Solving a Stochastic Queueing Control Problem with Constraint Programming

Daria Terekhov and J. Christopher Beck

Department of Mechanical and Industrial Engineering,
University of Toronto, Toronto, Ontario, Canada
{dterekho,jcb}@mie.utoronto.ca

Abstract. In a facility with front room and back room operations, it is useful to switch workers between the rooms in order to cope with changing customer demand. Assuming stochastic customer arrival and service times, we seek a policy for switching workers such that the expected customer waiting time is minimized while the expected back room staffing is sufficient to perform all work. Three novel constraint programming models and a shaving algorithm are presented. Experimental results show that the best constraint programming model, using shaving, is able to find and prove optimal solutions for almost all problem instances within a reasonable run-time, but that an existing heuristic algorithm performs better in terms of solution quality over time. A hybrid method combining the heuristic and the best constraint programming method is shown to perform better than either of these approaches separately. This is the first work of which we are aware that solves a queueing control problem with constraint programming.

1 Introduction

Many retail facilities, such as stores or banks, have back room and front room operations. In the front room, workers have to serve arriving customers, and customers form a queue and wait to be served when all workers are busy. In the back room, work is less time-sensitive, and may include such tasks as sorting or processing paperwork. All workers in the facility are cross-trained and are assumed to be able to perform back room tasks equally well and serve customers with the same service rate. Therefore, it makes sense for the managers of the facility to switch workers between the front room and the back room depending both on the number of customers in the front room and the amount of work that has to be performed in the back room. These managers are thus interested in finding a switching policy that minimizes the expected customer waiting time in the front room, subject to the constraint that the expected number of workers in the back room is sufficient to complete all required work. This queueing control problem has been studied in detail by Berman et al. [3], who propose a heuristic for solving it.

In this paper, a constraint programming (CP) approach is proposed for the problem. Thus, the contributions of this paper are twofold. Firstly, CP is, for the first time, used to solve a stochastic queueing control problem. Secondly, a complete approach for a problem for which only a heuristic algorithm existed previously is presented.

The paper is organized as follows. Section 2 presents a description of the problem and the work done by Berman et al. [3]. Section 3 presents three CP models for this problem. Section 4 describes a shaving procedure used for improving the efficiency of the CP models. In Section 5, the performance of the three CP models is compared and the best CP approach is contrasted with the Berman et al. heuristic. Based on these results, a hybrid method is proposed and evaluated in Section 6. In Section 7, a discussion of the results is presented. Section 8 describes related problems and states some directions for future work. Section 9 concludes the paper.

2 Problem Description

Let N denote the number of workers in the facility, and let S be the maximum number of customers allowed in the front room at any one time.[1] When there are S customers present, arriving customers will be blocked from joining the front room queue. Customers arrive according to a Poisson process with rate λ. Service times in the front room follow an exponential distribution with rate μ. The minimum expected number of workers that is required to be present in the back room in order to complete all of the necessary work is assumed to be known, and is denoted by B_l, where l stands for 'lower bound'. Only one worker is allowed to be switched at a time, and both switching time and switching cost are assumed to be negligible. The goal of the problem is to find an optimal approach to switching workers between the front room and the back room so as to minimize the expected customer waiting time, denoted W_q, while at the same time ensuring that the expected number of workers in the back room is at least B_l. Thus, a policy needs to be constructed that specifies how many workers should be in the front room and back room at a particular time, and when switches should occur.

2.1 Problem Definition

Berman et al. define a policy in terms of quantities k_i, for $i = 0, \ldots, N$. This policy states that there should be i workers in the front room whenever there are between $k_{i-1} + 1$ and k_i customers (inclusive) in the front room, for $i = 1, 2, \ldots, N$. As an illustration, consider the policy $(k_0, k_1, k_2, k_3) = (0, 2, 3, 6)$, with $N = 3$. This policy states that when there are $k_0 + 1 = 1$ or $k_1 = 2$ customers in the front room, there is one worker in the front room; when there are 3 customers, there are 2 workers; and when there are 4, 5, or 6 customers, all 3 workers are employed in the front. This definition of a policy forms the basis of the model proposed by Berman et al., with the switching points k_i, $i = 0, \ldots, N - 1$, being the decision variables of the problem, and k_N being fixed to S, the capacity of the system. In this paper, we follow Berman et al. and define an optimal policy as a set of values for the switching points, k_i, which minimize the expected waiting time subject to the back room constraint.[2]

In order to determine the expected waiting time and the expected number of workers in the back room given a policy defined by particular values of k_i, Berman et al. first

[1] The notation used by Berman et al. [3] is adopted throughout this paper.

[2] The term *optimal policy* is used in queueing control literature [6] to mean both the optimal *type* of policy and the optimal parameter values for a given policy type. In particular, for our problem, it is possible that an alternative type of policy (e.g., one that allowed randomization in the switching decision) may lead to a smaller expected waiting time.

define a set of probabilities, $P(j)$, for $j = k_0, k_0 + 1, \ldots, S$. Each $P(j)$ with $j > k_0$ denotes the steady-state (long-run) probability of there being exactly j customers in the facility. Since k_0 may not necessarily be 0 in a particular policy, $P(k_0)$ has a different interpretation – it is the probability of having between 0 and k_0 (inclusive) customers in the front room. Berman et al. define a set of balance equations for the determination of these probabilities:

$$P(j)\lambda = P(j+1)i\mu \quad j = k_{i-1}, k_{i-1}+1, \ldots, k_i - 1 \quad i = 1, ..., N. \tag{1}$$

An additional constraint on the values of $P(j)$ is $\sum_{j=k_0}^{S} P(j) = 1$.

All quantities of interest can be expressed in terms of the probabilities $P(j)$. Expected number of workers in the front room is

$$F = \sum_{i=1}^{N} \sum_{j=k_{i-1}+1}^{k_i} iP(j) \tag{2}$$

while the expected number of workers in the back room is $B = N - F$. The expected number of customers in the front room is

$$L = \sum_{j=k_0}^{S} jP(j). \tag{3}$$

Expected waiting time in the queue can be expressed as

$$W_q = \frac{L}{\lambda(1 - P(S))} - \frac{1}{\mu}. \tag{4}$$

This expression is derived using Little's Laws [6,8] for a system of capacity $k_N = S$.

Given a family of switching policies $\mathbf{K} = \{K; K = \{k_0, k_1, ..., k_{N-1}, S\}, k_i$ integer, $k_i - k_{i-1} \geq 1$, $k_0 \geq 0, k_{N-1} < S\}$, the problem can formally be stated as:

$$\begin{gathered} minimize\ W_q \\ s.t\ B \geq B_l \\ equations\ (1), (2), (3), (4). \end{gathered} \tag{5}$$

Note that B, F and L are expected values and can be real-valued. Consequently, the constraint $B \geq B_l$ states that the expected number of workers in the back room resulting from the realization of any policy should be greater than or equal to the minimum expected number of back room workers needed to complete all back room work. At any particular time point, there may, in fact, be fewer than B_l workers in the back room.

2.2 Berman et al.'s Heuristic

Berman et al. propose a heuristic method for the solution of this problem based on the following theorem, the details and proof of which are presented in [3].

Theorem 1 (Berman et al.). *Consider two policies K and K' which are equal in all but one k_i. In particular, suppose that the value of k'_J equals $k_J - 1$, for some J from the set $\{0, ..., N-1\}$ such that $k_J - k_{J-1} \geq 2$, while $k'_i = k_i$ for all $i \neq J$. Then (a) $W_q(K) \geq W_q(K')$, (b) $F(K) \leq F(K')$, (c) $B(K) \geq B(K')$.*

In addition, Berman's heuristic is based on two policies having special properties. Firstly, consider the policy $\hat{K} = \{k_0 = 0, k_1 = 1, \ldots, k_{N-1} = N-1, k_N = S\}$. This policy results in the largest possible F, and the smallest possible B and W_q. Because this policy yields the smallest possible expected waiting time, if it is feasible, then it is optimal. On the other hand, the smallest possible F and the largest possible W_q and B are obtained by applying the policy $\hat{\hat{K}} = \{k_0 = S-N, k_1 = S-N+1, \ldots, k_{N-1} = S-1, k_N = S\}$. Thus, if this policy is infeasible, the problem (5) is infeasible also.

Heuristic P_1 of Berman et al. starts with the policy $\hat{\hat{K}}$. If this policy is feasible, then the switching point k_i with the smallest index i satisfying the condition $k_i - k_{i-1} > 1$ for $0 < i < N$ and $k_i > 0$ for $i = 0$ is decreased by 1. By Theorem 1, this results in a policy with a better W_q value but smaller B. The heuristic continues decreasing switching points with this property until the resulting policy becomes infeasible (or is the policy $\hat{K}$, in which case P_1 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 $i < N$, is increased by 1. By Theorem 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 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 that is feasible with respect to the B_l constraint. Each time an 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. Assuming the problem is feasible, P_1 stops when it is unable to find any more switching points to decrease or increase, in which case it returns the best feasible policy that it has been able to find. The heuristic guarantees optimality only when the policy it returns is $\hat{K}$ or $\hat{\hat{K}}$.

Empirical results regarding the performance of heuristic P_1 are not presented in [3] and so the ability of P_1 to find good switching policies is not explicitly evaluated. In particular, it is not clear how close policies provided by P_1 are to the optimal policies.

3 Constraint Programming Models

Some work has been done on extending CP to stochastic problems [12,13,16]. Our problem is different from the problems addressed in these papers because all of the stochastic information can be explicitly encoded as constraints and expected values, and there is no need for either stochastic variables or scenarios. The major motivation for our work is to investigate whether CP can be successfully used to solve such problems. To this end, we investigate three CP models for our queue control problem:

– The *If-Then* model is a CP version of the formal definition of Berman et al.

- The *PSums* model uses some slightly different sets of variables, and some constraints are included which are based on closed-form expressions derived from the constraints that are used in the *If-Then* model.
- The *Dual* model includes a set of dual decision variables in addition to the variables used in the *If-Then* and *PSums* models. Most of the constraints of this model are expressed in terms of these dual variables.

Implementation of our models uses the predefined constraints available in standard CP solvers.

The proposed models have some similarities. Firstly, all of them have a set of decision variables k_i, $i = 0, 1, \ldots, N$, representing the switching policy. Each k_i with $i < N$ has the domain $[i, i+1, \ldots, S-N+i]$ (since $k_i < k_{i+1}$) and k_N is constrained to equal S. Secondly, a set of auxiliary variables is included in each model to represent the probabilities needed for the calculation of the quantities of interest. In addition, a constraint stating that $B \geq B_l$, a set of constraints $k_i < k_{i+1}$, for all i from 0 to $N-1$, (since the number of workers in the front room, i, increases only when the number of customers, k_i, increases) and constraint (6) are included in all three models. Constraint (6) ensures that an assignment of all decision variables leads to a unique solution of the balance equations.

$$P(k_0)\sum_{i=0}^{N}\beta Sum(k_i) = 1 \tag{6}$$

The calculation of $\sum_{i=0}^{N}\beta Sum(k_i)$ requires some auxiliary variables, which are defined in Equations (7) and (8). The derivation of these equations, based on expressions presented in [3], can be found in [14].

$$\beta Sum(k_i) = \begin{cases} X_i\left(\dfrac{\lambda}{\mu}\right)^{k_{i-1}-k_0+1}\left(\dfrac{1}{i}\right)\left[\dfrac{1-\left(\dfrac{\lambda}{i\mu}\right)^{k_i-k_{i-1}}}{1-\left(\dfrac{\lambda}{i\mu}\right)}\right] & \text{if } \frac{\lambda}{i\mu} \neq 1 \\ X_i\left(\dfrac{\lambda}{\mu}\right)^{k_{i-1}-k_0+1}\left(\dfrac{1}{i}\right)(k_i - k_{i-1}) & otherwise. \end{cases} \tag{7}$$

$$X_i = \prod_{g=1}^{i-1}\left(\frac{1}{g}\right)^{k_g-k_{g-1}} \qquad i = 1, \ldots, N; \quad (X_1 \equiv 1) \tag{8}$$

All models also include constraints for representing F, L and W_q. However, the expressions for F and L differ slightly depending on the model, as described below.

3.1 *If-Then* Model

The initial model is based directly on the formulation of Berman et al. The model includes the variables $P(j)$ for $j = k_0, k_0+1, \ldots, k_1, k_1+1, \ldots, S-1, S$, each representing the probability of there being j customers in the front room. These are floating point variables with domain $[0..1]$. The balance equations are represented by a set of if-then constraints. For example, the first balance equation, $P(j)\lambda = P(j+1)\mu$ for $j = k_0, k_0+1, ..., k_1-1$, is represented by the constraint: $(k_0 \leq j \leq k_1 - 1) \rightarrow P(j)\lambda = P(j+1)\mu$. Thus, somewhat inelegantly, an if-then constraint of this kind has to be added for each j between 0 and $S-1$ in order to represent one balance equation. In order to represent the rest of these equations, this technique has to be applied for each pair of switching points k_i, k_{i+1} for i from 0 to $N-1$. In addition, such if-then constraints are used for Equation (2), due to the dependence of this constraint on sums of variables between two switching points.

3.2 *PSums* Model

In order to avoid the if-then constraints, closed-form expressions[3] for the sums of probabilities between two switching points were derived and used as the basis of the second model. The set of $P(j)$ variables from the formulation of Berman et al. is replaced by a set of $PSums(k_i)$ variables for $i = 0, ..., N-1$, together with a set of probabilities $P(k_i)$ for $i = 0, 1, 2, \ldots, N$. The $PSums(k_i)$ variable represents the sum of all probabilities between k_i and $k_{i+1}-1$ and is defined in Equation (9). Equation (10) is a recursive formula for computing $P(k_i)$. $P(k_0)$ can be computed using Equation (6).

$$PSums(k_i) = \begin{cases} P(k_i)\dfrac{1-\left[\dfrac{\lambda}{(i+1)\mu}\right]^{k_{i+1}-k_i}}{1-\dfrac{\lambda}{(i+1)\mu}} & \text{if } \frac{\lambda}{(i+1)\mu} \neq 1 \\[2ex] P(k_i)(k_{i+1}-k_i) & otherwise. \end{cases} \tag{9}$$

$$P(k_{i+1}) = \left[\frac{\lambda}{(i+1)\mu}\right]^{k_{i+1}-k_i} P(k_i) \tag{10}$$

All quantities of interest can be expressed in terms of the $PSums(k_i)$ variables and the switching point probabilities, $P(k_i)$. In particular, the expected number of workers in the front room is

$$F = \sum_{i=1}^{N} i\left[PSums(k_{i-1}) - P(k_{i-1}) + P(k_i)\right]. \tag{11}$$

L, the expected number of customers in the front room, is

$$L = \sum_{i=0}^{N-1} L(k_i) + k_N P(k_N) \tag{12}$$

[3] The derivation of most of these expressions is based on expressing each $P(j)$ in terms of $P(k_i)$ for $k_i \leq j$ via the balance equations. In some derivations, such as that of Equation (9), the geometric series formula is used. Details can be found in [14].

where

$$L(k_i) = k_i PSums(k_i) +$$
$$P(k_i)\left[\frac{(\frac{\lambda}{(i+1)\mu})^{k_{i+1}-k_i}(k_i - k_{i+1}) + (\frac{\lambda}{(i+1)\mu})^{k_{i+1}-k_i+1}(k_{i+1} - k_i - 1) + \frac{\lambda}{(i+1)\mu}}{\left[\frac{(i+1)\mu - \lambda}{(i+1)\mu}\right]^2}\right]. \quad (13)$$

3.3 *Dual* Model

The problem can be alternatively formulated using variables w_j, which represent the number of workers in the front room when there are j customers present. Several expressions and constraints of the above models can be simplified by using these variables. Firstly, the balance equations can be stated as

$$P(j)\lambda = P(j+1)w_{j+1}\mu \qquad j = 0, 1, \ldots, S-1. \quad (14)$$

This formulation of the balance equations avoids the inefficient if-then constraints.

Secondly, F, the expected number of workers in the front room, can be stated as

$$F = \sum_{j=0}^{S} w_j P(j). \quad (15)$$

The difficulty with this model arises from the fact the $P(j)$ variables should be defined only for $j \geq k_0$ (since $P(k_0)$ is the probability of having from 0 to k_0 customers in the front room). It is hard to express this condition without explicitly having the variable k_0 in the model. Because of this, and since it is known that adding redundant variables to a model may be beneficial [11], it was decided that both the k_i and the w_j variables would be included in this model. In order to use two sets of redundant variables, the following channelling constraints[4] have to be included:

$$w_j < w_{j+1} \;\leftrightarrow\; k_{w_j} = j \qquad j = 0, 1, \ldots, S-1, \quad (16)$$

$$w_j = w_{j+1} \;\leftrightarrow\; k_{w_j} \neq j \qquad j = 0, 1, \ldots, S-1, \quad (17)$$

$$w_j = i \;\leftrightarrow\; k_{i-1} + 1 \leq j \leq k_i \qquad j = 0, 1, \ldots, S, \;\; i = 1, \ldots, N. \quad (18)$$

Additional constraints on the worker variables that are included in the model are: $w_0 = 0$, $w_S = N$ and $w_j \leq w_{j+1}$ for all j from 0 to $S-1$.

Preliminary experiments with these models showed poor performance. As one might expect from a problem with few constraints between decision variables, there was little constraint propagation, and search was required to essentially investigate the entire branch-and-bound tree. As a consequence, we examine shaving [4,9].

[4] Constraints (16) and (17) are redundant given the constraint $w_j \leq w_{j+1}$. However, such redundancy can often lead to increased propagation [7]. In future work, we will examine the effect that removing one of these constraints may have on the performance of the program.

4 Shaving

Shaving is a procedure for enforcing consistency in CSPs. It is based on temporarily adding constraints to the problem, performing propagation and making inferences according to the resulting state of the problem [5,15]. In our proposed shaving algorithm, *AlternatingSearchAndShaving*, two shaving procedures are run initially until they are no longer able to make domain reductions. Search is then performed until a better solution is found, at which point the shaving procedures are applied again. Subsequently, search and shaving alternate until one of them proves optimality of the best solution found. The first of the two shaving procedures makes inferences based on the feasibility of policies with respect to the B_l constraint, while the second one is based on the constraint $W_q \leq bestW_q$, where $bestW_q$ is the W_q value of the best policy found up to that point. In both shaving procedures, if the inferred constraint violates the current upper or lower bound of a k_i, then the best policy found up to that point is optimal.

B_l Shaving. Let $\min(k_i)$ and $\max(k_i)$ be, respectively, the smallest and largest values in the current domain of variable k_i. At each step of the B_l-based shaving procedure, $k_i = \min(k_i)$ or $k_i = \max(k_i)$ is temporarily added to the model for $i \in \{0, ..., N-1\}$. If $k_i = \min(k_i)$ is added, then all other switching points are assigned the maximum possible values subject to the condition that $k_n < k_{n+1}, \forall n \in \{0, ..., N-1\}$. If the resulting policy is infeasible, the constraint $k_i > \min(k_i)$ can be permanently added: if all variables except k_i are set to their maximum values, and the problem is infeasible (based on the B_l constraint), then, by Theorem 1, in any feasible policy k_i must be greater than $\min(k_i)$. If $k_i = \max(k_i)$ is added, all other switching points are assigned the minimum possible values. If the resulting policy is feasible, the constraint $k_i < \max(k_i)$ can be permanently added to the model. Since all variables except k_i are at the minimum values already, and k_i is at its maximum, it must be true, again by Theorem 1, that in any better solution the value of k_i has to be smaller than $\max(k_i)$. In both cases, after the resulting policy is checked for feasibility, the temporary constraint is removed.

W_q Shaving. The W_q-based shaving procedure makes inferences based strictly on the constraint $W_q \leq bestW_q$. The constraint $B \geq B_l$ is removed prior to running this procedure in order to eliminate the possibility of incorrect inferences. Similarly to the B_l-based shaving 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 the best W_q that has been encountered so far. Since all switching points except k_i are assigned their smallest possible values, this implies that in any solution with a better W_q, the value of k_i has to be strictly smaller than $\max(k_i)$.

5 Experimental Results

Several sets of experiments were performed in order to evaluate the efficiency of the proposed models and the shaving procedure, as well as to compare the performance of the best CP model with the performance of the heuristic proposed by Berman et al. All

CP models were implemented in ILOG Solver 6.2, while Berman et al.'s heuristic was implemented using C++. In all models, search assigns switching points in increasing index order and the smallest value in the domain of each variable is tried first.

The experimental results presented here are based only on the instances for which the optimal is between $\hat{K}$ and $\hat{\hat{K}}$, as instances in which either of these policies is optimal, or $\hat{\hat{K}}$ is infeasible, are easily solved both by the heuristic and the CP models with the elementary B_l-based shaving procedure. Preliminary experiments indicated that the value of S has a significant impact on the efficiency of the algorithms since higher values of S result in larger domains for the k_i variables for all models and also a higher number of w_j variables for the *Dual* model. Therefore, we considered instances for each value of S from the set $\{10, 20, \ldots, 100\}$ in order to gain an accurate understanding of the performance of the model and the heuristic.[5] Thirty feasible instances for which the optimal policy is neither $\hat{K}$ nor $\hat{\hat{K}}$ were generated for each S. A 10-minute time limit on the overall run-time of the program was enforced in the experiments. 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.

In order to perform comparisons between the CP models and the heuristic, we look at the number of instances in which the optimal solution was found and in which optimality was proved, and the mean relative error (MRE). MRE is a measure of solution quality that allows one to observe how quickly a particular algorithm is able to find a good solution. MRE is defined as $\frac{1}{|M|}\sum_{m \in M} \frac{c(a,m)-c^*(m)}{c^*(m)}$, where a is a particular algorithm used to solve the problem, M is the set of problem instances on which the algorithm is being tested, $c(a,m)$ is the cost of a solution found for instance m by algorithm a, and $c^*(m)$ is the best solution for instance m found during our experiments.

5.1 Comparison of Constraint Programming Models

Table 1 presents, for each model, the number of instances in which it finds the best solution (out of 300), the number of instances in which it finds the optimal solution (out of the 240 instances for which the optimal solution is known), and the number of times it proves optimality. It can be seen that all models find the optimal solution in the 240 instances for which it is known. However, the *PSums* model outperforms the other two models in the rest of the performance measures, proving optimality in 79% of all instances, and finding the best-known solution of any algorithm in 97.3% of all the instances considered.

Observations from Table 1 can be further confirmed by looking at Figure 1 (Left). The figure shows how MRE changes over the first 50 seconds of run-time for *If-Then*, *PSums* and *Dual* models with *AlternatingSearchAndShaving*, and for P_1 (we comment on the performance of P_1 in Section 5.2). It can be seen that *PSums* is, on average, able to find better solutions than the other two models given the same amount of run-time.

[5] For most instances with S greater than 100, neither our method nor Berman et al.'s heuristic P_1 may be used due to numerical instability. The maximum value of S used in the experiments of Berman et al. is also 100.

Table 1. Comparison of three CP models with *AlternatingSearchAndShaving* with Berman's Heuristic P_1. The Hybrid model is presented in Section 6.

	# best found (/300)	# optimal found (/240)	# optimal proved (/300)
PSums	292	240	238
If-Then	280	240	234
Dual	281	240	234
P_1	282	239	0
PSums-P_1 Hybrid	300	240	238

5.2 P_1 vs. the Best Constraint Programming Approach

It can be seen, from Table 1, that the heuristic performs extremely well, finding the best-known solution in only ten fewer instances than the *PSums* model, in two more instances than the *If-Then* model and in one more than the *Dual*. Moreover, it finds, but, of course, cannot prove, the optimal solution in 79.6% of all instances (239 out of the 240 instances for which the optimal is known). Its run-time is negligible, while the mean run-time of the best CP model, *PSums*, is approximately 130 seconds.

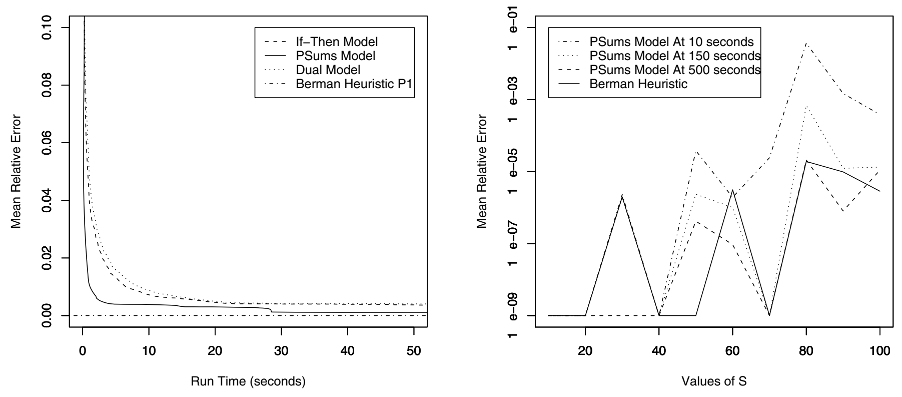

Fig. 1. Left: Comparison of MRE of three CP models with *AlternatingSearchAndShaving* with Berman's P_1 heuristic. Right: MRE for each value of S for P_1 and the $PSums$ model.

From Figure 1 (Left), it can be observed that the heuristic achieves a very small MRE in a negligible amount of time. After about 50 seconds of run-time, the MRE over 300 instances resulting from *PSums* with *AlternatingSearchAndShaving* becomes comparable to that of the heuristic MRE.

In Figure 1 (Right), the MRE for the 30 instances for each value of S is presented for P_1 and for *PSums* with *AlternatingSearchAndShaving* at 10, 150 and 500 seconds of run-time. After 10 seconds, the performance of *PSums* is comparable to that of the heuristic for $S \leq 40$, but the heuristic appears to be quite a bit better for higher values of S. At 150 seconds, the performance of *PSums* is comparable to that of the heuristic except at S of 50 and 80. After 500 seconds, *PSums* has a smaller or equivalent MRE over the 300 instances and also a lower MRE for each value of S except 50 and 100.

Overall, these results indicate that P_1 performs extremely well—its run-time is negligible, it finds the optimal solution in 79.6% of the instances and the best-known solution in 94%. Moreover, it results in very low MRE. Although *PSums* with *AlternatingSearchAndShaving* is able to achieve better performance in both of these measures, it is clear that these improvements are quite small given that the *PSums* run-time is so much higher than the run-time of the heuristic.

6 *PSums*-P_1 Hybrid

Naturally, it is desirable to create a method that would be able to find a solution of high quality in a short amount of time, as does Berman's heuristic, and that would also have the same high rate of being able to prove optimality within a reasonable run-time as does *PSums* with *AlternatingSearchAndShaving*. It is therefore worthwhile to experiment with a *PSums*-P_1 Hybrid, which starts off by running P_1 and then, assuming the instance is feasible, uses the *PSums* model with *AlternatingSearchAndShaving* to find a better solution or prove the optimality of the solution found by P_1 (infeasibility of an instance is proven if the heuristic determines that policy $\hat{\hat{K}}$ is infeasible).

Since P_1 is very fast, running it first incurs almost no overhead. We have also shown that P_1 provides solutions of very high quality (in 94% of instances used in the experiments, it found the best-known solution). Therefore, the first iteration of the W_q-based procedure should be able to significantly prune the domains of switching point variables because of the good quality solution found by the heuristic. Continuing by alternating the two shaving techniques and search, which has also been shown to be an effective approach, should result in at least as many instances for which optimality is proven as for the *PSums* model with *AlternatingSearchAndShaving*.

The proposed hybrid algorithm was tested on the same set of 300 instances that was used above. Results of the hybrid are presented in Table 1. The hybrid was able to find the best-known solution in all 300 cases while still being able to prove optimality in as many cases as *PSums*. Thus, in spite of the good quality solutions that are discovered quickly because of the heuristic, the domains of switching points in some instances are not reduced enough to increase the number of cases in which optimality is proved. The mean run-time for the hybrid is 130.18 seconds, which is essentially identical to the mean run-time of 129.98 seconds for *PSums* with *AlternatingSearchAndShaving*.

Thus, the hybrid is the best choice for solving this problem: it finds as good a solution as the heuristic in as little time (close to 0 seconds), it is able to prove optimality in as many instances as the best pure CP method, and it finds the best-known solution in all instances considered. Moreover, all these improvements are achieved with a negligible increase in the average run-time over the *PSums* model with shaving.

7 Discussion

In this section, we examine some of the reasons for the poor performance of the CP models without shaving and suggest reasons for the observed differences among them.

7.1 Lack of Back-Propagation

In our experiments, we have some instances for which even the *PSums*-P_1 hybrid with *AlternatingSearchAndShaving* is unable to find and prove optimality within the 10-minute time limit. Further analysis of the algorithm's behaviour suggests that this performance can be explained by the lack of back-propagation. Back-propagation refers to the pruning of the domains of the decision variables due to the addition of a constraint on the objective function: the objective constraint propagates "back" to the decision variables, removing domain values and so reducing search. In the CP models presented above, there is very little back-propagation. We illustrate this by focusing on the *PSums* model without shaving.

Throughout search, if a new best solution is found, the constraint $W_q \leq bestW_q$, where $bestW_q$ is the new objective value, is added to the model. However, the domains of the switching point variables are usually not reduced in any way after the addition of such a constraint. This can be illustrated by observing the amount of propagation that occurs in the model when W_q is constrained.

For example, consider an instance of the problem with $S = 6$, $N = 3$, $\lambda = 15$, $\mu = 3$, and $B_l = 0.32$. The initial domains of the switching point variables are $[0..3]$, $[1..4]$, $[2..5]$ and $[6]$. The initial domains of the probability variables $P(k_i)$ for each i, after the addition of W_q bounds provided by $\hat{K}$ and $\hat{\hat{K}}$, are listed in Table 2. The initial domain of W_q, also determined by the objective function values of $\hat{K}$ and $\hat{\hat{K}}$, is $[0.22225..0.425225]$. The initial domains of L and F, are $[2.8175e^{-7}..6]$ and $[0..2.68]$, respectively. Upon the addition of the constraint $W_q \leq 0.306323$, where 0.306323 is the known optimal value for this instance, the domain of W_q is reduced to $[0.22225..0.306323]$, the domain of L becomes $[1.68024..6]$ and the domain of F remains $[0..2.68]$. The domains of $P(k_i)$ after this addition are listed in Table 2. The domains of both types of probability variables are reduced by the addition of the new W_q constraint. However, the domains of the switching point variables remain unchanged. Therefore, even though all policies with value of W_q less than 0.306323 are infeasible, constraining W_q to be less than this value does not result in any reduction of the search space. It is still necessary to enumerate all remaining policies in order to show that no better feasible solution exists.

One of the reasons for the lack of pruning of the domains of the k_i variables due to the W_q constraint is likely the complexity of the expression for W_q. In particular, recall that W_q is expressed in all models as $W_q = \frac{L}{\lambda(1-P(S))} - \frac{1}{\mu}$. In the example above, when W_q is constrained to be less than or equal to 0.306323, we get the constraint $0.306323 \geq \frac{L}{15(1-P(S))} - \frac{1}{3}$, which implies that $9.594845(1 - P(S)) \geq L$. This explains why the domains of both L and $P(S)$ change upon this addition to the model. The domains of the rest of the $P(k_i)$ variables change because of the relationships between the $P(k_i)$s (Equation (10)) and because of the constraint that the sum of all probability variables has to be 1. Similarly, the domains of $PSums(k_i)$ change because these variables are expressed in terms of $P(k_i)$ (Equation (9)). However, because the actual k_i variables mostly occur as exponents in expressions for $PSums(k_i)$, $P(k_i)$, and $L(k_i)$, the minor changes in the domains of $PSums(k_i)$, $P(k_i)$, or $L(k_i)$ that happen due to the constraint on W_q have no effect on the domains of the k_i. This analysis suggests

Table 2. Domains of $P(j)$ and $PSums(j)$ variables for $j = k_0, k_1, k_2, k_3$, before and after the addition of the constraint $W_q \leq 0.306323$

	Before addition of $W_q \leq 0.306323$		After addition of $W_q \leq 0.306323$	
j	$P(j)$	$PSums(j)$	$P(j)$	$PSums(j)$
k_0	$[4.40235e^{-6}..0.979592]$	[0..1]	$[4.40235e^{-6}..0.979592]$	[0..0.683666]
k_1	$[1.76094e^{-7}..1]$	[0..1]	[0.000929106..1]	[0..0.683666]
k_2	$[2.8175e^{-8}..0.6]$	$[2.8175e^{-8}..1]$	[0.0362932..0.578224]	[0.0362932..0.71996]
k_3	$[4.6958e^{-8}..1]$	N/A	[0.28004..0.963707]	N/A

that it may be interesting to investigate a CP model based on log-probabilities rather than on the probabilities themselves. Such a model may lead to stronger propagation.

7.2 Differences in the Constraint Programming Models

Experimental results demonstrate that the best CP model of those proposed is *PSums*. In all models, the shaving procedures make the same number of domain reductions because shaving is based on the W_q and B_l constraints. However, the time that each shaving iteration takes is dependent on the model. Our empirical results show that each iteration of shaving takes a smaller amount of time with the *PSums* model than with the other two. This appears to be the primary reason for the *PSums* model finding good solutions faster than the other models, as shown in Figure 1 (Left). *PSums* is radically different from the other two models because it does not include an explicit representation of the balance equations. This model thus avoids the if-then constraints required in the *If-Then* model. Moreover, *PSums* has a smaller number of probability variables than the other two models, because it calculates sums of probabilities between two switching points rather than the probability of j customers being present in the front room for all j from 0 to S. This reduces the number of probability variables from $S + 1$ to $2N + 1$. In addition, the probability variables included in this model are more tightly linked by the closed-form expressions. Thus, because of the tighter links between variables, and a smaller number of variables, each iteration of shaving in *PSums* takes a smaller amount of time than in the other two models.

A comparison of the *If-Then* model with the *Dual* using Figure 1 shows that the *If-Then* model is usually able to find good solutions in a smaller amount of time. This is slightly surprising because the *Dual* model uses a much simpler representation of the balance equations and the expression for F, avoiding the use of if-then constraints. One possible explanation for the *Dual* sometimes taking more time to find a good solution is that, at each shaving iteration, it has to assign more variables (via propagation) than the other two models. In particular, in order to represent a switching policy, the *Dual* has to assign S w_j variables in addition to N k_i variables (usually, S is much larger than N).

On the other hand, within the given time limit, the *Dual* found the best solution in one more instance than the *If-Then* model. This may be due to an increase in the amount of propagation which results from the use of dual variables. In fact, an examination of the initial domains of the probability variables for the example instance of Section 7.1 shows that these domains are quite a bit smaller in the *Dual* model than in the *If-Then* model (e.g. the domain of $P(0)$ is $[0..1]$ in the *If-Then* model, while it is [0..0.00926208] in the *Dual*). This examination also shows that the initial domains of probability variables in the *If-Then* and *Dual* models are actually smaller than those in the *PSums* model. This implies that there exist some instances in which more initial propagation occurs in the *If-Then* model or the *Dual* model than in *PSums*.

8 Related Work and Possible Extensions

Several papers exist that deal with similar types of problems as the one considered here. For example, Berman & Larson [2] study a similar problem of switching workers between two rooms in a retail facility where the customers in the front room are divided into two categories, those "shopping" in the store and those at the checkout. Similarly, Palmer & Mitrani [10] consider the problem of switching computational servers between different types of jobs where the randomness of user demand may lead to unequal utilization of resources. Batta et al. [1] study the problem of assigning cross-trained customer service representatives to different types of calls in a call centre, depending on estimated demand patterns for each type of call. These three papers provide examples of problems for which CP could prove to be a useful approach. Investigating CP solutions to these problems is therefore one possible direction of future work. In particular, it may be interesting to look at problems with more complex constraints (e.g., on capacities or between workers) that may be naturally suitable for the CP approach. A complementary direction is to study the basic models of queueing theory in order to understand the applicability of CP.

9 Conclusions

In this paper, a constraint programming approach is proposed for the problem of finding the optimal times to switch workers between the front room and the back room of a retail facility under stochastic customer arrival and service times. This is the first work of which we are aware that examines solving such stochastic queueing control problems using constraint programming. The best pure CP method proposed is able to prove optimality in a large proportion of instances within a 10-minute time limit. Previously, there existed no non-heuristic solution to this problem aside from naive enumeration. As a result of our experiments, we hybridized the best pure CP model with the heuristic proposed for this problem in the literature. This hybrid technique is able to achieve performance that is equivalent to, or better than, that of each of the individual approaches alone: it is able to find very good solutions in a negligible amount of time due to the use of the heuristic, and is able to prove optimality in a large proportion of problem instances within 10 CPU minutes due to the CP model.

This work demonstrates for the first time that constraint programming can be a good approach for solving a stochastic optimization problem based on queueing theory.

References

1. R. Batta, O. Berman, and Q. Wang. Balancing staffing and switching costs in a call/service center. *European Journal of Operations Research*. To Appear. Available at: http://www.acsu.buffalo.edu/~batta/papers/Batta_et_al.pdf.

2. O. Berman and R. Larson. A queueing control model for retail services having back room operations and cross-trained workers. *Computers and Operations Research*, 31(2):201–222, 2004.
3. O. Berman, J. Wang, and K. P. Sapna. Optimal management of cross-trained workers in services with negligible switching costs. *European Journal of Operations Research*, 167(2):349–369, 2005.
4. Y. Caseau and F. Laburthe. Cumulative scheduling with task intervals. In *Proceedings of the Joint International Conference and Symposium on Logic Programming*, pages 363–377. MIT Press, 1996.
5. S. Demassey, C. Artigues, and P. Michelon. Constraint-propagation-based cutting planes: An application to the resource-constrained project scheduling problem. *INFORMS Journal on Computing*, 17(1):52–65, 2005.
6. D. Gross and C. Harris. *Fundamentals of Queueing Theory*. John Wiley & Sons, Inc., 1998.
7. B. Hnich, B. Smith, and T. Walsh. Dual modelling of permutation and injection problems. *Journal of Artificial Intelligence Research*, 21:357–391, 2004.
8. J. D. C. Little. A proof of the queueing formula $L = \lambda W$. *Operations Research*, 9:383–387, 1961.
9. P. Martin and D. B. Shmoys. A new approach to computing optimal schedules for the job shop scheduling problem. In *Proceedings of the Fifth Conference on Integer Programming and Combinatorial Optimization*, pages 389–403, 1996.
10. J. Palmer and I. Mitrani. Optimal server allocation in reconfigurable clusters with multiple job types. In *Proceedings of the Computational Science and Its Applications International Conference*, pages 76–86, 2004.
11. B.M. Smith. Modelling for constraint programming. Lecture Notes for the First International Summer School on Constraint Programming, 2005. Available at: http://www.math.unipd.it/˜frossi/cp-school/.
12. S.A. Tarim, S. Manandhar, and T. Walsh. Stochastic constraint programming: A scenario-based approach. *Constraints*, 11(1):53–80, 2006.
13. S.A. Tarim and A. Miguel. 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*, pages 133–148, 2005.
14. D. Terekhov and J. C. Beck. Solving a stochastic queueing control problem with constraint programming. Technical Report MIE-OR-TR2006-06, Department of Mechanical and Industrial Engineering, University of Toronto, 2006. Available from http://www.mie.utoronto.ca/labs/ORTechReps/.
15. M.R.C. van Dongen. Beyond singleton arc consistency. In *Proceedings of the 17th European Conference on Artificial Intelligence*, pages 163–167, 2006.
16. T. Walsh. Stochastic constraint programming. In *Proceedings of the 15th European Conference on Artificial Intelligence*, pages 111–115, 2002.