

Staff Scheduling for Inbound Call Centers and Customer Contact Centers

Alex Fukunaga, Ed Hamilton, Jason Fama, David Andre, Ofer Matan, and Illah Nourbakhsh

- The staff scheduling problem is a critical problem in the call center (or, more generally, customer contact center) industry. This article describes DIRECTOR, a staff scheduling system for contact centers. DIRECTOR is a constraint-based system that uses AI search techniques to generate schedules that satisfy and optimize a wide range of constraints and service-quality metrics. DIRECTOR has successfully been deployed at more than 800 contact centers, with significant measurable benefits, some of which are documented in case studies included in this article.

Staff scheduling is the following classic operations research problem: Given a set of employees, assign them to a schedule such that they are working when they are most needed but ensuring that certain constraints are maintained (for example, employees must work no more than 40 hours a week and must have at least 12 hours between work shifts). Even the simplest variations of this problem are known to be NP-complete (Garey and Johnson 1978).

Although staff scheduling has long been an important operations research problem, scheduling has recently become an important component of an emerging class of business software applications known as *work-force management software*. The need for effective work-force management systems has been driven primarily by the recent, rapid growth of the call center—customer contact center industry, in which efficient deployment of human resources is of crucial, strategic importance. Traditionally, in this industry, staff scheduling has been performed using ad hoc methods and operations research techniques (Cleveland and

Mayben 1997).¹ However, we found that this domain is particularly amenable to the application of constraint-based and heuristic scheduling techniques from AI.

This article describes Blue Pumpkin DIRECTOR, a recently developed staff scheduling system, which is currently being used by hundreds of contact centers. First, we describe the staff scheduling problem for call centers and contact centers. Then, we describe the design and implementation of DIRECTOR. Finally, examples of successful deployments of the application are given.

Staff Scheduling in Contact Centers

When a consumer calls a software vendor to ask for technical support or if he/she calls a credit card company with a billing inquiry, the call is often routed to an inbound *call center* (or, more generally, contact center), a large, centralized pool of trained agents (contact center employees) who are qualified to address the customer's inquiry.²

If all agents who can handle the call are busy, then the customer's call waits in a queue until an agent becomes available. Naturally, long wait times result in frustrated, dissatisfied customers, and it is therefore important for call centers to be staffed so that the wait times experienced by customers are acceptable. At the same time, businesses want to avoid overstaffing (having idle agents when few customer calls arrive) to minimize the cost of operating the call center and maximize overall business profitability.

It is well known that acquiring a new customer is several times more expensive (in terms of marketing and sales expenses) than deriving revenues from an existing customer. Therefore, maintaining customer satisfaction by achieving good service levels has a significant impact on corporate revenues. In addition, personnel costs account for 60 to 70 percent of the operational cost of a contact center. Efficient contact center staff scheduling is therefore important to a business both from the perspective of revenue ("the top line") and from operating margins and profitability ("the bottom line").

Internal corporate call centers are the centralized customer-service organizations that serve as the foci of customer contact for businesses. There is also a large industry of outsourced call centers. Businesses regularly outsource some of their customer-service functions to outsourcers, who are committed by the terms of a service-level agreement in the contract to achieve specified service goals (for example, outsourcer *X* agrees to handle manufacturer *Y*'s sales inquiries and promises that 80 percent of the calls will be answered within 20 seconds). Therefore, efficient staff scheduling is particularly critical for these outsourcers, so that they can deliver the contractually agreed-on service levels while they operate profitably.

Although most interactive contact between customers and businesses still takes place through the telephone, customer contact through other media such as e-mail, online chat, and instant messaging is rapidly increasing. A *contact center* is a generalization of a call center, where agents handle these other media, in addition to traditional media such as phone calls and faxes. Contact centers offer some new challenges for staff scheduling systems, as described later.

Because the call center industry is not well known in the AI and computer science communities, it is worth noting some relevant market statistics. In the beginning of 2001, there were over 82,000 contact centers (employing over 1.5 million agents) in the United States alone, expected to almost double by 2004.^{3,4,5,6} Approximately seven percent of U.S. call centers were using a work-force management system. Note that the market penetration of work-force management software is still very low, in part because modern work-force management systems with the full capabilities and ease of use required by the call center market are relatively new. However, because of the clear economic benefits, the market for work-force management software is growing rapidly (the annual revenues for the call center work-force management software market were \$175 mil-

lion in 2001, expected to grow to more than \$500 million by 2006).

The contact center scheduling problem poses a challenging problem. Meeting the demand profile implied by the forecasts of incoming calls and contacts is by itself a difficult combinatorial optimization problem, especially considering that the forecasts are probabilistic. At a minimum, a 1-week schedule with a 15-minute granularity must be generated. Typically, contact centers have hundreds of agents that need to be scheduled; some have thousands of agents. In addition to service goals, numerous hard and soft constraints reflecting the contact center's operational constraints, local labor rules, and employee preferences must be satisfied. The agents' schedules must be specified at a minimum of 15-minute granularity; in addition to specifying the start time and duration of a work shift, all the "off-telephone" activities such as breaks also need to be scheduled. Furthermore, the recent advent of multi-skilled scheduling and multicontact scheduling (see later) has significantly complicated the problem of optimizing service goals. Traditional methods (manual scheduling and mathematical programming approaches) have been unable to keep up with the rapidly evolving, increasingly difficult scheduling requirements of the modern contact center.

The typical contact center scheduling process can be described as follows:

Schedules are usually generated on a weekly basis, with a granularity of 15 minutes. A forecast of incoming contact volume (number of calls in a 15-minute period) and expected handling time of the contacts are used to generate a demand profile.

A standard goal for call center operations is to achieve a certain service level; that is, answer *X* percent of calls within *Y* seconds and minimize overstaffing.

Guided by the forecast and service goal, the scheduler (traditionally, a human contact center manager) generates a schedule that satisfies various hard constraints (for example, labor laws, company policies) and optimizes service goals and satisfies as soft constraints as much as possible.

The DIRECTOR System

We now describe the DIRECTOR application. After a brief discussion of the overall system architecture, we describe the major components most relevant to the algorithmic and AI aspects of the system.

...
acquiring a new customer is several times more expensive ... than deriving revenues from an existing customer. Therefore, maintaining customer satisfaction by achieving good service levels has a significant impact on corporate revenues.

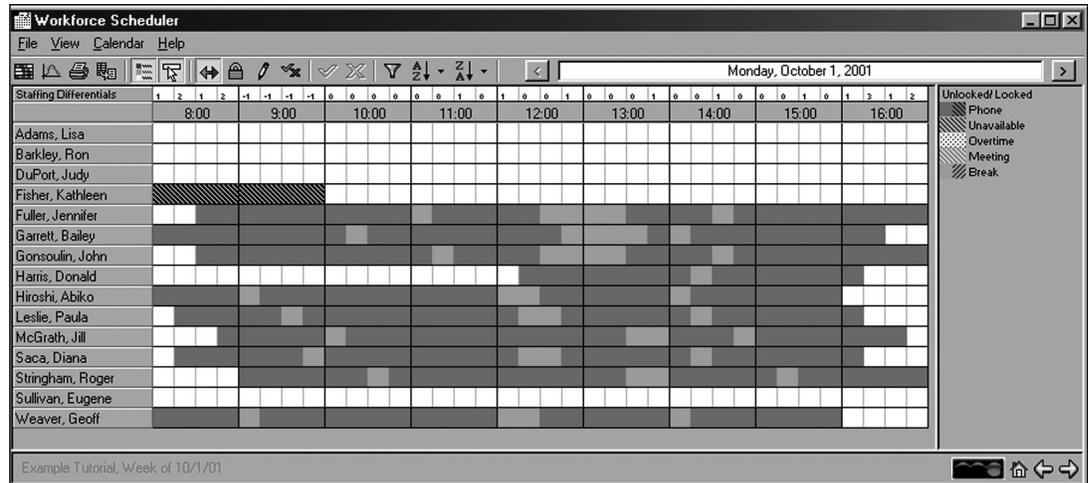


Figure 1. The DIRECTOR Graphic User Interface for Displaying and Manipulating a Schedule.

It currently shows a sample schedule generated by DIRECTOR for a single day for a call center open from 8:00 to 16:00. Each row in the grid represents an agent's schedule for the day. The lightly shaded regions indicate 15-minute intervals when the agent is scheduled to be answering telephone calls. The darkly shaded regions indicate scheduled breaks.

System Architecture

From a scheduling-centric point of view, DIRECTOR consists of the scheduling engine, which loads an input scenario and generates a schedule that satisfies hard constraints and optimizes schedule quality metrics; an infrastructure for persisting scheduling scenario input and output in a relational database; and a graphic user interface (GUI) (figures 1 and 2).

In addition, there is a major software component required for integration with *automatic call distributors* (ACDs), which are the hardware and software routers that route incoming calls and contacts to the appropriate agent in the contact center.

Work-force management software systems for contact centers include many more functions, such as the real-time monitoring of agent adherence to the published schedule and an extensive reporting facility; however, these other features in DIRECTOR are beyond the scope of this article, which focuses on the scheduling functions.

The current version of DIRECTOR (3.1) is implemented as a set of Microsoft COM components, mostly implemented in C++. Figure 3 shows the DIRECTOR ENTERPRISE architecture. It is a traditional client-server system, which consists of a back-end database (Microsoft SQL server or ORACLE relational database) running on a server, and a client, which consists of business logic components (including the scheduling engine) and GUI components. The next version of DIRECTOR ENTERPRISE (to be released in 2002) is based on a more modern multitiered web-oriented architecture (a relational data-

base, a J2EE application server running business logic and other middle-tier services, and a "thin" web-based GUI client).

In addition, there is another version of Blue Pumpkin DIRECTOR, called DIRECTOR ESSENTIAL, which is designed for use by small-and medium-sized contact centers (typically with fewer than 100 agents). Its scheduling engine is implemented in C++, the scheduling scenarios are stored in a Microsoft ACCESS relational database, and the GUI is implemented in VISUAL BASIC. The emphasis of ESSENTIAL is on ease of use and installation. DIRECTOR ESSENTIAL is actually the predecessor of DIRECTOR ENTERPRISE, and development on ESSENTIAL has continued, focusing on its target user base of smaller contact centers. Many algorithmic ideas used in ENTERPRISE originated in ESSENTIAL. In the rest of this article, we focus on the ENTERPRISE version because it provides a superset of the features of ESSENTIAL.

Using DIRECTOR to schedule contact center agents generally involves the following work flow: First, a model of the contact center is built in the client and is stored in the relational database. The main model elements are the characteristics of the contact center, the agents (resources), and the operational constraints. Then, rules and constraints that apply to the agents (for example, how many hours a week he/she can work, which days he/she is available, what times he/she prefers to work) are entered and linked. Typically, this part of the scheduling scenario is relatively static from week to week.⁷ For each week, the user (contact center manager) generates a forecast of the incoming calls and contacts (the demand pro-

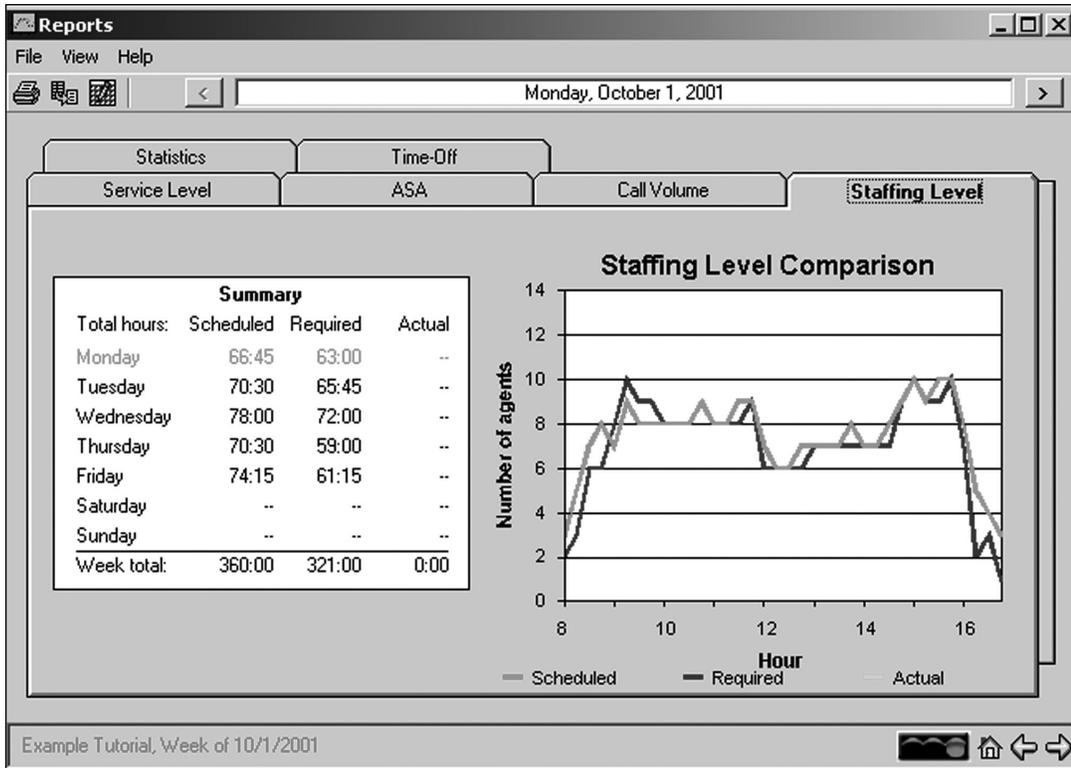


Figure 2. Required Staffing (according to Erlang-C) versus Scheduled Staffing Levels for the Schedule in Figure 1.

Note that the scheduled staffing closely matches the required staffing.

file). Then, he/she specifies a target service goal that the schedule should satisfy and runs the scheduling algorithm to generate the schedule. The schedule is posted and distributed to the contact center agents. Each of the major steps and components is described here.

Forecasting

In the forecasting step, users create a prediction of the series of contacts that will arrive in the contact center during the time period to be scheduled. A basic forecast can be specified as a sequence of tuples $(t, numContacts_t, AHT_t)$, where $numContacts$ is the number of contacts that arrive during the time period t , and AHT is the average handling time for the contacts (for example, the amount of time a contact center agent will spend talking on the telephone or the time it takes to write a reply to an e-mail inquiry). In DIRECTOR, forecasting is done with a 15-minute granularity. For example, the user might enter a forecast that specifies that from 8:00 AM to 8:15 AM, 10 calls arrive, with an AHT of 200 seconds, then from 8:15 to 8:30, 15 calls arrive with an AHT of 205 seconds, and so on.

Currently, DIRECTOR uses a simple forecasting model, where the user can either manually enter a forecast or create a forecast by combining (using weighted averaging) forecasts from pre-

vious scheduling periods. Although it might be possible to improve the accuracy of the forecasts by applying more sophisticated learning techniques, users report satisfaction with the current approach.

Service Goals, Computation of Agent Requirements, and Modeling Overstaffing and Understaffing

Given the forecast for a contact queue, the next step in scheduling is to specify a service goal for the queue. The following are some service goals: answer 90 percent of the incoming calls within 20 seconds, send a reply to 99 percent of the e-mail inquiries within 24 hours, answer calls within 30 seconds on average; limit abandoned calls to 5 percent of the incoming calls (calls are *abandoned* when a customer hangs up the phone before an agent becomes available to talk to the customer), and no agent should be idle more than 25 percent of the time.

Combinations of these goals are possible; for example, answer 80 percent of all incoming calls within 30 seconds, no more than 5 percent of the calls can be abandoned, and no agent should be idle more than 20 percent of the time.

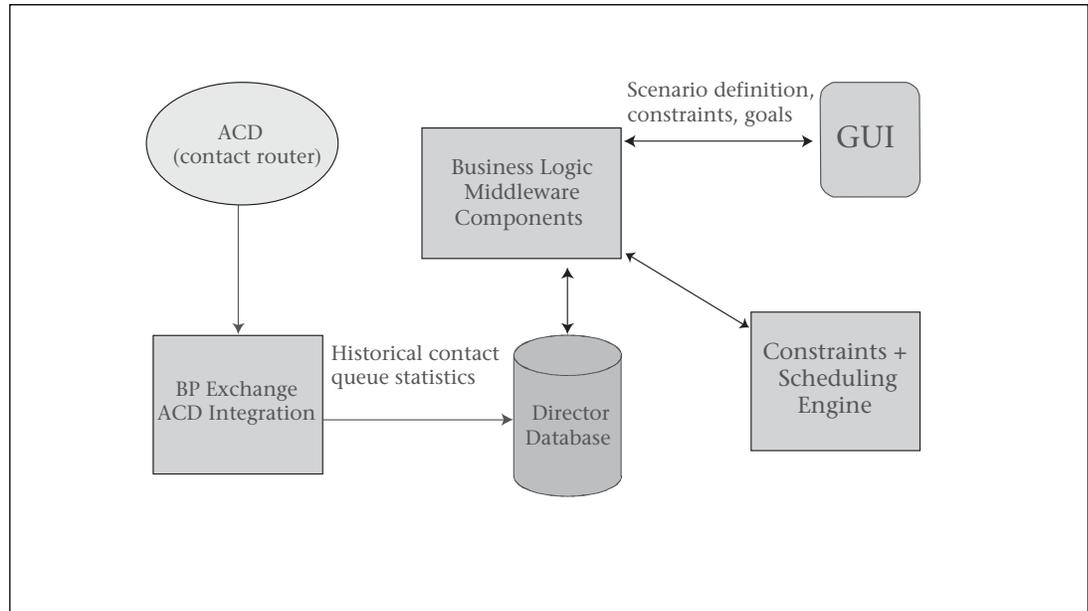


Figure 3. DIRECTOR ENTERPRISE Software Architecture.

In a scenario where there is a single queue of calls, and any agent in the contact center can answer the call, it is possible to compute an agent requirement for a time period, that is, the number of agents who must be working during the time period to satisfy the service goal, given the forecast. This is a classical $M/M/s$ queuing model, and agent requirements are computed by applying the well-known Erlang-C formula from operations research and queueing theory (c.f. Kleinrock [1976]) and some straightforward extensions. Given a candidate schedule, we say a time interval is *understaffed* if the number of agents scheduled to be working during the interval is less than the agent requirements and *overstaffed* if there are fewer agents scheduled than required. By computing the overstaffing and understaffing for each time interval in the scheduling period, we have the basis for an objective function for evaluating a candidate schedule with respect to service goals. Figure 2 shows DIRECTOR screenshots showing the schedule (and service goals and results) for a very small, example scenario.

Now, consider the following case: There are two queues: (1) the widget sales inquiry queue and (2) the widget tech support queue. There are three agents, Bob (who is qualified to answer sales inquiries), John (qualified to answer technical support inquiries), and Mary (qualified to answer either sales or support inquiries). This multiskilled scenario differs from the previously described single-queue case because it is no longer possible to straightforwardly compute how overstaffed or understaffed the

schedule is for a particular time interval because of the interaction between the queues. For example, suppose all agents are initially available, and three calls arrive in rapid succession. The first call arrives on the sales queue and is answered by Bob. The second call arrives on the tech support queue and is immediately followed by a third call, which is a sales call. If John answers the call, the third call will be answered by Mary. However, suppose that Mary answers the second call. Then, the third call will be put on hold (even though John is available, he is not able to respond to sales calls).

These interactions between the agents, their skills, the order of calls arriving on the queues, and the way in which the calls are routed make it very difficult to answer whether the schedule is understaffed or overstaffed. In fact, there is currently no known, closed-form formula (such as the Erlang-C formula) for computing the service level for the multiskilled scheduling problem (Koole and Mandelbaum 2001). It is possible to compute the service level by simulating the schedule and the call-routing algorithm. However, simulations are expensive (in the context of generating and optimizing a schedule by a generate-and-test framework such as iterative repair).

Another important case where the traditional operations research approaches do not apply is when modeling queues are significantly different from telephone queues, such as e-mail contact queues (and similar types of media such as faxes). E-mail contacts differ from telephone calls in several important ways. First,

the service goal usually involves much longer time periods than telephone calls (an e-mail reply is usually expected within a day or so, but people expect telephone calls to be answered within seconds or minutes). Second, e-mail inquiries are usually partitioned into many, sparse, virtual queues. Third, although a telephone call is abandoned and leaves a queue when the customer becomes frustrated after waiting too long on hold, e-mail contacts are never abandoned. Because of these factors, the standard Erlang formulas are not applicable when modeling scheduling agents to staff e-mail queues.

An increasing number of contact centers now handle a mixture of telephone and e-mail contacts simultaneously. For example, a contact center agent might typically answer telephone calls from the set of queues for which he/she is skilled, and when no calls are pending, he/she would reply to e-mail inquiries. Therefore, a modern contact center agent can no longer be modeled as a generic staffing unit that can simply be aggregated into the input of an Erlang-C formula.

A scheduling system for the modern contact center must simultaneously solve both the multiskilled scheduling and the nontelephone-media scheduling problems described earlier, in addition to the traditional single-telephone queue scheduling problem. This complexity makes it difficult to apply traditional operations research approaches (mathematical programming) because all known existing solutions (proprietary algorithms in commercial systems, including DIRECTOR) rely on some form of simulation model. Therefore, constraint-based and iterative scheduling approaches from AI are appealing techniques for the contact center scheduling problem.

Constraints

Employees have various constraints that determine how and when they can be scheduled. Some constraints are a result of the policies of the contact center. Some constraints are mandated either by law or by labor union agreements. Other constraints reflect the personal preferences of the staff.

The primitive building block of a schedule is a *shift*, which represents a class of object representing a contiguous span of time for which an agent is scheduled to answer telephone calls.⁸ A shift can contain a number of off-telephone activities during which he/she is not available to pick up calls (for example, 1-hour meal breaks, 15-minute breaks).

The basic constraints in DIRECTOR specify parameters such as the duration and possible

start times of shifts and the duration and possible start times of off-telephone activities. For example, we can specify that an “8-hour standard shift” is 8 hours long, starts between 9 AM and 1 PM. Furthermore, we can specify that this class of shift contains a *lunch break*, 1-hour off-telephone activity, which begins between 3 and 4 hours after the start of the shift as well as a 15-minute “break” that can be scheduled at any time during the shift. DIRECTOR builds on these building blocks with shift pattern constraints that constrain which shifts can be worked on which day. For example, I can say that Joe can either work an 8-hour standard shift or a 4-hour special shift on Monday, must work a 4-hour special shift on Tuesday, and must not work any shifts on Sundays.

The user can also specify constraints on the number of occurrences of various objects, for example, Bob must work between 3 and 4 weekend shifts a month, Alice must work no more than 80 hours every 2 weeks, and John cannot work more than 5 consecutive days in a row.

Most constraints involve only a single agent. However, there are constraints that can involve more than one employee. For example, we can specify that John, Mary, and Robert must all have the same number of weekend shifts between 1/1/02 and 6/1/02.

Agents can express their preferences about their own schedules, and these preferences are treated as soft constraints by DIRECTOR. One type of preference is a rank ordering on the start times of the shifts, for example, John prefers to start between 8 and 9 AM on Mondays, but if that’s not possible, he prefers to start between 9 and 10 AM and would really prefer not to start shifts in the evenings. Agents can also express preferences about the set of shifts they work; for example, I would much rather work on the day shifts Monday through Friday than on the night shifts.⁹

Although most planning and scheduling systems with a highly expressive constraint system use a programming language-like textual modeling language to specify constraints, such a modeling language would make the system excessively complex for the intended users of our system who are not engineers. The most commonly used rules are specified using various GUI elements, and the less frequently used constraints are entered using a pseudo-natural language “sentence builder” interface, similar to those used by some commercial rule-based systems such as the Versata LOGIC SUITE and ILOG RULES. This interface enables most of the end users of DIRECTOR to specify complete scheduling scenarios with little, if any, assis-

... constraint-based and iterative scheduling approaches from AI are appealing techniques for the contact center scheduling problem.

... brute-force search is insufficient to solve large problems with difficult constraints. Therefore, the DIRECTOR algorithm is an iterative procedure, which repeatedly selects some set of variables and optimizes the value bindings by applying some search algorithm to the limited search space.

tance from Blue Pumpkin consultants or technical support staff.¹⁰

The constraint system in DIRECTOR is very expressive and can express almost all constraints currently required by the contact center market (because of the lack of space, we limited this discussion to a basic subset that illustrates the capabilities of the system).

The Scheduling Algorithm

Once the scenario is defined, the process of schedule generation and optimization can begin.

The major design goal of the DIRECTOR scheduling algorithm is to allow users to quickly generate satisfactory schedules with the absolute minimum amount of hassle. Therefore, the scheduling algorithm needs to be an extremely robust “black box” with acceptable performance.

The only user-adjustable parameter that influences the scheduling algorithm’s behavior is a switch that determines whether the algorithm terminates after satisfying an internal termination criterion or continues to search for better solutions until explicitly interrupted by the user (normal scheduling mode versus schedule until interrupted mode).

Internally, the scheduling problem is formulated as a hybrid constraint-satisfaction–global optimization problem. There is a global objective function, which is a prioritized vector of scoring terms. For each class of constraint, there is a corresponding score term that represents the degree to which this class of constraint is being violated. The score terms corresponding to hard constraints have higher priority than soft constraints and terms corresponding to service goals.

For each agent, there is a slot variable, which represents the shift (if any) that the agent is scheduled to work on this day. Instantiating a shift in a slot results in the instantiation of variables representing off-telephone activities (thus, there is a one-level abstraction hierarchy consisting of slot and off-telephone activity variables). A *schedule* is therefore a complete assignment of variables to values. The scheduling algorithm tries to generate a schedule with a maximal score.

The DIRECTOR scheduling algorithm is a hybrid algorithm, combining elements from standard iterative repair and heuristic global optimization algorithms.

The foundation of the DIRECTOR scheduler is a library of search algorithms, including depth-first backtracking, beam search, and iterative sampling. A search algorithm takes a set of variables and returns a new set of value bind-

ings for those variables that maximize the value of the global objective function. The objective function is incrementally updated after each variable binding, which enables a flexible framework where arbitrary search pruning and backtracking control policies can be implemented in the search algorithms. We currently make heavy use of a heuristic algorithm inspired by simulated annealing.

In this framework, the simplest scheduling algorithm would be

Instantiate a search algorithm that takes as input all the slots for all the agents, then run the search algorithm until some termination criterion is met.

Although this strategy (using the annealing algorithm as the search algorithm) actually works for small, relatively unconstrained scenarios, brute-force search is insufficient to solve large problems with difficult constraints. Therefore, the DIRECTOR algorithm is an iterative procedure, which repeatedly selects some set of variables and optimizes the value bindings by applying some search algorithm to the limited search space. In classical iterative repair (Minton et al 1992), the goal of each “repair” is to resolve a constraint violation, but the DIRECTOR algorithm is similar in spirit to recent repair-based optimization scheduling systems such as OPIS (Smith 1994), DCAPS (Chien et al. 1999), and ASPEN (Rabideau et al. 1999). Rather than only repair constraint violations, a search algorithm could be run on a set of variables either for optimization (for example, one by one, unbind each slot variable, and try to locally slide the start time of the shift to improve the service goal score) or because it is good policy to run some heuristic periodically (for example, once in a while, unbind all slots for an agent and reschedule him/her).

The time required for the scheduling algorithm to generate a satisfactory schedule depends largely on the size of the contact center (number of agents), the number and types of queues, and the complexity of the constraints. A one-week schedule for a typical 150-agent scenario (at a 15-minute granularity) can be scheduled in under 5 minutes on a 500-megahertz PENTIUM III desktop machine; a 1000-agent multiskilled scenario takes 30 to 60 minutes. The complexity of the algorithm scales roughly linearly with the number of skills times the number of agents (assuming a fixed set of constraints). Interestingly, if the number of agents increases without a corresponding increase in the number of skills, then the scaling is better than linear because there are few hard constraints that involve more than a single agent; thus, the more agents there are, the

more flexibility the algorithm has with respect to meeting the service goals, which makes the problem easier in some sense.

Besides the scheduling algorithm itself, a great deal of effort has gone into the development of efficient data structures and algorithms that enable the incremental computation of the objective function. The major computational bottleneck in DIRECTOR is incremental, on-demand recalculation of the service goal terms in the objective function. For example, when the start time of a shift is changed from 8 AM to 9 AM, what is the impact on the service goals? For a single telephone queue scenario, this computation is relatively inexpensive (but still the major bottleneck); for multiskilled scenarios with e-mail queues, this evaluation becomes a major bottleneck, which must be alleviated using various lazy evaluation, caching, and approximation algorithms.

As we noted already, almost all hard constraints involve only one agent, meaning that in practice, satisfying hard constraints is relatively easy for the majority of the scenarios encountered by DIRECTOR. Most of the search effort is spent optimizing the soft constraints such as the service goals and agent preferences. Therefore, the current scheduling algorithm does not attempt to perform much constraint propagation, focusing instead on brute-force, rapid generation and evaluation of candidate schedule states. This approach contrasts with constraint-directed refinement search methods (c.f. Jonsson et al. [2000]; Smith et al. [2000]), which make heavy use of constraint propagation.

In addition to the standard scheduling problem described earlier, there are a number of related scheduling problems that are addressed by DIRECTOR. We describe some of these in the following subsections.

Event Scheduling In addition to scheduling agent work schedules, DIRECTOR also schedules various events attended by one or more of the agents. Examples of events are training sessions and group meetings. Traditional, manual meeting scheduling systems such as Microsoft OUTLOOK rely on the user finding a time when all attendees are available. More advanced, agent-based systems (c.f. Maes [1994]) automatically schedule a meeting and notify attendees but only consider the availability and preferences of the attendees. However, in contact centers, it is dangerous to schedule an event based only on availability or individual preferences because it can have a direct, negative impact on the center's service goals.

When scheduling events after the agents' schedules have already been finalized, DIRECTOR takes into consideration the impact on service

goals. In other words, DIRECTOR will schedule an event at a time where all attendees are available and when the contact queues on which the agents are working are least understaffed.

In addition, if the agent schedules are not finalized yet, DIRECTOR goes one step further and simultaneously reschedules the agent schedules and the event schedules to minimize the negative impact on service goals.

Work-Force Planning To date, we have assumed a version of the scheduling problem in which the task is to generate schedules for a group of existing agents.

A related scheduling problem is, Given a forecast of future contacts, a set of employee class profiles that represent typical subclasses of agents (and are linked to various constraints) and some additional constraints (for example, restrictions on the percentage of class profile instances, budget constraints), generate a schedule consisting of *phantom agents* (instances of the employee class profiles) that optimizes the global objective function.

This work-force planning problem is important for users who need to plan future hiring of contact center agents; that is, how many agents need to be hired, and what skills should they have?

In some sense, this optimization problem is more difficult than the standard staff scheduling problem because of the combinatorial explosion. Suppose that there are two employee class profiles. Profile 1 represents an agent who can only answer widget sales calls, costs \$15 an hour, and works 40 hours a week. Profile 2 represents an agent who answers both widget sales and technical support calls, works 20 hours a week, and earns \$25 an hour. There are many combinations of instances of profile 1 and profile 2, and for each combination, there is a different optimal schedule.

DIRECTOR solves this problem with a modified version of its standard scheduling algorithm, but work-force planning is a new application where there is clearly a need for further research.

Multiweek Constraints and Scheduling Currently, DIRECTOR schedules one week at a time because a week is a natural unit, and weekly scheduling is standard contact center industry practice. Most contact centers create and publish schedules on a weekly basis, regardless of whether they use work-force management software.

However, there are various constraints that have a time period other than one week; for example, Joe must work between two to three weekend shifts every four weeks. The DIRECTOR scheduling algorithm handles such multiweek

...
almost all hard constraints involve only one agent, meaning that in practice, satisfying hard constraints is relatively easy for the majority of the scenarios encountered by DIRECTOR.

DIRECTOR enabled Borders Group to increase agent productivity by 53 percent, with a 33-percent reduction in expenses by allocating agent time more effectively over operating hours.

constraints by assuming that the shifts can be distributed evenly among four weeks, but it is clear that such heuristics can fail. It might seem that if we scheduled all four weeks at a time that these multiweek constraints would not be an issue as long as the algorithm scales up. However, aside from any algorithmic problems related to scheduling longer time periods, there is a modeling problem in that the longer the time period being scheduled, the higher the probability that assumptions about the forecast and agent availability (because of unscheduled absences) become invalid (or the data required to make reasonable assumptions might be unavailable). Therefore, scheduling with multiweek constraints is another area where we will focus further research and development efforts in the future.

Application Deployment and Case Studies

Blue Pumpkin *DIRECTOR* (including both the *ENTERPRISE* version and the *ESSENTIAL* version) is currently in use at more than 800 contact centers combined in a wide range of industries; over 110,000 contact center agents are being scheduled by *DIRECTOR*. *DIRECTOR ENTERPRISE* (the version of *DIRECTOR* that is the focus of this article) is in use by approximately 400 customers, including 3M, Apple Computer, Federal Express, GE, AT&T, Kaiser Permanente, Time-Warner Cable, Verizon, and Yahoo!. *DIRECTOR ENTERPRISE* is also widely used by major outsourced contact centers, which handle inbound calls for companies such as AOL and Canon. The typical *DIRECTOR ENTERPRISE* user is a large contact center with 150 to 1000 agents. In addition, *ESSENTIAL* (described in the System Architecture subsection) is also in use by more than 400 customers, including AOL/CompuServe Europe, Peoplesoft, Airborne Express, and EDS. *DIRECTOR ESSENTIAL* users are typically small to mid-sized contact centers with fewer than 200 agents.

Like other enterprise-class business application software, deployment of *DIRECTOR* involves a team of implementation specialists and includes some end user training. It is worth noting that in most cases, the deployment complexity is in integrating the software with the ACD (see System Architecture subsection) and setting up the server. In many cases, the end users create the scheduling scenarios (including all constraints) and run the scheduling algorithm by themselves, using the *DIRECTOR* GUI. In some cases, it only requires several hours of training for a contact center manager to become proficient with *DIRECTOR ESSENTIAL*.

For *DIRECTOR ENTERPRISE*, it is typically several days before the users become proficient in modeling and scheduling. For complex scenarios, Blue Pumpkin consultants assist the users with building the first models, but subsequent models are usually built by the customers themselves. We believe that this relative simplicity represents a significant step forward in the popularization of constraints and AI scheduling technology.

Here, we describe several case studies of customers using *DIRECTOR ENTERPRISE*.

Borders Group

Borders Group is a leading global retailer of books, music, movies, and related items. The seasonal nature of the Borders Group's business, combined with a multiskilled contact center, made optimizing its work force a formidable challenge. Borders Group plans for its staffing needs well in advance of the holiday season where customer expectation is higher than usual. Meeting these expectations is critical because Borders Group transacts a high volume of its business during the holiday season. During this period, there is a surge of more than 35 percent in call volume, making optimizing available resources and staff essential.

After deploying *DIRECTOR*, Borders Group evaluated various staffing scenarios to design a work-force optimization strategy that accurately reflected all Borders' business goals. Based on a selected schedule generated by *DIRECTOR*, Borders Group knew how many seasonal workers to hire, covering which hours and requiring what skills—making the hiring process much easier. In addition, by focusing on the two most needed skills instead of cross-training agents on multiple skills, Borders Group was able to get seasonal staff on the telephones 33 percent faster, allowing them to be productive in 1 week instead of 3.

DIRECTOR enabled Borders Group to increase agent productivity by 53 percent, with a 33-percent reduction in expenses by allocating agent time more effectively over operating hours. Customer-service levels of 88 percent were achieved during the holiday period, with most calls answered in under 10 seconds. Borders Group claims that “[*DIRECTOR*] enabled us to clearly drive down our costs and deliver a high level of customer service not experienced before at Borders Group” (Charlie Moore, director of Customer Service, Borders Group). Borders was also able to reduce turnover of nonseasonal employees from 15 percent to 10 percent. These factors contributed to a 25-percent reduction in overall recruiting and training expenses.

SGI

SGI recently created a virtual contact center by installing a new switch that connected its four facilities located throughout the country. In the past, SGI developed schedules manually, relying on local critical needs assessment to develop a plan. Now they needed a more efficient and accurate method for accommodating the complexities of a work force physically located in four time zones. SGI also decided to bring all customer contact in house, increasing call volumes by 50 percent to 2500 to 3000 calls a week. Budget constraints discouraged increasing the percentage of staff to accommodate the added influx of new calls. Thus, SGI needed to improve service metrics without increasing its budget.

When call volumes doubled from bringing all contacts in house, head count had been a concern. However, by using DIRECTOR to generate schedules, the new volumes were handled with only an eight-percent increase in staffing. The new optimized plan resulted in a 37-percent increase in agent productivity. SGI was also able to improve customer-service levels by 40 percent and avoid millions of dollars in additional agent-related expenses. In addition, SGI increased caller satisfaction ratings by 47 percent.

Timberline Software

Timberline Software Corporation is an international supplier of accounting and estimating software for construction and property management companies. Timberline's work-force manager for client services previously spent a full 40-hour work week creating a 1-week schedule. Despite her long hours, creating the schedule manually could not accommodate last-minute changes and made it difficult to predict future staffing needs. DIRECTOR enabled Timberline to reduce the schedule creation time by 80 percent. This time savings allows Timberline management to focus on other duties, such as reporting, forecasting, and analysis.

Prior to deploying DIRECTOR, one of Timberline's greatest challenges was predicting future staffing needs. Using its traditional manual scheduling model, it predicted that it would need to increase its staff to 138 full-time specialists in 2000 to support its call volume. However, once it performed the analysis using DIRECTOR, it discovered it only needed as few as 107 full-time specialists. This reduction in future staffing represents substantial potential savings for Timberline, totaling more than \$1,000,000.

Compaq Canada

CONSUMER HELPDESK

Compaq's Canadian CONSUMER HELPDESK had already been recognized for operational excellence as "Call Center of the Year" by industry media. Recently, by deploying DIRECTOR, it was able to optimize its work-force processes even further and saw an immediate increase in customer-service performance and, correspondingly, in financial returns. In just the first quarter after deployment, Compaq Canada experienced the following performance and productivity improvements: call abandonment rate decreased 65.3 percent, average hold time decreased 57.3 percent, net service levels increased 16.3 percent, operational expenses decreased 15 percent, point-of-sale revenue for an agent increased by 17 percent, and gross margins increased 18 percent.

Conclusions

Staff scheduling has always been a problem of great practical importance. The recent growth in the contact center industry has highlighted the need for effective staff-scheduling systems. With their numerous complexities, real-world staff-scheduling problems have proven to be a fruitful application for AI-based techniques.

This article described Blue Pumpkin DIRECTOR, a staff-scheduling system for contact centers. DIRECTOR represents a significant application of AI techniques to solve a critical problem for an important industry.

DIRECTOR ENTERPRISE and its predecessor, DIRECTOR ESSENTIAL, have successfully been deployed at more than 800 contact centers worldwide and have provided significant, quantified benefits to their users. In addition, DIRECTOR is used daily (for scenario creation, modification, and scheduling) by call center managers with less than a week of training. This result demonstrates that powerful, expressive, constraint-based systems can successfully be used by users without an engineering or operations research background.

Acknowledgments

DIRECTOR represents the work of a large engineering and product marketing team at Blue Pumpkin Software. Thanks to Serdar Uckun, Rich Frainier, and Steve Chien for helpful comments and suggestions on this article.

Notes

1. A survey of classical operations research approaches to staff scheduling is Tien and Kamiyama (1982). A more recent survey focusing on call center operations is Koole and Mandelbaum (2001).

DIRECTOR is used daily ... by call center managers with less than a week of training.

2. Note that by *centralized*, we refer to organizational centralization. Call centers are frequently geographically distributed, with calls being routed to the most appropriate resource around the world. One of the challenges in modern call center scheduling is creating a coordinated schedule that uses resources from distributed call centers
4. Saddletree Research Report 0101, The U.S. Work-force Management Software Market, 2000-2004. Scottsdale, Arizona, 2001.
5. Datamonitor Corporation Research Report, New York, 1998.
6. Frost and Sullivan Research Report 6317-62, Agent Performance Optimization Software Markets, San Jose, California, 2001.
7. The standard period for which DIRECTOR is used to generate schedules is one week.
8. For clarity, we restrict this discussion to the simple scenario when agents only answer telephone calls. The definition of shifts and shift activities is slightly more complex when considering that agents can partition their time among several media types (for example, we can specify that an agent only answers telephone calls during a shift, or he/she can fully "blend" his/her phone and e-mail answering activities during a shift).
9. Preferences are entered either using the call center manager's DIRECTOR GUI client or the agents themselves using a web-based interface.
10. The underlying, structured scenario model in DIRECTOR can be manipulated as an XML document. However, it is hidden from end users.

References

- Chien, S.; Rabideau, G.; Willis, J.; and Mann, T. Automating Planning and Scheduling of Shuttle Payload Operations. *Artificial Intelligence* 1(14): 239-255.
- Cleveland, B., and Mayben, J. 1997. *Call Center Management on Fast Forward*. Annapolis, Md.: Call Center Press.
- Garey, M., and Johnson, D. 1979. *Computers and Intractability*. New York: Freeman.
- Jonsson, A.; Morris, P.; Muscettola, N.; Rajan, K.; and Smith B. 2000. Planning in Interplanetary Space: Theory and Practice. Paper presented at the Fifth International Conference on Artificial Intelligence Planning Systems, 14-17 April, Breckenridge, Colorado.
- Kleinrock, L. 1976. *Queueing Systems, Volume 1*. New York: Wiley.
- Koole, G., and Mandelbaum, A. 2001. Queueing Models of Call Centers: An Introduction, Technical Report, 2001-7, Department of Statistics, Vrije Universiteit.

Maes, P. 1994. Agents That Reduce Work and Information Overload. *Communications of the ACM* 3(7): 31-40, 146.

Minton, S.; Johnston, M.; Philips, A.; and Laird, P. 1992. Minimizing Conflicts: A Heuristic Repair Method for Constraint-Satisfaction and Scheduling Problems. *Artificial Intelligence* 58(1-3): 161-205.

Rabideau, G.; Chien, S.; Willis, J.; and Mann, T. 1999. Using Iterative Repair to Automate Planning and Scheduling of Shuttle Payload Operations. Paper presented at the Eleventh Innovative Applications of Artificial Intelligence (IAAI), 18-22 July, Orlando, Florida.

Rabideau, G.; Knight, R.; Chien, S.; Fukunaga, A.; and Govindjee, A. 1999. Iterative Repair Planning for Spacecraft Operations in the ASPEN System. Paper presented at the International Symposium on Artificial Intelligence Robotics and Automation in Space, 1-3 June, Noordwijk, The Netherlands.

Smith S. F. 1994. OPS: A Methodology and Architecture for Reactive Scheduling. In *Intelligent Scheduling*, eds. M. Fox and M. Zweben. San Francisco, Calif.: Morgan Kaufmann.

Smith, D.; Frank, J.; and Jonsson, A. 2000. Bridging the Gap between Planning and Scheduling. *Knowledge Engineering Review* 15(1): 61-94.

Tien, J. M., and Kamiyama, A. On Manpower Scheduling Algorithms. *SIAM Review* 24(3): 275-287.

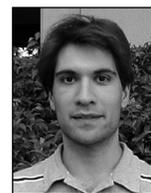


Alex Fukunaga is currently a senior member of the technical staff in the Artificial Intelligence Group at the Jet Propulsion Laboratory (JPL), California Institute of Technology. He was previously a senior software engineer at Blue Pumpkin Software. He received his M.S. and A.B. in computer science from the University of California at Los Angeles and Harvard University, respectively. His current research interests include search, combinatorial optimization, automated planning and scheduling, and machine learning. He has published over 30 technical papers.



Edward Hamilton is a senior software engineer at Blue Pumpkin Software. Since joining Blue Pumpkin in 1997, he has been working on the STAFF SCHEDULING ENGINE. Prior to working at Blue Pumpkin, he developed a conference scheduling system for Intel. He received a B.A. in mathematics and B.S. in computer science from the University of California at Santa Cruz.

His interests include computer chess, stock market prediction, and financial data sonification.



Jason Fama is currently a software engineer at Blue Pumpkin Software. He is working on the next iteration of the Blue Pumpkin DIRECTOR software and its scheduling algorithm. He previously worked at Rockwell Palo Alto Laboratory. He received B.A.s in computer science and economics from the University of California at Berkeley.



David Andre received his Ph.D. from the University of California at Berkeley in the fall of 2002 with a specialization in machine learning and reinforcement learning. Now at BodyMedia, Inc., he is heading up the Informatics Group, which is responsible for making BodyMedia's body monitor detect the wearer's context, energy expenditure, and other interesting biometric measures. Andre is a Hertz fellow, has written more than 50 technical papers, holds 6 patents, and is the author of a book on the automated synthesis of complex structures.



Ofer Matan is a cofounder of Blue Pumpkin Software. His research interests include statistical modeling, optimization, and machine learning techniques. Matan has conducted research in optical character recognition and computer learning systems at Bell Labs. He received his Ph.D. in computer science from Stanford University.



Illah R. Nourbakhsh is an assistant professor of robotics at The Robotics Institute at Carnegie Mellon University. He received his Ph.D. in computer science from Stanford University in 1996. He is cofounder of the Toy Robots Initiative at The Robotics Institute. His current research projects include electric wheelchair sensing devices, robot learning, theoretical robot architecture, believable robot personality, visual navigation, and robot locomotion. His past research has included protein structure prediction under the GENOME project, software reuse, interleaving planning and execution, and planning and scheduling algorithms. He is a cofounder and chief scientist at Blue Pumpkin Software, Inc., and Mobot, Inc.