Staff Scheduling for Inbound Call Centers and Customer Contact Centers

Alex Fukunaga, Ed Hamilton, Jason Fama, David Andre, Ofer Matan, Illah Nourbakhsh
Blue Pumpkin Software
884 Hermosa Court, Suite 100
Sunnyvale, CA 94086
{afukunaga,ehamilton,jfama,dandre,ofer, illah}@blue-pumpkin.com

Abstract

The staff scheduling problem is a critical problem in the call center (or more generally, customer contact center) industry. This paper 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 been successfully deployed at over 800 contact centers, with significant measurable benefits, some of which are documented in case studies included in this paper.

1. Introduction

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, while ensuring that certain constraints are maintained (e.g., 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).

While 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 *workforce management software*. The need for effective workforce 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. However, we found that this domain is particularly amenable to the application of constraint-based and heuristic scheduling techniques from artificial intelligence.

This paper 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 will be given.

Copyright © 2002, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

2. Staff Scheduling in Contact Centers

When a consumer calls a software vendor to ask for technical support, or if he 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.[1]

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 wish to avoid overstaffing (having idle agents when few customer calls arrive) in order to minimize the cost of operating the call center and maximize overall business profitability.

A standard goal for call center operations is to achieve a certain *service level*, i.e., answer X% of calls within Y seconds, while minimizing overstaffing.

It is well known that acquiring a new customer is several times more expensive (in terms of marketing/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 60-70% 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") as well as for 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

[1] 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 utilizes resources from distributed call centers.

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 (e.g., Outsourcer X agrees to handle Manufacturer Y's sales inquiries, and promises that 80% 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 upon service levels while operating profitably.

Although most interactive contact between customers and businesses still takes place through the telephone, customer contact through other media such as e-mail, on-line chat / 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 below.

Since the call center industry is not well-known in the AI/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 U.S. alone, expected to almost double by 2004 (Frost and Sullivan, 2001; Saddletree, 2001, Datamonitor, 1998). Approximately 7% of U.S. call centers were using a workforce management system. Note that the market penetration of workforce management software is still very low, in part because modern workforce 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 workforce management software is growing rapidly (the annual revenues for the call center workforce management software market were $175 million in 2001, expected to grow to over $500 million by 2006). (Frost and Sullivan, 2001; Saddletree, 2001).

The contact center scheduling problem poses a very challenging problem. Meeting the demand profile implied by the forecasts of incoming calls/contacts is by itself a difficult combinatorial optimization problem, especially considering that the forecasts are probabilistic. At minimum, a one-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 of the "off-phone" activities such as breaks also need to be scheduled. Furthermore, the recent advent of multi-skilled scheduling and multi-contact scheduling (see below) 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.

3. The Director System

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

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;
- Infrastructure for persisting scheduling scenario inputs and outputs in a relational database; and
- A GUI.

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

Workforce management software systems for contact centers includes much more additional functionality, such as 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 paper, which focuses on the scheduling functionality.

The current version of Director (3.1) is implemented as a set of Microsoft COM components, mostly implemented in C++. 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. A new version of Director Enterprise will be released in 2002 which is based on a more modern multi-tiered web-oriented architecture (a relational database, 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 was 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 paper, we will focus on the Enterprise version, since it provides a superset of the features of Essential.

Using Director to schedule contact center agents generally involves the following workflow. 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/constraints that apply to the agents (e.g., how many hours per week she can work, which days she is available, what times he prefers to work, etc) are entered and linked. Typically, that part of the scheduling scenario is relatively static from week to week.[2] For each week, the user (contact center manager) generates a forecast of the incoming calls/contacts (the demand profile). Then, she specifies a target service goal which 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 below.

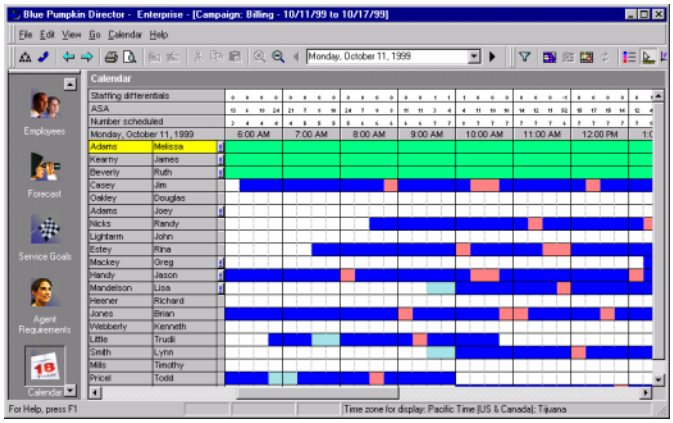

Figure 1: A screenshot of the Director GUI for manipulating schedules. Each row display's an agent's schedule (currently showing part of a day).

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 (e.g., the amount of time a contact center agent will spend talking on the phone, 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 which specifies that from 8:00AM-8:15AM, 10 calls arrive, with an AHT of 200 seconds, then from 8:15-8:30, 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 previous scheduling periods. Although it may 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 Over/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% of the incoming calls within 20 seconds.
- Send a reply to 99% of the e-mail inquiries within 24 hours.
- Answer calls within 30 seconds on average.
- Limit abandoned calls to 5% of the incoming calls (calls are abandoned when a customer hangs up the phone before an agent becomes available to talk to the customer).
- No agent should be idle more than 25% of the time.

Combinations of the above are possible, e.g., "Answer 80% calls within 30 seconds, no more than 5% of the calls can be abandoned, and no agent should be idle more than 20% of the time."

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 that time period to satisfy the service goal, given the forecast. Agent requirements are computed by applying the well-known Erlang-C formula from operations research/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 over/understaffing for each time interval in the scheduling period, we have the basis

[2] The standard period for which Director is used to generate schedules is one week.

for an objective function for evaluating a candidate schedule with respect to service goals.

Now, consider the following case: There are two queues, the "Widget Sales" inquiry queue, and 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 *multi-skilled* scenario differs from the previously described single-queue case, because it is no longer possible to straightforwardly compute how over/understaffed the schedule is for a particular time interval, due to 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 makes it very difficult to answer the question, "is the schedule 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 multi-skilled scheduling problem. It is possible to compute the service level by simulating the schedule and the call routing algorithm. However, simulations are very expensive (in the context of generating and optimizing 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 that are significantly different from phone queues, such as e-mail contact queues (and similar types of media such as faxes). E-mail contacts differ from phone calls in several important ways. First, the service goal usually involves much longer time periods than phone calls (an e-mail reply is usually expected within a day or so, while people expect phone calls to be answered within seconds or minutes). Second, e-mail inquiries are usually partitioned into many, sparse, virtual queues. Third, while phone calls are "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 phone and e-mail contacts *simultaneously*. For example a contact center agent might normally answer phone calls from the set of queues for which he is skilled, and when no calls are pending, she would reply to e-mail inquiries. Therefore, a modern contact center agent can no longer be modeled as a generic staffing unit who 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 multi-skilled scheduling and the non-phone-media scheduling problems described above, in addition to the traditional single, phone queue scheduling problem. This complexity makes it difficult to apply traditional operations research approaches (mathematical programming), since all known existing solutions (proprietary algorithms in commercial systems, including Director) rely on some form of simulation model. This makes constraint-based and iterative scheduling approaches from AI particularly appealing for the contact center scheduling problem.

Constraints

Employees have various constraints that determine how/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 phone calls.[3] A shift may contain a number of *off-phone activities* during which she is not available to pick up calls (e.g., 1-hour meal breaks, 15-minute breaks, etc).

The basic constraints in Director specify parameters such as the duration and possible start times of shifts, the duration and possible start times of off-phone activities. For example, we can specify that an "8-hour standard shift" is 8 hours long, starts between 9AM and 1PM. Furthermore, we can specify that this class of shift contains a "lunch break", 1-hour off–phone activity, which starts between 3-4 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 upon these building blocks with *shift pattern* constraints that specify constrain which shifts can be worked on which day. For example, we can say that Joe can either work an "8-hour Standard Shift" or "4-hour Special Shift" on Monday, must work a 4-hour Special Shift on Tuesday, and must not work any shifts on Sundays.

[3] For clarity, we restrict this discussion to the simple scenario when agents only answer phone calls. The definition of shifts and shift activities is slightly more complex when considering that agents can partition their time among several media types (e.g., we can specify that an agent only answers phone calls during a shift, or he can fully "blend" his phone and e-mail answering activities during a shift).

The user can also specify constraints on the number/amount of occurrence of various objects, e.g., "Bob must work between 3 and 4 weekend shifts per month", "Alice must work no more than 80 hours per 2 weeks", "John can not 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 are treated as soft constraints by Director. One type of preference is a rank-ordering on the start times of the shifts, e.g., John prefers to start between 8-9AM on Mondays, but if that's not possible, start between 9-10AM, and would really prefer not to start shifts in the evenings. Agents can also express preferences about the set of shifts they work, e.g., "I would much rather work on the day shifts Monday through Friday than on the night shifts".[4]

Although most planning/scheduling systems with a highly expressive constraint system use a programming-language-like textual modeling language to specify constraints, this 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 enables most of the end users of Director to specify complete scheduling scenarios with little, if any, assistance from Blue Pumpkin consultants or technical support staff.[5]

The constraint system in Director is very expressive, and can express almost all constraints currently required by the contact center market (due to lack of space, we have 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 which 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 vs., "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 that 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 that day. Instantiating a shift in a slot results in the instantiation of variables representing off-phone activities (thus, there is a one-level abstraction hierarchy consisting of slot and off-phone 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 bindings for those variables that maximizes 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 which takes as input all of the slots for all the agents, then run the search algorithm until some termination criterion is met.

While 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 that limited search space. In classical

[4] Preferences are entered either using the call center manager's Director GUI client, or by the agents themselves using a web-based interface.

[5] The underlying, structured scenario model in Director can be manipulated as an XML document. However, it is hidden from end users.

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), in that rather than only repairing constraint violations, a search algorithm could be run on a set of variables either for optimization (e.g., "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 we have observed that it is a good policy to run some heuristic periodically (e.g., "once in a while, unbind all slots for an agent and reschedule him").

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 500Mhz Pentium-III desktop machine; a 1000-agent multi-skilled scenario takes 30-60 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 increases in the number of skills, then the scaling is better than linear. This is because there are few hard constraints that involve more than a single agent, which means that 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 recomputation of the service goal terms in the objective function. For example, when the start time of a shift is changed from 8AM to 9AM, what is the impact on the service goals? For a single phone queue scenario, this computation is relatively inexpensive (but still the major bottleneck); for multi-skilled scenarios with e-mail queues, this 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. This means 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 constrasts 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 above, there are a number of related scheduling problems that are addressed by Director. We describe some of these below.

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, since 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 in order to minimize the negative impact on service goals.

Workforce Planning

So far, 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" which represent typical subclasses of agents (and are linked to various constraints), and some additional constraints (e.g., restrictions on the number of percentage of class profile instances, budget constraints), generate a schedule consisting of "phantom agents" (instances of the employee class profiles) which optimizes the global objective function.

This *workforce planning* problem is important for users who need to plan future hiring of contact center agents, i.e., 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 2

employee class profiles. Profile#1 represents an agent who can only answer Widget Sales calls, costs $15 per hours, and works 40 hours per week. Profile#2 represents an agent who answers both Widget Sales and Technical Support calls, works 20 hours per week, and earns $25 per 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 workforce planning is a new application where there is clearly a need for further research.

Multi-week Constraints and Scheduling

Currently, Director schedules one week at a time. This is 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 workforce management software.

However, there are various constraints that have a time period other than one week, e.g., "Joe must work between 2-3 weekend shifts every 4 weeks." The Director scheduling algorithm handles such multi-week constraints by assuming that the shifts can be distributed evenly among 4 weeks, but it is clear that such heuristics can fail. It might seem that if we scheduled all four weeks at a time, then this is not 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 (due to unscheduled absences) become invalid (or the data required to make reasonable assumptions might be unavailable). Therefore, scheduling with multi-week constraints is another area where we will focus further research and development efforts in the future.

4. Application Deployment and Case Studies

Blue Pumpkin Director (including both the Enterprise version and the Essential version) is currently in use at over 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 which is the focus of this paper) 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 centers with 150-1000 agents. In addition, Essential (described in the "System Architecture" section) is also in use by over 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 the integration of the software with the ACD (see System Architecture section), and setting up the server. In many cases, the end users create the scheduling scenarios and run the scheduling algorithm themselves (including all constraints) 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, the training period is typically several a few days before the users become proficient with 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.

Below, we describe several cases 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 business combined with a multi-skilled contact center made optimizing its workforce 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 as Borders Group transacts a high volume of its business during the holiday season. During this period, there is a surge of over 35% in call volume, making optimizing available resources and staff essential.

After deploying Director, Borders Group evaluated various staffing scenarios to design a workforce optimization strategy that accurately reflected all of Borders' business goals. Based upon 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 required skills instead of cross-training agents on multiple skills, Borders Group was able to get seasonal staff on the phones 33% faster, allowing them to be productive in one week instead of three.

Director enabled Borders Group to increase agent productivity by 53%, with a 33% reduction in expenses by allocating agent time more effectively over operating hours. Customer service levels of 88% 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 non-seasonal employees from 15 percent to 10 percent. These factors contributed to a 25% 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 workforce physically located in four time zones. SGI also decided to bring all customer contact in-house, increasing call volumes 50% to 2,500 to 3,000 calls per 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, headcount had been a concern. However, by using Director to generate schedules, the new volumes were handled with only an 8% increase in staffing. The new optimized plan resulted in a 37% increase in agent productivity. SGI was also able to improve customer service levels by 40% and avoid millions of dollars in additional agent-related expenses. In addition, SGI increased caller satisfaction ratings by 47%.

Timberline Software

Timberline Software Corporation is an international supplier of accounting and estimating software for construction and property management companies. Timberline's workforce manager for client services previously spent a full 40-hour work week creating a one-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%. 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 their traditional manual scheduling model, they predicted that they'd need to increase their staff to 138 full-time specialists in 2000 in order to support their call volume. However, once they performed the analysis, using Director, they discovered they only needed as few as 107 full-time specialists. That 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 previously recognized for operational excellence by being named "Call Center of the Year" by industry media in recent years. Recently, by deploying Director, they were able to optimize their workforce 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%
- Average hold time decreased 57.3%
- Net service levels increased 16.3%
- Operational expenses decreased 15%
- Point of sale revenue per agent increased by 17%
- Gross margins increased 18%

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.. Real-world staff scheduling problems, with their numerous complexities have proven to be a fruitful application for artificial intelligence-based techniques.

This paper 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 been successfully deployed at over 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 demonstrates that powerful, expressive constraint-based systems can be used successfully by users without an engineering or operations research background.

Acknowledgements

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 paper.

References

Chien S, Rabideau G, Willis J, Mann T, Automating Planning and Scheduling of Shuttle Payload Operations," Artificial Intelligence, 114, pp.239-255, 1999.

Frost & Sullivan Research Report 6317-62, Agent Performance Optimization Software Markets, San Jose, CA, 2001.

Datamonitor Corporation Research Report, New York, 1998.

Garey M, Johnson D. *Computers and Intractability*. New York: W. H. Freeman. 1979.

Jonsson A, Morris P, Muscettola N, Rajan K, Smith B, Planning in interplanetary space: Theory and practice," Proc.5th Intl Conf Artificial Intelligence Planning Systems, CO. April, 2000.

Kleinrock, L. *Queueing Systems*, Vol 1, New York: Wiley, 1976.

Minton S, Johnston M, Philips A, Laird P. Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. Artificial Intelligence, 58:161--205, 1992.

P. Maes. Agents that Reduce Work and Information Overload. Communications of the ACM. Vol. 37, No.7,pp. 31-40, 146, ACM Press, July 1994.

Rabideau G, Chien S, Willis J, Mann T. "Using Iterative Repair to Automate Planning and Scheduling of Shuttle Payload Operations," Innovative Applications of Artificial Intelligence (IAAI), Orlando, Florida, July 1999.

Rabideau G, Knight R, Chien S, Fukunaga A, Govindjee A. "*Iterative Repair Planning for Spacecraft Operations in the ASPEN System*," International Symposium on Artificial Intelligence Robotics and Automation in Space, Noordwijk, The Netherlands, June 1999.

Saddletree Research Report 0101, The U.S. Workforce Management Software Market, 2000-2004. Scottsdale, AZ, 2001

Smith D, Frank J, Jonsson A. Bridging the gap between planning and scheduling. Knowledge Engineering Review, 15(1):61--94, 2000.

Smith SF. OPIS: A Methodology and Architecture for Reactive Scheduling. In M. Fox and M. Zweben, ed., Intelligent Scheduling. Morgan Kaufmann, 1994.