Workforce Planning Policy Optimization with Trajectory Transformers

Jimmy Rigby
People Experience and Technology
rigbjame@amazon.com

Nooshin Yousefi
Amazon Sponsor Brands
nyousefi@amazon.com

Peter Zhu
People Experience and Technology
zhilongb@amazon.com

Jeremy Curuksu
Amazon Web Services
curukj@amazon.com

Abstract

While machine learning is now increasingly used for workforce planning at Amazon to produce talent forecasts, identifying an optimal talent strategy based on these forecasts still relies exclusively on human expertise. We propose a reinforcement learning method for workforce planning which can explore talent scenarios offline and identify optimal talent strategies, with a customizable agent's objective defined by the user (human-in-the-loop). We gathered a dataset covering nearly a decade of historical workforce data at Amazon and show that the workforce policy learned by the agent systematically outperforms a behavioral cloning baseline during planning on held-out data. Compared to the baseline which more naively infers policy to imitate historical human planning methods, the RL policy achieves between 2.8% to 12.3% higher rewards, with an improvement of 8.0% on average across all model-based evaluations. This corresponds to over $26 million in potential cost saving annually across the Amazon world-wide talent population, as a result of headcount plans that more closely align with headcount targets.

1 Introduction

Workforce planning (*a.k.a.* talent planning) at Amazon coordinates the acquisition and management of talents to achieve the desired staffing levels over time and across employee segments. Talent planners typically work backward from individual team leader's headcount targets to create plans defining how to bring employees in (promotions in, hires, transfers in) and move employees out (promotions out, transfers out, attrition) of employee segments. High quality i.e., optimal talent plans and strategies allow Amazon to achieve the desired staffing levels enabling the business to consistently operate while maintaining low cost.

In this paper, we propose an original reinforcement learning (RL) method with human-in-the-loop for workforce planning at Amazon, which can autonomously explore talent scenarios offline, identify optimal talent strategies, and recommend specific talent levers. The exact objective of the RL agent encoded in the reward function is customizable by the user's preferences. Thereby, the user controls the nature of the talent strategies and talent segments explored by the agent. The proposed method also implements some recent Transformer generative models to simulate the Amazon workforce offline,

which gives total control to humans, talent planners and business leaders, both upstream *and* downstream of the talent policy optimization process.

1.1 Current Workforce Planning Method at Amazon and Limitations

Today, workforce planning at Amazon combines knowledge on business targets with forecasts of talent movements, which are often done using linear extrapolation (e.g., 3-month trailing rates) or machine learning predictive models, to manually define workforce strategies to meet these targets. By "manually", we mean using deductive reasoning and expert intuitions. In other words, the actual identification of talent strategies remains entirely on the shoulders of human planners and decision makers. Let us immediately clarify our aim in this paper is not to replace humans in this decision process, but to assist them with recommendations derived from machine learning. In addition to a talent planners', financial planners, talent specialists, and business leaders, talent planners also monitor the incremental growth (month-to-month) of financial headcount targets, adjust their forecast, and quickly react to unexpected events by adjusting talent plans on-the-fly, and propose strategic levers. These levers include strategies such as transferring employees to areas of the business where there is growth, laying off portions of the workforce, or waiting for natural attrition to correct headcount overage. Concurrent to these processes, promotion levels are being determined through merit-based processes and business needs. All together, these processes provide a constantly evolving policy to coordinate hiring, transfers, attrition, and promotions across Amazon employee segments.

To above planning method currently used at Amazon has at least three primary limitations. First, non-standardized processes and evaluation criteria make measuring plan quality and improving the planning process difficult. Planners require a solution that standardizes the contextual factors considered during planning and the plans' objectives. Second, the current process relies heavily on human input at each step in the planning process, which limits the processes scalability. Planners need a planning solution that scales across all levels of analysis. Finally, there are no in-built mechanisms to balance short-term and long-term goals. Planners need a solution with inbuilt mechanisms that systematically balance short-term and long-term tradeoffs. A solution that meets these requirements will enable workforce plans' ability to achieve desired staffing levels while controlling cost.

1.2 Workforce Planning Policy Optimization with Trajectory Transformers

Optimizing talent policies using offline reinforcement learning can help address the limitations described above. Offline RL [1, 2] implements an agent exploring its environment and receiving feedback from it to learn strategies (sequences of actions called "policies") that fulfil a predefined goal, as in online RL, but in offline RL the environment is simulated using models derived from some pre-existing dataset. A model, in the RL terminology, is a predictive function that takes the agent state and action as input, and output its next state (e.g., transition function in a Markov Decision Process) and/or reward. Thereby, for workforce planning, all we need to optimize talent policies by offline RL, is to collect past talent planning experiences. A model can then be trained using supervised learning and *queried* to simulate experience beyond what was sampled in the original dataset. Using a RL algorithm, the agent can then identify good and bad actions learned from past experiences, generalizes good actions to new settings, and identifies compositions of behaviors that work well together [2].

Offline RL is especially well-suited for addressing customer problem's related to workforce planning. First, offline RL formalizes the talent planning problem as an MDP, standardizing the business context considered during planning, the planning objectives, and defining mechanisms that balance short- and long-term objectives. Second, the agent can explore, in the service of the customers, a much larger number (up to millions) of scenarios to identify promising scenarios given their objectives. This helps planners scale their service across Amazon. The agent's exploration poses minimal risk to Amazon because plans are explored offline and talent planners and business leaders retain full control over both the goal of the agent and whether to implement the recommended strategy or not. Finally, Amazon has nearly a decade of experiences sampled by human experts (Amazon's workforce planners) providing one of the largest talent planning dataset in the world. This provides RL agents with a wealth of information to learn from.

In this paper we first describe the offline RL method used for talent policy optimization and to assess performance. We formalize the policy optimization problem as a MDP problem, and train a trajectory transformer [3] on the Amazon global workforce dataset as this model is currently considered a best

performer offline RL model [2]. The goal of the agent is to recommend cost efficient workforce plans that achieve desired staffing levels. To evaluate its performance, we implement offline policy evaluation (OPE) using a behavioral cloning algorithm as baseline which naively mimics the behavior of human experts in a sample of the pre-existing dataset. Finally, we present our key results. The offline RL agent improved upon the naïve planner's policy in 37 of 37 sampled segments. Compared to the baseline, the RL policy achieves between 2.82% to 12.33% higher rewards, with an improvement of 7.99% on average across all model-based evaluations, corresponding to a $26 Million in cost saving annually across the Amazon world-wide talent population. Nearly all savings were driven by delivering talent plans that reduced cost associated with understaffing and overstaffing.

2 Methodology

2.1 Workforce Planning as a Markov Decision Process

A MDP is a mathematical formalization of a dynamic system. Here, the Amazon workforce dynamics is modeled as a MDP by defining the workforce planners' objectives (reward function), the actions that workforce planners can use to meet their objectives (actions), and the information that the planner has access to, every month, to decide which action to take (state). Figure 1 provides an illustration of how the different components of the MDP offline RL system (defined individually below) interact.

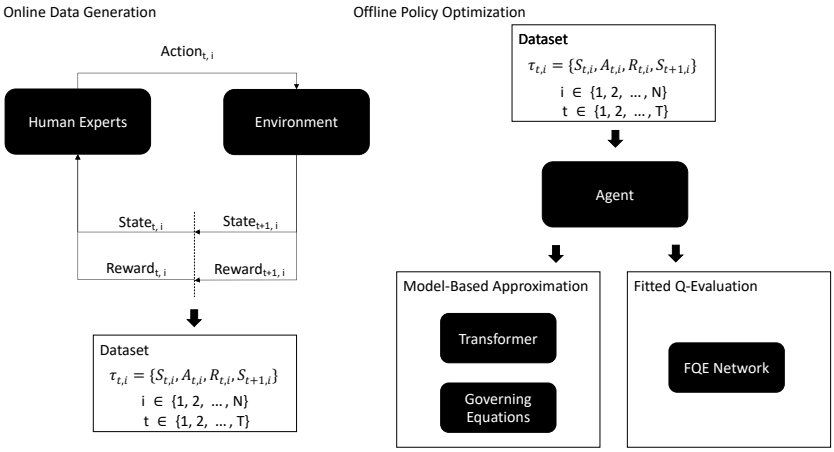

Figure 1. Workforce planning as a MDP offline RL problem. The left pane illustrates the online data generation process where human experts interact with the real world to produce a dataset of trajectories across time and segments. The right pane shows offline policy optimization, where this dataset is used to train an agent that aims to improve or replicate the human expert's policy. The quality of this agent is then evaluated using off-policy evaluation (OPE) methods such as Model-Based Evaluation and Fitted-Q Evaluation described below.

2.2 Action Space

The action space defines the strategic talent levers which the agent can pull to optimize rewards and achieve its goals. In this paper, the agent can take four actions for each segment including: opening external job postings, promoting employees out of their level, terminating employees, and creating transfers-in request.

2.3 Reward Function

The reward function quantifies the goals of the agent, numerically. A fundamental need of talent planners and business leaders at Amazon is to achieve a desired staffing level (user-defined headcount target) while minimizing the cost of talent strategy. At a high level, this is defined by:

$$\boldsymbol{r_{ti} = -f_i(d_{ti} - s_{ti}) - v_i(A_{ti}, O_{ti})}$$

where $\boldsymbol{f_i(d, s)}$ is the cost of the gap for a talent segment *i*, $\boldsymbol{d_t(A, O)}$ is the targeted headcount, $\boldsymbol{s_t}$ is the actual headcount, and $\boldsymbol{v_i}$ is the cost of executing the recommended actions ***(A)*** and resulting states (***O***). *i* indexes employee segments and *t* indexes time. Appendix 1 provides more details about how the cost associated with talent gap and the cost associated with workforce plans were operationalized. This reward specification allows us to express the value of policies in dollars.

2.4 State Space

As the agent makes decisions about each segment, it has access to information about the segment's attributes (i.e., job level), funnel metrics into the segment (i.e., requisitions), talent movements, supply and demand, the number of promotion-ready employees, and finally some information on supply and demand about other segments that could be transfer partners and promotion partners. Table 1 provides a complete list of the state information available to the agent each month.

Table 1. Description of state information available to the agent during planning.

Grouping	State Dimensions
Historical Actions	Requisitions, Promotions Out, URA, Transfer-In Requests
Non-Actionable Talent Movements	Regretted Attrition, Transfer Outs, Hiring Volumes, Promotions In
Funnel Metrics	Internal Applicant Volumes, External Applicant Volumes, On-Sites, Offers, and Pending Starts
Segment Identifiers	Job Level, Tech Indicator, Leader, and Corporate Flag
Headcount	Desired and actual headcount
Forward looking Slate	Employees ready for promotion

2.5 Offline Simulation of Amazon Workforce using Trajectory Transformer

We use an adapted version of the trajectory transformer method [3] to produce talent plans that maximize future reward under the MDP introduced above. The trajectory transformer [3] is an offline RL method that treats planning as a sequence-to-sequence modeling problem, where every dimension of states, actions, and rewards, in the pre-existing dataset represent a step in one global sequence (referred to as *quantization of information* in [3]). A Transformer model is then trained to predict the next step in the sequence based on prior states, actions, and rewards. As in the original Transformer architecture [4], the trajectory transformer can use sections of observed trajectories and thus scale to arbitrarily large datasets. Figure 2 illustrates how the trajectory transformer operates.

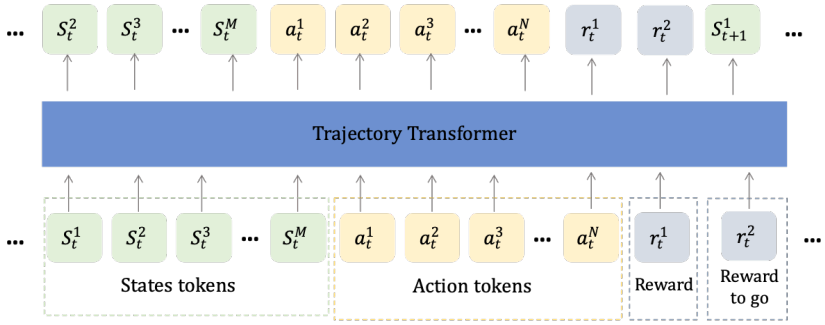

Figure 2. The trajectory transformer uses observed trajectories to predict future states, actions, and rewards

Since the agent is planning for a large number of segments, we trained a global trajectory transformer similar to a global timeseries models [5, 6]. The global model learns from training examples spanning multiple employee segments. By learning across multiple segments simultaneously the global trajectory transformer (1) allows the model to transfer knowledge about system dynamics across employee segments and (2) reduces the agent's training time and operation burden enabling the agent to scale across an arbitrarily large workforce.

The trajectory Transformer model can be queried both to predict next states and rewards, and to plan actions for policy optimization. Here it was used to improve upon the behavioral policy by, first, generating future trajectories, constrained to the principles underlying the governing equations (see below), using beam search (same as in [3]). After generating a set of candidate trajectories, planning is done by selecting the action with the highest sum of reward and "*reward to go*" (see [3] for more details). By first sampling likely actions under the behavioral policy and then selecting the highest reward outcome, the trajectory transformer can recommend actions that do not fall too far beyond the support of the sampled historical data [2].

2.6 Constrained System Dynamics using Governing Equations

In addition to using a global model, we constrained the trajectory transformer to follow three known principles that underly Amazon's talent dynamics. This constrains the trajectory transformers' stochasticity using known physical principles. Not only do they help constrain stochasticity, but the governing equations introduce interdependence across the employee segments. Interdependence allows the agent to make plans in consideration of actions taken in other segments and guarantee consistency of in- and out-flows between segments. The governing equations and their corresponding principles are as follows:

Principle 1: Changes in headcount are fully characterized by the movements that bring employees into and take employees out of their segment:

$$\mathbf{HC}_{i,t+1} = \mathbf{HC}_{i,t} + \mathbf{Hires}_{i,t} + \mathbf{Promos\ In}_{i,t} - \mathbf{Promos\ Out}_{i,t} - \mathbf{Attrition}_{i,t} + \mathbf{NetTransf}_{i,t}$$

Principle 2: Promotions out of one job level are equal to promotions into another job level of the same tech indicator and leader.

$$\mathbf{Promos\ In}_{l+1,t} = \mathbf{Promos\ In}_{l,t}$$

Principle 3: Transferring employees across segments cannot create new employees, only redistribute employees across leaders in the same tech indicator and job level.
See Appendix 2 for implementation details.

2.7 Amazon Workforce Historical Dataset

The pre-existing dataset used for the offline RL method described above consisted in Amazon internal talent movement data with monthly time granularity data, spanning the period from 01/2016 to 09/2023. Employee-segments were defined using four employee dimensions, to produce employee segments characterized by different Senior Vice Presidents, job levels (levels 3 - levels 6), technical indicators (technical (non-SDE), non-technical, SDE), and corporate statuses (corporate or not corporate). In total, this required the agent to simultaneously plan for 37 segments of employees with a total of 3,370 sampled trajectories and a global sequence made of 112,299 elements.

2.8 Baseline Policy

The trajectory transformer's policy was compared to a naive behavioral cloning algorithm which approximates the human-experts' behavioral policy as sampled in the pre-existing dataset. In other words, behavioral cloning does not explore and attempt to generalize through alternative state-action space, as the offline RL attempt to do. Behavioral cloning was selected as a baseline because it allows us to approximate the quality of the trajectory transformer's learned policy relative to how Amazon has typically acted. The behavioral cloning algorithm replicated the trajectory transformer architecture and hyper parameters. The key technical difference between behavioral cloning and the trajectory transformer is that behavioral cloning does not use any planning method.

2.9 Off Policy Evaluation

Off-policy evaluation (OPE) is the problem of estimating the value of a target policy only using data pre-collected from some behavioral policy. Two OPE methods, model-based approximation and Fitted-Q Evaluation (FQE), were selected because no single OPE method outperforms others in all settings [7]. Furthermore, the two approaches offer complementary benefits and limitations.

2.9.1 Model-based OPE

Model-based OPE evaluates policies by learning a state transition function and predicting future states and rewards given a target policy. Higher quality policies have higher expected cumulative rewards. Model-based approaches can be inaccurate when model dynamics are unknown or poorly approximated [7, 8]. Despite this, they can be useful in situations when aspects of the environment are known, such as the governing equations, when planning is done for shorter horizons, or when state-action transitions are practically useful. In this study, we used the constrained trajectory transformer itself to approximate state transitions. Appendix 3 describes how this model was trained along with model performance.

After training the model, 12-month trajectories were rolled out for each segment using either the behavioral cloning baseline using the last 12-months as initial states. This process was repeated 25 times to provide a reliable estimate of model performance. Finally, the policies' returns were estimated by calculating the total discounted rewards across trajectories and averaging across samples.

2.9.2 Fitted-Q Evaluation

Fitted-Q Evaluation (FQE) is deep-learning model that directly predicts Q-values for a target policy by learning from a set of experiences sampled under a different behavioral policy [9]. FQE was utilized in addition to model-based OPE because it outperforms other OPE methods more consistently than alternative approaches [7, 8]. Note that its incremental performance in OPE is not universal across all setting - ranking in the top 10% of OPE methods across 23% of experimental conditions in one experimental study [7]. We implemented FQE as described in [9] using an autoregressive feed forward network with the same context length as the model-based OPE and a discount factor of 0.75. Policies were then scored using the learned Q function and the last 12-months as initial states to predict the value of the initial state under the learned policy as described in [8].

3 Results

3.1 Reward Accumulation

The trajectory transformer described above improved upon the behavioral policy across all employee segments and evaluation methods tested in this paper. At aggregate, the trajectory transformer produces 7.99% higher rewards according to the model-based OPE and 3.04% higher rewards according to FQE when compared to behavioral cloning. Considering FQE as the most conservative of the estimate, the talent policies learned and proposed by the agent would result in $26 million of cost savings per year. Table 2 reports the marginal rewards across the two policies using model-based OPE and Fit Q Evaluation. Policy improvements varied across segments and evaluation methods with the greatest percent improvements being observed for SDE (Avg. Improvement: 6.99% and $33 million), Leader C (Avg. Improvement: 7.58%, $80 Million), and Level 5 (Avg. Improvement: 6.48%, $21 Million).

Table 2. Comparison of Behavioral Cloning and the Trajectory Transformer algorithms' cumulative discounted rewards across employee segment and evaluation methods.

	Model-Based Evaluation		Fitted-Q Evaluation	
Employee Segment	Behavioral Cloning ($\times 10^8$)	Trajectory Transformer ($\times 10^8$)	Behavioral Cloning ($\times 10^8$)	Trajectory Transformer ($\times 10^8$)
Tech Indicator				
Non-Tech	-$7.41	-$6.86	-$3.56	-$3.45
SDE	-$2.45	-$2.18	-$2.03	-$1.97
Tech (Non-SDE)	-$2.28	-$2.13	-$2.95	-$2.86
Leaders				
Leader A	-$8.53	-$7.90	-$2.76	-$2.68
Leader B	-$2.87	-$2.63	-$3.28	-$3.18
Leader C	-$0.73	-$0.64	-$2.48	-$2.41
Job Level				
L3	-$1.65	-$1.56	-$2.11	-$2.04
L4	-$2.20	-$2.01	-$2.52	-$2.43

L5	-$3.68	-$3.33	-$2.03	-$1.96
L6	-$4.60	-$4.28	-$1.88	-$1.82

Note: The planning method with higher rewards under each evaluation method is underlined

These costs were primarily realized through reduction in cost attributable to under and overstaffing. Reaching desired headcount was often done using relatively more expensive talent plans. For example, the agent, on average, incurred $5 million dollars more in costs associated with the talent plan while saving $102 million dollars in cost with the talent gap according to model-based OPE. Appendix 4 reports these results in more detail.

It is important to note that model-based evaluation is relatively more optimistic about the agent's policy relative to FQE, which we plan to investigate in the future. Overall, model-based evaluation suggests the trajectory transformer improved over behavioral cloning was $71 million or 4.95% greater than the improvement estimated by FQE. This over may be due to its explicit use of the state-transition function learned by the trajectory transformer to evaluate the policy. This means that errors in state transitions are present in both the planning an evaluation method. In contrast, FQE uses the concept of Q-values which are estimated without explicitly modeling state-transitions. This allows the policy to be evaluated using a method that does not share the same errors in model dynamics as the planning method.

3.2 Policy Comparison

To better understand the policy implications of the algorithm, we examined the actions recommended by behavioral cloning and trajectory transformer algorithms in some key segments. Figure 3 provides an illustration of the actions taken by the agent and the subsequent states and rewards returned by the model across the behavioral cloning baseline and the trajectory transformers for one run and one initial state. Relative to behavioral cloning, for this segment the trajectory transformer had on average 33.05% higher rewards. The plan proposed by the trajectory transformer had 17.37% higher regretted attrition. These increased vacancies were filled by promoting 18% more employees into level 5 and hiring 9% more external employees. Both agents produced approximately the same transfer out requests (approximately a l.8% difference).

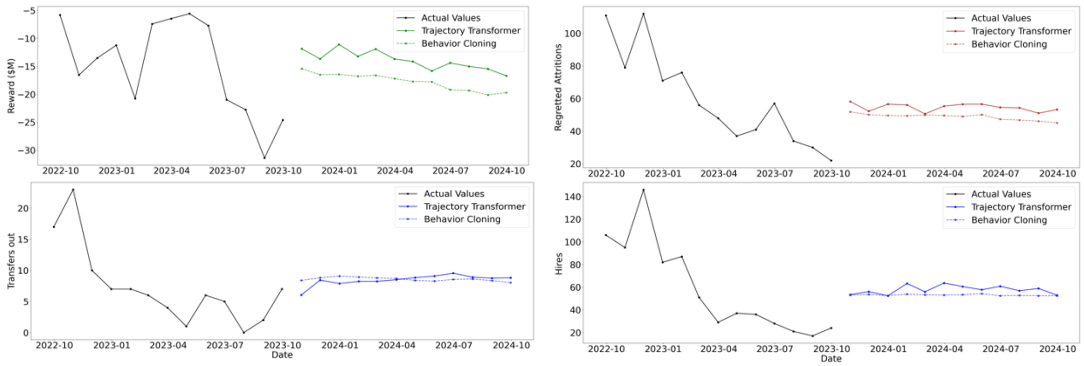

Figure 3. Rewards (Green), States (Red) and Actions (Blue) for one initial state for the baseline behavioral cloning algorithm and the trajectory transformer for Leader B, L5, SDE

4 Conclusion

We developed a RL method for workforce planning which can explore talent scenarios offline and identify optimal talent strategies. Our method uses a trajectory transformer architecture to learn from a training dataset generated by human experts, and a customizable reward function to help align the agent's behavior to human preferences. We gathered a dataset covering nearly a decade of historical workforce data at Amazon and show that the workforce policy learned by the agent systematically outperforms a behavioral cloning baseline which more naively infers policy from the historical dataset. The RL policy achieves between 2.82% and 12.33% higher rewards, with an improvement of 7.99%

on average across all model-based evaluations. This corresponds to a potential improvement of over $26 million compared to the current approach used at Amazon - a substantial cost savings.

Future research will address some of the limitations of the proposed offline RL approach: the simplifications inherent to using an MDP, which restricts the scope of the action space and state representation, can be addressed by increasing the fidelity of the state and expanding the controllable actions (e.g. compensation changes, recruiter activities). In particular, we are researching Graph methods to *plan more globally i.e.,* take into account more global ripple effects at decision time, by defining latent states as Graphs characterizing the global Amazon population and market conditions. These could better capture real-world complexities and identify novel strategies.

Another limitation and future research direction is the current definition of the reward function. In the current paper, we used deterministic reward components to align the agent with the most fundamental needs of workforce planning: minimize talent gaps and minimize costs. This, of course, doesn't capture more subtle human preferences and evolving strategic priorities of business leaders and talent planners across Amazon. To increase alignment of our agent to our customer preferences, we will follow two avenues of research in parallel. We will continue to identify and incorporate more detailed proxy measures in the reward function, and we will collect a dataset of human feedback (pairwise preferences between talent scenarios) on the current agent's behavior as in [10] to learn a reward model that can generalize and predict more subtle human preferences.

Given the objective of the agent in our method is defined by the user, and the exact nature of its reward function can be extended with arbitrary human preferences using Reinforcement Learning with Human Feedback (RLHF) [10], we hope the results presented in this paper will contribute to a wider adoption of RL methods for workforce planning policy optimization at Amazon.

5 Customer Problem Statement

Amazon team leaders, talent planners, and financial analysts, produce talent plans every year based on forecasts of talent movements so they can meet future staffing needs, but also regularly adjust these plans on the go based on internal and external talent and financial indicators. While machine learning is now increasingly used to produce forecasts, there does not exist any machine learning recommendation system at Amazon to explore, identify, or recommend talent plans. A talent planning problem is by definition a policy optimization problem, which can be addressed using RL. In particular, offline RL has few drawbacks compared to currently used methods at Amazon because stakeholders retain full control over both the goal of the agent, and whether to implement the recommended strategy or not. The agent has the potential to identify complex data-driven talent strategies under a variety of possible internal and external talent scenarios, but the entire optimization process happens *offline*, using generative models to simulate the Amazon workforce. In a RL framework, the goal of the agent is identical to the goal of a human talent planner i.e., to identify strategies to reduce talent gaps and minimize costs, and possibly to reach other user-specific goals. The key difference is that instead of customers having to explore scenarios and identify optimal talent strategies on their own, the agent can explore, in the service of these same customers, a much larger number (up to millions) of scenarios to identify optimal talent strategies. The agent then recommends which levers to pull and when to pull them, to minimize talent gaps and costs. Using the popular method of RL from Human Feedback, the exact nature of the agent's reward function could even be extended with arbitrary customer preferences, all the RL agent needs is some customer feedback on its behavior. The pioneering work in [10] showed RLHF can be scaled to practical and complex real-world applications beyond just language models, which we see as a formidable and untapped potential for the talent planning problem. The amount of feedback required to fine tune the agent's behavior may take time to collect. As a first step, the current paper aligns the agent using deterministic reward components corresponding to known fundamental needs in workforce planning: minimize talent gaps and minimize costs.

References

[1] Levine, S., Kumar, A., Tucker, G., & Fu, J. (2020). Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*.

[2] Prudencio, R. F., Maximo, M. R., & Colombini, E. L. (2023). A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*.

[3] Janner, M., Li Q., & Levine, S. (2021) Offline Reinforcement Learning as One Big Sequence Modeling Problem. *Advances in Neural Information Processing Systems. 34, 1273-1286.*

[4] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017) Attention is all you need. *Advances in Neural Information Processing Systems*. 30

[5] Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, *36*(3), 1181-1191.

[6] Januschowski, T., Gasthaus, J., Wang, Y., Salinas, D., Flunkert, V., Bohlke-Schneider, M., & Callot, L. (2020). Criteria for classifying forecasting methods. *International Journal of Forecasting*, *36*(1), 167-177.

[7] Voloshin, C., Le, H. M., Jiang, N., & Yue, Y. (2019). Empirical study of off-policy policy evaluation for reinforcement learning. *arXiv preprint arXiv:1911.06854*.

[8] Paine, T. L., et al. (2020). Hyperparameter selection for offline reinforcement learning. *arXiv preprint arXiv:2007.09055*.

[9] Le, H., Voloshin, C., & Yue, Y. (2019). Batch policy learning under constraints. In *International Conference on Machine Learning*. 3703-3712.

[10] Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., & Amodei, D. (2017). Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Appendix 1: Detailed Reward Information

We set the relative cost of the talent gap based on customer anecdote. Business partners report that they incur more cost when they are understaffed than overstaffed. This is because talent deficits make it difficult to produce planned deliverables and reduce future trust. Talent surpluses still incur cost by reducing the ROI realized from compensation expenditure. We differentially account for the cost of surplus and deficits by defining $\boldsymbol{c_d}$ as a function of the direction of the gap.

$$\boldsymbol{f(x) = \begin{cases} .5 \times x_{ti} \times m_i, & if\ x \leq 0 \\ .6 \times x_{ti} \times m_i, & otherwise \end{cases}}$$

Where $\boldsymbol{m_i}$ is the employee segment's total compensation. For example, if a segment were to have $100,000 total compensation and were to have a 10-employee talent surplus, the agent would incur a $500,000 penalty. If the segment were to have a 10-employee deficit the agent would incur a $600,000 penalty. For the sake of this paper, we assume that the cost associated with a talent plan is a linear function of the actions and states taken in the recommended talent plan.

$$\boldsymbol{v_i(A_t, O_t) = \overrightarrow{C_{A_t}^T}\,\overrightarrow{A_{ti}} + \overrightarrow{C_{O_t}^T}\,\overrightarrow{O_{ti}}}$$

Where $\boldsymbol{\overrightarrow{C_A}}$ is a vector of weights that describe the relative cost of each action and $\boldsymbol{\overrightarrow{C_O}}$ describes the relative cost of non-actionable talent movements (i.e., regretted attrition) that occur as state transitions.

$$\boldsymbol{\overrightarrow{C_{A_t}} = [C_P, C_{URA}];\ \overrightarrow{A_t} = [P, URA]}$$
$$\boldsymbol{\overrightarrow{C_{O_t}} = [C_H, C_{TI}, C_{RA}, C_{TO}];\ \overrightarrow{O_t} = [H, TI, RA, TO]}$$

The cost of talent strategies explored by the agent are defined by the relative cost of the talent movements. Relative costs were derived based on the financial and time expenditures necessary to complete an action and are reported in Table 2. For example, transfer in's require a relatively smaller pre-employment screening process and tend to require a lower training period when compared to external hires.

Table 1.1. Costs of actions and resulting talent movements per HC as percentage of average total compensation. Numbers are estimated and a more precise value will be aligned with business input

Promo (C_P)	URA (C_{URA})	Hires (C_H)	Transf-in (C_{TI})	RA (C_{RA})	Transf-out (C_{TO})
15%* m	20%* m	15%* m	8%* m	10%* m	5%* m

Appendix 2: Transfer Governing Equations

In addition to the aforementioned principles, we constrained the maximum number of transfers out per month from a segment as 20% of the segment's total headcount. Our approach to estimating transfers at the transaction level uses transfers-in conditional probabilities derived from historical transfers data. The conditional probability P(A|B) represents the likelihood that an employee transferred to team B originated from team A. We use this conditional probability as the basis for attributing total transfers into all other teams. Since we have a constraint on total transfers out from a team, the allocation may only be feasible for some segments. In such cases, we (1) identify teams that violated the constraint, (2) set their transfers out to the maximum transfers out, (3) exclude these teams and update transfers-in for remaining teams that did not violate the constraint, and finally, renormalize the conditional probability after excluding teams that violated the constraint. This process is repeated until no more teams violate the maximum transfer out constraints. Algorithm 1 illustrates the pseudocode for the constrained transfer allocation.

Algorithm 1: Pseudo code for iterative method to transfer allocation under constraint

Required transfers in, max transfers out for all segments stored in DF, and P as transfers-in conditional probability
Initialize Compute unconstrained transfers out by allocating using P
DF1 = teams violated transfers out > max transfers out constraint
While num teams in DF1 > 0
1. **For** each team i in DF1
    1. Set transfer out = max transfers out
    3. Subtract transfers-in and transfers-out team i contributed from DF
    4. Remove team i from P and re-normalize P
2. Identify teams with excess HC to transfer out as DF2 = set(DF) – set(DF1)
3. Compute unconstrained transfers out using updated P and DF2
4. Update DF1 = teams violated transfers out > max transfers out constraint
Update transfers in of each team based on transfer out that satisfy all constraint

Appendix 3: Agent Training

The trajectory transformer was trained to predict next state, action, and reward transitions. Hyperparameters were selected by conducting a Bayesian search of the parameter space implemented using Amazon SageMaker HPO. Parameters tuned include layers, heads, embeddings, and context. A manual search of batch sizes, learning rates, and numbers of bins for the discretization. We found that a batch size of 512, Adam optimizer with a learning rate of 1e-3, and 200 bins for the discretization were the most effective parameters.

In this paper, we use the Mean Absolute Percentage Error (MAPE) as a metric for evaluating the accuracy of our forecasting model. MAPE is a reliable indicator by quantifying the average absolute percentage deviation between predicted and actual values. Using the proposed trajectory transformer, governed by specific equations, the overall MAPE for all the state dimension for all the segments is 118.98%, overall MAPE for all actions dimension for all the segments is 256.69%, and for reward is 72.51%.

Appendix 4: Breakdown of Rewards

Table 4.1. Comparison of Behavioral Cloning and the Trajectory Transformer algorithms' cumulative discounted rewards across employee segments using model-based estimation.

	Talent Gap Cost		Talent Plan Cost	
Employee Segment	Behavior Cloning ($\times 10^8$)	Trajectory Transformer ($\times 10^8$)	Behavior Cloning ($\times 10^8$)	Trajectory Transformer ($\times 10^8$)
Tech Indicator				
Non-Tech	\$6.95	\$6.37	\$0.45	\$0.50
SDE	\$2.37	\$2.11	\$0.070	\$0.074
Tech (Non-SDE)	\$2.24	\$2.08	\$0.035	\$0.039
Leaders				
A	\$8.08	\$7.41	\$0.45	\$0.50
B	\$2.78	\$2.53	\$0.083	\$0.088
C	\$0.70	\$0.62	\$0.023	\$0.024
Job Level				
L3	\$1.49	\$1.39	\$0.16	\$0.17
L4	\$2.04	\$1.84	\$0.16	\$0.17
L5	\$3.56	\$3.18	\$0.13	\$0.14
L6	\$4.49	\$4.15	\$0.11	\$0.13