Solving a Stochastic Queueing Design and Control Problem with Constraint Programming

Daria Terekhov and **J. Christopher Beck**
Department of Mechanical & Industrial Engineering
University of Toronto, Toronto, Ontario, Canada
{dterekho,jcb}@mie.utoronto.ca

Kenneth N. Brown
Cork Constraint Computation Centre
Department of Computer Science
University College Cork, Cork, Ireland
k.brown@cs.ucc.ie

Abstract

A facility with front room and back room operations has the option of hiring specialized or, more expensive, cross-trained workers. Assuming stochastic customer arrival and service times, we seek a smallest-cost combination of cross-trained and specialized workers satisfying constraints on the expected customer waiting time and expected number of workers in the back room. A constraint programming approach using logic-based Benders' decomposition is presented. Experimental results demonstrate the strong performance of this approach across a wide variety of problem parameters. This paper provides one of the first links between queueing optimization problems and constraint programming.

Introduction

Scheduling and resource allocation problems have long been studied in artificial intelligence and constraint programming (CP) (Fox 1983). In the real world, these problems are often subject to uncertainty and change: it is not known in advance what orders will arrive or what machines will need repair. Hence, there has been recent work in scheduling under uncertainty and a more general interest in applying CP to stochastic optimization (Brown & Miguel 2006). One area that has intensively studied the design and control of systems for resource allocation under uncertainty is queueing theory (Gross & Harris 1998). Therefore, we are beginning an investigation of the links between queueing theory and CP in order to determine whether the techniques of one can enhance the other, and to extend the range and richness of problems that can be addressed by CP.

In this paper, we generalize an existing queueing design and control problem (Berman, Wang, & Sapna 2005) and propose a complete hybrid solution technique combining logic-based Benders' decomposition, an extension of an existing heuristic, and CP. This paper has two main contributions. Firstly, we provide the first complete method for a generalization of a queueing control problem from the literature. Secondly, this paper is initial work towards a link between constraint programming and queueing theory, demonstrating the applicability of CP and logic-based Benders' decomposition for solving queueing optimization problems.

Problem Description

We consider a facility, such as a bank, with back room and front room operations. In the front room, work depends on a stochastic process of customer arrivals and service times. If all front room workers are busy, customers form a queue and wait to be served. In the back room, tasks include sorting of material and paperwork, and do not directly depend on customer arrivals. The facility can hire either cross-trained workers, able to perform tasks in both rooms, or specialized workers, able only to serve customers or only to work in the back room. Cross-trained workers provide flexibility since they may be switched between the back room and the front room depending on demand. However, cross-trained workers are more expensive than specialized ones because they possess more skills. Managers of the facility are therefore interested in finding the optimal number of workers of each type and, if this combination includes cross-trained workers, in knowing when to switch them between the two rooms.

Using the notation of (Berman, Wang, & Sapna 2005), let S denote the maximum number of customers allowed in the front room at any time. When there are S customers present, arriving customers are blocked from entering. In order to complete all the back room work, there is a known minimum requirement, B_l, for the expected number of workers in the back room. W_u denotes the upper bound on the expected customer waiting time, W_q. It is assumed that only one worker is allowed to be switched at a time, and both switching time and cost are negligible. Customers arrive according to a Poisson process with rate λ. Customer service times follow an exponential distribution with rate μ.

Berman et al. study two related problems with only cross-trained workers. The objective of problem P_1 is to determine when to switch workers between the two rooms so that expected customer waiting time is minimized, but the requirement on the minimum number of back room workers is met. In problem P_2, the goal is to find the minimum number of cross-trained workers such that a switching policy exists to meet constraints on the expected customer waiting time and on the expected number of back room workers.

We extend P_2 to allow specialized front and back room workers in addition to cross-trained ones. Given a different staffing cost for each type of worker, the goal of this problem is to find the lowest-cost combination of specialized and cross-trained workers so as to ensure that the expected wait-

ing time of customers does not exceed W_u and that there are enough workers in the back room to ensure that all back room work is completed.

Let f be the number of specialized front room workers, b the number of specialized back room workers, and x the number of cross-trained workers in the facility. It is assumed that the service rate of a worker in the front room is equal to μ regardless of whether the worker is specialized or cross-trained. Similarly, cross-trained and specialized employees working in the back room are assumed to be able to perform back room tasks equally well. Denote the staffing costs as c_f, c_b and c_x, respectively for front room, back room and cross-trained workers. We assume that $c_x \geq c_f > 0$, $c_x \geq c_b > 0$ and $c_x \leq c_f + c_b$.

Given particular values of x, f and b, and a policy specifying when cross-trained workers are to be switched between the two rooms, one can calculate the expected customer waiting time, W_q, and the expected number of workers in the back room, B. (See below for the definition of a switching policy and expressions for the calculation of B and W_q). Since we are required to find both the optimal staff mix and a satisfying switching policy, this problem is both a queueing design and a queueing control problem.

The problem of finding the optimal staff mix given front room and back room constraints can be stated as

$$\begin{aligned} &minimize\ c_f f + c_b b + c_x x \\ &s.t.\ W_q \leq W_u, B \geq B_l. \end{aligned} \tag{1}$$

Related Work

The problem addressed in this paper belongs to the class of problems usually referred to as "optimal design and control of queues," which has applications in many areas such as communication systems and scheduling (Tadj & Choudhury 2005). It is also a problem which deals with the management of cross-trained workers. Various problems involving this subject have been considered in the literature using simulation (Chevalier & Tabordon 2003), and linear and mixed-integer programming (Cezik & L'Ecuyer 2007; Batta, Berman, & Wang 2007).

As noted, our paper is closely linked with the work presented in (Berman, Wang, & Sapna 2005). In addition, (Terekhov & Beck 2007) apply constraint programming to the P_1 queue control problem. To our knowledge, no papers have examined applying CP to a problem which deals with both optimal queue design and control.

Benders' Decomposition

Benders' Decomposition was originally developed for solving mixed-integer programming problems (Benders 1962). More generally, the method can be applied to any problem in which the variables and constraints can be separated into a master problem and a sub-problem, which are solved in an alternating fashion until an optimality criterion is satisfied. The master problem and the sub-problem may be modelled and solved using linear or integer programming, or CP, as in logic-based Benders' decomposition (Hooker & Ottosson 2003; Tarim & Miguel 2005).

It is natural to decompose our problem into the master problem of finding a combination of cross-trained and specialized workers (a queueing design problem) and the sub-problem of finding a switching policy that satisfies the back room and front room service level constraints given an employee configuration (a queueing control problem).

Our approach is, first, to derive a set of constraints on the values of x, f and b by solving problem (1) with $x = 0$. These constraints are then used to form the master problem, which is solved in order to identify a cost-optimal, but possibly infeasible, combination of cross-trained and specialized workers, say $x = x'$, $f = f'$ and $b = b'$. If a feasible sub-problem solution can be found, then the master solution is optimal. Otherwise, the cut $(x > x' \ || \ f > f' \ || \ b > b')$ is added to the master problem. The master problem and the sub-problem are then re-solved in order to determine if a feasible policy exists for the new master problem solution.

The "Specialized-Only" Solution

When $x = 0$, the number of workers required for the back room is independent of the number of workers required for the front room. Thus, to find the minimum-cost specialized-only solution, one can independently determine F_{total}, the smallest number of specialized front room workers sufficient to satisfy the waiting time constraint, and B_{total}, the smallest number of specialized back room workers needed to satisfy the back room constraint.

In order to find F_{total}, W_q is calculated for each value of $f \geq 1$ using the equation

$$W_q = \frac{P_0 \sum_{i=1}^{i=S} (\frac{\lambda}{\mu})^{i-1} \frac{i}{D(i)}}{\mu[1 - (\frac{\lambda}{\mu})^S P_0 \frac{1}{D(S)}]} - \frac{1}{\mu},$$

where $D(i) = \prod_{j=1}^{i} d(j)$ with $d(i) = i$ for $i \leq f$, and $d(i) = f$ for $i > f$, and $P_0 = 1 + \sum_{i=1}^{i=S} (\frac{\lambda}{\mu})^i \frac{1}{D(i)}$, until the constraint $W_q \leq W_u$ is satisfied for some value of f. This equation is a re-formulation of the expression for W_q in an M/M/f/S queue (Gross & Harris 1998). B_{total} is simply $\lceil B_l \rceil$ because, when there are no cross-trained workers, this is the smallest possible number of back room workers required to complete all of the back room work.

By definition of F_{total} and B_{total}, there have to be at least F_{total} cross-trained and front room workers in the facility in order for the constraint on W_q to be satisfied, and at least B_{total} cross-trained and back room workers in order for the back room constraint to be satisfied. Thus, constraints $f + x \geq F_{total}$ and $b + x \geq B_{total}$ are valid for problem (1). In any feasible solution, the number of specialized front room workers does not ever have to exceed F_{total}, and the number of specialized back room workers does not ever have to exceed B_{total} because these values already satisfy the constraints in their respective rooms, and having more workers would only incur additional cost. Therefore, $F_{total} - 1$ is an upper bound for f, $B_{total} - 1$ is an upper bound for b, and $F_{total} + B_{total} - 1$ is an upper bound for x.

Additionally, since one needs at least F_{total} workers in the facility in order to satisfy the front room constraint, and at least B_{total} workers in the facility in order to satisfy the back

room constraint, it is clear that at least $\max(F_{total}, B_{total})$ workers need to be hired. The lower bound on x is 1, since the best solution with $x = 0$ has already been determined.

The cost of the specialized-only solution is an upper bound on the cost of the optimal solution. The lower bound on the optimal cost is the maximum of $c_f F_{total}$ and $c_b B_{total}$.

Master Problem

Given the constraints derived from the specialized-only solution, the master problem can be stated as

$$\begin{gathered} minimize\ cost = c_f f + c_b b + c_x x \\ s.t.\ f + x \geq F_{total}, b + x \geq B_{total} \\ 0 \leq f \leq F_{total} - 1, 0 \leq b \leq B_{total} - 1 \\ 1 \leq x \leq F_{total} + B_{total} - 1 \\ f + b + x \geq max(F_{total}, B_{total}) \\ max(c_f F_{total}, c_b B_{total}) \leq cost \leq c_f F_{total} + c_b B_{total} \\ cuts \end{gathered} \tag{2}$$

where $cuts$ are constraints that are added to the master problem each time the sub-problem is not able to find a feasible solution. These cuts remove the current optimal solution of the master problem because it does not result in a feasible policy. The master problem is re-solved each time a new cut is added, and the resulting values of f, b and x define the sub-problem. We solve the master problem with a CP model identical to (2). As it is a simple minimization problem with three integer variables, finding a solution is trivial.

Solving the Sub-Problem

Given values for f, b and x, the goal of the sub-problem is to find a policy K such that the constraints $W_q \leq W_u$ and $B \geq B_l$ are satisfied. The method we use for solving the sub-problem is a modification of the *PSums*-Heuristic Hybrid method proposed in (Terekhov & Beck 2007), which combines a CP model with Berman et al.'s heuristic.

Switching Policy A policy is a sequence of "switching points," k_i for $i = 0, 1, \ldots, x + f$, with the interpretation that the number of "busy" workers (ones who are currently serving a customer) in the front room is i whenever the number of customers in the front room is between $k_{i-1} + 1$ and k_i. Switching points with index $i < f$ state that as another customer arrives to the front room, the number of specialized workers who are busy increases; switching points with index $i \geq f$ specify when to switch cross-trained workers between the front room and the back room. Switching point k_{x+f} is a constant and equal to S. Thus, a policy is vector of k_is, $(k_0, k_1, \ldots, k_{x+f})$, such that $k_i < k_{i+1}$ for $i = 0, 1, \ldots, x + f - 1$, $k_{x+f} = S$, $k_i = i\ \forall i < f$ and $k_i \geq i\ \forall i \geq f$.

For example, if $f = 2$, $x = 1$ and $S = 6$, the policy $(0, 1, 3, 6)$ states that whenever the number of customers is 1, there is 1 busy worker in the front room, whenever the number of customers is 2 or 3, there are 2 busy workers in the front room. When the number of customers becomes $k_2 + 1 = 4$, a cross-trained worker is switched to the front room. Other policy definitions have been considered. However, these either result in more complicated models or were less efficient in preliminary experiments.

Modified Berman et al.'s Heuristic In order to solve the sub-problem, we first run a modified version of the heuristic Berman et al. use for problem P_1. This modified heuristic starts with the policy $\hat{K} = \{k_0 = 0, k_1 = 1, ..., k_{f-1} = f - 1, k_f = S - x, k_{f+1} = S - x + 1, ..., k_{x+f-1} = S - 1, k_{x+f} = S\}$, which yields the greatest possible values of W_q and B. If this policy does not satisfy the constraint $B \geq B_l$ (is B-infeasible), then the sub-problem is infeasible. If the policy is B-feasible, then the switching point k_i with the smallest index i satisfying the condition $k_i - k_{i-1} > 1$ for $0 < i < x + f$, or $k_i > 0$ for $i = 0$, is decreased by 1. This results in a policy with both a smaller W_q and a smaller B. The heuristic continues decreasing switching points satisfying this property until the resulting policy becomes B-infeasible (or is the policy $\hat{K} = \{k_0 = 0, k_1 = 1, k_2 = 2, ..., k_f = f, ..., k_{x+f-1} = x + f - 1, k_{x+f} = S\}$, in which case the heuristic stops because this policy is optimal). Once infeasibility is reached, a switching point k_i having the smallest index and satisfying the condition $k_{i+1} - k_i > 1$, for $i < x + f$, is increased by 1. Increasing a switching point with such properties allows the policy to become more feasible in terms of the back room constraint, but also increases W_q. Once a B-feasible policy is found again, the heuristic tries to find switching points to decrease. Thus, the heuristic alternates between trying to reach a policy with smaller W_q and a policy with higher B. Each time a B-infeasible policy is found, the set of switching points that can be increased or decreased at subsequent steps is reduced in order to prevent cycling. The heuristic stops when it is unable to find any more switching points to decrease or increase, in which case it returns the B-feasible policy with the best value of W_q that it has been able to find. If this W_q value is smaller than W_u (the policy is W_q-feasible), then the current sub-problem and master solutions are optimal. Otherwise, the CP method based on the *PSumsSubproblem* model and shaving is applied.

The *PSumsSubproblem* Model The *PSumsSubproblem* model has, as its decision variables, the set of k_i, $i = 0, 1, \ldots, x + f$, each with an initial domain of $[0..S]$. Additionally, it has two types of probability variables: $PSums(k_i)$ for $i = 0, ..., x + f - 1$, and $P(j)$ for $j = k_0, k_1, k_2, ..., k_{x+f}$ (Terekhov & Beck 2007), which are necessary for the calculation of W_q and B given a switching policy. We relate these three types of variables via a set of constraints which ensure that the probability values satisfy the steady-state conditions of the front room queue. We also include constraints for expressing W_q and B in terms of the variables k_i, $PSums(k_i)$ and $P(k_i)$.

In order to simplify the presentation of equations necessary for the definition of the probability variables and quantities of interest, let $r_i = \frac{\lambda}{i\mu}$, $i = 1, \ldots, x + f + 1$, and $\delta_i = k_{i+1} - k_i$, $i = 0, \ldots, x + f$. $PSums(k_i) = \phi_i P(k_i)$ represents the probability of there being between k_i and

$k_{i+1}-1$ customers in the front room, where ϕ_i is defined in Equation (3). Equation (4) is a recursive formula for computing $P(k_{i+1})$, the probability of having k_{i+1} customers in the facility, where $P(k_0) = (\sum_{i=0}^{x+f} \beta Sum(k_i))^{-1}$ and $\beta Sum(k_i)$ is defined using Equations (5)–(6).

$$\phi_i = \begin{cases} \frac{1-[r_{i+1}]^{\delta_i}}{1-r_{i+1}} & \text{if } r_{i+1} \neq 1 \\ \delta_i & otherwise \end{cases} \quad (3)$$

$$P(k_{i+1}) = [r_{i+1}]^{\delta_i} P(k_i) \quad (4)$$

$$\beta Sum(k_i) = \phi_{i-1} X_i \left[\frac{\lambda}{\mu}\right]^{k_{i-1}-k_0+1} \frac{1}{i} \quad (5)$$

$$X_i = \prod_{g=1}^{i-1} \left(\frac{1}{g}\right)^{k_g - k_{g-1}} \quad (6)$$
$$(X_1 \equiv 1), i = 1, \ldots, x+f$$

The expected total number of cross-trained workers in the front room is $F_{cross} = \sum_{i=1}^{x} i[PSums(k_{i+f-1}) - P(k_{i+f-1}) + P(k_{i+f})]$. The expected number of workers in the back room, B, can therefore be defined as $b+x-F_{cross}$.

The expected number of customers in the front room, L, is defined as $\sum_{i=0}^{x+f-1} L(k_i) + k_{x+f} P(k_{x+f})$ where

$$L(k_i) = k_i PSums(k_i) + P(k_i) \quad (7)$$
$$\times \frac{[(r_{i+1})^{\delta_i}(-\delta_i) + (r_{i+1})^{\delta_i+1}(\delta_i - 1) + r_{i+1}]}{\left[\frac{(i+1)\mu-\lambda}{(i+1)\mu}\right]^2}.$$

$W_q = \frac{L}{\lambda(1-P(k_{x+f}))} - \frac{1}{\mu}$ is the expected customer waiting time.

The equations defining $PSums(k_i)$, $P(k_i)$, L, B and W_q, together with $W_q \leq W_u$ and $B \geq B_l$ are the major constraints of the CP model. The set of constraints defining a policy (presented above) is also included in the model.

Shaving Shaving is a consistency-enforcing procedure for constraint programs based on temporarily adding constraints to the problem, performing propagation, and making inferences according to the resulting state of the problem (Demassey, Artigues, & Michelon 2005; van Dongen 2006). We use two shaving procedures: $B_l Shaving$, which makes inferences based on the feasibility of policies with respect to the B_l constraint, and $W_u Shaving$, which makes inferences based on feasibility with respect to the W_u constraint.

Let $\min(k_i)$ and $\max(k_i)$ be, respectively, the smallest and largest values in the current domain of variable k_i, and suppose that the constraint $W_q \leq W_u$ is temporarily removed. At each step of the $B_l Shaving$ procedure, $k_i = \min(k_i)$ or $k_i = \max(k_i)$ is temporarily added to the model for $i \in \{0, ..., x+f-1\}$. If $k_i = \min(k_i)$ is added, then all other switching points are assigned the maximum possible values subject to the condition that $k_n < k_{n+1}$, $\forall n \in \{0, ..., x+f-1\}$. If the resulting policy is B-infeasible, the temporary constraint is removed, and the constraint $k_i > \min(k_i)$ is permanently added: if all variables except k_i are set to their maximum values, and the problem is B-infeasible, then, by Theorem 1 of (Berman, Wang, & Sapna 2005), in any feasible policy k_i must be greater than $\min(k_i)$. Otherwise, we check if the policy is W_q-feasible, in which case the procedure stops since feasibility of the sub-problem has been proved.

If $k_i = \max(k_i)$ is added, all other switching points are assigned the minimum possible values. If the resulting policy is B-feasible but W_q-infeasible, the constraint $k_i < \max(k_i)$ is permanently added. Since all variables except k_i are at their minimum values already, and k_i is at its maximum, it must be true, again by Berman et al.'s Theorem 1, that in any solution with smaller W_q, the value of k_i has to be smaller than $\max(k_i)$. If the policy is both B-feasible and W_q-feasible, the procedure stops, and the current master and sub-problem solutions are optimal.

The $W_u Shaving$ procedure makes inferences based strictly on the constraint $W_q \leq W_u$. The constraint $B \geq B_l$ is removed prior to running this procedure. A constraint of the form $k_i = \max(k_i)$ is added, and the smallest possible values are assigned to the rest of the variables. As the B_l constraint has been removed, the only reason why the policy could be infeasible is because it has a W_q value greater than W_u. Since all switching points except k_i are assigned their smallest possible values, this implies that in any solution with a better expected waiting time, the value of k_i has to be strictly smaller than $\max(k_i)$.

$B_l Shaving$ and $W_u Shaving$ are iteratively run until a policy satisfying both constraints is found, until a constraint is inferred that violates the current upper or lower bound of a k_i, or until no further inferences can be made. In the final case, standard CP search is performed to determine whether a feasible policy exists.

Experimental Results and Analysis

Since our problem has not been previously solved, our experiments focus on determining some of the reasons for the CPU times required to solve various problem instances and on evaluating the effect of different parameter values on the efficiency of our method. Additionally, we evaluate the usefulness of the three main components of the method for solving the sub-problem. Our approach was implemented in ILOG Solver 6.2, and all experiments were performed on a Dual Core AMD 270 CPU with 1 MB cache, 4 GB of main memory, running Red Hat Enterprise Linux 4.

Problem Size Recall that S is the maximum number of customers allowed in the system at any one time and thus is the prime determinant of problem size. We use a set of 300 instances, 30 for each value of S from {10, 20, ..., 90, 100}, with costs $c_x = 32$, $c_f = 31$ and $c_b = 30$. The values of the rest of the parameters are taken from the experiments of (Terekhov & Beck 2007), for which they were generated in such a way as to ensure non-trivial solutions for Berman's problem P_1. The best W_q values found in those experiments were used as the W_u values for our instances. Since P_1 is similar to our sub-problem, we expected that using these parameters would give us instances of various difficulty.

Our method solved all but one instance (at S of 80) within

Statistic/Value of S	10	20	30	40	50	60	70	80	90	100
CPU Time (seconds)	0.07	0.81	2.72	0.30	1.12	0.52	0.30	87.54	4.64	5.74
# of Iterations	5.07	7.83	16.20	6.23	8.23	7.23	8.23	34.73	27.60	23.83
Total # of Workers	4.37	6.43	9.20	4.87	5.47	5.60	5.27	15.33	8.83	8.93
% Diff. Compared to Specialized-only	6.47	2.87	0.20	3.04	4.15	1.28	2.91	0.09	0	0
% Diff. Compared to Cross-trained-only	7.26	9.38	7.42	13.22	12.27	11.84	11.47	7.74	8.18	7.54

Table 1: Mean CPU time (seconds), mean number of iterations, mean total number of workers in the optimal solution, and mean percentage differences between the cost of the optimal solution and the two 'naive' solutions for each value of S. For the one instance (at S of 80) that was not solved within the time limit, we assume a run-time of 600 seconds, as many iterations as were completed in 600 seconds, and the specialized-only solution to be optimal.

10 CPU minutes. In Table 1, the mean CPU time, mean number of times the sub-problem is solved (number of iterations) and the mean total number of workers in the optimal solution over 30 instances for each value of S are presented. The mean run-times show a significant peak at $S = 80$ and are approximately correlated with both the total number of workers and the number of iterations. This is not surprising: the more workers are needed in the facility, the more staff combinations usually need to be examined. An increase in the total number of employees in the optimal solution also leads to higher run-times for each sub-problem. This is because the higher the total number of workers in the facility, the larger the size of the policies, and the longer shaving and search will take to prove feasibility or infeasibility. In our problem set, when $S = 80$, there are several instances which have both a large number of iterations (a maximum of 141) and difficult sub-problems (maximum sub-problem CPU time of approximately 77.6 seconds).

Table 1 shows that higher values of S do not necessarily lead to higher run-times: the mean CPU times at S of 90 and 100 are substantially lower than that at S of 80. Therefore, problem difficulty is not simply a result of the problem size.

Cost Combinations Using the above instances, five different cost combinations, (c_x, c_f, c_b), are examined. When the costs are (32, 1, 31), (32, 31, 1) and (32, 24, 24), all 300 instances are solved, with a mean run-time of under 1 second. For cost combinations (32, 30, 31) and (32, 32, 32), 298 and 288 instances, respectively, are solved, and the mean run-times are 8.07 and 35.10 seconds, respectively. These results indicate that as the difference between the cost of a cross-trained worker and the sum of the costs of specialized workers increases, the time needed to solve the instance increases. In these cases, the master problem is likely to consider solutions with a larger number of cross-trained workers, resulting in larger domains for the k_is and longer shaving and search times. The general pattern in the mean CPU times for each S seen in Table 1 also occurs under these five cost combinations.

Other Parameters In order to examine the effect of the rest of the problem parameters, we use a set of instances with S of 100 and the λ, μ, B_l and W_u values from the 54 instances used in (Berman, Wang, & Sapna 2005). Ten different cost combinations which satisfy our cost assumptions are used, giving us 540 instances. Although the values of the parameters vary, these instances can be grouped into 9 types according to the value of $\frac{\lambda W_u}{B_l}$, which is an indication of how difficult it will be to satisfy the W_u and B_l constraints simultaneously, adjusted by the arrival rate. Smaller values of this ratio lead to tighter problem instances. In addition, each triple out of these 9 types has an equal value of λ/μ, which is a representation of the workload of the facility.

All 540 instances were solved within 10 minutes. From Table 2, it can be seen that as λ/μ increases, mean CPU times increase. This happens simultaneously with an increase in the mean number of iterations and an increase in the total number of workers in the optimal solution, confirming the observations made from our experiment with different S values. As the value of $\frac{\lambda W_u}{B_l}$ decreases, while λ/μ is held constant, mean CPU times, number of iterations and total number of workers in the optimal solution increase. Therefore, as the workload of the facility increases and/or the expected waiting time bound becomes tighter, the problem becomes harder to solve.

Cost of the Optimal Solution Both Table 1 and 2 present the mean percentage difference between the cost of the optimal solution, and the costs of the specialized-only solution and the cross-trained-only solution (i.e. with $f = b = 0$). We calculate the difference between each pair of solutions as a percentage of the cost of the optimal solution. In most cases, there is a non-zero difference between the costs of the optimal solution and the two 'naive' solutions, indicating that our approach may be valuable in practical applications.

Influence of Algorithm Components The sub-problem solution technique has three main components: the heuristic, the alternating shaving procedure and CP search. We tested three modifications of our method: *heuristicOnly*, in which shaving and search are removed, *noShaving*, in which shaving is removed, and *noSearch*, in which search is removed.

Experiments with our set of 540 instances indicate that the *heuristicOnly* method finds the optimal solution only in the 215 instances for which the optimal solution is the specialized-only solution. In 52 additional instances, this method finds a better-quality, but sub-optimal, solution than the specialized-only one. The *noSearch* method is able to find the optimal-cost solution in 516 instances: 92.6% of instances for which the specialized-only solution is not optimal. The *noShaving* method finds the optimal-cost solution in 389 instances, 53.5% of instances where the specialized-

λ/μ	10			15			20		
$\frac{\lambda W_u}{B_l}$	55.83	21.87	6.15	38.34	10.84	3.55	28.56	8.12	2.37
CPU Time (seconds)	0.31	12.47	11.98	0.59	8.72	37.10	40.37	58.21	88.41
# of Iterations	1.00	3.20	11.30	1.00	5.70	22.58	3.20	14.10	36.50
Total # of Workers	11.00	11.60	12.70	16.00	17.00	18.82	21.60	22.80	25.10
% Diff. Compared to Specialized-only	5.27	1.42	0.75	3.69	3.41	1.03	0.75	0.34	1.44
% Diff. Compared to Cross-trained-only	61.46	55.22	46.85	67.92	59.35	48.36	68.01	60.26	47.61

Table 2: Mean CPU time (seconds), number of iterations, total number of workers in the optimal solution, and percentage differences between the cost of the optimal solution and the two 'naive' solutions for each value of $\frac{\lambda W_u}{B_l}$.

only solution is not optimal. Recall that the full algorithm found the optimal-cost staff mix in all 540 instances. The mean run-times of the *heuristicOnly*, the *noSearch* and the *noShaving* methods are approximately 0.07, 28.62 and 300.39 seconds, respectively, whereas the mean run-time of our overall approach is approximately 28.69 seconds.

These results indicate, firstly, that the shaving procedure is an essential part of our method: it finds the optimal solution in most of our test instances and significantly reduces the run-time. Secondly, we see that the heuristic is usually not helpful in finding the optimal solution. However, given its negligible run-time, it is still worthwhile to use as part of our overall method. CP search guarantees completeness and is useful in those cases in which shaving is unable to find a feasible policy for the optimal master solution.

Conclusions

In this paper, the applicability of CP for solving queueing design and control problems is demonstrated by showing that a Benders' decomposition-based method is able to solve an example of such a problem over a wide range of parameter settings. This work provides one of the first links between queueing theory and CP. In future work, we intend to further investigate the integration of these areas.

Acknowledgments

This work has received support from the Science Foundation Ireland under grant 00/PI.1/C075, the Natural Sciences and Engineering Research Council of Canada, Canada Foundation for Innovation, and ILOG, SA.

References

Batta, R.; Berman, O.; and Wang, Q. 2007. Balancing staffing and switching costs in a service center with flexible servers. *European Journal of Operations Research* 177(2):924–938.

Benders, J. F. 1962. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik* 4:238–252.

Berman, O.; Wang, J.; and Sapna, K. P. 2005. Optimal management of cross-trained workers in services with negligible switching costs. *European Journal of Operations Research* 167(2):349–369.

Brown, K. N., and Miguel, I. 2006. Uncertainty and change. In Rossi, F.; van Beek, P.; and Walsh, T., eds., *Handbook of Constraint Programming*. Elsevier. chapter 21, 731–760.

Cezik, T., and L'Ecuyer, P. 2007. Staffing multiskill call centers via linear programming and simulation. *Management Science*. To Appear. Available from http://www.iro.umontreal.ca/~lecuyer/myftp/papers/sbrms.pdf.

Chevalier, P., and Tabordon, N. 2003. Overflow analysis and cross-trained servers. *International Journal of Production Economics* 85:47–60.

Demassey, S.; Artigues, C.; and Michelon, P. 2005. Constraint-propagation-based cutting planes: An application to the resource-constrained project scheduling problem. *INFORMS Journal on Computing* 17(1):52–65.

Fox, M. S. 1983. *Constraint-Directed Search: A Case Study of Job-Shop Scheduling*. Ph.D. Dissertation, Carnegie Mellon University, Intelligent Systems Laboratory, The Robotics Institute, Pittsburgh, PA. CMU-RI-TR-85-7.

Gross, D., and Harris, C. 1998. *Fundamentals of Queueing Theory*. John Wiley & Sons, Inc.

Hooker, J. N., and Ottosson, G. 2003. Logic-based Benders' decomposition. *Mathematical Programming* 96:33–60.

Tadj, L., and Choudhury, G. 2005. Optimal design and control of queues. *Sociedad de Estadìstica e Investigación Operativa, Top* 13(2):359–412.

Tarim, S. A., and Miguel, A. 2005. A hybrid Benders' decomposition method for solving stochastic constraint programs with linear recourse. In *Joint ERCIM/CoLogNET International Workshop on Constraint Solving and Constraint Logic Programming*, 133–148.

Terekhov, D., and Beck, J. C. 2007. Solving a stochastic queueing control problem with constraint programming. In *Proceedings of the Fourth International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR'07)*, 303–317. Springer-Verlag.

van Dongen, M. R. C. 2006. Beyond singleton arc consistency. In *Proceedings of the Seventeenth European Conference on Artificial Intelligence (ECAI'06)*, 163–167.