Workforce Scheduling in Inbound Customer Call Centres With a Case Study

Author(s) anonymised

ABSTRACT

Call centres are an important tool businesses use to interact with their customers. Their efficiency is especially significant, since long queuing times can reduce customer satisfaction. Assembling the call centre work schedule is a complex task that needs to take various and often mutually conflicting goals into account. In this paper, we present a workforce scheduling system suited for small to medium call centres and adjusted to the needs of a real-world example. The demand is forecasted based on data gathered from the call activity history and the staff schedule is constructed using a GRASP metaheuristic. The scheduling problem is to minimise the difference between allocated and forecasted number of staff members while also caring for numerous legal and organisational constraints as well as staff preferences. Additionally, a flexible constraint handling framework is devised to enable rapid prototyping methodology used during the development. Algorithm performance analysis on several realistic problem examples is provided. The devised system is successfully implemented in a real world setting of (*Anonymised financial institutions*).

1. INTRODUCTION

Virtually all types of human organisations occasionally face a certain type of a scheduling problem. Staff scheduling is a classic operations research problem that consists of assigning a set of employees to a set of working times, subject to various constraints and with the goal of finding the schedule of the best quality. Solving them to optimality in a way that comprehensively takes costs, regulations as well as the employee preferences into account can be difficult [7]. Moreover, such problems are $\mathcal{NP} - hard$ [15, 11] in their very simplest forms, which imposes tractability issues as well. Contact centre staff scheduling is an example of this type of a problem [15, 7, 8, 1].

Call centres are an instrument many companies use for communication with their clients. They are commonly used to provide technical support, perform sales, handle various customer inquiries etc. They typically consist of a centralised pool of trained staff members called *agents*. When a customer dials the number of a company, if there are available agents in the centre, her call is answered immediately. However, if all staff members are busy at that moment, the customer will have to wait in the queue until an agent becomes available. Long waiting times can cause customer dissatisfaction and it is critical for the contact centres to keep the waiting times as low as possible. On the other hand, to keep the operating cost reasonable, hiring too much staff is also undesired.

Contact centres can be classified in various ways. In a traditional call centre, staff members are communicating exclusively via telephone. With the increasing popularity of the Internet, many companies allowed their clients other means of contact, such as e-mail or chat, extending call centres into their contemporary generalisation called *contact centres*. *Multi–channel* contact centres are supporting multiple channels, while call centres are an example of a *single–channel* centre. Centres that are answering incoming inquiries, but are never actively initiating communication are called *inbound* contact centres, while a centre that is only initiating communication with the customers is called an *outbound* contact centre. If both answering and calling (or sending e-mails etc.) is performed, the service is classified as a *mixed* contact centre. With regard to the staff skills, a contact centre can be *single–skilled* or *multi–skilled*. In a simple, single skilled centres, all of the employees have received the same training and theoretically, provide a homogeneous service level, independent of the agent. In a multi–skilled centre, there are various profiles of agents, depending on their skill sets.

In this paper, we describe a call centre scheduling algorithm based on a GRASP metaheuristic implemented in a commercial software package. It is suited for the needs of a small to moderately sized single–skill inbound call centres. The system is based on a flexible constraint management framework that allows easy addition of new company–specific constraints and a robust, scalable optimisation algorithm based on the GRASP metaheuristic.

Related work proposes a diverse range of techniques such as dynamic programming, linear and mixed integer programming, relaxations of the linear programs [12, 2, 13, 3, 10, 4]. Several heuristic methods are also proposed such as [14]. However, the aforementioned work is focused on the optimisation part, and less on the constraints imposed by the organisation. As an exception, a hybrid heuristic approach with several algorithms including an algorithm inspired by

GECCO'15, July 11-15, 2015, Madrid, Spain.

simulated annealing proposed in [5] is applied to a real–world problem. Constraint handling appears to be performed through the means of the objective function, therefore defining them as ranked soft constraints that can be violated in final solutions.

We are focused on GRASP, which was, to the best of our knowledge, never successfully applied to this type of problems. While a direct heuristic to schedule weekend working days is also used, times of arrival are scheduled relying exclusively on a metaheuristic method, instead of using hybrid algorithms. As compared to related work, the problem we are solving is a *highly constrained* example of a *real world* problem. A flexible constraint handling system is a highly prominent feature of our system. Moreover, our approach deals with the constraints differently than any other proposed method we are aware of. Instead of implementing them in the objective function as in [5], we devised a rule based ensurance system in each component of the algorithm. In that manner all but one of the constraints is guaranteed to be satisfied in solutions the algorithm is working on throughout the execution.

To conclude this analysis, we are focused on improving small to medium contact centres with 20-40 employees, while related work in real world scheduling mentions higher number of staff, up to a thousand. Contrary to common first notion, small contact centres might prove to be more difficult to schedule than big ones [9]. As a further differentiation, in this stage of our work, we are investigating single–channel, single–skilled centres only, not taking into account complexities of multi–channel and multi-skilled ones. The rationale behind that decision was to provide a quick–to–develop solution for such centres. Solutions suited for larger and more complex centres surely do exist, but our goal was to produce a simple product with low development cost, and therefore competitive price, affordable even for small businesses.

As noted in [6], a noticeable gap was observed between the research output from academia and the needed expertise that can be directly utilised by the industry. Solving real problems is difficult since it necessitates modeling and handling various kinds of constraints which are usually simplified in academic problems. This paper aims to contribute to bringing the worlds of academia and practical implementations together.

The remainder of this paper is organised as follows: in Section 2, the formal problem description, including the objective function and constraints are given. Section 3 gives a brief overview of the demand forecasting used in our approach. The optimisation algorithm for generating schedules is described in Section 4. Section 5 brings the results and some ideas for future improvements and the paper is concluded in Section 6.

2. PROBLEM DESCRIPTIPON

The call centre workforce scheduling problem ($CCWSP$) is a scheduling problem whose goal is to satisfy all the imposed constraints and maximise the *service level*. It is typically defined as a percentage of answered calls (more is better) and percentage of dropped calls (less is better) during a time period. In our system, schedules are typically generated one month ahead, with adjustable target service levels. Times are quantized, with minimum quantum duration given as a parameter. 30 minutes quanta were used in this case study, meaning that all events in the schedule occur on the full hour or full hour and 30 minutes. If needed, finer granularity could be used. This naturally means that the optimisation process might take more time.

The problem is defined as an ordered triplet:

$$CCWSP = (S, sd(t), C),$$

that consists of a set of staff members S, ideal staff time distribution $sd(t)$ and a set of constraints. The set of staff members is a fixed set of workers at the contact centre, with their permanent workplace assigned for each staff member (seating scheduling is therefore not a part of the problem). Each staff member is defined by its identifier in the system and data about the staff member is stored in a suitable database.

The ideal staff distribution $sd(t) : \mathbb{N} \rightarrow \mathbb{N}$ is a function which defines a minimum number of employees needed at time t, in order to achieve the desired service level. As the frequency of calls varies throughout the days of the month, staff number that is needed to handle such demand will also vary.

The set of constraints is inspired by the labour law in the (*Anonymised country*), company specific policies and regulations. Furthermore, it's easy to notice that many of the constraints were defined out of a desire to keep the employees satisfied by giving as much flexibility as currently possible. The set of constraints can be divided into general contact centre constraints, general employee constraints and company specific constraints. Most of the constraints are related to a single worker but there are some dealing with aggregate number of working staff during weekends.

Contact centre constraints include:

- Contact centre working hours during working days, Saturdays, Sundays and holidays,
- Shifts intervals, e.g. morning shift with arrival to work 7–10am, afternoon shift with arrival 11–14h.
- Maximum allowed daily work time (8 hours according to the (*Anonymised country*) labour law applied while the system was implemented).
- Minimum daily break for staff members (minimum time between consecutive presences at work, 12 hours according to (*Anonymised country*) labour law applied while the system was implemented).

Basic staff constraints include:

- Working days for the staff member (e.g. some agents are working Mon-Sat, while some, part-time agents, are only working Mon–Wed).
- Duration of the work time (can be adjusted for each day of the week and holidays, e.g. worker can work 8 hours Mon-Fri and 7 hours on Saturdays).
- Prearranged absences (vacations, free days, sick leaves).
- Prearranged presences (for example, an employee might arrange her schedule in advance with her manager).

Some company specific constraints in our case study include:

- Workers arrive to work at the same hour throughout Mon-Fri, but if they are working Saturdays and Sundays, they can be scheduled at a time different from their work Mon-Fri.

- Simple night shift rotating rules for employees are supported, which defines night shifts as a specific case of prearranged presences.
- Employees working on Saturdays need to have at least two working Saturdays, if needed (during 5 Saturday months), an employee can work three Saturdays, but only if he or she didn't work three Saturdays during the previous month.
- Shift work constraint – employees working in both morning and afternoon shifts need to have at least 5 days in the morning shift and at least 5 days in the afternoon shift to ensure they are entitled to their monthly shift work bonus.
- To ensure equal service levels during weekends, number of staff members during Saturdays needs to be roughly equal.
- To increase staff satisfaction, an employee can be scheduled in highly undesired shifts (e.g. going to work at 10, 11 and 14h) only one week per month.
- To increase staff satisfaction, an employee cannot be scheduled in undesired afternoon shifts during two consecutive weeks.

While these constraints are highly specific for the case study call centre, the implementation of the constraint management system is modular and parametrised, which is intended to enable easy integration into similar call centres and further customisation.

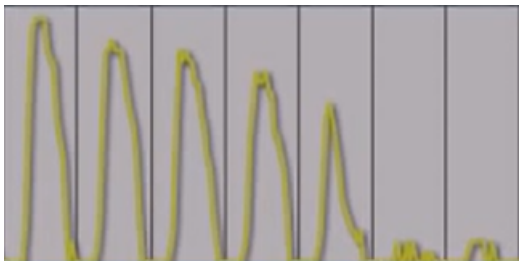

Figure 1: An example staff distribution curve for one week as seen in the user interface (Mon-Sun)

3. METHODOLOGY OVERVIEW

Similarly to related approaches, the process of scheduling the contact centre workforce is performed in three stages: (1) future calls forecasting, (2) staffing and (3) scheduling. While staff and constraints are in that Unlike in [4], scheduling and rostering is both done in the same stage and performed by the metaheuristic. In the remainder of this Section, the first two stages will be briefly described, while the scheduling algorithm is elaborated in detail in the following Section. The process starts with the contact centre manager telling their staff to enter their preferences and absence days, like arranged vacations etc in the scheduling system. Therefore, S and C component of the contact centre are defined by the users through a suitable graphical user interface.

The ideal staff distribution $s(t)$, however, isn't known in advance and is determined during call forecasting and staffing phases. The call forecasting is based on the of inbound calls records and is estimated based on the call centre activity history. Currently, a simple forecasting model is used that estimates number of calls during time quanta based on the history data for the current month of a previous year. January from the previous year is used to forecast this year's January, with the days shifted to ensure that the days of the week in the previous January match the week days of the current month. In that way, working days, weekends and most non working days are aligned. The greatest variation in the volume of calls is dependent on the day of the week and the season of the year. For example, late December peaks and low intensity summer weeks are common in our case study every year, as well as intense Mondays and low intensity weekends. Therefore, even such a simple model provided a good estimate of the future calls customers were satisfied with.

After call intensity forecasting, the ideal staff distribution is calculated. Since our call centre has a single queue and is single–skilled (any agent can answer any call), there is an analytical solution to calculating the number of needed staff members. This calculation is computed using the Erlang-C [7] formula frequently used in operations research and queuing theory. An example of an ideal staff distribution curve calculated by the system for an example real–world week is shown on Figure 1. It is visible that while Monday has a considerable peak, just a few staff members are needed to successfully handle the weekend calls.

Since the calculated curve is based on the previous demand, the system allows manual staff distribution curve adjustments. For example, a call centre manager might know that due to the new campaign, a lot of calls might be expected late evening and therefore she can adjust the curve to meet this new demand. Furthermore, specific events such as promotions of new services might increase demand during several days in the month, which can be easily modified using the graphical user interface.

After defining an ideal staff distribution, it is needed to find the workforce schedule that matches this distribution as close as possible. This is a highly complex problem for which simple approaches such as exhaustive search are not suitable for practical use. The size of the search space grows very fast and such approaches are prohibitively slow for all but the smallest problem instances. Metaheuristic approaches have been especially successful when dealing with this type of issues. While they usually give no warranty on solution quality and do not assure that an optimum solution will be reached, they have shown very good results on various types of difficult combinatorial optimisation problems. The development methodology and the scheduling algorithm is described in more detail in the following sections.

4. DEVELOPMENT APPROACH

While the project was initiated, an initial screening revealed that the contact centre schedules were built by hand. The contact centre managers relied mostly on their subjective judgement, as there was no time for detailed analyses. Furthermore, since the beginning of the development, the clients insisted that the scheduling system needs to guarantee that most, if not all constraints are satisfied in each of the produced schedules. Therefore, the usual approach of implementing constraints by the objective function was not suitable for this purpose. Additionally, while general

ideas about the constraints were agreed upon, they were in general not detailed enough to allow simple translation into scheduling rules to be implemented in a scheduling system that needs to be able to respond to various possible outcomes in a sophisticated way, in essence, acting as a replacement for the human that performed scheduling.

Agreeing on the definition of the necessary constraints for the workforce scheduling system is a difficult communication issue. The knowledge of the persons who was scheduling the workforce manually can be characterised as *tacit knowledge*, something that is intuitively clear, but difficult to explain, formalise and automate in a way that works satisfactory. Therefore, a prototyping approach was chosen. Prototypes of a scheduling system were presented on regular meetings with the call centre management and further improvements were discussed and agreed upon. Since the definition of the problem varied through time, a flexible constraint assurance system was needed.

We decided to use a system in which all of the constraints except one are satisfied throughout the search process. The only exception is the shift–work constraint that was added later and is a combinatorial optimisation problem on it's own, in current version partially solved by postprocessing. The devised system is a modular object oriented set of classes and interfaces written in Java. Constraint management system is based on *constraint objects* working on a common list of suitable times where a staff member can be scheduled. For each staff member, an independent set of constraint objects can be added. After scheduling an agent to a certain time slot, all of the constraints assigned to him/her are triggered and the list of suitable times is updated to reflect the effects of each change in the schedule.

That type of system also enables tracking of quanta which are currently suitable for scheduling. In each stage of the search, the algorithm can easily determine which quanta are suitable to schedule a staff member by simple querying of the suitable times list. For example, the unpopular times constraint is implemented in such a way that if a staff member is scheduled at an unpopular time, it removes other unpopular times throughout the month for that staff member, making such quanta "invisible" to the search algorithm in the remainder of the run, thus further reducing the size of the search space.

The algorithm was planned to be built as a set of components that can be connected into a desired metahuristic algorithm. The plan was to stop the development as soon as satisfiable optimisation results have been achieved. First, a random solution generator was devised which schedules staff at random feasible times. Afterwards, a local search and perturbation operator were developed. These two basic components are a foundation for several metaheuristics and were then combined into a GRASP and Iterated Local Search algorithms that showed very good performance for the contact centre in our study.

5. SCHEDULING ALGORITHM

In this work, a GRASP (Greedy Randomised Adaptive Search Procedure) metaheuristic is implemented to solve the $CCWSP$. This metaheuristic uses a randomised solution construction component and a local search operator to build different locally optimal solutions while keeping record of the best solution found so far. The metaheuristic restarts the local search from a different solution in each iteration in order to escape local optima. The GRASP metaheuristic runs only after several initial preprocessing steps have been performed. The algorithm works in three stages. First, preprocessing is done to check if the problem to be solved contains any misconfigurations that render it infeasible. Prearranged presences and absences are also handled during the preprocessing stage. Since all schedules have some preexisting presences and absences, working days for each staff member are first split into *statically* and *dynamically* scheduled days. For statically scheduled days, the schedule is known in advance, due to sick leaves or prearranged working time for example. Such days are not subject to optimisation and are not modified by the algorithm.

After the preprocessing step is successfully done, for each staff member, weekend working days are determined using a direct heuristic algorithm that ensures weekend work constraints are satisfied. Note that this stage only determines which Saturdays and Sundays a person is working, but not at which time during weekends. When all the working days are known, an optimised data structure representing the search space is built. For example, since staff always comes at the same time Monday – Friday, for most staff members, Monday can be used as a single representative of the five weekdays. Such preprocessing significantly reduces the size of the search space. The second phase is finished when search space is successfully built. Note that at this stage only the working days are known, but the schedule is still empty (working times have not yet been determined).

After the search space is built, an optimisation algorithm determines suitable working hours for the dynamically scheduled days (those that are subject to optimisation). It is clear that scheduling a staff member at the time t during the day d has consequences for the work times during the remainder of the month due to a complex set of imposed constraints. To ensure that complete schedules are feasible, the robust and extensible constraint assurance system described in the previous Section is used. As already mentioned, in the current implementation, throughout the search process all except the shift work constraint is always satisfied. Three individual components of the optimisation system were built: random solution generator, local search operator and the perturbation operator. Using these components, three metaheuristics were built: *random restart search*, *Random restart local search* and *Iterated local search*.

5.1 Objective function

The objective function for evaluating solutions is based on the ideal staff distribution. The closer real staff distribution over time in a candidate schedule is, the solution is better.

An ideal timetable will have exactly $sd(t)$ staff members available at time t. The number of staff at time t in candidate schedules is denoted $s(t)$ Since ideal staffing curve can frequently have one hour spikes and irregularities, and the typical working time is around 8 hours, it's generally impossible to achieve absolute accordance to the ideal staffing curve. However, the goal of the optimisation algorithm is to create a schedule for which the number of staff will be as close to the ideal as possible.

This leads us to the definition of the objective function, which defines penalty for each candidate schedule as a sum

of penalties for all time quanta in the schedule:

$$p = \sum_{t=0}^{t_{max}} (sd(t) - s(t))^2,$$

where t_{max} is the last quantum in the scheduling period and $s(t)$ is the number of staff in the generated candidate schedule, as opposed to the goal number of staff members $sd(t)$ during time t.

Note that the number of staff missing is squared. That way, the penalty function is always positive. The rationale for such objective function is the decision to emphasise bigger deviations. A bigger penalty means lower solution quality and bigger difference from the ideal staff distribution. The best possible solution would perfectly follow the ideal distribution and have a penalty of zero.

5.2 Random solution generator

The random solution generator uses the described constraint ensurance system to produce an initial solution in which all of the working times are chosen randomly but from a set of feasible times. The scheduling is done sequentially, for all of the working days in the schedule, for each staff member that works on the work day. As each scheduling decision is made, the constraints handling system is keeping updated information about which working times are suitable for the employee. As the schedule is having more working times determined, less and less time intervals are suitable, since many constraints contradict each–other. In case of a failed constraint setup for an employee, in very rare occassions, the schedule might be empty since a constellation of a conflicting constraints determined no working hours are suitable for the staff member. In such cases, users are advised to carefully revise the input data or ask for help from the support team.

5.3 Local search and perturbation operators

Local search operator is a simple greedy operator that tests if moving a staff member earlier or later during the day helps make a better schedule. It works only on feasible times for the staff member and does nothing if there are no other feasible times during that day. Local search is performed for each staff member, for each day in the search space. To promote discovery of good solutions, the order in which local search is applied on staff members is randomised. The local search operator has one parameter: number of passes through the entire schedule. If number of passes is set to 2, the local search algorithm will be performed twice.

Note that the order in which the staff schedules are optimised is always randomised, so that in general the sequence of optimizing staff members is different for each call. In that manner, unfair schedules are avoided, since initial solutions tend to have different distribution than the goal staff distribution. For that reason, some bias was discovered in the agents that were the first to be optimised had a tendency to be scheduled to unpopular afternoon times when the demand is in general higher.

The perturbation operator performs modifications of a solution that should extend behind the scope of local search. It modifies the solution, in a parametrisable manner so that the local search does not converge back to the initial solution, but not as much to degrade the algorithm to GRASP. The parameter of staff percentage to be modified determines the percentage of staff members for which the schedules will be modified. For each staff member, and for each time in the dynamic schedule, a random choice of a feasible work time is performed. In that manner, for some quanta, the perturbation operator might not modify the working time, depending on the size of suitable work times.

5.4 Random restart search

After having an initial solution generator, a trivial metaheuristic of random restart search can be easily implemented. It is frequently used as a baseline to check initial versions of more advanced algorithms. The algorithms should, at their very least, be able to significantly outperform random restart search. As the end condition, a timeout of iterations is used, currently fixed to 1 minute, as described in the parameter tuning subsection.

5.5 GRASP

After having an initial solution algorithm and the local search, a complete GRASP metaheuristic can easily be built. Our implementation starts by building a random initial feasible solution, on which the local search operator is applied. After that, the procedure is repeated, all the time keeping a record of the best solution found so far. The algorithm pseudocode is given in Algorithm 1. As the end condition, a timeout of iterations is used, currently fixed to 1 minute.

```
generate initial solution S_0 ;
S = local search (S_0) ;
S_best = S ;
while end condition not met do
    generate initial solution S_0 ;
    S = local search (S_0) ;
    if eval(S) < eval(S_best) then
        S_best = S ;
    end
end
```

Algorithm 1: GRASP metaheuristic pseudocode

5.6 Iterated local search

After having two components previously mentioned and a perturbation operator, the Iterated local search can be built. Iterated local search uses the perturbation operator instead of using new solutions to escape local optima. As with the previous two approaches, a timeout of one minute is used. The algorithm pseudocode is given in Algorithm 2.

```
generate initial solution S_0 ;
S = local search (S_0) ;
S_best = S ;
while end condition not met do
    generate initial solution S_0 ;
    S = local search (S_0) ;
    if eval(S) < eval(S_best) then
        S_best = S ;
    end
    perturb (S);
end
```

Algorithm 2: Iterated local search metaheuristic pseudocode

6. RESULTS

To achieve good performance, basic parameter tuning was performed before testing the algorithm. The metaheuristics were implemented in Java 1.8 programming language. All experiments were run on a computer equipped with a 2.4 GHz Intel Core i7-4700HQ processor and 16 GB of RAM. The algorithm was running using Java 1.8.0, subversion 11-b12 runtime environment.

Since random restart local search is a very simple method, it has shown to produce relatively low quality results, as expected. Running it longer does not help much, as our initial tests with 30 runs of 100000 iterations shown that 95% of the improvement is on average achieved during first 22 thousand iteration. 90% of all the improvement can be achieved in even lower 6200 iterations (just 6% of the tested number of iterations). An average performance ranges around 30000 penalty points, as opposed to almost 50% lower values for the GRASP and iterated local search methods.

A more advanced GRASP metaheuristic based on local search has achieved much better performance. The tuning of the local search iterations parameter was done with 7 different values of the parameter. For each value, 30 one minute runs were performed. It was shown that the success of the method does not significantly depend on the number of iterations. However, a tendency to have a quality drop at very high values is evident in the table. This might be explained by the fact that the local search does not discover features of the problem that lead to global optimum and the increased duration of detailed local search inhibits the restart mechanism that gives the search a needed ”kick” to escape the local optimum. As a final recommendation, 2 iterations are given as a guideline. The time limit of one minute was determined by a similar progress analysis like in the example of the random restart search.

	Iterations						
	1	2	5	10	20	50	100
average	14585,8	14383,87	14453,13	14559,93	14650,33	14779,47	15123,27
median	14592	14384	14448	14600	14664	14790	15152
min	14206	14096	14202	14176	14080	14238	14686
max	14802	14638	14762	14820	14920	15272	15654
std.dev	143,8432	130,9435	154,9529	161,3439	192,0324	218,1167	218,4052

Figure 2: An example staff distribution curve for one week as seen in the user interface (Mon-Sun)

Preliminary tests with the iterated local search have shown it is comparable to its performance. It appears higher perturbation rates help improve the performance, but even with the percentage of modified staff set to 100%, the performance was slightly worse. The reason for such effect might be the fact that the large number of restrictive constraints prevent the operator from moving the solution sufficiently to escape local optima. As the perturbation works only in the feasible neighbourhood, its ability to significantly change the solution is limited. This might not be the case in such extent for the GRASP, since in this approach the mechanism to escape local optima is starting from a new solution. Solution generator, as described has more freedom in choosing times, especially during early phases of solution construction. It should be noted that a more sophisticated perturbation could improve the performance and this cannot be interpreted as a evidence that iterated local search is in general inferior to GRASP.

7. FUTURE WORK AND CONCLUSION

The described system has been successfully implemented in (*Anonymised company*), the largest (*Anonymised country*) credit card vendor and in (Anonymised bank). Previously, the call centre schedules were created manually, which is highly labour intensive. Managers needed more than one full working day each month to create an initial version of the schedule. They were creating schedules while mostly having organisational issues in mind and without any way to assess the impact of the scheduling on the service quality. Furthermore, the implementation at the second institution proven the system to be flexible and generic enough for a relatively quick implementation without major constraint changes.

Performance analysis has shown that the forecasts are highly accurate and that the devised schedules in general achieve desired performance levels. While the comparisons to the time before the system was used are difficult due to staffing fluctuations, some performance indicators have increased after implementation. The devised system produces schedules that are ready for use in around a minute. While the schedules generated by the system usually still need some minor manual adjustments, the automation has brought the institution both great savings in time as well as a well defined set of rules according to which the schedules are created. The graphical user interface intuitively displays current schedules and distributes it to the staff and the effect of each change in the schedule can be visualised in a simple schedule visualiser. In case of understaffing, the graphical tools highlight parts of the schedule that are understaffed and might have service quality lower than expected. The devised system can also provide suggestions for staff planning in case of frequent understaffing.

While the early version of the system has shown promise, a lot more can still be done to make the system more flexible and suited to the needs of modern contact centres, such as implementing the mentioned shift–work rule in the constraint ensurance system. First of all, lunch breaks and short breaks are now handled outside of the system. Break scheduling could further improve service quality. As a major feature, multi–channel and multi–skilled contact centres should also be supported. These types of contact centres are especially interesting since nowadays there is no known analytical way to calculate their ideal staff distribution. Most of the existing systems are using some sort of a heuristic or a simulation to evaluate the quality of their schedules.

We have shown that a very simple metaheuristic such as GRASP can achieve performance that is good enough to be implemented in a commercial setting of a credit card vendor customer service. The implemented system helped the contact centre management to save time on scheduling and focus on other important tasks. Furthermore, in contrast to manual scheduling based on intuition and experience of a person doing the schedules, the automated scheduling system and related graphical tools have enabled the management to base their schedules on analyses of real data from their centre.

References

[1] Z. Aksin, M. Armony, and V. Mehrotra. The modern call center: A multi-disciplinary perspective on operations management research. *Production and Operations Management*, 16(6):665–688, 2007.

[2] J. Atlason, M. A. Epelman, and S. G. Henderson. Call center staffing with simulation and cutting plane methods. *Annals of Operations Research*, 127(1-4):333–358, 2004.

[3] A. N. Avramidis, W. Chan, M. Gendreau, P. L'ecuyer, and O. Pisacane. Optimizing daily agent scheduling in a multiskill call center. *European Journal of Operational Research*, 200(3):822–832, 2010.

[4] S. Bhulai, G. Koole, and A. Pot. Simple methods for shift scheduling in multiskill call centers. *Manufacturing & Service Operations Management*, 10(3):411–420, 2008.

[5] C. Centers. Staff scheduling for inbound call centers and customer contact centers. 2002.

[6] EPSRC/ESRC. Review of research status of operational research in the uk. *European Journal of Operational Research*, 125(2):359–369, 2004.

[7] A. T. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier. Staff scheduling and rostering: A review of applications, methods and models. *European journal of operational research*, 153(1):3–27, 2004.

[8] A. Fukunaga, E. Hamilton, J. Fama, D. Andre, O. Matan, and I. Nourbakhsh. Staff scheduling for inbound call and customer contact centers. *AI Magazine*, 23(4):30, 2002.

[9] N. Gans, G. Koole, and A. Mandelbaum. Telephone call centers: Tutorial, review, and research prospects. *Manufacturing & Service Operations Management*, 5(2):79–141, 2003.

[10] N. Gans, H. Shen, Y. Zhou, K. Korolev, A. McCord, and H. Ristock. Parametric stochastic programming models for call-center workforce scheduling. Technical report, working paper, 2009.

[11] M. R. Garey and D. S. Johnson. Computers and intractability. 1979. *F reeman, San Francisco*, 1979.

[12] S. Henderson, A. Mason, I. Ziedins, and R. Thomson. A heuristic for determining efficient staffing requirements for call centres. Technical report, Citeseer, 1999.

[13] A. Ingolfsson, E. Cabral, and X. Wu. Combining integer programming and the randomization method to schedule employees. *University of Alberta Research Report 02*, 1, 2002.

[14] R. M. Saltzman. A hybrid approach to minimize the cost of staffing a call center. *International Journal of Operations and Quantitative Management*, 11(1):1, 2005.

[15] N. Slack, S. Chambers, and R. Johnston. *Operations management*. Pearson Education, 2009.