gap between the fact that people cannot be treated as pure mathematical objects and the requirement for a mathematical procedure for optimization. Indeed, one of the most compelling features of CP is that rules are stated in a high-level language derived from the domain of application, and not as a mathematical formulation. Once the rules are stated, there are rigorous mathematical algorithms that can interpret the rules and optimize the solution according to all rules defined. As we demonstrate, this approach eliminates the drawbacks of manual handling that are listed above. The only previous work we are aware of that applied CP to the ID&Assign problem [6] considered only the most simple matching rules; it was motivated by the scheduling aspects when applied to a combination of full-time and part-time employees. Our focus is on complex rules and the large number of significantly different individual professionals available for assignments. We are also interested in the case in which people may be assigned to highly specific jobs of relatively short duration (a few weeks to a few months) and may need to move from job to job quickly, without wasting time. The technology presented here may not be as useful in a context in which people are assigned to jobs only once or infrequently.

The scope of this paper is the ID&Assign problem for highly skilled employees. Early in the WM cycle, planning and prediction of the expected future workforce is performed. In that phase, the entities of interest are aggregates of people and jobs, where the actual people and jobs are not yet available or known. A well-defined mathematical model can be built for the aggregates and solved by traditional OR and supply chain techniques. The IBM Resource Capacity Planning tool [7] does just that. The scheduling problem of low-level employees can also be solved in an aggregate fashion, because the entire workforce can be partitioned into a small set of homogeneous groups, and OR techniques may be readily applied to take into account the simple scheduling constraints of each employee in the group. A typical example of such a case is call-center scheduling, and the IBM SWOPS tool [8], based on linear programming, is designed to meet that need.

The next section provides a more detailed summary of existing work in workforce optimization. We then provide a short overview of constraint programming—in particular, its modeling and algorithmic aspects. We go on to describe a basic constraint satisfaction problem (CSP) model for the ID&Assign problem and delve deeper into the details of workforce management rules. We present use cases and results related to IBM service organizations and then conclude the paper.

Survey of existing work

Workforce scheduling problems are traditionally classified into three types: shift scheduling, days-off scheduling, and tour scheduling. Shift (or *time-of-day*) scheduling determines each employee's work and break hours per day. Days-off (or *days-of-week*) scheduling determines each employee's workdays and off-days per week or on a multiple-week work cycle. Tour scheduling combines the shift and days-off scheduling problems by determining each employee's daily work hours and weekly workdays. An introduction to the problem, classification of its types, and the difficulty in solving it can be found in [9, 10]. More recent reviews can be found in [11, 12].

In general, most WM solutions can be divided into two approaches: using a generic method to solve the problem or using some specific algorithm created for a particular problem. We are interested primarily in the first approach. The second approach is often used for problems that have complex or unique features, such as discontinuous objective functions [13] or others [14].

Traditional OR approaches are often used for WM [15]. Linear and integer programming techniques [16, 17] are examples of such approaches. Another approach to solving WM problems is trying to find reductions of those problems to other OR domains (e.g., routing) [18].

Significant work is also being done to apply modern metaheuristics techniques to WM. Tabu search is often used [19, 20]. Genetic programming is sometimes used with special features of the WM problem structure to efficiently solve problems that are computationally hard [21–23]. An interesting variant of WM problems is known as *mobile workforce management* [24]. To solve this problem, a special multi-agent information system

technology has been developed. Additional discussion of metaheuristics and heuristics-oriented solutions for WM appears in [25].

CP and constraint logic programming (CLP) have also been used to solve WM problems. One case of a simple real-life WM problem was noted by British Telecom [26]. This problem was later approached using simple CP [6] and two local search methods [27]. Other uses of CP and CLP for WM are presented in [28] and [29]. In all of these cases, CP is used to solve traditional WM problems in which the main difficulty is scheduling employees to shifts under rules governing the work hours and workdays. However, in these approaches, individual employees are not well distinguished from one another and are in general interchangeable with other employees. In this paper we concentrate on the other extreme: Each employee has unique attributes that are defined for each individual. Similarly, each job has a unique set of requirements. The objective is to fit as many employees to as many jobs as possible while maintaining the best match between the individual employees and the jobs to which they are assigned. This ID&Assign problem is significantly more difficult to automate. To the best of our knowledge, it has not previously been studied using traditional OR or CP techniques.

Constraint programming

CP deals with modeling and solving CSPs. A CSP is an abstract problem that captures the relations between some entities (variables) and the constraints that restrict the values those entities can assume. For example, in an exam-scheduling problem, the variables may be the time and location of each exam in a given semester, and a constraint may specify that no two four-hour exams may be scheduled on consecutive days.

Mathematical formalism

Mathematically, a CSP P is a triplet (V, D, C) consisting of a set of variables V, a corresponding set of domains D, and a set of constraints C. A solution to a CSP is an assignment of a value to each variable out of the domain of the variable such that all constraints are satisfied. A CSP is *satisfiable* if it has at least one solution and *unsatisfiable* otherwise. In the exam-scheduling example, assuming there are N exams, we would have $2N$ variables: one date variable and one location variable for each exam. The domain of the date variables may be the list of days in the exam period, while the domain of the location variables may be the list of rooms in the building. Constraints may specify such things as blackout days for any particular exam, requirements on room sizes, and mutual requirements on neighboring exams. Mathematically, constraints are known as *relations*. A relation on a set of k variables is the list of all legal combinations of k values, each taken from the domain of the corresponding variables. For example, consider three variables a, b, c, with domains $\{1, 2\}$, $\{1, 2, 3\}$, $\{1, 2, 3\}$, respectively. A constraint requiring that the three variables assume different values may be represented by the mathematical relation

$$\{(1,2,3),(1,3,2),(2,1,3),(2,3,1)\}. \quad (1)$$

CSP modeling

CSP modeling is the process of translating a real-world constraint problem into a CSP. It involves identifying the variables in the problem, the variable domains (i.e., the values each variable can have before considering conflicts due to the constraints), and the constraints. There is usually more than one way to choose the variables and domains. For example, in the exam-scheduling problem, we could have chosen the variables to be all combinations of dates and locations. Under this choice, the domains of all variables may be the names of the exams plus "null," signifying that no exam is taking place at this particular date and location.

Choosing an appropriate model is a crucial step in CP. The most important aspect of choosing a model is to make it simple. Variables should correspond to physical entities that are easily identified in the real-world problem. Auxiliary variables, sometimes added to make the constraints appear simpler, should almost always be avoided. Domains should be of the same type as in the real-world problem. For example, if the possible colors of a shoe in a shoe-manufacturing plant are white, black, gray, yellow, and red, the domain of the variable *shoe-color* should be {white, black, gray, yellow, red} and not a numerical coding such as {0, 1, 2, 3, 4}. Finally, the constraints should be specified in a language understandable to the end user, which often excludes sophisticated mathematical notations. Following these guidelines allows simple maintenance of the model when the problem evolves and when variables and constraints are changed or added.

Constraint languages

The constraints in the CSP model should be specified in a way that is close to the physical reality. Specifying the mathematical relation of the constraint is almost always out of the question because the intuition behind the constraint is usually lost this way and because the relation may become very large (i.e., a very large number of legal combinations). The more common way to represent constraints is to use some constraint language. A constraint language may be generic, such as Numerica [30] and the Optimization Programming Language [31], or designed for a specific problem domain. Generic constraint languages usually support arithmetic and logic

operators and comparators, so two constraints in the shoe-plant model may be written as

$$(\textit{shoe-color} = \textit{red})\ \textbf{\textit{implies}}\ (\textit{lace-color} = \textit{red}\ \textbf{\textit{or}}\ \textit{lace-color} = \textit{white})$$

and

$$\textit{number-of-boxes} \geq (\textit{number-of-shoes} + \textit{number-of-sandals})/2\,.$$

Here, *shoe-color*, *lace-color*, *number-of-boxes*, *number-of-shoes*, and *number-of-sandals* are all names of variables; *red* and *white* are domain values; and the reserved symbols (,), ***implies***, ***or***, $=$, $\geq$, /, and $+$ are all part of the generic constraint language.

In addition to simple expressions, constraint languages contain operators (or *global constraints*), which are known to be useful in many constraint problems. The best known example of a global constraint is ***all-different***, which specifies that all variables in a given set must assume different values. By using ***all-different***, the numerical constraint represented by Equation (1) may be specified simply as

$$\textbf{\textit{all-different}}(a, b, c). \tag{2}$$

For any expression or global constraint, a *constraint propagator* must be implemented. A propagator is an algorithm that accepts n domains, where n is the number of variables affected by the constraint, and outputs the same domains after all *unsupported* values are removed. A value of a domain is unsupported if it cannot be extended into a legal combination of the constraint by using any choice of $n - 1$ values, one from each of the other domains. For example, consider the constraint $a + b = c$, where a, b, c are variables with domains $\{1, 2, 4\}$, $\{3, 5, 8\}$, and $\{0, 3, 6, 9\}$, respectively. When a propagator for this constraint accepts these domains as input, it returns as output the new domains: $\{1, 4\}$, $\{5, 8\}$, and $\{6, 9\}$, respectively. If all values of an input domain are unsupported, the propagator returns an empty set for all domains.

Algorithms

Maintain-arc-consistency (MAC) algorithms [32] accept a CSP in which constraints are represented by their propagators, and they output a solution to the CSP, a proof that the CSP is unsatisfiable, or a timeout. They do so by interleaving two types of steps: First, they iteratively call each of the propagators to reduce the domains of variables accepted by the propagator until no domains can be further reduced; second, they instantiate a variable with a value from the reduced domain of the variable, as in a regular search. The propagation step may greatly prune the underlying search tree, thus making the problem tractable. In general, global constraints have a greater pruning ability than an equivalent set of simple-expression constraints.

Sometimes a specific problem calls for special constraints that are not defined in a general-purpose constraint language. In this case, the user must provide a new propagator corresponding to this constraint. Once the propagator is available, it can be used by the MAC algorithm in conjunction with all other constraints. This expandability of the constraint language allows both simpler modeling of the problem at hand and better pruning of the search tree, hence shorter runtimes of the algorithm. In the next section, we discuss a new global constraint that is extremely useful in the ID&Assign problem.

When the best propagation algorithm for a constraint either is intractable or has a weak pruning ability, MAC algorithms become inefficient. In these cases, stochastic local search [33] is often used. Stochastic local search is also used for flexibility, i.e., to repair a solution obtained by MAC when the problem is slightly perturbed and when the solution of the perturbed problem is required to be close enough to the original solution [34]. This is usually the case with the ID&Assign problem, since a small number of people or jobs are likely to leave or change after the problem has been solved, and we do not want to reshuffle the assignments of all other professionals when this happens.

Soft CSP

Pure CSP deals with finding a satisfying solution to the problem, i.e., an assignment to all variables out of the variable domains such that all constraints are satisfied. However, most real-world problems require an optimal solution, not just any solution. There are many ways to expand the CSP definition to incorporate optimality into the problem [35]. One of the most appealing is the soft CSP framework. A soft CSP is the list (V, D, C, C_1, C_2, C_3, $\cdots$, C_N) where (V, D, C) is a regular CSP. The constraints in C are called *hard* constraints, and C_i represents sets of prioritized, or *soft*, constraints. Roughly speaking, a solution to the soft CSP is an assignment to the variables out of the domains such that 1) all constraints in C are satisfied, 2) as many constraints as possible are satisfied in each C_i, and 3) when conflicts between constraints occur, the constraint with the highest priority in the set of conflicting constraints (i.e., the one belonging to C_i with the smallest i) is satisfied [36].

This scheme allows us to specify the optimization criteria in a natural way, as a set of prioritization rules, rather than defining a rigid mathematical cost function that attempts to associate a well-defined numerical cost with the much laxer notion of business preferences.

Applications

Since its inception, CP has been used to solve many real-world problems. This section includes a brief summary of the most important applications. Reviews of practical applications of CP can be found in [37, 38]. Interactive graphics was one of the earliest applications to apply computers to constraint problems. Sketchpad [39] and the follow-on ThingLab [40] were interactive graphics applications that allowed the user to draw and manipulate constrained geometric objects. These systems contributed to the development of local propagation methods and constraint compiling. The scene-labeling problem [41] is probably the first CSP that was formalized as such. The goal was to recognize the objects in a three-dimensional scene by interpreting lines in the two-dimensional drawings.

Assignment and allocation problems were the first type of industrial application solved with constraint tools [37]. Typical examples are counter allocation for departure halls [42] and berth allocation to ships in a harbor [43]. Typical scheduling problems solved by constraint tools are petroleum-well activity scheduling [44], forest treatment scheduling [45], production scheduling in the plastics industry [37], and production planning of military and business jets [46]. Network management and configuration problems include planning and configuration of telecommunication or electric power networks [47] and optimal placement of base stations in wireless indoor telecommunication networks [48]. Database applications use related CP ideas [49, 50], and CP methods are employed in relation to program testing [51–53]. Hardware verification is a large modern application field of CP. A full-fledged industrial application based entirely on CP is presented in [54].

There have been many works aimed at reducing problems from other domains of knowledge to the CP framework in order to utilize existing powerful CP algorithms. The two broadest cases are the reformulation of optimization problems [35, 55] and satisfiability problems [56] as CSPs.

CSP model for the ID&Assign problem

We model the most basic ID&Assign problem as a soft CSP.

Variables: The set of variables V corresponds to the set of job positions. For each job, there is a variable in V.

Domains: The domain of each variable in V is the set of professionals who can perform the job. This set of professionals is found by iterating over all professionals, checking each individual to see whether he or she can perform the job according to the specifications of the job and the credentials of the professional. For example, if the only requirement specified by some job is for the professional to be a C++ or a Java** programmer, the domain of the variable corresponding to this job will include all professionals skilled in either C++ or Java.

Constraints: The only hard constraint in our basic model is a ***some-different*** constraint applied to all variables. The ***some-different*** constraint is discussed below. It is used to ensure that the same professional is not assigned to two jobs taking place at the same time.

Soft constraints: Any preference defined in the problem is modeled as a set of soft constraints on any of the variables. For example, if a job requires either a C++ or a Java programmer but prefers a C++ programmer, a soft constraint with priority 1 is added, and its propagator removes all professionals without C++ skills from the input domain. This way, if a C++ programmer is available, he or she will be matched by the constraint. However, if no C++ programmer is available, the constraint will eventually drop out of the model (because it is only a soft constraint and cannot be satisfied), and a non-C++ programmer may be assigned to the job. More complex is the case of a continuous preference. For example, suppose there is a preference that the professional live as close as possible to the job location. In this case, the following set of soft constraints, with their listed priorities and legal domain values, is added to the model:

Priority 1: Person closest to the job location.
Priority 2: Two people closest to the job location.
...
Priority $n-1$: $n-1$ people closest to the job location.

In the case in which there are two (or more) preference criteria (e.g., prefer both a C++ programmer and a professional living nearby), the problem should specify which criterion is more important. The corresponding soft constraints are then given the appropriate priority number.

The *some-different* constraint

The ***all-different*** constraint is a fundamental primitive in CSP [57]. This constraint is defined over a subset of the variables and requires that they are assigned different values. Many classical CSPs are modeled using this constraint, the n-queen problem[1] being the canonical example, along with air traffic management [58, 59], rostering problems [60], and many more. The semantics of a single ***all-different*** constraint over n variables may be preserved by replacing it with $n(n-1)/2$ binary ***not-equal*** constraints. However, in the context of MAC algorithms, the single ***all-different*** constraint is vastly more powerful in pruning the search space, leading to a great reduction

[1]The n-queen problem is to find a configuration of n chess queens on an $n \times n$ board so that no queen attacks any other queen; ***all-different*** constraints enforce that no two queens are on the same row, column, and diagonal.

in the number of backtracks, and hence reaching the solution more rapidly.

For our ID&Assign problem, ***all-different*** is too restrictive. In fact, we would like only some of the pairs of variables to be assigned different values. For example, suppose our list of job positions specifies that different jobs start and end at different times or require only partial availability of a worker. In these cases, the same worker may be assigned, in general, to multiple jobs, thus violating ***all-different*** on the full set of variables.

Within the framework of our basic model, one way to model this type of behavior is to add a binary ***not-equal*** constraint for any two jobs that overlap in their time of execution (assuming for simplicity that all jobs require full-time workers). However, this model suffers from the same disadvantages of modeling an ***all-different*** problem by multiple ***not-equal*** constraints. Partitioning the jobs according to their periods of execution and adding an ***all-different*** constraint for each partition does not help because job B may overlap with both jobs A and C, even though jobs A and C may not overlap and can therefore be assigned the same professional. All of this calls for a generalization of ***all-different***.

We recently defined a new type of global constraint that we call ***some-different*** [61]. This constraint was designed to address the above modeling problem, but in actuality its scope is wider and it is relevant to many other real-world problems. The ***some-different*** constraint is defined over a set of variables $X = x_1 \cdots x_n$ with domains $D = D_1 \cdots D_n$, respectively, and an underlying graph $G = (X, E)$. That is, the nodes of the graphs are the set of variables, and the edges are given explicitly. The legal combinations allowed by the constraint are all values out of the domains so that no two values of variables connected by an edge are equal:

$$\textbf{\textit{some-different}}(X, D, G) = [(a_1, \cdots, a_n): a_i \in D_i, a_i \neq a_j \text{ for all } (i, j) \in E(G)]. \quad (3)$$

The ***all-different*** constraint is the special case of ***some-different*** in the case in which G is a clique[2].

For ***all-different***, there exists a polynomial propagation algorithm [62]. Its runtime is $O(mn^{1/2})$, where n is the number of variables and m the sum of all domain sizes. Unfortunately, there is little hope of finding a similar polynomial algorithm for ***some-different***, since it contains the NP-hard problem of graph three-colorability as a special case. Nevertheless, we designed special heuristics into the ***some-different*** propagator. This allowed us to show that for all real ID&Assign problems we worked on, the ***some-different*** propagator was not only tractable but extremely fast.

[2]A *clique* is a set of nodes in a graph such that there exists an edge between any two nodes in the set.

In [61], a detailed theoretical analysis of the ***some-different*** propagator was performed. The results can be summarized thus: We introduced an exact propagation algorithm for hyper-arc consistency of the ***some-different*** constraint. The algorithm has time complexity of $O(n^3\beta^n)$, with $\beta \approx 3.5$, and depends on the domain sizes only for unavoidable deletion operations. We implemented the algorithm (with multiple additional heuristics) and tested it on two kinds of data: our real-life WM instances and synthetic data generated through a random graph model. In both cases the implementation performed very well, much better than expected from the theoretical bounds. Specifically, the implementation propagated instances that included 250 to 300 variables in less than a second.

The results of [61] that are relevant to this paper are summarized in **Figures 1(a)–1(c)**, which show the potential efficiency of using ***some-different*** compared with using the equivalent model composed of a multitude of binary ***not-equal*** constraints. Figure 1(a) shows the runtime of the ***some-different*** propagator on WM data instances as a function of the ***some-different*** graph size. Some of the instances shown were satisfiable, while others were unsatisfiable. Figure 1(b) shows the runtime of the CSP solver on the ***some-different*** model, and Figure 1(c) shows the speedup factor relative to an equivalent model composed of ***not-equal*** constraints on WM data instances. The rising curve in Figure 1(b) is composed of satisfiable instances, while the flat curve is composed of unsatisfiable ones. Figure 1(c) shows that most instances are solved much more rapidly using the ***some-different*** propagator. However, some instances are up to a factor of 2.5 slower with this model. Experiments were performed on a Linux** machine running an Intel Pentium** 4 at 3.6 GHz. A more detailed discussion of Figures 1(a)–1(c) can be found in [61].

Beyond the basic model

The basic model includes only two types of constraints. One type is constraints on the match quality of a specific professional to a given job. These are soft constraints, because the hard constraints the professional must match are taken into account implicitly by having the domain of each job include only the legally matched professionals. The other type is the ***some-different*** constraint, which ensures that a professional is not simultaneously assigned to two jobs.

The basic model is at the heart of all ID&Assign problems with which we have dealt. However, in almost all cases, there are additional, more-complex constraints that are part of the model and coexist with the basic constraints mentioned above. These additional constraints reflect the set of business rules that govern

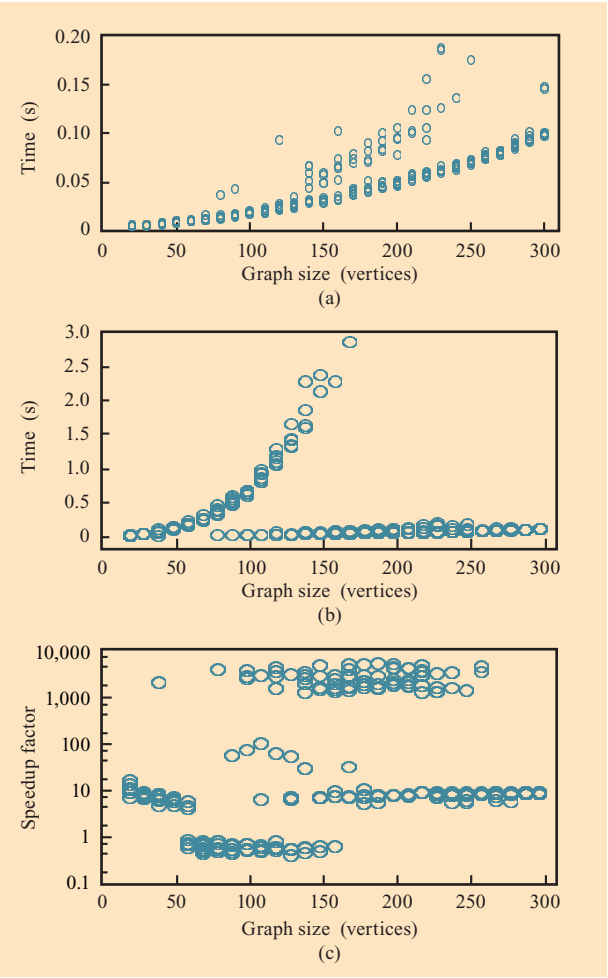


Figure 1

Using ***some-different*** compared with using the equivalent model composed of a multitude of binary ***not-equal*** constraints: (a) Runtime of the ***some-different*** propagator on workforce management data instances as a function of the ***some-different*** graph size; (b) runtime of the CSP solver on the ***some-different*** model; (c) speedup factor relative to an equivalent model composed of ***not-equal*** constraints on workforce management data instances. From Figures 1 and 2 of [61], © 2006 Springer Science and Business Media, reproduced with permission.

the assignments, as discussed below in the section on workforce matching rules. The simplest example: A team-forming rule which specifies that two specific persons (Dave and Mary) cannot work on the same team may be modeled as a constraint of the form ***at-most-one***(Dave, Mary) and applied to all variables forming a single team. Similarly, all of the different rules discussed in the section on workforce matching rules may be modeled as constraints (either hard or soft) and applied to the variables relevant to those rules.

Unsatisfiable problems

In general, a CSP may be unsatisfiable: i.e., two or more constraints conflict. In many cases, a CSP solver can identify that the problem is unsatisfiable. In this case it reports "unsatisfiable" and stops. In fact, most complex instances of the ID&Assign problem are bound to be unsatisfiable. There are two explanations for this. First, many conflicting rules can come from the various levels and organizations in the business. Second, even without any conflict in rules, for many job positions a single professional who matches the requirements may not be found. This means that the domain is left empty, and technically this implies the unsatisfiability of the CSP.

Of course, any real ID&Assign application cannot just report "unsatisfiable." The user obviously prefers to see a partial assignment to the set of jobs as opposed to no assignments at all. We enhanced the original CSP model so that it is always satisfiable. In addition to the regular professionals, we define fictitious professionals who are initially part of the domain of all variables. We also add soft constraints favoring actual professionals over the fictitious ones. There are no explicit constraints acting on the fictitious values. Implicitly, the soft constraints favoring actual professionals may be seen as acting on the fictitious professionals and removing them from the domain. However, all other matching constraints do not remove any fictitious person from any domain. Under this model, because of the soft constraints, a real professional is chosen whenever possible, but a fictitious professional is chosen when all real persons have been removed by the regular constraints. Because the CSP solver treats both real and fictitious professionals as legal domain values, it does not report "unsatisfiable" when the domain of a variable is left with only a fictitious person and continues to solve the problem. After the solver returns with a solution, a simple procedure removes all fictitious persons and reports to the user only the assignments of actual professionals.[3]

Workforce matching rules

This section presents the plethora of workforce matching rules that are at the basis of the ID&Assign problem. These rules can be classified according to several characteristics:

- *Rigidity*—Are the rules mandatory or merely nice to have?
- *Scope*—Do they apply to single individuals or to whole teams?

[3]Another way to solve the "unsatisfiable" problem is to run the solver a few times, each time removing all jobs whose domain became empty in the previous run until the problem becomes satisfiable. This procedure is likely to result in fewer matches than by using our fictitious persons scheme, because when the search is backtracked, jobs that were removed in the previous iteration are no longer available to be filled.

- *Level of definition*—Are they derived from corporate strategy or from an RDP's decision?
- *Complexity*—Are they a simple matching of attributes or do they represent a complex process?
- *Explicitness*—Are they defined explicitly by the user or are they expected to be taken into account implicitly?

Below we follow this classification and demonstrate it with concrete examples. We also discuss the impact of the various types of rules on the CSP model.

Rigidity of rules

Rules can be mandatory. For example, a job may require a person with project management skills. However, rules can also be nonmandatory, or nice to have. For example, a job may have preference for a person with project management skills, but will settle for one without such skills if all mandatory rules are satisfied.

All mandatory rules are of the same importance, and all should be satisfied equally. Nonmandatory rules can be defined in a ladder of importance. For example, it may be more important to live close to the place of employment than to have experience in programming (or vice versa). Whoever sets the nonmandatory rules should also specify this ladder of importance between them.

Deciding on the relative importance of rules is a rather intuitive task. This amounts to prioritization, which is what people naturally do when they have to decide informally among a few imperfect options. Once the prioritization is known, the optimality of a solution can be crudely defined as the solution that best satisfies the prioritization scheme.

In contrast, other optimization schemes do not allow the user to define the relative importance of rules but require a numeric cost defined for any complete assignment of professionals to all jobs. In many cases, it is quite unnatural to quantify a violation of a rule with a specific numeric cost. In the example given above, it is unreasonable to assume that the user can actually quantify the cost of living far away from the job, because, in addition to the dollar cost of travel, the cost includes such factors as the dissatisfaction of the worker and the reduced likelihood of his or her working extra hours.

The natural scheme of specifying mandatory rules and a prioritized list of nonmandatory rules fits nicely with the formalism of soft CSP. Mandatory rules are mapped to hard constraints, while the list of nonmandatory rules, together with the prioritization of each such rule, is mapped to the Borning hierarchy [36] of prioritized soft constraints. This way, no cost function has to be defined. Indeed, once we have a soft-CSP model of the problem, it is up to the soft-CSP solver to produce an optimized solution.

Scope of rules

Rules may apply to individuals or to teams. For example, a skills rule may state that a particular job requires a person with medical qualifications. However, it may also state that for a given set of n jobs, at least two persons filling the jobs should have medical qualifications, but it need not specify which two.

Rules applied to sets of jobs may be used to form teams or to find a suitable professional to fill a vacant position in an already existing team. As with rules on individual matches, team-matching rules can be either mandatory or nonmandatory. For example, a nonmandatory rule may be applied to a previously successful team to keep the team together in the next assignment. However, if the team as a whole cannot fit any of the new job opportunities, this nice-to-have rule may be violated, and the team can be spread to different projects.

As a CSP, rules on matching individuals may be modeled as unary constraints on the jobs to which the rule applies, whereas rules on matching teams to projects can be modeled as k-ary constraints on the jobs comprising the project, with $k \leq n$, where n is the number of individual jobs in the project.

Definition level of rules

Rules can be defined at the lowest RDP level. For example, an RDP who owns a particular job position may impose rules derived directly from the specifications of the job. The rules discussed above are all examples of such rules (special requirements for skills or experience, live near the job location, form a winning team by combining certain individuals, and so on).

However, rules can also be imposed by higher authority levels in the organization. For example, an organization within the corporation may impose a reservation rule stating that no more than 90 percent of top professionals should be assigned at any given time. This implies the constant availability of ten percent of the leading professionals in case an emergency request from a must-fill job arrives.

An example of a corporate-level team-building rule may be that it is forbidden for spouses to be assigned to the same team. Note that both of our examples of rules defined at the higher level are in conflict with the low-level RDP rules. In fact, the RDP would prefer to use the top professional from the reserve if he or she fits the job requirements and would also want to assign both spouses if they are the best match for the team. This illustrates a common phenomenon in which organization-level, or *strategic*, rules often violate the decisions made by RDPs—hence the importance of specifying those rules. Without them, it is likely that low-level decisions would violate many of the standards of the corporation and its vision and strategy. Needless to say, the potential conflict

among rules at different levels is bound to generate some level of friction among the various position holders. Therefore, in addition to implementing robust practices to ensure enforcement of the higher-level rules, it is absolutely necessary to create a rewarding system for RDPs that renders it in their own best interest to abide by all levels of rules.

CSP modeling lends itself naturally to rules originating from different levels or different organizations. The CSP model is declarative, with constraints being added to the model while the solution algorithm remains the same. Therefore, it allows the seamless addition of constraints from different sources. In fact, the different organizations need not even know of the participation of all other organizations in building the CSP model. Constraints can be added in any order without affecting the solution process as long as they are all added before the solver is called to solve the model. Finally, within the soft-CSP framework, it is easy to enforce the overriding of rules by some level over another. This can be done by allocating different ranges of priorities to soft constraints entered by different organization levels. By applying a set of different prioritization levels, we can define a complete hierarchy of overriding rules that mirror the complete hierarchical structure of the organization.

Complexity of rules

Rules can be as simple as matching a single attribute of a job to a corresponding attribute of a professional. For example, a rule for a job requiring a person with a pay scale lower than an annual salary of $90K can be implemented by comparing the max-pay-level attribute of the job with the pay-level attribute of the person.

However, rules can also be quite complex and sometimes require access to databases or elaborate calculations. Note that this complexity can arise even with rules defined on individual jobs. The complexity we are addressing here is different from the implicit complexity of rules defined on more than one job. We illustrate this with two examples of complex rules.

The first example is the rule which states that the professional must reside within a specific distance from the place of employment. While easy to state, this rule is difficult to enforce automatically, for two reasons. First, in order to enforce this rule, the locations of the job and the professional must be known. However, it is sometimes difficult to obtain trusted information about this location. In many cases, zip codes, longitude and latitude data, or other well-defined location definers are not available as part of the person or job descriptors. In these cases, the location entry may be entered as street address, city, county, state, and country. Since the number of cities is huge and since many cities (especially in non-English-speaking countries, and assuming that data is entered in English characters) have alternative spellings, it is possible that the same location may be spelled differently for the job and the professional. A naive application will then recognize the two locations as different. A more reasonable application will invest in complex text analysis as part of the matching algorithm. A second problem is that even if two locations are recognized correctly, it is not clear how to calculate the distance between them. Here again, the problem may be solved by increasing the complexity of deciding whether the rule is violated, e.g., by looking at huge interlocation-distance databases. Another approach, which requires less computer space but provides only an approximate result, is to calculate the distance. Given the name of the location, we can determine the longitude and latitude information by searching existing databases. The shortest distance between any two locations can then be calculated according to the formula

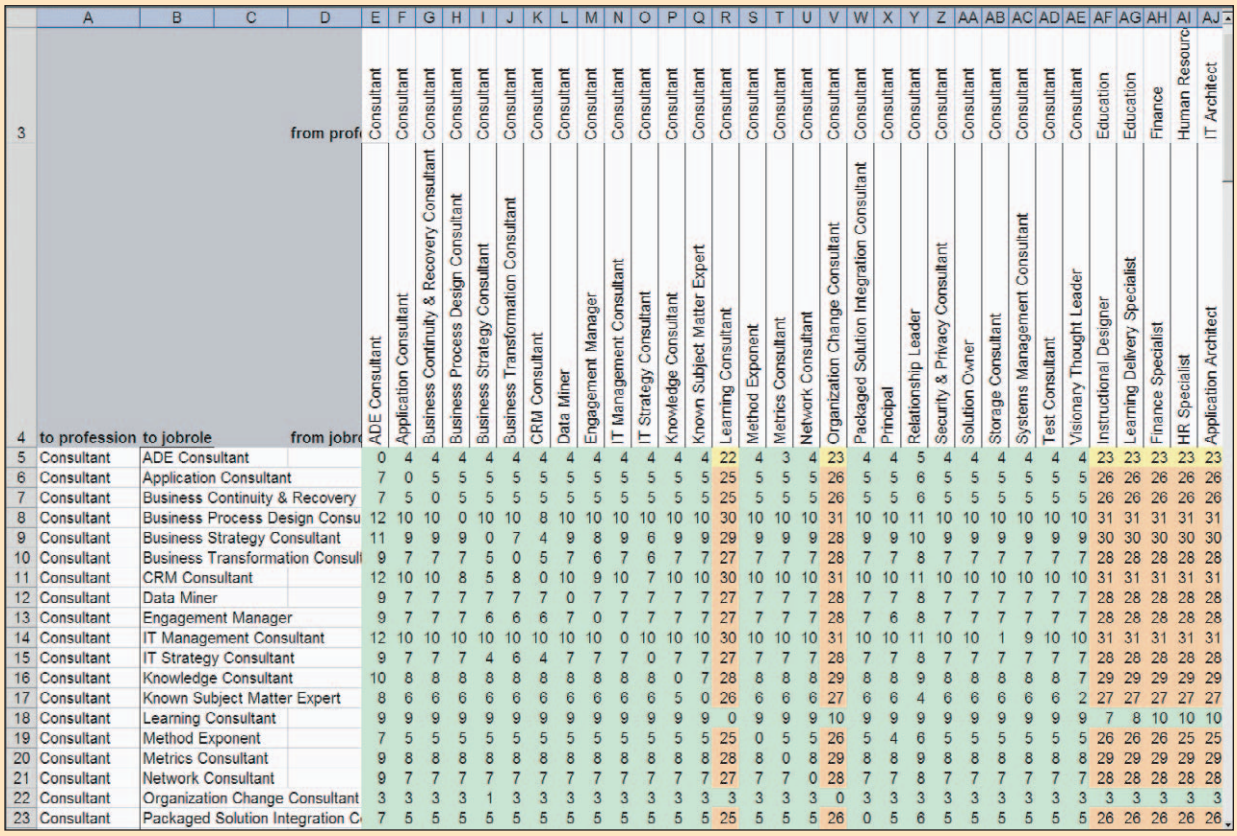
$$Distance = R\,\mathrm{acos}[\sin\Theta_1 \sin\Theta_2 + \cos\Theta_1 \cos\Theta_2 \cos(\varphi_1 - \varphi_2)],$$

where Θ_1 and Θ_2 are the latitudes of the first and second locations, respectively, and φ_1 and φ_2 are the longitudes of those locations.

In practice, we multiply this number by a factor of 1.3 (found empirically for suburban areas) to estimate the actual road distance. The rule must be made even more complex if the area contains water barriers, such as rivers or lakes.[4]

The second example is a rule stating that the skills required by the job must be "close enough" to skills attained by the professional. Here the problem is in the definition of "close enough." How can one obtain a clear definition of the proximity between any two sets of skills? In our initial model, we required that the job role and skill set of an individual be an exact match to the job requirements. As the model matured, our users asked if we could incorporate the raising of personal skill levels or retraining as matching factors. We implemented a rule stating that skills required by the job must be "close enough," which was defined by our human resources (HR) subject-matter experts. **Figure 2** shows one approach to this problem. In this example, our HR partners defined the distance between any two job roles by the number of additional skills one would have to acquire in order to raise one's skill level from one of the job roles to the other, and by color, which specifies more crudely whether the transition is easy (green), medium (yellow), or hard (orange). Our HR partners continue to refine this distance matrix to

[4]Although there are popular Internet-based applications in various countries that offer travel directions and distance calculators, there are a number of obstacles to solving this problem on the basis of such applications. However, it is a good direction for further exploration.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ
3				from prof	Consultant	Consultant	Consultant	Consultant	Consultant	Consultant	Consultant	Consultant	Consultant	Consultant	Consultant	Consultant	Consultant	Consultant	Consultant	Consultant	Consultant	Consultant	Consultant	Consultant	Consultant	Consultant	Consultant	Consultant	Consultant	Consultant	Consultant	Education	Education	Education	Finance	Human Resourc	IT Architect
4	to profession	to jobrole		from jobr	ADE Consultant	Application Consultant	Business Continuity & Recovery Consultant	Business Process Design Consultant	Business Strategy Consultant	Business Transformation Consultant	CRM Consultant	Data Miner	Engagement Manager	IT Management Consultant	IT Strategy Consultant	Knowledge Consultant	Known Subject Matter Expert	Learning Consultant	Method Exponent	Metrics Consultant	Network Consultant	Organization Change Consultant	Packaged Solution Integration Consultant	Principal	Relationship Leader	Security & Privacy Consultant	Solution Owner	Storage Consultant	Systems Management Consultant	Test Consultant	Visionary Thought Leader	Instructional Designer	Learning Delivery Specialist	Finance Specialist	HR Specialist	Application Architect
5	Consultant	ADE Consultant			0	4	4	4	4	4	4	4	4	4	4	4	4	22	4	3	4	23	4	4	5	4	4	4	4	4	4	23	23	23	23	23
6	Consultant	Application Consultant			7	0	5	5	5	5	5	5	5	5	5	5	5	25	5	5	5	26	5	5	6	5	5	5	5	5	5	26	26	26	26	26
7	Consultant	Business Continuity & Recovery			7	5	0	5	5	5	5	5	5	5	5	5	5	25	5	5	5	26	5	5	6	5	5	5	5	5	5	26	26	26	26	26
8	Consultant	Business Process Design Consu			12	10	10	0	10	10	8	10	10	10	10	10	10	30	10	10	10	31	10	10	11	10	10	10	10	10	10	31	31	31	31	31
9	Consultant	Business Strategy Consultant			11	9	9	9	0	7	4	9	8	9	6	9	9	29	9	9	9	28	9	9	10	9	9	9	9	9	9	30	30	30	30	30
10	Consultant	Business Transformation Consult			9	7	7	7	5	0	5	7	6	7	6	7	7	27	7	7	7	28	7	7	8	7	7	7	7	7	7	28	28	28	28	28
11	Consultant	CRM Consultant			12	10	10	8	5	8	0	10	9	10	7	10	10	30	10	10	10	31	10	10	11	10	10	10	10	10	10	31	31	31	31	31
12	Consultant	Data Miner			9	7	7	7	7	7	7	0	7	7	7	7	7	27	7	7	7	28	7	7	8	7	7	7	7	7	7	28	28	28	28	28
13	Consultant	Engagement Manager			9	7	7	7	6	6	6	7	0	7	7	7	7	27	7	7	7	28	7	6	8	7	7	7	7	7	7	28	28	28	28	28
14	Consultant	IT Management Consultant			12	10	10	10	10	10	10	10	10	0	10	10	10	30	10	10	10	31	10	10	11	10	10	1	9	10	10	31	31	31	31	31
15	Consultant	IT Strategy Consultant			9	7	7	7	4	6	4	7	7	7	0	7	7	27	7	7	7	28	7	7	8	7	7	7	7	7	7	28	28	28	28	28
16	Consultant	Knowledge Consultant			10	8	8	8	8	8	8	8	8	8	8	0	7	28	8	8	8	29	8	8	9	8	8	8	8	8	7	29	29	29	29	29
17	Consultant	Known Subject Matter Expert			8	6	6	6	6	6	6	6	6	6	6	5	0	26	6	6	6	27	6	6	4	6	6	6	6	6	2	27	27	27	27	27
18	Consultant	Learning Consultant			9	9	9	9	9	9	9	9	9	9	9	9	9	0	9	9	9	10	9	9	9	9	9	9	9	9	9	7	8	10	10	10
19	Consultant	Method Exponent			7	5	5	5	5	5	5	5	5	5	5	5	5	25	0	5	5	26	5	4	6	5	5	5	5	5	5	26	26	26	25	25
20	Consultant	Metrics Consultant			9	8	8	8	8	8	8	8	8	8	8	8	8	28	8	0	8	29	8	8	9	8	8	8	8	8	8	29	29	29	29	29
21	Consultant	Network Consultant			9	7	7	7	7	7	7	7	7	7	7	7	7	27	7	7	0	28	7	7	8	7	7	7	7	7	7	28	28	28	28	28
22	Consultant	Organization Change Consultant			3	3	3	3	1	3	3	3	3	3	3	3	3	3	3	3	3	0	3	3	3	3	3	3	3	3	3	3	3	3	3	3
23	Consultant	Packaged Solution Integration C			7	5	5	5	5	5	5	5	5	5	5	5	5	25	5	5	5	26	0	5	6	5	5	5	5	5	5	26	26	26	26	26

Figure 2

Provisional distance to upgrade skills between any two job roles.

consider factors such as the cost of the course work and the time required to complete the course. The availability of this table permits rules to be defined that relate to the fitness of the professional with respect to the required job role. For example, it is now simple to include individual preferences for career paths as an additional matching factor.

Before building a CSP model, one must define and implement the types of constraints required by that model. A constraint is implemented by implementing its propagator, which can be very simple or very complex. Indeed, in the case of the complex rules defined above, the propagator may become quite complex in itself, performing such activities as analyzing text, accessing databases, and using expert knowledge. However, the complexity is localized in a single function in the model. This ensures the simplicity of the design and the ease of CSP model maintenance, even when the underlying logic is complex.

Explicit and implicit rules

The rules discussed above were defined explicitly by a person or persons whose responsibility is to find a good assignment under the various conditions and regulations in the company. However, there are other types of rules that are not stated explicitly by anyone but should still be enforced.

One such rule is the requirement that the same professional not be assigned to two jobs that overlap in time and that together require more total time than the professional has available. This rule is mandatory. An example of a nonmandatory implicit rule is that as many job positions as possible will be assigned a professional.

Such rules can easily be enforced within the CSP framework by having the application builder incorporate constraints into the CSP model. These implicit constraints are not seen by the application users who define the explicit rules and run the solver to find the assignments. However, since these constraints are part of

Table 1 Definitions of (a) rules and (b) priorities for the Workforce Matching Tool. Table inputs are provided by a special-purpose graphical user interface.

(a) *Matching rules*

Index	*Name*	*Value*
1	Match on primary and secondary skill sets	Yes
2	Match on location	Yes
3	Consider part-time jobs	No
4	Consider part-time employees	No
5	Minimum duration of job (in days)	30
6	Match on required languages	No
7	Maximum travel distance (in kilometers)	50
8	Maximum upskilling allowed	10
9	Maximum slack in band	1
10	Maximum absolute slack in late arrival (in days)	14
11	Maximum relative slack in late arrival (percentage)	10
12	Late arrival determined by larger or smaller between absolute and relative forms	larger

(b) *Priorities*

Index	*Category*
1	Late arrival
2	Band in range
3	Criticality
4	Lowest band

the CSP model, they are enforced by the solver just like any other constraint.

Use by IBM service organizations

Workforce matching tool

In 2005, we began implementing the concepts and methods discussed here in the Workforce Matching Tool (WMT) designed to solve the ID&Assign problem. The tool is based on the IBM state-of-the-art constraint solver [54]. We applied this tool to a series of assignment problems encountered by IBM service organizations. These organizations employ more than 100,000 highly skilled professionals. The professionals are typically assigned to jobs at customer locations and engage in either information technology (IT) infrastructure work or business consulting. A typical work assignment can last from a few weeks to a few years. Teams of professionals are commonly formed to address project needs. Many of the examples of rules and constraints discussed above were specified by actual users and were required in order to resolve some real-world scenarios.

ID&Assign problem at IBM

Since the implementation of the WMT, we have conducted many pilot projects, experiments, and actual work with the various service organizations. Each such experience was unique with respect to the type of data we received and the set of rules and prioritization schemes defined by the users. However, some common attributes were the same in almost all of the experiments.

The common job attributes included in the open seats table are unique-identifier, required job role, required skill set, lowest pay rate, highest pay rate, start date, end date, location (city, state, country), indication for the possibility of working remotely, and contact e-mail.

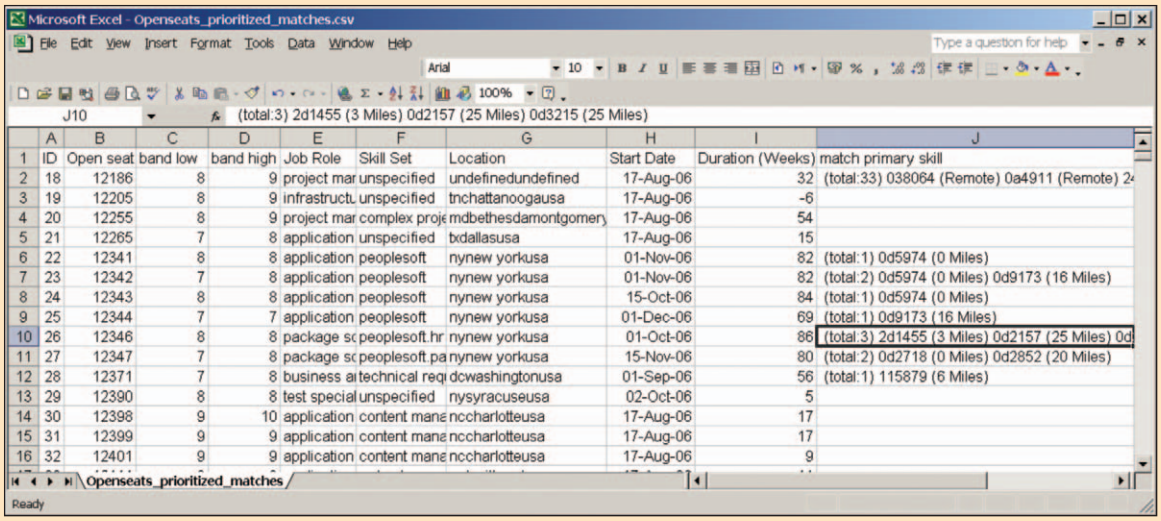
The common attributes included in the professionals table are unique identifier, name, primary job role, secondary job roles, skill set, availability date, pay rate, location (city, state, country), and contact e-mail.

Table 1(a) shows the definition of rules; here mandatory rules are defined and parameters for those rules are set. For example, rule number 10 states that a person can still be considered for a job even if he or she is not available to start working until up to 14 days after the job has started. In **Table 1(b)**, the prioritization scheme is set; it shows that the first priority is to find the professional who is least late to the job, the second priority is to have the professional's band (or pay rate) be within the job specification range [note that a slack of 1 in band is allowed by rule number 9 in Table 1(a)], and so on. (We use the term *slack* to indicate the allowed discrepancy from an exact match between the professional and the job specification.) Users can change the values of rules and add or remove prioritization criteria before pushing "save and execute" to run the tool.

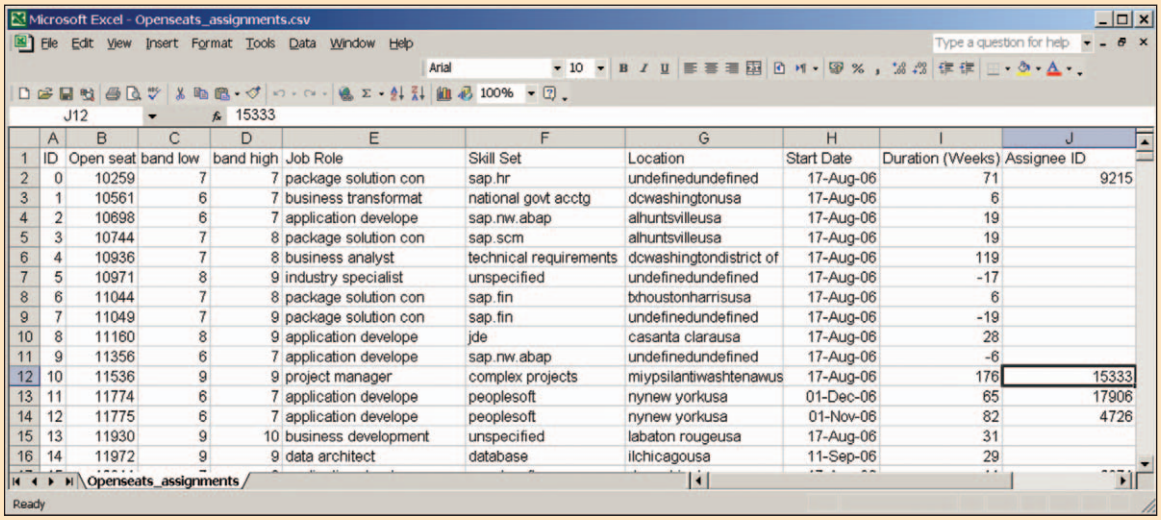
Results

The matching tool can work in two modes: prioritized matching or assignment. In *prioritized matching* mode, the output of the tool is a list of possible matches for any job, prioritized according to the prioritization scheme defined by the user. An example of the output of this mode is shown in **Figure 3(a)**. For any job, a list of matching professionals is given in column J; the list may be empty if no professional matches the job. This list is prioritized so that in a typical use, an RDP seeking to fill a job would start by considering the first professional in the list for the job, then the second, and so forth. A similar output is created for all professionals, except that in the results column, all *jobs* matching the professional are listed in order of priorities.

Technically, the lists are generated by reaching arc consistency over all explicit mandatory constraints



(a)



(b)

Figure 3

Results for (a) prioritized matches; (b) assignments to jobs.

defined by the user and then ordering the lists by considering all prioritization criteria defined by the user. This approach ensures that only legal matches are listed (otherwise, arc consistency is violated on some explicit constraint) and that all such matches appear (because no domain reduction was performed beyond arc consistency). Implicit constraints, such as ***some-different*** on all overlapping jobs, are not considered in this model because they do not enforce user-defined rules, but rather consistency of the full problem.

In *assignment* mode we go one step further and let the automation perform the actual assignments for the jobs. An example of the output of this mode is shown in **Figure 3(b)**. For any job, at most one matching professional is listed in column J. In contrast to the prioritized matching case, a professional is never assigned here to two jobs that overlap in time (assuming that all jobs are full-time). Technically, the list of assignments is just the solution of the soft CSP outlined in the constraint programming section above, which takes into account the user-defined rules, the prioritization scheme, and

the implicit built-in constraints. This ensures that the assignments in Figure 3(b) are near-optimal in the sense that the largest number of possible assignments is met and the prioritization scheme is respected.

To provide concrete numbers, we report on a single use case performed on a large set of service organizations that can share professionals among them. Altogether, the problem specified 24,480 professionals to be matched to 703 jobs under the set of rules specified in Table 1. The results obtained showed that 218 jobs had at least one matching professional, and 574 professionals were found to match at least one job. Globally consistent assignments were found between 176 jobs and professionals. This number is high because the original 706 jobs came from a source of jobs with particularly high demands. The runtime to solve both prioritized matching and complete assignment modes was 146 seconds on a Linux machine running an Intel Pentium 4 processor at 3.6 GHz. Much of this time is spent on input and output analysis, including loading of databases. The runtime to solve the CSP by itself was less than a second, which implies the possibility of a real-time mode of operation.

To demonstrate the power of CP for this problem, we also analyzed the case in which jobs are considered one at a time, according to some predefined random order. For each job considered, the best-matching professional is found and assigned to the job. Once a professional is assigned, he or she is no longer available to be assigned to another job that overlaps in time with the first, even if this professional is more suited for the second job. This process of *sequential assignment* simulates the actual process often deployed by RDPs when they find assignments for jobs. With this process, and considering the full data (24,480 professionals, 703 jobs), only 152 jobs were assigned. This is a reduction of 13 percent in the number of assignments compared with the 176 assignments obtained by CP.

Feedback from RDPs within the service organizations with which we worked gave us additional insight. First, our match quality can be only as good as the data we obtain. Sometimes the data was inaccurate or incomplete. In such cases, the matching results were not of great value, since the RDP may have had to check three or four names listed before finding a professional that truly matched.

Second, many of the rules should be defined by the lowest-level person performing the match, i.e., the RDP. The reason is that different RDPs have different ways of looking at the data and therefore require different types of rules. For example, some RDPs prefer that slack in the availability of the professional be defined in absolute numbers of days, while others prefer to express this value in terms relative to the duration of the work—hence the two types of rules (9 and 10) seen in Table 1.

Third, there was some uncertainty about the type of output that is preferred: prioritized matches or assignments. RDPs usually first preferred the prioritized match format because they felt they had more control over the actual choices performed. However, after becoming familiar with the tool, some reported that the assignments format was preferable because it made their decision process simpler. Supporting this was the fact that the number of false positives (i.e., suggestions for assignments that were found to be wrong after considering data not available to the tool) on the full assignments reported was small enough to be tolerable, especially in the cases in which the quality of data was good.

Finally, after gaining some experience with the tool, the RDPs recognized how they can work with the tool in a robust interactive way by disabling, enabling, and changing rules. This allowed for overrides and exceptions to rigid rules, which inevitably occur in this domain.

Summary and conclusions

This paper discusses a CP approach to the ID&Assign problem of workforce management. This problem has severe consequences if it is not solved properly, and it is expected to be even more crucial as the IT industry shifts toward services and as business requirements become increasingly demanding.

For various reasons, CP is found to be highly appropriate for modeling and solving this problem. First, the rules of the ID&Assign problems are complex and are changing over short time scales. This implies frequent maintenance of the model, which in turn requires modeling that is close to the problem domain and not stated in mathematical form. CP provides exactly this type of modeling construct. Second, CP, through the notion of soft constraints, permits optimization of a solution even without defining a rigid mathematical cost function. Third, since powerful pruning algorithms and search heuristics exist, in most cases runtime remains well below the worst case of this NP-hard problem.

To provide the best assignments, input data must be accurate and well defined. This statement is true of any automated process, not just those that are CP-based. Hence, data architects should make every effort to have data provided in a trusted-source manner and not as free text. For example, architects should always choose the use of pull-down menus with enumerated options over free-text description fields. It is also important to specify geographical locations in terms of zip codes or work-location codes, rather than city or street names, which are bound to have multiple spellings and are natural inhibitors of automation (e.g., entering "Northern Georgia" for a city name). Once all data comes from

trusted sources, it is best to have as many data fields as possible: the more the better. In contrast to human agents, the automatic process is never overwhelmed by a large number of fields, and adding more fields can only make the definition of rules and prioritization closer to what the user has in mind.

Finally, resumes should become much more structured documents, possibly created by resume-building tools with predefined options and pull-down menus as trusted sources. We are beginning to see this trend, and it should be highly encouraged. While this structuring may reduce the personal touch somewhat, enabling automated identification and assignment may greatly increase the quality of the jobs found for the resume writer, and it can speed up the process of finding jobs for them. Eventually, professionals should understand that it is in their own best interest to have at least one version of their resume written in a precise machine-readable manner.

Acknowledgments

We are grateful to Dan Forno from the IBM Workforce Management Initiative, who sponsored the research and championed the work. We also thank Eric Andersen, Steve Heise, Mike McInnis, and numerous other individuals in IBM service organizations who invested much time in analyzing our technology and providing valuable feedback. We thank Ari Freund for participating in the study of the ***some-different*** constraint.

References

1. R. Cerulli, M. Gaudioso, and R. Mautone, "A Class of Manpower Scheduling Problems," *Math. Methods Oper. Res.* **36**, No. 1, 93–105 (1992).
2. D. Lesaint, C. Voudouris, N. Azarmi, I. Alletson, and B. Laithwaite, "A Field Workforce Scheduling," *BT Technol. J.* **21**, No. 4, 23–26 (2004).
3. R. L. Kliem and H. B. Anderson, "Teambuilding Styles and Their Impact on Project Management Results," *Project Manage. J.* **27**, No. 1, 41–50 (1996).
4. G. Eitzen, D. Panton, and G. Mills, "Multi-Skilled Workforce Optimisation," *Ann. Oper. Res.* **127**, No. 1/4, 359–372 (2004).
5. R. Dechter, *Constraint Processing*, Morgan Kaufmann Publishers, San Francisco, CA, 2004.
6. R. Yang, "Solving a Workforce Management Problem with Constraint Programming," *Proceedings of the 2nd International Conference on the Practical Application of Constraint Technology*, London, U.K., 1996, pp. 373–387.
7. D. L. Gresh, D. P. Connors, J. P. Fasano, and R. J. Wittrock, "Applying Supply Chain Optimization Techniques to Workforce Planning Problems," *IBM J. Res. & Dev.* **51**, No. 3/4, 251–261 (2007, this issue).
8. D. Gilat, A. Landau, A. Ribak, Y. Shiloach, and S. Wasserkrug, "SWOPS (Shift Work Optimized Planning and Scheduling)," *Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling*, Brno, Czech Republic, 2006, pp. 518–523.
9. R. Nanda and J. Browne, *Introduction to Employee Scheduling*, Van Nostrand Reinhold, New York, 1992.
10. J. P. van den Berg and D. M. Panton, "Personnel Shift Assignment: Existence Conditions and Network Models," *Networks* **24**, No. 7, 385–394 (1994).
11. A. S. Appelblad and S. Lönn, "A Study of Workforce Arrangement," Master's Thesis, Department of Informatics, Göteborg University, Göteborg, Sweden, 2004.
12. A. T. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier, "Staff Scheduling and Rostering: A Review of Applications, Methods, and Models," *Euro. J. Oper. Res.* **153**, No. 1, 3–27 (2004).
13. E. K. Burke and E. Soubeiga, "A Real-World Workforce Scheduling Problem in the Hospitality Industry: Theoretical Models and Algorithmic Methods" see *http://webhost.ua.ac.be/eume/workshops/reallife/burke.pdf*.
14. T. H. Hultberg and D. M. Cardoso, "The Teacher Assignment Problem: A Special Case of the Fixed Charge Transportation Problem," *Euro. J. Oper. Res.* **101**, No. 3, 463–473 (1997).
15. S. E. Bechtold, M. J. Brusco, and M. Showalter, "A Comparative Evaluation of Labor Tour Scheduling Methods," *Decision Sci.* **22**, No. 4, 683–699 (1991).
16. H. K. Alfares, "Optimum Workforce Scheduling Under the (14, 21) Days-Off Timetable," *J. Appl. Math. & Decision Sci.* **6**, No. 3, 191–199 (2002).
17. A. Billionnet, "Integer Programming to Schedule a Hierarchical Workforce with Variable Demands," *Euro. J. Oper. Res.* **114**, No. 1, 105–114 (1999).
18. J. C. Beck, P. Prosser, and E. Selensky, "Vehicle Routing and Job Shop Scheduling: What's the Difference?," *Proceedings of the 13th International Conference on Automated Planning and Scheduling*, Trenton, Italy, 2003; see *http://tidel.mie.utoronto.ca/pubs/icaps03.pdf*.
19. B. Cao and G. Uebe, "Solving Transportation Problems with Nonlinear Side Constraints with Tabu Search," *Computers & Oper. Res.* **22**, No. 6, 593–603 (1995).
20. M. Sun, J. E. Aronson, P. G. McKeown, and D. Drinka, "A Tabu Search Heuristic Procedure for the Fixed Charge Transportation Problem," *Euro. J. Oper. Res.* **106**, No. 2, 441–456 (1998).
21. U. Aickelin and K. A. Dowsland, "Exploiting Problem Structure in a Genetic Algorithm Approach to a Nurse Rostering Problem," *J. Scheduling* **3**, No. 3, 139–153 (2000).
22. F. F. Easton and N. Mansour, "A Distributed Genetic Algorithm for Employee Staffing and Scheduling Problems," *Proceedings of the 5th International Conference on Genetic Algorithms*, Urbana-Champaign, IL, 1993, pp. 360–367.
23. A. Wren and D. O. Wren, "A Genetic Algorithm for Public Transport Driver Scheduling," *Computers & Oper. Res.* **22**, No. 1, 101–110 (1995).
24. D. K. W. Chiu, S. C. Cheung, and H.-F. Leung, "A Multi-Agent Infrastructure for Mobile Workforce Management in a Service Oriented Enterprise," *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, Big Island, HI, 2005, p. 85.3.
25. P. Cowling, G. Kendall, and E. Soubeiga, "A Parameter-Free Hyperheuristic for Scheduling a Sales Summit," *Proceedings of the 4th Metaheuristics International Conference*, Porto, Portugal, 2001, pp. 127–131.
26. D. Munaf and B. Tester, "And/Or Parallel Programming in Practice," *Technical Report WP12:1203*, British Telecom Research Laboratory, Project 1251, London, U.K., 1993.
27. E. Tsang and C. Voudouris, "Fast Local Search and Guided Local Search and Their Application to British Telecom's Workforce Scheduling Problem," *Oper. Res. Lett.* **20**, No. 3, 119–127 (1997).
28. F. Kokkoras and S. Gregory, "D-WMS: Distributed Workforce Management Using CLP," *Proceedings of the 4th International Conference on the Practical Application of Constraint Technology*, London, U.K., 1998, pp. 129–146.
29. A. Meisels and N. Lusternik, "Experiments on Networks of Employee Timetabling Problems," *Proceedings of the 2nd*

International Conference on the Practice and Theory of Automated Timetabling, selected papers, Toronto, Canada, 1997, pp. 130–141.
30. P. Van Hentenryck, L. Michel, and Y. Deville, *Numerica: A Modeling Language for Global Optimization*, MIT Press, Cambridge, MA, 1997.
31. P. Van Hentenryck, *The OPL Optimization Programming Language*, MIT Press, Cambridge, MA, 1999.
32. A. Mackworth, "Consistency in Networks of Relations," *Artif. Intell.* **8**, No. 1, 99–118 (1977).
33. H. H. Hoos and T. Stützle, *Stochastic Local Search: Foundations and Applications*, Morgan Kaufmann Publishing, San Francisco, CA, 2004.
34. G. Verfaillie and N. Jussien, "Constraint Solving in Uncertain and Dynamic Environments: A Survey," *Constraints* **10**, No. 3, 253–281 (2005).
35. P. Meseguer, N. Bouhmala, T. Bouzoubaa, M. Irgens, and M. Sanchez, "Current Approaches for Solving Over-Constrained Problems," *Constraints* **8**, No. 1, 9–39 (2003).
36. A. Borning, B. Freeman-Benson, and M. Wilson, "Constraint Hierarchies," *Lisp Symbol. Computation* **5**, No. 1, 223–270 (1992).
37. R. Barták, "Constraint Programming: In Pursuit of the Holy Grail," *Proceedings of the Week of Doctoral Students*, Prague, Czech Republic, June 1999, pp. 555–564.
38. M. Wallace, "Practical Applications of Constraint Programming," *Constraints* **1**, No. 1/2, 139–168 (1996).
39. I. E. Sutherland, "Sketchpad: A Man–Machine Graphical Communication System," *Proceedings of the SHARE Design Automation Workshop, Annual ACM–IEEE Design Automation Conference*, 1964, pp. 6.329–6.346.
40. A. Borning, "The Programming Language Aspects of ThingLab, a Constraint-Oriented Simulation Laboratory," *ACM Trans. Programming Lang. & Syst.* **3**, No. 4, 252–387 (1981).
41. D. L. Waltz, "Understanding Line Drawings of Scenes with Shadows," *Psychol. Computer Vision*, P. H. Winston, Editor, McGraw-Hill, New York, 1975.
42. K. P. Chow and M. Perrett, "Airport Counter Allocation Using Constraint Logic Programming," *Proceedings of the 3rd International Conference on Practical Application of Constraint Technology*, London, U.K., 1997.
43. M. Perett, "Using Constraint Logic Programming Techniques in Container Port Planning," *ICL Tech. J.* **7**, No. 33, 537–545 (1991).
44. G. Hasle, R. C. Haut, B. S. Johansen, and T. S. Ølberg, "Well Activity Scheduling—An Application of Constraint Reasoning"; see *http://www.ilog.com/products/optimization/tech/custpapers/sintef.pdf*.
45. J. Adhikary, G. Hasle, and G. Misund, "Constraint Technology Applied to Forest Treatment Scheduling," *Proceedings of the 3rd International Conference on the Practical Application of Constraint Technology*, London, U.K., 1997; see *http://www.cs.sfu.ca/research/groups/ISL/papers/adhikary-etal-PACT.pdf*.
46. J. Bellone, A. Chamard, and C. Pradelles, "PLANE: An Evolutive Planning System for Aircraft Production," *Proceedings of the 1st International Conference on Practical Application of Prolog*, London, U.K., 1992.
47. T. Creemers, L. R. Giralt, J. Riera, C. Ferrarons, J. Rocca, and X. Corbella, "Constraint-Based Maintenance Scheduling on an Electric Power-Distribution Network," *Proceedings of the 3rd International Conference on Practical Applications of Prolog*, Paris, France, 1995, pp. 135–144.
48. T. Frühwirth and P. Brisset, "Optimal Placement of Base Stations in Wireless Indoor Telecommunication," *Proceedings of Principles and Practices of Constraint Programming*, Pisa, Italy, 1998, pp. 476–480.
49. A. Brodsky, J. Jaffar, and M. J. Maher, "Toward Practical Query Evaluation for Constraint Databases," *Constraints* **2**, No. 3/4, 279–304 (1997).
50. P. C. Kanellakis and D. Q. Goldin, "Constraint Programming and Database Query Languages," *Theoretical Aspects of Computer Software*, J. C. Mitchell and M. Hagiya, Editors, Springer-Verlag, Berlin, Germany, 1994, pp. 96–120.
51. B. Beizer, *Software Testing Techniques*, Van Nostrand Reinhold Co., New York, 1990.
52. C. Brzoska, "Temporal Logic Programming and Its Relation to Constraint Logic Programming," *Proceedings of the International Symposium on Logic Programming*, San Diego, CA, 1991, pp. 661–677.
53. A. P. Sistla, M. Y. Vardi, and P. Wolper, "The Complementation Problem for Buchi Automata with Applications to Temporal Logic," *Theoret. Computer Sci.* **49**, 217–237 (1987).
54. Y. Naveh, M. Rimon, I. Jaeger, Y. Katz, M. Vinov, E. Marcus, and G. Shurek, "Constraint-Based Random Stimuli Generation for Hardware Verification," *AI Magazine*, in press.
55. E. Tsang and C. Voudouris, "Constraint Satisfaction in Discrete Optimisation," presented at theUNICOM Seminar, 1998; see *http://www.cs.essex.ac.uk/CSP/papers/TsaVou-GLSOpt-Unicom98.pdf*.
56. T. Walsh, "Reformulating Propositional Satisfiability as Constraint Satisfaction," *Proceedings of the 4th International Symposium on Abstraction, Reformulation, and Approximation*, Horseshoe Bay, TX, 2000, pp. 233–246.
57. W.-J. van Hoeve, "Operations Research Techniques in Constraint Programming," Ph.D. dissertation, University of Amsterdam, Institute for Logic, Language, and Computation, Amsterdam, The Netherlands, 2005.
58. N. Barnier and P. Brisset, "Graph Coloring for Air Traffic Flow Management," *Proceedings of the International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimisation Problems*," LeCroisic, France, 2002, pp. 133–147.
59. M. Gronkvist, "A Constraint Programming Model for Tail Assignment," *Proceedings of the International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimisation Problems*," Nice, France, 2004, pp. 142–156.
60. E. Tsang, J. Ford, P. Mills, R. Williams, and P. Scott, "ZDC-Rostering: A Personnel Scheduling System Based on Constraint Programming," *Technical Report CSM-406*, University of Essex, Department of Computer Science, Essex, U.K., 2004.
61. Y. Richter, A. Freund, and Y. Naveh, "Generalizing AllDifferent: The SomeDifferent Constraint," *Proceedings of the 12th International Conference on Principles and Practice of Constraint Programming, Lecture Notes in Computer Science* **4204**, 468–483 (2006).
62. J.-C. Régin, "A Filtering Algorithm for Constraints of Difference in CSPs," *Proceedings of the 12th National Conference on Artificial Intelligence*, Seattle, WA, 1994, pp. 362–367.

Received September 21, 2006; accepted for publication December 15, 2006; Internet publication May 11, 2007

Yehuda Naveh *IBM Haifa Research Laboratory, Haifa University Campus, Haifa 31905, Israel (naveh@il.ibm.com)*. Dr. Naveh received a B.S. degree in physics and mathematics, an M.S. degree in experimental physics, and a Ph.D. degree in theoretical physics, all from the Hebrew University of Jerusalem, Israel. He joined IBM Research in 2000 after working for four years as a research associate at Stony Brook University in New York. His current research interests include the theory and practice of constraint programming and the theory and practice of workforce optimization.

Yossi Richter *IBM Haifa Research Laboratory, Haifa University Campus, Haifa 31905, Israel (richter@il.ibm.com)*. Dr. Richter received a B.A. degree in computer science and economics, and M.S. and Ph.D. degrees in computer science specializing in algorithms, all from Tel Aviv University, Israel. Since 2005, he has been a Research Staff Member at the IBM Haifa Research Laboratory, working on the theory and practice of constraint programming.

Yaniv Altshuler *IBM Haifa Research Laboratory, Haifa University Campus, Haifa 31905, Israel (yanival@il.ibm.com)*. Mr. Altshuler received a B.A. degree in computer science from the Israeli Institute of Technology (IIT), the Technion, under the framework of the Chais Family Foundation Technion Excellence Program. He is currently a Ph.D. candidate in the computer science department of IIT, where he specializes in multiagent systems in dynamic environments and swarm intelligence. In 2004 Mr. Altshuler joined the IBM Research Division, where he works on constraint satisfaction and optimization problems.

Donna L. Gresh *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (gresh@us.ibm.com)*. Dr. Gresh received her B.S. degree in engineering in 1983 from Swarthmore College and her M.S. and Ph.D. degrees in electrical engineering in 1985 and 1990 from Stanford University, where she studied the rings of Uranus using data from the spacecraft Voyager. She joined the IBM Thomas J. Watson Research Laboratory as a Research Staff Member in 1990 and spent twelve years conducting research in scientific and information visualization. Since 1992, Dr. Gresh has been a member of the Mathematical Sciences Department, with research interests in the area of workforce optimization.

Daniel P. Connors *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (dconnors@us.ibm.com)*. Dr. Connors received his B.S.E. degree in electrical engineering from the University of Michigan in 1982, and his M.S. and Ph.D. degrees in electrical engineering from the University of Illinois in 1984 and 1988, respectively. Since 1988, he has been a Research Staff Member at the IBM Thomas J. Watson Research Center. Dr. Connors has worked on modeling, simulating, and designing business processes and developing decision support tools for manufacturing and supply chain logistics. He is a member of the Mathematical Sciences Department at the Research Center, where he is currently working on developing business processes and workforce management optimization tools.