Simulation-Based Optimization of Agent Scheduling in Multiskill Call Centers

Athanassios N. Avramidis\textsuperscript{1}, Michel Gendreau\textsuperscript{1}, Pierre L’Ecuyer\textsuperscript{1}, Ornella Pisacane\textsuperscript{2}

1. Département d’Informatique et de Recherche Opérationnelle Université de Montréal, Canada
2. Dipartimento di Elettronica Informatica e Sistemistica Università della Calabria, Italia

Sponsored by Bell Canada via the Bell University Laboratories.

Outline

▶ Motivation: importance of call centers.
▶ Problem formulation: scheduling agents under stochastic service-level constraints.
▶ Two-step vs single-step solution approaches.
▶ Simulation-based optimization algorithms.
▶ Numerical illustration.
Importance of Call Centers

Include sales by telephone, customer service, billing/recovery services, marketing centers, governmental or other public services, 911 services, taxis, pizza, fast food car order, etc.

In USA+Canada: about 50,000 call centers with 3 million agents. Call centers employ about 3% of the workforce in North America. More than in agriculture!
Importance of Call Centers

Include sales by telephone, customer service, billing/recovery services, marketing centers, governmental or other public services, 911 services, taxis, pizza, fast food car order, etc.

In USA+Canada: about 50,000 call centers with 3 million agents. Call centers employ about 3% of the workforce in North America. More than in agriculture!

They are “the face” of the organization: ⇒ important for keeping a good image and customer satisfaction, acquiring new customers, increasing sales (for commercial organizations) and providing efficient service (for public organizations).

Standard industry practice for operations management of call centers leaves room for large improvements and savings.

How can we use simulation effectively to improve management of call centers in industry?
Framework: A multiskill center

$K$ call types. Depends on required technical skill, language, importance, etc.

$I$ agent types. Each has skills to handle certain call types.
Service time distribution may depend on pair \langle call type, agent type \rangle.

Arrival rates

\[ \lambda_1 \quad \lambda_2 \quad \ldots \quad \lambda_K \]

Abandonments

CRM (routing logic and queues)

Agent types

\[ S_1 \quad \ldots \quad S_I \]

Service rates

\[ \mu_{1,1} \ldots \mu_{K,1} \quad \mu_{1,I} \ldots \mu_{K,I} \]
A small example of call-to-agent assignment
Typical call center models

**Arrival process** is nonstationary and much more complicated than Poisson. **Service times** are not exponential. **Abandonments** (balking + reneging), retrials, returns, etc.
Typical call center models

Arrival process is nonstationary and much more complicated than Poisson. Service times are not exponential. Abandonments (balking + reneging), retrials, returns, etc.

Skill-based routing: Rules that control in real time the call-to-agent and agent-to-call assignments. Can be complex in general. Here we assume they are fixed.
Typical call center models

Arrival process is nonstationary and much more complicated than Poisson. Service times are not exponential. Abandonments (balking + reneging), retrials, returns, etc.

Skill-based routing: Rules that control in real time the call-to-agent and agent-to-call assignments. Can be complex in general. Here we assume they are fixed.

Agents using fewer skill tend to work faster. Also less expensive. Compromise between single-skill agents (specialists) vs flexible multiskill agents (generalists). A small amount of flexibility usually gives the best compromise.

Problem considered here: minimize total cost of agents under service-level (SL) constraints.
Scheduling optimization

Suppose the routing rules are fixed (we do not optimize them here). Several call types, several agent types, several time periods.
Scheduling optimization

Suppose the routing rules are fixed (we do not optimize them here).
Several call types, several agent types, several time periods.

A shift type specifies the time when the agent starts working, when he/she finishes, and all the lunch and coffee breaks.

\[ c_{i,q} = \text{cost of an agent of type } i \text{ having shift type } q. \]
Scheduling optimization

Suppose the routing rules are fixed (we do not optimize them here). Several call types, several agent types, several time periods.

A shift type specifies the time when the agent starts working, when he/she finishes, and all the lunch and coffee breaks. $c_{i,q} =$ cost of an agent of type $i$ having shift type $q$.

The decision variables $x$ and $z$ are:

(i) $x_{i,q} =$ number of agents of type $i$ having shift type $q$;
(ii) $z_{j,i,p} =$ number of agents of type $j$ that work as type-$i$ agents in period $p$, with $S_i \subset S_j$ (they use only part of their skills).

This determines indirectly the staffing vector $y$, where $y_{i,p} =$ num. agents of type $i$ in period $p$, and $a_{p,q} = 1$ iff shift $q$ covers period $p$:

$$y_{i,p} = \sum_{q=1}^{Q} a_{p,q} x_{i,q} + \sum_{l \in S_i^+} z_{l,i,p} - \sum_{l \in S_i^-} z_{i,l,p} \quad \text{for all } i, p.$$
Scheduling Optimization Problem

\( \mathbf{x} = \text{vector of shifts}; \quad \mathbf{c} = \text{their costs}; \quad \mathbf{y} = \text{staffing vector}; \)

**service level** for type \( k \) in period \( p \) (depends on entire vector \( \mathbf{y} \)):

\[
g_{k,p}(\mathbf{y}) = \frac{\mathbb{E}[\text{num. calls type } k \text{ in period } p \text{ answered within time limit}]}{\mathbb{E}[\text{num. calls type } k \text{ in period } p, \text{ ans., or abandon. after limit}]}.
\]

\[
(P0) : \quad \begin{aligned}
\text{min} & \quad \mathbf{c}^T \mathbf{x} = \sum_{i=1}^{I} \sum_{q=1}^{Q} c_{i,q} x_{i,q} \\
\text{subject to} & \quad \mathbf{A} \mathbf{x} + \mathbf{B} z = \mathbf{y}, \\
& \quad g_{k,p}(\mathbf{y}) \geq l_{k,p} \quad \text{for all } k, p, \\
& \quad g_p(\mathbf{y}) \geq l_p \quad \text{for all } p, \\
& \quad g_k(\mathbf{y}) \geq l_k \quad \text{for all } k, \\
& \quad g(\mathbf{y}) \geq l, \\
& \quad \mathbf{x} \geq 0, \ z \geq 0, \ y \geq 0, \ \text{and integer}.
\end{aligned}
\]
Here we have virtual agents; we think of medium-term planning, e.g., for hiring and training decisions. In the rostering problem, one has a fixed set of employees instead.
Here we have virtual agents; we think of medium-term planning, e.g., for hiring and training decisions.
In the rostering problem, one has a fixed set of employees instead. For given staffing, we can estimate the (average) service levels, abandonment ratios, etc., either

- via simplified queueing models (⇒ unreliable), or
- by simulation.
Single-Period Staffing Problem

Take one period at a time and assume that the system is in steady-state.

\[(P1) : \text{[Staffing problem]} \]
\[\begin{align*}
\min \quad & c^t y = \sum_{i=1}^{l} c_i y_i \\
\text{subject to} \quad & \tilde{g}_k(y) \geq l_k \quad \text{for all } k, \\
& \tilde{g}(y) \geq l, \\
& y \geq 0, \text{ and integer.}
\end{align*}\]
Scheduling by a two-step method (industry standard)

First step: solve the staffing problem for each period (easier).
Second step: find admissible set of shifts that cover the staffing at minimal cost, allowing skill transfer.
Easy to solve, by linear programming with integer variables.

(P2) : [Second step]
\[
\begin{align*}
\min & \quad c^T x \\
\text{subject to} & \\
Ax + Bz & \geq \hat{y} \\
x & \geq 0, z \geq 0 \quad \text{and integer}
\end{align*}
\]
This does not give an optimal solution to the original scheduling problem. And the gap could be significant if there are discrepancies in mix of agent types across successive periods.
This does not give an optimal solution to the original scheduling problem. And the gap could be significant if there are discrepancies in mix of agent types across successive periods.

The skill-transfer decision variables help, but there are still cases where the two-step approach yields very poor solutions.

**Small example:** 3 periods, 3 call types, one shift type, 3 agent types: \( S_1 = \{1, 2\} \), \( S_2 = \{1, 3\} \), and \( S_3 = \{2, 3\} \). Poisson arrivals with rate 100, split between call types \( \{1, 2\} \) in period 1, \( \{1, 3\} \) in period 2, \( \{2, 3\} \) in period 3. Exponential service times with mean 1. Want 80–20 SL.

Two-step solution: 312 agents (104 of each type).

This does not give an optimal solution to the original scheduling problem. And the gap could be significant if there are discrepancies in mix of agent types across successive periods.

The skill-transfer decision variables help, but there are still cases where the two-step approach yields very poor solutions.

Small example: 3 periods, 3 call types, one shift type, 3 agent types: $S_1 = \{1, 2\}$, $S_2 = \{1, 3\}$, and $S_3 = \{2, 3\}$. Poisson arrivals with rate 100, split between call types $\{1, 2\}$ in period 1, $\{1, 3\}$ in period 2, $\{2, 3\}$ in period 3. Exponential service times with mean 1. Want 80–20 SL.

Two-step solution: 312 agents (104 of each type).


So we must really attack the scheduling problem head-on.
Sample-path optimization via simulation

We simulate $n$ independent operating days of the center, to estimate the functions $g$.

Let $\omega$ represent the source of randomness, i.e., the sequence of independent uniform r.v.’s underlying the entire simulation ($n$ runs).

The empirical SL’s over the $n$ simulation runs are:
- $\hat{g}_{n,k,p}(y,\omega)$ for call type $k$ in period $p$;
- $\hat{g}_{n,p}(y,\omega)$ aggregated over period $p$;
- $\hat{g}_{n,k}(y,\omega)$ aggregated for call type $k$;
- $\hat{g}_{n}(y,\omega)$ aggregated overall.

For a fixed $\omega$, these are deterministic functions of $y$.

We replace the (unknown) functions $g(\cdot)$ by $\hat{g}(\cdot,\omega)$ and optimize.

To compute them at different values of $y$, we use simulation with well-synchronized common random numbers.
Empirical (sample) scheduling optimization problem

\[
(\text{SP0}_n): \quad \text{[Sample scheduling problem]}
\]
\[
\begin{align*}
\min & \quad c^T x = \sum_{i=1}^I \sum_{q=1}^Q c_{i,q} x_{i,q} \\
\text{subject to} & \quad A x + B z = y, \\
& \quad \hat{g}_{n,k,p}(y) \geq l_{k,p} \quad \text{for all } k, p, \\
& \quad \hat{g}_{n,p}(y) \geq l_p \quad \text{for all } p, \\
& \quad \hat{g}_{n,k}(y) \geq l_k \quad \text{for all } k, \\
& \quad \hat{g}_n(y) \geq l, \\
& \quad x \geq 0, z \geq 0, \text{ and integer.}
\end{align*}
\]

**Theorem:** Under mild conditions, when \( n \to \infty \), the optimal solution of \( \text{SP0}_n \) converges w.p.1 to that of P0.
Solving the sample optimization problem

Integer programming with cutting planes.
[Atlason, Epelman, and Henderson, 2004; Cezik and L’Ecuyer 2005]
Replace the nonlinear constraints in SP0_n by a set of linear constraints. This gives an integer program (IP).
Empirical (sample) scheduling optimization problem

\[(SP0_n) : \text{[Sample scheduling problem]}\]
\[
\begin{align*}
\text{min} & \quad c^t x \\
\text{subject to} & \quad Ax + Bz = y, \\
& \quad \hat{g}_{n,k,p}(y) \geq l_{k,p} \quad \text{for all } k, p, \\
& \quad \hat{g}_n,p(y) \geq l_p \quad \text{for all } p, \\
& \quad \hat{g}_n,k(y) \geq l_k \quad \text{for all } k, \\
& \quad \hat{g}_n(y) \geq l, \\
& \quad x \geq 0, z \geq 0, \text{ and integer.}
\end{align*}
\]
Empirical relaxed integer program

\[ (\text{SIP}_n) : \quad \text{[Sample relaxed IP]} \]

\[
\begin{align*}
\min & \quad c^t x \\
\text{subject to} & \quad Ax + Bz = y, \\
& \quad Cy \geq b, \\
& \quad x \geq 0, z \geq 0, \text{ and integer.}
\end{align*}
\]
Solving the sample optimization problem

Integer programming with cutting planes. [Atlason, Epelman, and Henderson, 2004; Cezik and L’Ecuyer 2005]

Replace the nonlinear constraints in SP0_n by a set of linear constraints. This gives an integer program (IP).

Start with a relaxation of the problem: Initial constraints are obtained simply by imposing that skill supply covers a given fraction of the load, for each call type (max flow problem).
Solving the sample optimization problem

Integer programming with cutting planes. [Atlason, Epelman, and Henderson, 2004; Cezik and L’Ecuyer 2005]
Replace the nonlinear constraints in $\text{SP}_0^n$ by a set of linear constraints. This gives an integer program (IP).

Start with a relaxation of the problem: Initial constraints are obtained simply by imposing that skill supply covers a given fraction of the load, for each call type (max flow problem).

At each step, use simulation to compute the service levels in $\text{SP}_0^n$ for the optimal solution $\bar{y}$ of the current IP.
At each step, use simulation to compute the service levels in SP0_n for the optimal solution \( \bar{y} \) of the current IP.

For each constraint that is not satisfied in SP0_n, say \( g(\bar{y}) < l \), we estimate a subgradient \( \bar{q} \) of \( g \) at \( \bar{y} \). Then we must have

\[
l \leq g(y) \leq g(\bar{y}) + \bar{q}^t(y - \bar{y}).
\]

So we add the linear cut

\[
\bar{q}^t y \geq \bar{q}^t \bar{y} + l - g(\bar{y}).
\]

This removes \( \bar{y} \) from the feasible solutions of the relaxed IP without removing any feasible solution of SP0_n, if \( \bar{q} \) is a true subgradient.
At each step, use simulation to compute the service levels in SP0\(_n\) for the optimal solution \(\bar{y}\) of the current IP.

For each constraint that is not satisfied in SP0\(_n\), say \(g(\bar{y}) < l\), we estimate a subgradient \(\bar{q}\) of \(g\) at \(\bar{y}\). Then we must have

\[
    l \leq g(y) \leq g(\bar{y}) + \bar{q}^t(y - \bar{y}).
\]

So we add the linear cut

\[
    \bar{q}^t y \geq \bar{q}^t \bar{y} + l - g(\bar{y}).
\]

This removes \(\bar{y}\) from the feasible solutions of the relaxed IP without removing any feasible solution of SP0\(_n\), if \(\bar{q}\) is a true subgradient.

We stop when all SL constraints of SP0\(_n\) are satisfied.
At each step, use simulation to compute the service levels in SP0\(_n\) for the optimal solution \(\bar{y}\) of the current IP.

For each constraint that is not satisfied in SP0\(_n\), say \(g(\bar{y}) < l\), we estimate a subgradient \(\bar{q}\) of \(g\) at \(\bar{y}\). Then we must have

\[
l \leq g(y) \leq g(\bar{y}) + \bar{q}^t(y - \bar{y}).
\]

So we add the linear cut

\[
\bar{q}^t y \geq \bar{q}^t \bar{y} + l - g(\bar{y}).
\]

This removes \(\bar{y}\) from the feasible solutions of the relaxed IP without removing any feasible solution of SP0\(_n\), if \(\bar{q}\) is a true subgradient.

We stop when all SL constraints of SP0\(_n\) are satisfied.

**Phase II:** run longer simulation to perform a local adjustment to the final solution, using heuristics (add, remove, switch).
Heuristic to estimate subgradients $\bar{q}$: choose $d_j \in \{1, 2, 3\}$ and compute (by simulation):

$$\bar{q}_j = \frac{\hat{g}(\bar{y} + d_j e_j) - \hat{g}(\bar{y})}{d_j}$$

for $j = 1, \ldots, IP$.

This is very time-consuming; for this reason, we simulate only $n_0 < n$ days here...

For large problems, we also solve the IP as an LP and then round the solution (at each step, to be able to simulate).
Heuristic to estimate subgradients $\bar{q}$: choose $d_j \in \{1, 2, 3\}$ and compute (by simulation):

$$
\bar{q}_j = \left( \hat{g} (\bar{y} + d_j e_j) - \hat{g} (\bar{y}) \right) / d_j
$$

for $j = 1, \ldots, IP$.

This is very time-consuming; for this reason, we simulate only $n_0 < n$ days here...

For large problems, we also solve the IP as an LP and then round the solution (at each step, to be able to simulate).

Because of simulation noise and several heuristics, the method does not always provide a near-optimal feasible solution. But we can repeat it a few times and retain the best found solution.
Example

\[ K = 20 \text{ call types,} \]
\[ I = 35 \text{ agent types (see paper),} \]
\[ P = 52 \text{ time periods (15 min.).} \]

Arrival rates vary from 5 to 27 calls per minute, per type. Patience times are exponential with rate 0.1 per minute.
Example

\( K = 20 \) call types, \\
\( I = 35 \) agent types (see paper), \\
\( P = 52 \) time periods (15 min.).

Arrival rates vary from 5 to 27 calls per minute, per type.
Patience times are exponential with rate 0.1 per minute.

We consider 123 different shift types; each lasts 7.5 hours and includes one 30-min. lunch break and two 15-min. coffee breaks.

This gives \( 35 \times 123 = 4305 \) decision variables \( x_{i,q} \), and many more skill transfer variables!

An agent with \( s \) skills costs \( 0.9 + s/10 \).

SL constraints: 80–20 for each period.
CP-LP: cutting planes + linear programming + rounding.
Solved SP0_n with n = 300 days; n_0 = 20 days for subgradients.

TS: two-step approach, optim. by simulation for each period.
Solved staffing problem with n = 1500 days for each period.

Each method used a CPU time budget of about 5 hours, and this was repeated 8 times, independently.
CP-LP: cutting planes + linear programming + rounding.  
Solved SP0_n with n = 300 days; n_0 = 20 days for subgradients.

TS: two-step approach, optim. by simulation for each period.  
Solved staffing problem with n = 1500 days for each period.

Each method used a CPU time budget of about 5 hours, and this was repeated 8 times, independently.

Each final solution was then simulated for n_\ast = 50000 days as a more stringent feasibility test.

<table>
<thead>
<tr>
<th>Algo</th>
<th>\text{min cost}</th>
<th>\text{median cost}</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP-LP</td>
<td>300</td>
<td>136.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>137.5</td>
</tr>
<tr>
<td>TS</td>
<td>1500</td>
<td>156.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>156.1</td>
</tr>
</tbody>
</table>

CP-LP found 4 feasible and 4 almost feasible solutions (the worst SL in any given period was 0.797). The cheapest solution, whose cost is 136.2, had a worst-case SL of 0.797.

TS found 8 feasible (although much more expensive) solutions.
**CP-LP**: cutting planes + linear programming + rounding.
Solved SP0\(_n\) with \(n = 300\) days; \(n_0 = 20\) days for subgradients.

**TS**: two-step approach, optim. by simulation for each period.
Solved staffing problem with \(n = 1500\) days for each period.

Each method used a CPU time budget of about 5 hours, and this was repeated 8 times, independently.

Each final solution was then simulated for \(n_* = 50000\) days as a more stringent feasibility test.

<table>
<thead>
<tr>
<th>Algo</th>
<th>(n)</th>
<th>min cost</th>
<th>median cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP-LP</td>
<td>300</td>
<td>136.2</td>
<td>137.5</td>
</tr>
<tr>
<td>TS</td>
<td>1500</td>
<td>156.1</td>
<td>156.1</td>
</tr>
</tbody>
</table>

CP-LP found 4 feasible and 4 almost feasible solutions (the worst SL in any given period was 0.797). The cheapest solution, whose cost is 136.2, had a worst-case SL of 0.797.

TS found 8 feasible (although much more expensive) solutions.
Summary of the **best feasible solutions** found by CP-LP and by TS

<table>
<thead>
<tr>
<th>agent type</th>
<th>num. skills</th>
<th>cost</th>
<th>CP-LP</th>
<th>TS</th>
</tr>
</thead>
<tbody>
<tr>
<td>4, 1, 5, 9, 11, 13</td>
<td>9</td>
<td>1.8</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7, 12, 14</td>
<td>7</td>
<td>1.6</td>
<td>3</td>
<td>32</td>
</tr>
<tr>
<td>0, 2, 3, 8</td>
<td>6</td>
<td>1.5</td>
<td>15</td>
<td>18</td>
</tr>
<tr>
<td>6, 10</td>
<td>5</td>
<td>1.4</td>
<td>19</td>
<td>33</td>
</tr>
<tr>
<td>15, ..., 34</td>
<td>4</td>
<td>1.3</td>
<td>23</td>
<td>23</td>
</tr>
<tr>
<td>total number</td>
<td></td>
<td></td>
<td>115</td>
<td>107</td>
</tr>
<tr>
<td>total cost</td>
<td></td>
<td></td>
<td>138.8</td>
<td>156.1</td>
</tr>
</tbody>
</table>

Simulations made with [ContactCenter](https://www.contactcenter.org), a Java library based on SSJ.
Conclusion and future work

- We have a simulation-based optimization methodology that can handle large realistic scheduling problems in call centers.
- We know of no other competitive solution method.
- Our approach remains time-consuming, contains heuristics, and is sometimes noisy. We are working on various improvements.

Current and future work:
  - Improvements and refinements of current algorithm.
  - Simultaneous optimization of routing rules.
  - Stochastic optimization with recourse.