Dimensioning an inbound call center using constraint programming

Cyril Canon[1,2], Jean-Charles Billaut[2], and Jean-Louis Bouquard[2]

[1] Vitalicom, 643 avenue du grain d'or, 41350 Vineuil, France
ccanon@fr.snt.com
[2] Université François-Rabelais de Tours, Laboratoire d'Informatique, 64 avenue Jean Portalis, 37200 Tours, France
{jean.billaut,jean-louis.bouquard}@univ-tours.fr

Abstract. One of the critical problems in the call center industries is the staffing problem since they must face variable demands and because staff costs represent a major part of the costs of this industry. The problem of dimensioning a call center is modeled as a particular scheduling problem, where the objective is to minimize the weighted number of resources needed to perform the jobs. We present in this paper a first constraint programming approach and some improvements, to solve this problem. We show that the results are very sensitive to the constraints definition and to the instanciation methods. Models are tested and results are discussed.

1 Introduction

Call Centers are used by organizations as an important channel of communication and transaction with their customers. The most prevalent form of communication is the telephone. When only the telephone is used, we call the company a *call center*. For this type of industry, personnel costs and specially the cost of staffing, account for around 70% of the cost of running a typical call center. It is essential to efficiently manage telephone call centers, so that the customer requests are met without excess staffing. One of the questions to answer is *How many agents are to be staffed in order to provide the required service quality?* and the related problem is called *the staffing problem* in the following. In this paper, we give an answer to this question.

From a modeling point of view, a call center can be viewed as a large system, operating in a stochastic environment and is generally modeled as a $M/M/N$ system, also called the Erlang-C model, "the most prevalent model that supports call center staffing". In [Koole and Mandelbaum, 2001],[Rottembourg, 2003], the authors present a survey of the state-of-the-art about possible models of a call center. They present the well known Erlang-C formulas and model a call center as a Markov Chain.

In this paper, we propose a deterministic model of the problem and we propose heuristic algorithms to solve it. To the best of our knowledge, no deterministic approach has been designed to deal with the staffing problem before.

The paper is organized as follows. In Section 2 the staffing problem is modeled as a scheduling problem and notations are introduced. In Section 3 we present an upper bound and a lower bound to our problem. In Section 4 we present a classical constraint programming formulation of the problem. In section 5 we propose some improvements of the classical model. In section 6 we detail the computational experiments and we discuss the results.

2 Problem modeling and notations

We consider an inbound call center, it means a call center that only receives calls. The outgoing calls are not considered, the staff requirement is supposed to be imposed by the client.

The staff dimensioning problem can be set as follows: the whole horizon is split into T periods of size τ. Generally, each period corresponds to a quarter of hour. For each period t, we know the number of expected incoming calls. These calls are of several types, depending on the client and on the activity.

The staffing levels are generally determined from a service level perspective. Indeed, when a client entrusts an activity to a call center, he provides the foreseen calls for each period and the call center has to reach an objective in terms of *quality of service*. In this paper, the quality of service is equal to the ratio of the average number of incoming calls taken in less than a given time over the horizon. It depends on the activity. For instance, one client imposes a mean quality of service greater than or equal to 85% of incoming calls taken in less than 20 seconds. This quality of service is a constraint for the staff dimensioning problem.

The problem consists in determining the minimum number of required agents at each period, respecting the quality of service of each activity.

We consider one given period t. In our model we assume that the calls of each type are regularly distributed over the period. This assumption is generally verified in practice if the arrival rate of calls (number of calls divided by the period length) is important. So, we assume that each call arrives at a determinated time and has to be answered according to the quality of service definition. It means that whatever the percentage of average number of calls to take is, we search for a solution such that all the calls are taken in the given time. The solution that we will obtain is a priori an over estimation of the minimum staff level.

Each agent j is considered as a resource denoted by M_j, $1 \leq j \leq m$. All the agents constitute a "parallel machine" environment, where all the resources are assumed to be identical. We associate to each resource a set of tools, where each tool corresponds to the skill which possesses the agent, i.e. to the type of calls the agent can take. A cost-in-use is associated to each resource M_j, depending on the number of associated tools. Each call is considered as an independent job i with a release date r_i, a processing time p_i and a deadline $\tilde{d}_i$, $1 \leq i \leq n$. N_k denotes the number of jobs corresponding to calls of type k. Finally, a tool is associated to each job, representing the skill required to perform the job. We denote by $EMSR$ the set of mono-skill resources.

The aim is to assign to the resources all the jobs and to schedule them in order to minimize the total cost.

We assume that the maximum slack time is smaller than the smaller average processing time. This hypotheses is realistic in call center contexts, it means that the waiting times allowed by the quality of service definition (around 20 seconds) is always smaller than the duration of the calls (at least one minute).

Proposition: Considering the previous hypotheses between duration of calls and waiting time, for all i and i' two jobs assigned to the same resource, if $r_i < r_{i'}$ then i' cannot precede i in a feasible solution.

Proof: can be easily demonstrated.

So, once the assignment of jobs to resources is known, the jobs are ordered by the SRT rule (Shortest Ready Time first). The problem is then reduced to an assignment problem. According to the three-field notation [Graham et al., 1979] of scheduling problems, our problem is denoted by $PMPM|r_i, \tilde{d}_i, s_{max} < p_{min}|m^w$ where m^w is the cost function due to machine use, s_{max} is the greatest slack time and p_{min} the smallest processing time.

3 Upper and lower bounds

We describe here an upper bound and a lower bound for our problem.

The upper bound is the result of a simple list algorithm: sort the jobs using SRT rule and assign the jobs to the first available machine (FAM), that is able to process it before its deadline. This algorithm as an $O(n \log(n))$ time complexity. We denote by UB the value of the solution returned by this heuristic algorithm and we denote by LA this algorithm in the following.

In [Simons, 1983], the author propose a simple algorithm to solve in polynomial time the problem $P|r_i, \tilde{d}_i, p_i = p|-$. The algorithm consists in sorting the jobs according to the SRT rule. Then it assigns resources to the jobs according to the FAM rule. If a job cannot be placed because of its deadline and if no job before has a greater deadline, then the problem is unfeasible.

Proposition: Considering that there is only mono-skill resources, LA algorithm is optimal.

Proof: The problem we solve here is the $P|r_i, \tilde{d}_i, p_i = p, r_i \geq r_{i'} \Rightarrow \tilde{d}_i \geq \tilde{d}_{i'}|m$. For this problem, the algorithm of Simons is equivalent to apply FAM assignment rule to a SRT list. Let m^* be the solution returned by LA. We prove that m^* is also a lower bound. Assume that $m = m^* - 1$ resources are sufficient to process all the jobs. If we apply the algorithm of [Simons, 1983] with exactly m resources, at least one job i cannot be assigned to a resource. But because all the jobs placed before i have a deadline lower than $\tilde{d}_i$, the problem is unfeasible. So m^* is a also lower bound and thus is the optimal value. ■

The lower bound is based on the notion of "mandatory part" [Lopez, 1992], [Le Pape et al., 2001]. Let consider one job i. This job has to be processed between r_i and d_i, and more precisely, whatever is its starting time, the job will be performed in the interval $[d_i - p_i, r_i + p_i]$, if $d_i - p_i < r_i + p_i$. We denote by $\mathcal{T}$ the set of dates $d_i - p_i$ for all jobs i. We denote by $\mathcal{S}_t$ the set of jobs that are processed simultaneously at time t, $\forall t \in \mathcal{T}$. The number $\max_{t \in \mathcal{T}} |\mathcal{S}_t|$ gives the minimum number of jobs that will be performed simultaneously. We assume that these jobs are assigned to the first cheaper resources, which gives a lower bound denoted by LB.

4 Classical constraint programming formulation

We denote by EJM_i the set of resources which can handle job i according to the tools associated to the resources.

The variables of the model are: for each job i, its starting time T_i and its assignment R_i. For each resource, Z_j is equal to 1 if machine M_j is used and 0 otherwise. The variable to minimize is denoted by C.

The constraints are: definition domains of variables T_i and R_i, the disjunctive constraints that indicate that if i and i' are assigned to the same resource and $i < i'$ then i precedes i' and constraints that give their values to variables Z_j.

Instanciation

To avoid the test of all equivalent configurations, weigths of resources have been modified. Resources having the same set of tools belong to one class. Then for all the resources of a class, a hierarchy is introduced in order to give a priority for their use. A small value is added to their cost according to the priority that is introduced. For example, if three resources M_1, M_2 and M_3 are of the same type, with a cost equal to 1500, then the first resource cost is set to 1501, the second to 1502 and the third to 1503.

Resource variables R_i are first instanciated. When the performing resource is known for job i, the starting time T_i is fixed to the earliest starting time, it means the first possible value in the definition domain of T_i. Finally, the Z_k variables are fixed to 0 if they are not fixed to 1 (due to constraints).

5 Improvements of the formulation

5.1 Model improvement

The first improvement consists in adding constraints on the objective function value, by using a lower bound and an upper bound, adding a global constraint *alldifferent* on the assignments of jobs and by modifying the disjunctive constraints. A constraint of type *alldifferent* is added. These constraints insure that

the assignment of all the jobs that belong to a same set $\mathcal{S}_t$ will not have the same assignment.

$$alldifferent(R_i, \forall i \in \mathcal{S}_t), \forall t \in \mathcal{T}$$

Constraints that imposes that if a resource M_j is not used ($Z_j = 0$), then so do the resources of the same class that have a higher cost ($M_{j'}$ with $j' > j$) are added.

At last, a new variable is introduced: X_i which is equal to 1 if job i is assigned to a mono-skill resource and 0 otherwise.

5.2 Instanciation improvement

The variables are instanciated according to two different ways.

1. all the variables Z_j are sorted according to their increasing cost of use, and set to 1 first ("cheaper machines first"); then all the couples of variables (R_i, T_i).
2. all the variables Z_j are sorted according to their decreasing cost of use, and set to 0 first ("expensive machines last"); then all the couples of variables (R_i, T_i).

In the case of model that contain X_i variables, all the variables X_i are instanciated, first to one and then to zero. If $X_i = 1$, only the first possible resource is choosen (according to algorithm LA).

6 results

The classical model never finds an optimal solution, but always finds a feasible one, even for an important number of jobs ($N = 30$ means instances with a number of jobs comprised between 120 and 240).

With the model that improves th instanciation but doesn't use the X_i varaibles, whatever the instanciation method, the problem is always solved optimaly within 2 minutes for $N = 5$, i.e. with a number of jobs comprised between 20 and 40. The results of the two first instanciations are similar: the result of the heuristic algorithm is improved for six instances with $N = 10$, and the results are not improved after. Note that the improvement is not significant for $N = 10$ in the second instanciation method. The third instanciation seems to be the best of the three: it allows to solve optimaly 50% of the instances for $N = 10$, i.e. a number of jobs comprised between 40 and 80. For $N = 15$, eigth instances are improved within two minutes with this instanciation method. But for $N \geq 20$, the model is not very efficient because it doesn't improve the upper bound in less than two minutes.

for the model that use X_i variables, we can notice that the first instanciation method does not give better results with this model than with previous ones. With the second instanciation method, the model returns more feasible solutions than all the other models. However, in all the cases, the average deviation from the upper bound is very small.

7 Conclusion

We consider the problem of dimensionning an inbound call center and we model this problem as a scheduling problem. We present different constraint programs to solve this multi-purpose machine scheduling problem, with the objective to minimize the weighted number of resources needed. A basic model is first presented. Then some improvements are proposed and tested on randomly generated instances. An upper bound based on a simple list algorithm and an lower bound based on the notion of mandatory parts are also presented and are used to improve the CP resolution. Finally, some instances of the problem are solved optimaly within two minutes for less than 80 jobs. For more than 100 jobs, the upper bound is never improved within two minutes by the best model.

In the near future, we will develop more sophisticated heuristic algorithms to solve this problem. The first will be based on Ant Colony Optimization, the second one will be a local search with the use of CP: a resource is replaced by another one (with a lower price) and the feasibility problem is solved using CP.

References

[Le Pape et al., 2001] P. Baptiste, C. Le Pape, W. Nuijten (2001). "Constraint-based Scheduling: Applying Constraints to Scheduling Problems." Kluwer Academic Publishers, Dordrecht.

[Brucker, 2001] P. Brucker (2001). Scheduling algorithms. Third edition, Springer-Verlag, Berlin.

[Call Center Statistics] Website, http://www.callcenternews.com/resources/statistics.shtml

[Canon et al., 2004] C. Canon, J-C. Billaut, J-L. Bouquard, J. Olivier, E. Néron (2004). "Dimensionnement d'un centre d'appels et validation par la simulation (in french)", Actes de la 5e conférence francophone de Modélisation et de Simulation (MOSIM'04), S. Dauzère-Pérès et A. Dolgui (Eds.) SCS Publishing House, 2004, pp. 431-437, Nantes, septembre 2004.

[Graham et al., 1979] R.L. Graham, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan (1979). "Optimisation and approximation in deterministic sequencing and scheduling: a survey." Annals of Discrete Mathematics. **5** (1979) 237-287.

[Koole and Mandelbaum, 2001] G. Koole and A. Mandelbaum (2002). "Queueing models of call centers: an introduction." Annals of Operations Research. **113**:41-59.

[Lopez, 1992] P. Lopez, J. Erschler et P. Esquirol (1992). "Ordonnancement de tâches sous contraintes : une approche énergétique" (in french). RAIRO APII volume 26, numro 6, 453-481.

[Rottembourg, 2003] B. Rottembourg. "Dimensionnement et planification de centres d'appels tlphoniques : problmatiques et solutions d'optimisation." MOSIM'03, Toulouse, April 2003.

[Simons, 1983] B. Simons (1983). "Multiprocessor scheduling of unit-time jobs with arbitrary release times and deadlines". SIAM journal of Computing, **12-2**:294-299.