International Journal of Advanced Artificial Intelligence Research (IJAAIR)
eISSN: 3087-4270
Volume03 Issue07 | Page No: - 69-81

Received: 27 May 2026, Revised: 23 June 2026, Accepted: 13 July 2026, Published: 31 July 2026

DOI: https://doi.org/10.55640/ijaair-v03i07-07

Dynamic Operator Allocation for Conversational AI Voice-Calling Systems

Nadezhda Shiroglazova [1]
[1]Northwestern University Chicago, USA

ABSTRACT: The article examines the dynamic allocation of outbound call launches in conversational AI voice-calling systems, where voice bots qualify customers before transferring them to human operators. The study aims to reduce customer hold time and operator idleness by replacing fixed-concurrency settings with a receding-horizon integer-programming controller. The relevance of the work stems from the growing use of AI voice agents in outbound contact centers, where stochastic answer behavior, delayed transfers, and scarce operator capacity create a control problem that static dialing rules cannot resolve. The novelty lies in modeling the bot-induced transfer delay as an empirical lag convolution and embedding this representation within a closed-loop optimization framework, validated through discrete-event simulation calibrated against production logs. The main results show that the proposed controller captures most of the throughput from aggressive dialing while reducing abandonment and hold times, and it unlocks unused operator capacity at higher staffing levels. The article will be useful to researchers, contact center engineers, operations managers, and developers of AI voice-calling platforms.

KEYWORDS: outbound call center, predictive dialing, integer programming, receding-horizon control, discrete-event simulation, AI voice agents.

1. Introduction

Outbound calling has changed under the influence of conversational artificial intelligence. In a growing class of contact-center operations, an AI voice bot places outbound calls, conducts an initial qualifying conversation with the customer, and transfers the live call to a human operator only when the customer shows genuine interest, after which the operator completes the high-value part of the interaction. This architecture inverts the economics of the traditional call center, since the scarce and expensive resource embodied in the human operator stays shielded from the large volume of unproductive dials in the form of no-answers, voicemail, early hang-ups, and uninterested parties, and engages at the moment a warm, conversation-ready customer is on the line.

AI voice systems already substitute for traditional interactive voice response in customer service, and field evidence shows a measurable effect on call outcomes (Wang et al., 2023). The architecture creates a new and subtle control problem: the bot can launch many calls at once, up to 20 parallel outbound lines in the system studied here, while each call that eventually transfers to an operator does so only after a variable bot-conversation delay and with low probability. Whether a transferred customer connects to a free operator immediately or ends up on hold depends on how many calls were launched a minute or two earlier, how many of them will transfer, when each finishes its bot conversation, and how many operators are free at that future moment.

The tension described here is operationally real. In the production system that motivated this study, the number of simultaneous launches is set to a static value of 6. Under this setting, operators are idle for much of the time, and the system still occasionally produces holds when several calls transfer in a burst. The static setting cannot protect both objectives at once, because the correct level of concurrency depends on the current number of available operators, the prevailing answer and transfer rates, and the duration of ongoing bot conversations, and

all of these quantities drift over the campaign as operators take breaks, log off, or finish calls of varying length.

The production system that motivates this study operates inside a large telephone-sales company that runs distributed call centers across Europe and Latin America and handles on the order of 10,000 calls per day with operators who work across several languages. The automated segment is repeat sales to a cooled customer base, namely customers who have worked with the company before and who plan no purchase at the moment of the call, which makes the conversation harder than an ordinary inbound lead. The voice bot opens the conversation, gauges interest, and transfers only a warm, conversation-ready customer to an operator, while a second operational layer decides in real time how many calls to launch so that warm customers avoid the line and operators avoid idleness. At a volume of thousands of calls per day, even a small improvement in the filtering of the funnel yields a tangible economic effect, which is the setting in which the launch-pacing problem acquires its weight.

The research question takes the following form. Given a pool of live operators with dynamic availability and a voice bot capable of placing up to 20 parallel outbound calls, how should the number of calls launched in each batch be set and continuously re-set so that transferred customers connect to a free operator with minimal hold time while operator capacity goes unused?

The problem is treated as one of sequential optimization under uncertainty, and the work makes three contributions. The first contribution is a formulation of the per-batch launch decision as an integer program whose objective balances expected customer hold against expected operator idleness, and whose constraints encode the empirical bot-conversation-delay distribution as a lag convolution, the long operator handle time as a cumulative capacity constraint, and the twenty-line ceiling. The second contribution is the embedding of this program in a receding-horizon controller that re-solves at every batch from the latest observed operator state, and that carries a hysteresis deadband and a queue-abort rule for self-regulation without thrashing. The third contribution is a discrete-event simulator calibrated on real production call logs that serves as both the evaluation ground truth and the vehicle for sensitivity analysis, with the calibrated model validated against the real call timeline.

Three hypotheses are tested. Under the first hypothesis, a static concurrency policy is structurally unable to use additional operator capacity, and its throughput saturates regardless of the number of operators added. Under the second hypothesis, a naive policy of always dialing the maximum maximizes throughput at the cost of long hold times and elevated abandonment rates. Under the third hypothesis, a dynamic optimization-based policy can capture nearly all of the throughput available to aggressive dialing while holding mean hold time and abandonment near the levels of the conservative static policy, and thereby dominates both baselines on the hold-throughput frontier.

2. Literature Review

The problem lies at the intersection of three established research streams: queueing-theoretic staffing of call centers, control of predictive dialers for outbound calling, and sequential decision-making under uncertainty via mathematical programming and simulation. The discussion is organized around the key characteristics of the problem and the way the literature treats each of them.

The classical analysis of inbound call centers models the operator pool as a many-server system with abandonment and uses it to set staffing toward a service-level target, while a recent practice-oriented overview of call center workforce planning systematizes the Erlang A and M|M|s+G models, the treatment of customer impatience, and loading regimes close to critical that still deliver short waits (Koole & Li, 2023). The operator subsystem of this work is exactly such a many-server queue with abandonment, and from it the evaluation vocabulary of hold time, abandonment rate, and utilization is borrowed. The crucial difference is that in inbound settings, the arrival stream is exogenous, whereas here it is controlled indirectly through the launch decision and the bot-delay convolution, turning a staffing problem into a pacing-and-control problem.

The outbound literature addresses how aggressively to dial so that operators stay busy without pushing abandonment rates for called parties past regulatory or service limits. Predictive dialing was introduced as an optimization of operator idle time against nuisance calls, and subsequent work models the dialer as a control problem balancing operator occupancy and abandonment. The present setting is a structural mirror of the classical predictive dialer, since over-dialing

there abandons called customers, whereas over-launching here produces transferred customers placed on hold. The bot conversation introduces a feature absent from the classical model in the form of a substantial, stochastic, and heavy-tailed delay between the launch of a call and the readiness of a customer for an operator, and for this reason, a memoryless steady-state dialer formula is insufficient, and an explicit lag distribution is required.

When the service time scale is large relative to the control horizon, fluid and deterministic approximations of queues become accurate and tractable, and applications of fluid models in service operations continue to develop (Hong & Chen, 2025). This view is adopted here, since within a planning horizon shorter than one handle time, each committed operator stays unavailable through the end of the horizon, and capacity is represented as a cumulative quantity. A complementary stream formulates contact-center control as integer programs solved on a rolling basis, and the appeal of receding-horizon control, under which a finite-horizon model is re-optimized as new states arrive and only the immediate action is applied, is documented in the process-control and service-system literature, including formulations for linear systems with uncertainty through integral quadratic constraints (Schwenkel et al., 2023) and computationally efficient robust formulations for nonlinear systems (Köhler et al., 2021). The contribution of this work within that stream is a compact, solver-friendly integer program tailored to the AI-calling lag structure, deployed in closed loop with damping via a deadband and a launch-smoothing term, providing sufficient stability for production use.

The launch decision is made under uncertainty about how many calls will transfer and when, and about operator availability. Two-stage stochastic programming and chance-constrained formulations are the natural rigorous treatments. A two-stage stochastic formulation of call-center staffing with integer recourse and chance constraints on service levels, solved through sample-average approximation and decomposition, demonstrates the feasibility of such an approach under arrival-rate uncertainty (Ta et al., 2021), while modern data-driven approximations of chance-constrained programs and adaptive decomposition methods for two-stage stochastic programs extend this toolkit (Rahimian & Pagnoncelli, 2023; Ramírez-Pico et al., 2023). A deterministic expected-value model with a buffered-capacity surrogate through a safety margin is adopted here, and through simulation and an explicit safety-margin sweep, the work quantifies how much of the target of a fully stochastic model the buffer recovers, which positions the work as a practical middle ground that stays simpler and more transparent than a two-stage stochastic program while remaining robust.

Relative to the state of the art, the investigation is distinctive in three respects. First, it explicitly models the bot-induced transfer lag as an empirical convolution kernel that drives the optimization model. Second, it couples a small integer program with a receding-horizon controller and anti-thrashing damping designed for the asynchronous launch-then-transfer pipeline. Third, it is calibrated and validated against real production voice-bot logs, whereas earlier formulations rely solely on synthetic assumptions. Where the classical predictive-dialer literature optimizes operator occupancy against called-party abandonment under memoryless assumptions, this work optimizes transfer hold against operator idleness under an empirically estimated heavy-tailed delay.

3. Methodology

The objective is to choose, at each batch launch, the integer number of outbound calls that minimizes a weighted combination of expected customer hold time at transfer and expected idle operator capacity, subject to the pipeline's physical and distributional constraints, and to do so repeatedly in a closed loop as the system state evolves. The modeling pipeline has five components: the estimation of empirical parameters and distributions from real call logs; a calibrated data generator; the per-batch integer program; a receding-horizon controller that wraps that program; and a discrete-event simulator for evaluation and sensitivity analysis.

The empirical parameters are estimated from production logs of a deployed voice agent and cover 491 calls across several test dates, of which 479 remain after excluding web test calls. The voice-agent platform is a commercial voice platform headquartered in the United States. A methodological subtlety arises from the telephony stack: the bot may begin speaking before the called party picks up, and call-status fields of the dialed-no-answer type turn out unreliable. A call counts as answered only when the customer produces genuine speech, consisting of at least 2 transcript utterances with real words, after the removal of the platform's inaudible-speech placeholder tokens.

Under this definition, the calibrated parameters are obtained. The answer rate per dial is 0.317, the transfer rate given an answer is 0.217, the transfer rate per dial is 0.069, and the mean bot-conversation duration on transferring calls is 167.5 s with a median of 164.7 s, a standard deviation of 169.4 s, and a range from 10 to 830 s. The most consequential empirical fact is that the bot-conversation duration on transferring calls is heavily right-skewed, with a standard deviation roughly equal to the mean and a maximum nearly five times the median, which rules out a single mean value for the launch-to-transfer delay and motivates carrying the full empirical distribution into the optimization model as a lag kernel.

The empirical parameters above are drawn from a single agent design, and the production history of that agent supplies a second piece of context for the metrics. The agent first ran on one large prompt, an arrangement that overloaded the context, complicated control of the scenario, and blocked targeted changes to individual parts of the dialogue. A move to a multi-block architecture followed, under which separate blocks handle greeting, qualification, interest and problem discovery, objection handling, the transfer decision, and the handoff to a live operator, with state passed between blocks through compact variables instead of the full dialogue history.

A direct comparison of the single-prompt and multi-block agents on real calls during the test period serves as production A/B context here, not as a controlled experiment, since the two designs were not run under matched conditions. The multi-block design holds the customer in conversation nearly three times as long and sounds warmer, at the cost of a higher response latency on the order of 0.4 s added to the median, which the team accepted as a deliberate engineering trade-off, since empathy and a genuinely multi-step conversation do not fit into one prompt. These figures describe the production behavior of the deployed agent that generated the call logs and serve as context for the optimization model, and they are reported without treatment as verified model-quality metrics.

From the transfer durations, a discrete lag probability mass function is constructed that describes the probability that a transferring call becomes ready a given number of periods after launch, and from all dial durations, a line-occupancy survival function is constructed. Two parameters are unobservable from the bot logs and are treated as explicitly stated synthetic assumptions. The operator handle time is set to a right-bounded lognormal distribution with a mean of 12.5 min and a range from 8 to 18 min, which reflects the transfer of interested customers only, and the number of operators on shift, equal to two in production, is treated as a decision and scenario variable swept from 2 to 20, since it is the principal lever of management. Customer patience is set to a 60 s abandonment threshold, and the campaign is set to a four-hour session.

Time is discretized into 15-s periods, chosen to match the resolution of the bot-duration data, while the optimization remains small. Batches are launched every 30 s, namely once per two periods. The planning horizon is 60 periods, namely 15 min, and is set to exceed one operator handle time so that the model accounts for the way that committing an operator to a transfer removes that operator's capacity through the end of the horizon. The hard concurrency ceiling is 20 lines.

At a given decision epoch, the controller plans launches at the batch epochs within the horizon, and the decision variables are the integer numbers of calls launched at each epoch, bounded by the line ceiling, while only the immediate action is applied, since the rest is re-planned at the next epoch. The expected number of transfer arrivals in a period is the convolution of past launches with the lag kernel plus a term from calls already in flight at the start of the solve. The number of occupied lines in a period is the sum of residual occupancy and the contribution of past launches through the occupancy survival function, and this sum does not exceed the line ceiling, which enforces the twenty-line constraint at all moments.

Because the operator handle time exceeds the horizon, operator capacity is treated as cumulative service slots. The number of operator slots available for a period combines operators who are free now, those who finish a prior call by that period, and those who join the shift, and it is reduced by a safety-margin buffer. With the number of already waiting customers, cumulative arrivals, and a nonnegative nondecreasing cumulative-served variable are defined, bounded above by cumulative arrivals and available capacity, and the number of held customers and the number of idle operator slots follow as the corresponding nonnegative differences, and the program minimizes their weighted sum over the horizon. The result is a small mixed-integer linear program with a dozen integer launch variables and several dozen

continuous service variables, solved with PuLP and the CBC solver, where the ratio of hold and idle weights reflects the managerial preference between protecting customers and using operators.

The integer program is embedded in a closed loop. At each batch epoch, the controller reads the live state composed of the number of free operators, the queue length, the ages of in-flight calls, and the remaining busy time of each occupied operator, constructs from that state the initial arrival, line-occupancy, and available-capacity quantities, solves the program, and applies the immediate action. Two mechanisms provide self-regulation without overreaction. The first mechanism is a hysteresis deadband, under which the applied launch level changes only when the newly optimized target differs from the current level by more than a threshold, which prevents minute-to-minute thrashing. The second mechanism is a queue-abort rule, under which the batch is suppressed, and zero calls are launched when the live queue exceeds a threshold, throttling inflow during a transient overload.

Together, these mechanisms realize the practical intent of a system that raises concurrency when operators are free, lowers it when a customer queue builds, and changes its setting only when a meaningful threshold is crossed. Because the controller resolves frequently and many live states recur in steady operation, solutions are memoized using a discretized state signature, reducing repeated solver invocations to cache lookups without changing policy.

Model behavior is evaluated using a period-stepped discrete-event simulator that simulates the full pipeline call by call. Each launched dial is independently answered and transferred according to the calibrated probabilities, draws its bot-conversation duration, and for non-transferring calls, a line-occupancy duration, by bootstrap from the empirical distributions, occupies a line accordingly, and on transfer joins a first-in-first-out queue, is served by the first free operator over a drawn handle time, or abandons if its wait exceeds customer patience. The simulator records the hold-time distribution, the fraction connected immediately, the abandonment rate, operator utilization, and throughput, and it is policy-agnostic, so the static baselines and the optimizer are evaluated on identical realized demand. Discrete-event simulation is an established tool for systems of this kind with heavy-tailed service times (Werner, 2021).

Verification rests on component self-checks: since the data generator reproduces the target answer and transfer rates on large samples, the integer program returns optimal solutions whose hold and idle accounting match an independent recomputation, and a trace confirms the invariant that no operator idles while a customer waits. Validation is twofold. A calibration check confirms that the synthetic world reproduces the statistics of the real sample, including the answer rate, transfer rate, and bot-duration distribution, via a two-sample Kolmogorov-Smirnov test. A real-demand replay takes the actual sequence of transfer-ready moments from the real call logs as the sum of the start time of each transferring call and its bot-conversation duration, and runs this real arrival stream through the operator model at varying operator counts, estimating on genuine demand the way hold time and abandonment respond to staffing.

Policies are compared on throughput (customers served per campaign), mean and 90th-percentile hold times at transfer, the fraction connected immediately, the abandonment rate, and operator utilization. The reported values are means over independent Monte Carlo replications with 95% confidence intervals, and the optimizer is compared with the current static policy that launches toward six lines and with an aggressive reference that launches toward a maximum of twenty lines.

4. Results

All experiments run the discrete-event simulator on the synthetic world calibrated to the real parameters of Section 3.1, and each reported value is a mean over 20 independent Monte Carlo replications of a four-hour campaign. Three policies are compared on identical realized demand. The first policy is baseline-6, the current production policy that caps concurrency at 6 lines. The second policy is baseline-20, an aggressive reference that always dials to the ceiling of twenty lines. The third policy is the optimizer, a controller based on a receding-horizon integer program with a 60-period horizon, re-optimization every 2 min, a hold-to-idle weight ratio of 5 to 1, a deadband of 2 lines, and a queue-abort at 5 waiting customers without a safety buffer, unless stated otherwise. The number of operators is swept from 2 to 20.

The two-layer system, in which the conversational layer transfers warm customers into the queue and

onward to operators, and the control layer reads the live state and resets the launch rate, is shown in Fig. 1.

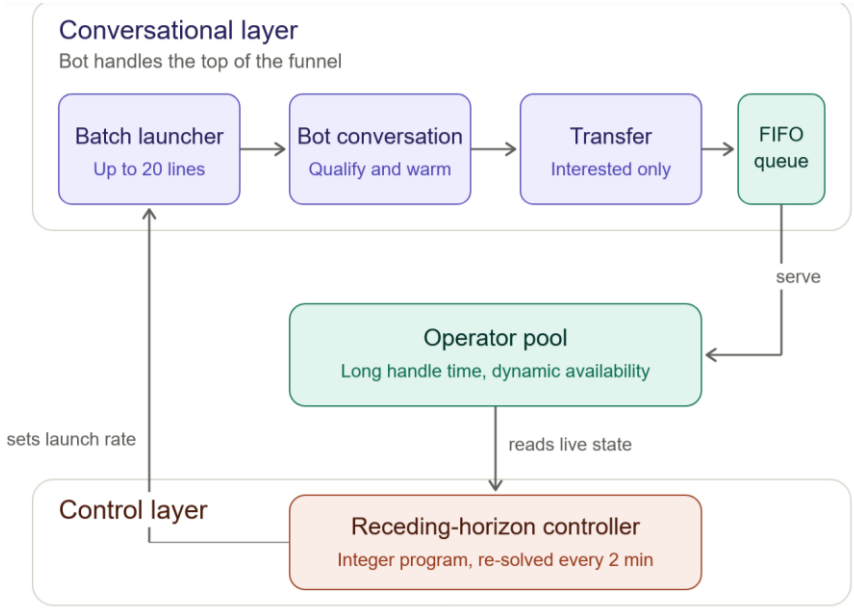

Fig. 1. Two-layer architecture of the AI voice calling system

Before the policies are compared, the simulated world is confirmed to reproduce production reality. The synthetic answer rate is 0.316 against 0.317, the transfer rate per dial is 0.071 against 0.069, the mean bot duration matches at 167.5 s, and a two-sample Kolmogorov-Smirnov test of the synthetic against the real distribution returns a p-value of 1.00, which indicates no detectable difference. The calibration check is summarized in Table 1.

Table 1. Calibration check of synthetic and real data

Quantity	Real, 491 calls	Synthetic
Answer rate	0.317	0.316
Transfer rate per dial	0.069	0.071
Mean bot duration, s	167.5	167.5
Median bot duration, s	164.7	164.7
Duration distribution, KS p	N/A	1.00

The comparison of the three policies across operator counts is collected in Table 2 and reveals three results that confirm the stated hypotheses.

Table 2. Policy comparison across operator counts means over 20 replications

Ops	Policy	Served	Util.	Mean hold, s	p90 hold, s	Abandon	Immed.
2	baseline-6	27.8	0.82	9.2	36.9	0.59	0.77
2	baseline-20	33.2	0.97	23.7	58.7	0.85	0.41
2	optimizer	28.6	0.86	3.7	12.2	0.60	0.90
4	baseline-6	47.9	0.72	5.2	20.8	0.31	0.86
4	baseline-20	63.8	0.96	21.7	58.9	0.72	0.47
4	optimizer	58.0	0.87	6.3	26.7	0.44	0.83
6	baseline-6	59.8	0.61	2.7	4.4	0.15	0.93
6	baseline-20	94.7	0.95	19.0	58.4	0.59	0.54
6	optimizer	86.7	0.87	7.3	35.3	0.35	0.82
8	baseline-6	65.5	0.50	1.3	0.0	0.06	0.97
8	baseline-20	122.8	0.93	16.2	55.3	0.47	0.59
8	optimizer	113.0	0.86	6.8	32.6	0.25	0.82
12	baseline-6	67.4	0.34	0.5	0.0	0.03	0.99
12	baseline-20	170.3	0.86	9.4	42.8	0.26	0.75
12	optimizer	163.1	0.83	4.7	17.7	0.13	0.87
16	baseline-6	68.0	0.25	0.2	0.0	0.02	0.99
16	baseline-20	201.7	0.77	3.9	12.2	0.12	0.89
16	optimizer	197.9	0.75	2.8	3.8	0.07	0.92
20	baseline-6	68.3	0.21	0.1	0.0	0.02	1.00
20	baseline-20	214.0	0.65	2.0	0.0	0.07	0.94
20	optimizer	212.0	0.65	1.1	0.0	0.04	0.97

The throughput saturation of the static policy confirms the first hypothesis. The throughput of baseline-6 plateaus near 68 customers per campaign from eight operators onward and shows no further increase, standing at 65.5, 67.4, 68.0, and 68.3 at 8, 12, 16, and 20 operators, respectively. Because this policy occupies no more than six lines, additional operators are idle, and utilization falls from 0.50 at eight operators to 0.21 at twenty, namely about four of every five operator-hours wasted. This is the production pathology that motivated the study: under which 6 is set, operators stay free most of the time, and the low hold and abandonment rates are symptoms of chronic underuse.

The cost of naive aggression confirms the second hypothesis. Baseline-20 attains the highest throughput at every staffing level, up to 214 per campaign, at a punishing service cost whenever operators are scarce, since 47% to 85% of warm, ready-to-buy customers abandon before an operator answers at operator counts from 2 to 8, with mean holds from 16 to 24 s. Maximizing dials creates more opportunities while destroying the very leads the architecture exists to protect, and only once operators become abundant at 16 and 20 does abandonment fall into single digits.

The dominance of the dynamic policy confirms the third hypothesis. The optimizer occupies the efficient frontier the baselines cannot reach, since it captures 92% to 99% of the throughput of baseline-20, from 92% at eight operators to 99% at twenty, roughly halving abandonment at every staffing level, for example 0.25 against 0.47 at eight operators and 0.13 against 0.26 at twelve, and cutting mean hold by 50% to 58%, for example 6.8 s against 16.2 s at eight operators. Against the current policy, the gain is large where baseline-6 stalls, since at eight operators the optimizer serves 73% more customers (113 against 65), and at twelve operators 142% more (163 against 67), converting idle operator capacity into served customers. The controller achieves this by adapting the launch rate to live capacity, gently when operators are scarce and aggressively when they are plentiful, so that among the three policies, it alone sits near the throughput ceiling while its abandonment stays far below that of aggressive dialing.

The placement of the three policies in the coordinates of abandonment and throughput as the number of operators varies from 2 to 20 is shown in Fig. 2, where the optimizer sits on the frontier at the upper left.

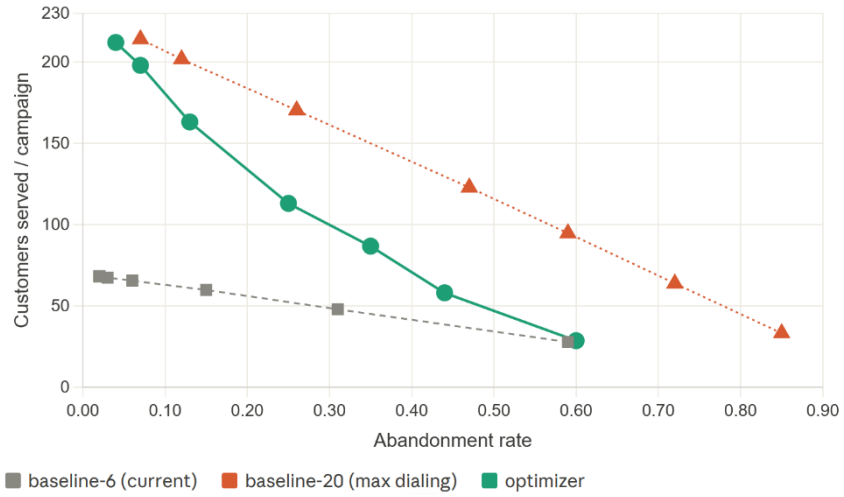

Fig. 2. Throughput and service-quality frontier of the three policies

The dependence of the number of served customers on the number of operators for the three policies is shown in Fig. 3, where the baseline-6 curve saturates while baseline-20 and the optimizer rise almost together.

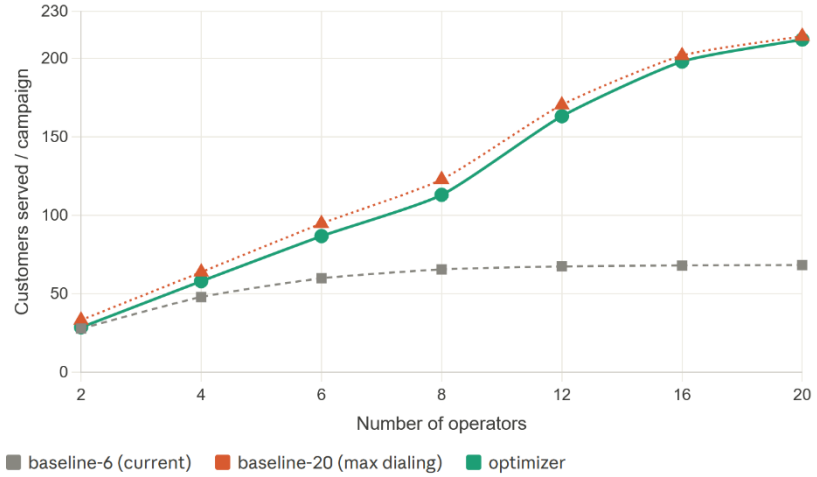

Fig. 3. Throughput saturation by operator count

4.1. Sensitivity analysis

The sweep of the operator safety margin at eight operators is collected in Table 3 and shows that the margin is an effective service-level lever.

Table 3. Safety-margin sensitivity, 8 operators, optimizer

Margin	Served	Util.	Mean hold, s	p90 hold, s	Abandon
0.0	113.0	0.86	6.8	32.6	0.25
0.5	112.6	0.85	6.4	30.6	0.24
1.0	106.6	0.81	3.8	9.7	0.16
1.5	102.7	0.78	2.8	0.6	0.14
2.0	96.9	0.74	1.9	0.0	0.12

Raising the margin from 0 to 2 operators lowers the mean hold from 6.8 s to 1.9 s and abandonment from 25% to 12% at a throughput cost of about 14%, from 113 to 97 served. Management can drive the controller to any point along this hold-throughput curve toward a service target, and a buffer of one operator is a balanced choice at an abandonment of 16%, a hold of 3.8 s, and 107 served. This confirms that the deterministic model with a buffered capacity estimate behaves as a controllable surrogate of the underlying stochastic problem, recovering much of what an explicit risk constraint would target while remaining a simple linear model.

The effect of scaling the dispersion of the launch-to-transfer delay over a range from half to twice the empirical value is collected in Table 4.

Table 4. Bot-duration-variance robustness, 8 operators

Variance scale	Policy	Served	Util.	Mean hold, s	Abandon
0.5×	optimizer	111.2	0.85	6.0	0.28
1.0×	optimizer	113.0	0.86	6.8	0.25
1.5×	optimizer	112.8	0.86	6.3	0.24

2.0×	optimizer	112.6	0.86	6.3	0.23
1.0×	baseline-6	65.5	0.50	1.3	0.06

Scaling the dispersion leaves the optimizer with no material change, since throughput holds within 111 to 113 served, mean hold near 6s, and abandonment in the band from 23% to 28%, while baseline-6 stays pinned near 63 to 66 across the range. The controller is robust to the heavy right tail of bot durations because it accounts for the full empirical lag kernel, whereas a point estimate provides no such robustness, and this is the single most important modeling choice for this problem.

The effect of sweeping the hold-to-idle weight ratio from 1 to 20 is collected in Table 5.

Table 5. Objective-weight-ratio sensitivity, optimizer

Ops	Weights	Served	Util.	Mean hold, s	Abandon
8	1	115.8	0.88	8.2	0.31
8	2	115.6	0.88	7.9	0.29
8	5	113.0	0.86	6.8	0.25
8	10	112.9	0.86	6.3	0.26
8	20	110.8	0.84	5.1	0.22

At eight operators, the mean hold falls from 8.2 s to 5.1 s and abandonment from 31% to 22% as the weight rises, at a small throughput cost of 5 served. The effect concentrates on the step from 1 to 2, since penalizing hold at all is what matters, and it saturates thereafter, because beyond a point the binding constraint becomes operator capacity, whereas the hold-idle trade-off stops binding. The weight ratio is a secondary tuning knob, while the safety margin in Table 3 is the stronger, more interpretable lever for setting a service level. A weight ratio of 5:1 is adopted as the balanced default.

The effect of the anti-thrashing deadband as it grows from 0 to 5 lines is collected in Table 6.

Table 6. Deadband and control stability, 8 operators, optimizer

Deadband, lines	Churn	Served	Util.	Mean hold, s	Abandon
0	3.10	113.0	0.86	6.5	0.25
1	3.10	113.0	0.86	6.5	0.25
2	3.09	113.0	0.86	6.8	0.25
3	3.09	113.0	0.86	6.8	0.25
5	3.01	113.6	0.87	7.2	0.26

Increasing the deadband changes outcomes by a small amount, since control churn falls only from 3.10 to 3.01 mean launch changes per decision, while throughput, hold, and abandonment stay flat. The practical reading is that in this regime, the controller's stability comes mainly from resolving on a 2-minute cadence, while the

deadband itself rarely binds, since the optimal target shifts by more than a few lines as the stochastic queue and the free-operator state evolve. The deadband is a cheap safety valve that can remain small at no cost.

The supply of the actual stream of transfer-ready moments from the real call logs, through the operator model at varying operator counts, is presented in Table 7.

Table 7. Real-demand replay across operator counts, 3 real test days

Operators	Abandon	Mean hold, s	p90 hold, s	Immed.
1	0.60	2.8	7.4	0.90
2 (current)	0.37	2.2	6.8	0.93
3	0.16	1.4	3.7	0.95
4	0.06	0.3	0.3	0.99
6	0.00	0.0	0.0	1.00

On the 3 substantive test days, with about nine transfers per day, the current two-operator configuration abandons about 37% of ready customers during bursts, and about four operators are needed to drive abandonment near zero. This staffing conclusion holds independently of the pacing policy and confirms, on genuine demand, the capacity-scarcity regime visible in the synthetic headline experiment.

5. Discussion

The central result is that the launch-concurrency decision in an AI voice-bot calling system is a control signal to be reset continuously against live capacity, whereas a once-tuned constant fails in that role. The two static policies expose the two horns of the conflict between customer experience and operator productivity: a low fixed setting keeps hold short only by capping throughput far below the pool's potential, while maximal dialing realizes that potential only by forcing a large fraction of warm leads onto hold and into abandonment. The dynamic controller is a dominant alternative on the frontier because it uses information that static policies ignore: the number of operators free now, the number of calls mid-conversation, and the moment at which, under the empirical bot-delay distribution, those calls become transfers.

A second managerially important pattern is the regime boundary between staffing-limited and pacing-limited operation. Below about four operators, the system is staffing-limited, since every policy abandons a large fraction of ready customers because operators are too few to serve the warm leads that even a handful of transfers create, and at two operators, even the optimizer abandons about 60%, which the real-demand replay confirms on genuine data through the loss of about 37% of ready customers in bursts. At about eight operators and above, the system becomes pacing-limited, and here the controller unlocks throughput that the static six-line policy is structurally unable to reach, serving 73% more at eight operators and 142% more at twelve without the holds that naive aggression would create. The practical recommendation is twofold: first, staffing must be brought above the scarcity threshold for the expected transfer volume, and then the dynamic controller is deployed with a margin of about one operator when a stricter abandonment target is required.

Several limitations bound these conclusions. First, the controller uses a deterministic expected-value model with a buffered-capacity surrogate, which is an effective and tunable approximation that provides no formal probabilistic service guarantee. The residual abandonment of about 25% at eight operators without a buffer reflects burst arrivals that the expected-value model only partly anticipates. Second, the operator handle time is a stated synthetic assumption in the form of a bounded lognormal with a 12.5 min mean, since it is unobservable from the bot logs, and the absolute numbers would shift under the true distribution while the comparative conclusions stay robust in direction. Third, the model treats operators as a single homogeneous single-skill pool with first-in-first-out service and a

constant-patience abandonment rule, setting aside skill-based routing, callbacks, and heterogeneous patience. Fourth, the real sample is modest at 491 calls and 33 transfers across a few test days, so the empirical distributions carry sampling uncertainty, and the real-demand replay reflects only 3 substantive days.

Three extensions follow naturally. The deterministic core can be promoted to a two-stage stochastic or chance-constrained program that bounds the probability of any customer waiting, for which the present safety-margin policy would serve as a warm start and a benchmark. The operator model can be enriched to support multi-class skill-based routing, so that the launch decision accounts for which operators can serve which campaigns. Because the transfer-lag kernel and the answer and transfer probabilities are the most influential inputs, an online-learning layer that re-estimates them from streaming call outcomes and feeds updated parameters into the receding-horizon solve would let the controller adapt as scripts, telephony, and customer mixes change.

6. Conclusion

The pacing-concurrency decision in an AI voice-bot calling system was modeled as a receding-horizon integer program that balances customer hold against operator idleness under an empirically estimated heavy-tailed transfer delay, and it was evaluated in a discrete-event simulator calibrated on real production logs and validated against the real call timeline. The dynamic controller dominates the current fixed six-line policy and aggressive maximal dialing on the throughput and service-quality frontier, serving up to 73% more customers than current practice at eight operators and far more as staffing grows, while roughly halving the abandonment of maximal dialing, all from a small, transparent, solver-friendly model fit for live deployment. The message for the practitioner is concrete: staffing must first be brought above the scarcity threshold, and the decision on the number of calls to launch is then entrusted to an optimization-based controller, whereas a static number cannot make that decision.

References

1. Hong, G.-Y., & Chen, X.-Y. (2025). Simulation optimization for queues with heavy-tailed service times. *Journal of the Operations Research Society of China, 13*(3), 810–836. https://doi.org/10.1007/s40305-024-00562-z

2. Köhler, J., Soloperto, R., Müller, M. A., & Allgöwer, F. (2021). A computationally efficient robust model predictive control framework for uncertain nonlinear systems. *IEEE Transactions on Automatic Control, 66*(2), 794–801. https://doi.org/10.1109/TAC.2020.2982585

3. Koole, G. M., & Li, S. (2023). A practice-oriented overview of call center workforce planning. *Stochastic Systems, 13*(4), 479–495. https://doi.org/10.1287/stsy.2021.0008

4. Rahimian, H., & Pagnoncelli, B. (2023). Data-driven approximation of contextual chance-constrained stochastic programs. *SIAM Journal on Optimization, 33*(3), 2248–2274. https://doi.org/10.1137/22M1528045

5. Ramírez-Pico, C., Ljubić, I., & Moreno, E. (2023). Benders adaptive-cuts method for two-stage stochastic programs. *Transportation Science, 57*(5), 1252–1275. https://doi.org/10.1287/trsc.2022.0073

6. Schwenkel, L., Köhler, J., Müller, M. A., & Allgöwer, F. (2023). Model predictive control for linear uncertain systems using integral quadratic constraints. *IEEE Transactions on Automatic Control, 68*(1), 355–368. https://doi.org/10.1109/TAC.2022.3171410

7. Ta, T. A., Chan, W., Bastin, F., & L'Ecuyer, P. (2021). A simulation-based decomposition approach for two-stage staffing optimization in call centers under arrival rate uncertainty. *European Journal of Operational Research, 293*(3), 966–979. https://doi.org/10.1016/j.ejor.2020.12.049

8. Wang, L., Huang, N., Hong, Y., Liu, L., Guo, X., & Chen, G. (2023). Voice-based AI in call center customer service: A natural field experiment. *Production and Operations Management, 32*(4), 1002–1018. https://doi.org/10.1111/poms.13953

9. Werner, C. (2021). Assessing and modelling tail dependencies between service times in discrete event simulation with minimum information copulas for a better understanding of maximum time in system risk. *Frontiers in Applied Mathematics and Statistics, 7*, Article 641245. https://doi.org/10.3389/fams.2021.641245