Proceedings of the 2016 Winter Simulation Conference
T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick, eds.

OUTPLACEMENT TIME AND PROBABILITY ESTIMATION USING DISCRETE EVENT SIMULATION

Sudhanshu S Singh
Rakesh R Pimplikar
Ritwik Chaudhuri
Gyana Parija

IBM Research - India
4, Block C, Phase II
ISID Campus, Vasant Kunj
New Delhi - 110070, INDIA

ABSTRACT

In today's rapidly changing technological scenario, tech giants revise their strategic alignment every couple of years. As a result, their workforce has to be adapted to the organization's strategy. Members of the workforce who are neither relevant to the strategic alignment, nor can be made relevant by reskilling, have to be either outplaced (i.e. placed in an another job within organization) or separated from the organization. In geographies like Europe, where the cost of separation is very high, it becomes very important to make the right decision for each employee. In this paper, we describe a simulation based methodology to find the probability and time of outplacement of an employee. These numbers are inputs to a global problem of making the optimal decision for the entire workforce.

1 INTRODUCTION

Outplacement means placing a candidate in an another job within organization. Estimation of outplacement time (*OT*) and outplacement probability (*OP*) of candidates in jobs is an analytically intractable problem due the inherent nature of the problem. Big technology organizations need to often solve this problem, due to frequent realignment of strategic directions in the rapidly changing tech landscape. Every time an organization decides to focus on certain technical domains, they need to make some tough decisions regarding their workforce. Each employee is tagged as being in one of the following categories

1. Suitable to the strategic alignment with some investment in reskilling (potentially zero for many employees)
2. Not suitable to the strategic alignment, but can be outplaced
3. Not suitable to the strategic alignment, can not be outplaced

Determining an appropriate category for each candidate, given the organization's budget constraints is the umbrella problem to be solved. In order to solve the problem, every employee's fitment to the strategic alignment has to be quantified. Those below a threshold should be considered for outplacement. Employees in category three have to be separated from the organization. In some geographies such as Europe, the cost of separation is very high. Hence it is imperative that the organizations be very careful in tagging employees in the last category. Technically, employees who have a very large expected *OT* and or very small *OP* are candidates in the third category. It is this problem of finding a candidate's outplacement time and probability that we are addressing here.

With an approximate reskilling cost vs relevance curve, and outplacement time/probability numbers available, a global problem to maintain an aligned headcount while staying close to the budget can be formulated and solved. The outplacement time can be monetized by multiplying it with an employee's salary, and adding the actual placement process cost to it. Those with outplacement cost more than severance cost might be separated instead of being chosen for outplacement. The overall problem is that of making one of the three decisions for each employee, keep with (potentially zero) reskilling cost, outplace or separate, with corresponding costs. The objective is to minimize the deviation from realignment budget (with more weight for deviation on higher side) with headcount constraints.

In order to estimate a candidate's outplacement time and probability, some measures on goodness of fit to available jobs, and expected time to place for each available job are required. The latter can be obtained from historical data, by using machine learning models such as CHAID Regression Tree, Neural Networks etc. Since it is an approximate estimate for time to fill the jobs, it can also be aggregated from historical data based on job attributes such as geographies, job level, job category, etc. As long as the aggregation is not way off the reality, and each candidate for a job has the same placement time with respect to it, this number is less critical than the former for calculations. The other, namely goodness of fit to available jobs is more critical, since it defines preference of candidates over jobs.

Although our focus in the paper is only on simulation to estimate *OT* and *OP*, we describe the entire process of outplacement, and our solution for different stages for the sake of completeness. Outplacement is typically carried out by an agency responsible for outplacing as many candidates as possible. The agency first identifies relevant jobs for each candidate, then recommends certain jobs to each candidate to apply for. Note that the agency's objective in recommending jobs is to place maximum number of candidates in matching jobs, and not to place candidates in the best possible job for them. A candidate may choose not to apply to all the recommended jobs, and to apply to not recommended jobs as well. The probability of a candidate getting an offer is therefore adjusted by "probabilities of applying" to different jobs. Our complete solution involves

1. A job description and candidate CV matching system based on semantic similarity.
2. An optimization algorithm to recommend jobs to candidates based on offer probability to maximize the number of candidates placed.
3. A simulation engine to estimate *OT* and *OP*

We assume that appropriate descriptions of jobs, which candidates can be outplaced to, are provided, and the most recent candidate CVs are provided. The candidate CVs can be matched to job descriptions using NLP based semantic text matching techniques to compute a job - candidate tech match score. This match score, in conjunction with other attributes such as location match, experience match etc. is used to calculate an offer probability ($\mathscr{P}_{ij}$ = probability that candidate i will be offered job j). The reason for calculating the offer probability on top of match score is that the match score just finds the suitability of job to candidate based on text matching between the job description and candidate CV. We use SVM to train a classification model using tech match score and other attributes. We apply this model on new job - candidate pairs to compute $\mathscr{P}_{ij}$. We have skipped the details of tech match score and $\mathscr{P}_{ij}$ computations, as it is not the focus of this paper.

2 LITERATURE REVIEW

The problem of likelihood based job matching for candidates has not been discussed in literature in the past. This, particular problem is not only very relevant for companies to determine how likely a candidate is to be placed, but also it gives an overview on how long it takes for candidates to be placed.

This particular problem can also be transformed into a problem of queuing theory on non-reusable servers where each server represents a particular job and the candidates are the people on the queue who have a match for that particular job. The servers are non-reusable because once a candidate chooses a job, that server becomes non-existent from the system. Moreover when a particular candidate applies for a job, a matching score is calculated based on the candidates profile and the job description. Based on the score the candidates profile is ranked against the other potential candidates and kept in the pipeline according to the order of the ranking obtained. In the context of queuing theory, the same can be analogously explained as, considering a customer appears in a queue pertaining to a particular server, instantly the server returns a probability of how likely the server will serve that customer. The probability is based on the similarity between, the help that the customer needs and the domain of services provided by the server. Better matches yield higher probabilities. Along with the probability the server also returns the expected waiting time that the customer needs to wait before her ask is processed. While the customer keeps on waiting, the server keeps on returning similar probabilities and the expected waiting time to other potential customers also. Also the server gives an option to only one of the customers (based on match probabilities) whether she is interested in being served by that particular server and also a time frame within which the customer needs to give her opinion. The moment that customers opinion is affirmative, the server becomes unavailable to all the existing customers including the ones who were on the waiting list. Otherwise, server extends the same offer to another customer in the pipeline. In addition, the arrival of customers to the server follows any arbitrary distribution. This particular notion gives rise to a very interesting set up which has not been discussed much in literature.

This particular problem is an advanced version of M/G/1 queuing theory where the arrival of a customer to a server is same as a new candidate application arriving for a job. But customers are being served based on the best match of the job type and the server will accommodate for only one customer. The basics of queuing theory can be found in Gross et al. (2008). Also in the context of hiring, often recruiters use technologies. In the paper Chapman and Webster (2003) the authors address the issues relating to what are the technologies that are used by recruiters, recruiter's goals for using such technologies and the extent to which the goals are being met. According to the findings in their paper, as much as one third of the technology users ended up having limited or moderate success. Also, Malinowski et al. (2006) describe an approach based on matching jobs with the candidates by using a bipartite graph where the best matches are obtained via preferences of the candidate and the preferences of the recruiter. Also, in the literature it can be found that there are quite a few patents that are based on different methodologies for matching jobs with the candidates. Defoor (2001) describes a methodology where candidates from a certain region or city comes to a web page and post their qualifications, skill level and domain of expertise in a specified skill matrix. Skill matrix contains specific information on competency and experiences of the candidates. Similarly, employers or recruiters enter the information of the required skills, competencies in the skill matrix. Hence via simple search employers can find the candidates who are best suited for a specific job and likewise, candidates can see the jobs for which they are most suited. On the other hand Baldwin and Baldwin (2009) describe a methodology for determining matching jobs based on candidate's personality traits and optional interests. Based on candidates' answers of questions from a questionnaire, personality traits are estimated and matched with the employers' choice of suitable personality traits that they need in the potential candidates. Based on the evaluated metric, a suitable pool of matching jobs are shown to the candidate. Puram and Sadagopal (2001) describe a methodology based on matching a candidate's profile to a pool of jobs and by readjusting the skills of the candidates the best matching job is determined. Also in the process, against each of the jobs candidates are given scores. Based on the adjusted skill scores candidates are shown a pool of jobs which have maximum matching with their profiles. Also, Yi et al. (2007) discuss an approach based on Structured Relevance Models (SRM), an extended relevance based language model for modeling and retrieving semi structured documents such as resumes of candidates. The model matches the resumes of the candidates with different jobs. In the paper the authors discuss that SRM based model performs better than the typical unstructured relevance models.

The approach described in this work is very different from the traditional approaches that are tried out in the past or that can be found in the existing literature. The novelty of this approach is the NLP based machine learning techniques which are used to assign scores for candidates CVs and jobs. Also probability of offers are also calculated using ML based techniques by using matching score and other attributes and in addition doing a simulation to estimate actual OP and OT given competition for jobs and candidates.

3 BASIC MODEL FOR OUTPLACEMENT

We define a very basic model to compute *OP* and *OT*. It is a very simple model that may not be realistic in several complex scenarios, but it definitely helps to understand the motivation behind more complex model that we have discussed in section 4.

Time to outplace a candidate directly affects the cost of outplacement. Longer the time, higher the cost will be. Probability of outplacement for a candidate indicates the chances of a candidate getting hired for at least one of the open jobs. Both time and probability depend on the jobs that a candidate is applying for. As described in section 1, placement agency recommends jobs to every candidate. Though candidates are expected to apply from jobs recommended to them, they cannot be restricted to do so. So it becomes essential to capture the uncertainty in candidates applying to jobs, while designing outplacement probability model.

Consider that a candidate C_i is recommended K jobs, while she is eligible for N jobs. A candidate applies for n jobs on an average per week. Let's assume that out of n, she applies to k jobs from the recommended list and to $(n-k)$ jobs from the remaining list that she is eligible for. So probability of C_i applying to job J_j is given by

$$\mathscr{A}_{ij} = \begin{cases} \frac{k}{K} & \text{if job } J_j \text{ is recommended} \\ \frac{n-k}{N-K} & \text{if eligible for } J_j \text{ but not recommended} \\ 0 & \text{if not eligible for } J_j \end{cases}$$

We combine this application probability with the offer probability as given in equation (1). So probability of a candidate C_i getting hired for a job J_j, considering application probability, can be calculated as

$$\mathscr{P}^{ea}_{ij} = \mathscr{A}_{ij} \times \mathscr{P}_{ij} \tag{1}$$

Consider T_j to be acceptance time to fill a job J_j. T_j is the acceptance period, that is, the number of days between the interview day and the day candidate has to revert back with the decision if the candidate is hired. For every candidate, we rank the jobs that she is eligible for in ascending order of T. Assumption here is that, if a candidate gets hired earlier, she will be ruled out from rest of the jobs. Let's denote rank of job J_j in the list of candidate C_i by $\mathscr{R}_{ij}$. Thus, the probability of a candidate C_i getting hired for job J_j considering everything (i.e. competition from all candidates, application probability and other jobs that she might have applied to) can be calculated as

$$\mathscr{P}^{*}_{ij} = \left[\prod_{r=1}^{\mathscr{R}_{ij}-1} (1-\mathscr{P}^{ea}_{rj})\right] \mathscr{P}^{ea}_{ij}$$

Probability of outplacement for a candidate is the probability of getting hired in at least one job considering everything. Once we have all $\mathscr{P}^{*}_{ij}$ calculated, probability of outplacement for C_i can be simply calculated as

$$P_i(\text{Outplacement}) = \sum_j \mathscr{P}^{*}_{ij} \tag{2}$$

Now consider a hypothetical job ˆ, which is offered to a candidate if she cannot be hired for any open job. Probability of a candidate C_i getting hired for ˆ can be calculated as

$$\mathscr{P}^{*}_{\hat{\imath}} = 1 - P_i(\text{Outplacement})$$

Time to fill the hypothetical job ˆ is actually the maximum time a candidate can spend in the system before getting considered for separation from organization. Let's denote that time by $\hat{T}$. With this, we can compute expected time to outplace a candidate C_i as

$$T_i(\text{Outplacement}) = \mathscr{P}_{\hat{i}}^{*} \times \hat{T} + \sum_{j} \left(\mathscr{P}_{ij}^{*} \times T_j\right) \tag{3}$$

3.1 Disadvantages

Basic model provides a closed form solution to compute probability and time for outplacement in polynomial time. However it inherently makes following strong assumptions. When these assumptions are not valid, outplacement probability and time could be significantly different.

1. It assumes that all job positions are open when the computations are carried out. Effect of jobs arriving at later point of time is not taken into account.
2. Once a job is offered to a candidate, candidate doesn't have option to decline the offer. It also means that a job, once offered to a candidate, cannot be offered to anyone else. In reality, candidates are given a time window to decide whether to accept the offer or not. If a candidate reject an offer, that offer it given to the next best candidate in the list.

Real business problem does not satisfy aforementioned assumptions and is analytically intractable. Therefore, we propose discrete event simulation based approach as described in the following sections.

4 DISCRETE EVENT MODEL FOR OUTPLACEMENT PROCESS

The model we describe here follows real world application and placement process. The process is described from both candidate's and job's perspective here. For the purpose of simplicity, we treat a job as an entity that interviews candidate and makes decisions. Please note that each open position of the same job requisition is treated as a separate job here.

Candidates apply to jobs as they become available/open and appear for interview a few days after applying. Each candidate is either offered the job, or not. If the job is offered to the candidate, she has to either accept or reject the job within a given time frame. The candidate may appear for other appropriate jobs during the response period and she might have multiple offers at the time when she has to decide to accept/reject the first job offered to her. Once a candidate accepts one offer, other offers made to her are considered as rejected. These jobs are then offered to other available candidates.

A job does not interview all the candidates that apply for it as all candidates are not equally good fit for it. Even among the candidates that are a good fit, the interview process follows an order of processing the best candidates first. Here we assume that candidates are preference ordered by offer probability. Based on the computational context, the offer probability of offer is a number between [0, 1] and and all candidates for a job can be ordered by it. One might be tempted to think that an $\mathscr{P}_{ij}$ values of 0.97 is better than 0.92, but these are numbers based on certain machine learning models, and prone to error. And in most recruitment scenarios there are multiple candidates who are all equally good fit for the job. Hence we club candidates in one bucket based on their $\mathscr{P}_{ij}$. Since we use SVM to predict $\mathscr{P}_{ij}$, the $\mathscr{P}_{ij}$ values for candidates who are likely to get placed are more than 0.50. We divide the $\mathscr{P}_{ij}$ in 5 bins (0.50, 0.60], (0.60, 0.70], (0.70, 0.80], (0.80, 0.90] and (0.90, 1.00]. Candidates in the highest bin are interviewed first, and if none of them is found good, then only candidates in the next bin are interviewed.

4.1 Model

In accordance with the processes mentioned above, we create discrete event simulation model that has two primary events - job events and candidate events. The model has been developed using Anylogic, which is a java based simulation tool. Anylogic provides an intuitive drag and drop UI with several built in objects to make discrete event simulation easy. In addition, it also allows users to define their own java classes as well for providing a better control to the user. We primarily use the event object provided by Anylogic, in conjunction with some custom defined objects.

We define two custom objects in our simulation model:

1. Requisition Object representing each candidate. A requisition object has following attributes:
   (a) ReqID, ReqName: Identifiers.
   (b) TimeToFill: Time (in days) from opening of this requisition to the day of interview. (filled after each simulation run)
   (c) TotalTimeToFill: Actual time the requisition takes over several simulation runs to be filled. Note that it is not just TimeToFill plus time a candidate gets to accept. It is usually higher since candidates may also reject the job.
   (d) Filled: Boolean to indicate if the job is filled in a simulation run.
   (e) ResList: List of resources that are matching to this job.
   (f) FilledStats: Statistics object to capture the status (filled or not) of the requisition over multiple runs.
2. Resource Object representing each job.
   (a) ResID, ResName: Identifiers.
   (b) TimeToPlace: Time to place the resource during a simulation run
   (c) PlacementProbability: Probability of the resource getting placed in a simulation run (1 or 0).
   (d) ReqList: List of resources that have $\mathscr{P}_{ij}$ more than 0.50 for this resource. The list is sorted by $\mathscr{P}_{ij}$.
   (e) JobsOffered: List of jobs that are offered to this resource during a simulation run
   (f) PlacedStats: Statistics object to capture the status (placed or not) of the resource over multiple runs.
   (g) TimeToPlaceStats: Statistics object to capture the time to place of the resource over multiple runs.

The attributes FilledStats, PlacedStats and TimeToPlaceStats are of type StatisticsDiscrete, which is an in built class in Anylogic. As more and more numbers are added to a StatisticsDiscrete object, it incrementally calculates the mean, standard deviation, number of observations, quartiles etc. of the collected statistics.

The simulation model consists of several events taking place in order. The Simulation model is governed by two primary events

- Job Event: A job event corresponds to an interview result declaration. A job event essentially makes the decision to offer the job to one candidate. The candidates are considered for being offered the job by bins.
  1. First the candidates in highest $\mathscr{P}_{ij}$ bin (0.90, 1.0] are considered. Order of candidates within a bin does not matter.
  2. Every candidate is assigned true value with probability $\mathscr{P}_{ij}$.
  3. If there are more than 1 candidates with true value, a candidate is picked randomly from the true set for making the offer.

  Initially, Job Events are set to execute at the time of interview. As a result of a job event, a Candidate Event is set to execute after acceptance time for the job.
- Candidate Event: A candidate event corresponds to a candidate accepting/rejecting a job. A candidate can have multiple job offers in hand at the time of a candidate event since she can get other offers within the acceptance period. Once a candidate accepts a job, both the job and candidate become unavailable (i.e., their filled/placed boolean attributes are set to true) for further processing. The jobs that a candidate rejects are again processed via Job Events taking place instantly. The process a

candidate follows to accept/reject a job is the same as that used by jobs to accept/reject candidates. That is

1. First the jobs in highest $\mathscr{P}_{ij}$ bin (0.90, 1.0] are considered. Order of jobs within a bin does not matter.
2. Every job is assigned true value with probability $\mathscr{P}_{ij}$.
3. If there are more than 1 jobs with true value, a job is picked randomly from the true set for accepting the offer.

After each simulation run, the StatisticsDiscrete objects get appended by the corresponding number. Note that for candidates that are not placed, it is filled by a very large number. The filled/placed boolean attributes of resources and requisitions are set to false before the start of each simulation run.

A basic diagram explaining the model logic and entities/events is shown in figure 1.

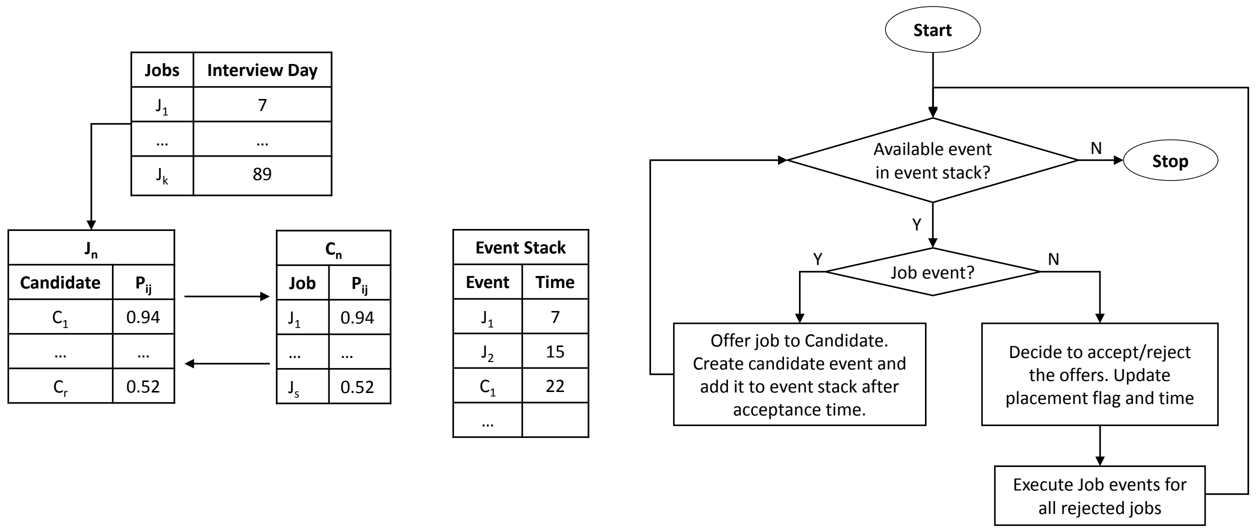

Jobs
Interview Day
J1 7
…
Jk 89
Jn
Candidate
Pij
C1 0.94
Cr 0.52
Cn
Job
J1 0.94
Js 0.52
Event Stack
Event
Time
J1 7
J2 15
C1 22
Start
Available event in event stack?
N
Stop
Y
Job event?
Y
N
Offer job to Candidate. Create candidate event and add it to event stack after acceptance time.
Decide to accept/reject the offers. Update placement flag and time
Execute Job events for all rejected jobs

Figure 1: Entities/Events and Model Logic.

4.2 Simulation Termination

In this section, we describe the termination time for a single simulation run and termination criteria for multiple replications of the simulation run. Even if the set of jobs and candidates is a small fixed set, this process might take a very long time to terminate due to bad matches and frequent rejections. Moreover, jobs and candidates keep coming into the mix with time. In order to get meaningful results, the simulation horizon needs to be fixed, and job/candidate availability has to be predicted/projected over this horizon.

Since job and candidate availabilities are predicted, making those predictions farther away in future would increase the prediction error. Also in order to get a reasonable number of jobs to simulate over, the simulation horizon should be large enough to accommodate the cycle time from interview to candidate acceptance for several jobs. In our case, business required a look ahead for a quarter, so that they can take timely action for those who are less likely to get a job in coming quarter. This would help them to reduce the cost of retaining idle employees. Hence we decided to run the simulation for 3 months plus some grace time for acceptance of jobs opening later in the horizon. If we stop the simulation at exact 3 months, then the jobs interviewing in last week with 2-3 weeks of acceptance period will not be filled for sure. Hence a grace period is allowed in the simulation. For simplicity we assume that no new jobs are arriving during the grace period, otherwise this becomes a recursive issue. For the business problem described in introduction section, we run this simulation model every two weeks with updated job and candidate information to get relevant results.

In order to get meaningful results with good confidence, the simulation model is run several times and results are aggregated. As the simulation is run more and more number of times, the *OP* numbers for candidates converge to their true values.The simulation model is replicated and results are collected till the change in *OP* numbers becomes very small. if there are n candidates, then the *OP* numbers form a n - dimensional vector in $[0, 1]^n$, referred to as $\hat{p}$. As more and more replications are done, the change in $\hat{p}$ gets smaller with each iteration. Hence we replicate the simulation model in batches of a fixed size. That is, depending upon the data instance, we may check the change in $\hat{p}$ after every 20 replications. This batch size may change from model to model, and is determined experimentally. The replication termination criteria used is to stop after the k^{th} batch if the manhattan norm does not change much between $k-1^{th}$ and k^{th} batch. That is

$$\sum_{i=1}^{n} |x_i| < n\varepsilon \qquad (4)$$

Where ε is a user defined parameter. We created two variants of our simulation model:

1. MS (Multi Shot acceptance): In version one, upon arrival of first acceptance deadline, the candidate decides about accepting/rejecting the first job only. If the decision is to accept, then both the job an candidate are tagged as taken. Otherwise the candidate keeps appearing for more interviews till the next deadline arrives.
2. SS (Single Shot acceptance): In version two, upon arrival of first acceptance deadline, the candidate decides to accept one of the offered jobs at the first deadline itself.

4.3 Verification and Experiments

As described earlier, this problem is analytically untractable, but it is possible to create instances wherein an intuitive solution to the system is available. We use one such instance to verify the correctness of our model. If there are n candidates for j jobs (n ¿ j), and each candidate has an offer probability of 1 with respect to every job, then the *OP* of each candidate is $\frac{j}{n}$. In addition, if each job opens the interview on same day, say day d of simulation, and has zero acceptance period (that is, candidates must accept reject it immediately) then outplacement time of each candidate will be$d$ days in simulation runs in which they get placed, and the large number, say L in other simulation runs. That is, the expected outplacement time for the candidates will be $d\frac{j}{n} + L(1-\frac{j}{n})$.

We chose 2 candidates and 3 jobs for specific instance that we used for verification of our model. We chose $d = 30$ for interview day of all the jobs, executed the simulation upto 90 days. The placement time for candidates not placed is set to be $L = 105$ days. Hence for the 2×3 case, we expect outplacement probability of both candidates to be $\frac{2}{3}$ and placement time to be 55 days. Irrespective of the model variant used, we get the same results.

Candidate1	*OP*	*OT*
1	0.6719	54.60
2	0.6645	55.16
3	0.6635	55.23

Figure 2 shows convergence of norm difference over iterations

Apart from testing on the verification set, we also ran the simulation experiment for a business scenario. This scenario contains 593 different job candidate combinations with 279 unique candidates and 182 different jobs. The offer probability ($\mathscr{P}_{ij}$) numbers' distribution for this scenario is shown in Figure 3. The Bins are same here as we define them in the beginning of section 4. Y-axis shows the number of job - candidate pairs for the corresponding bin. It can be seen that most job - candidate combinations are either in lowest or highest bin.

The job competition distribution can be seen in Figure 4. On the X axis, this graph shows number of candidates a job is matched to, and on the y axis it shows number of such jobs. E.g., there are 63 jobs

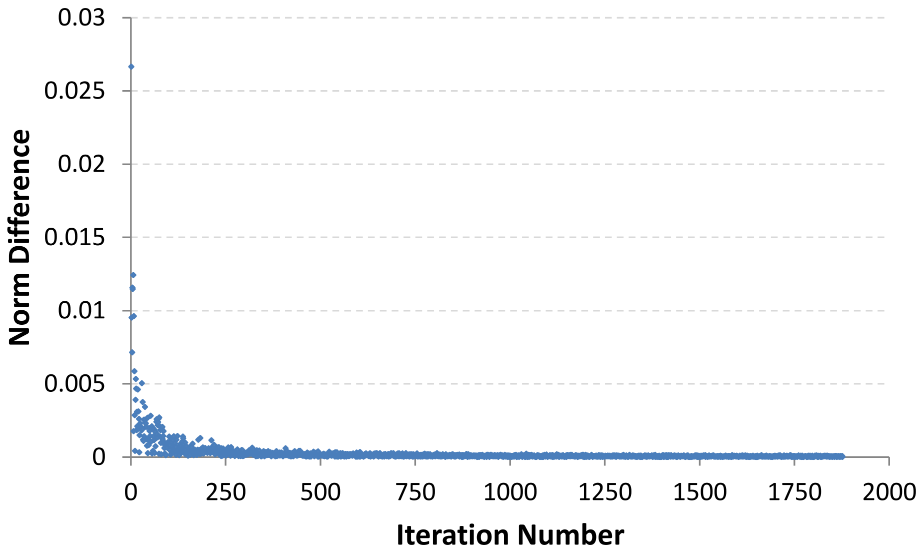


Figure 2: Norm difference vs iterations, 2 x 3 verification.

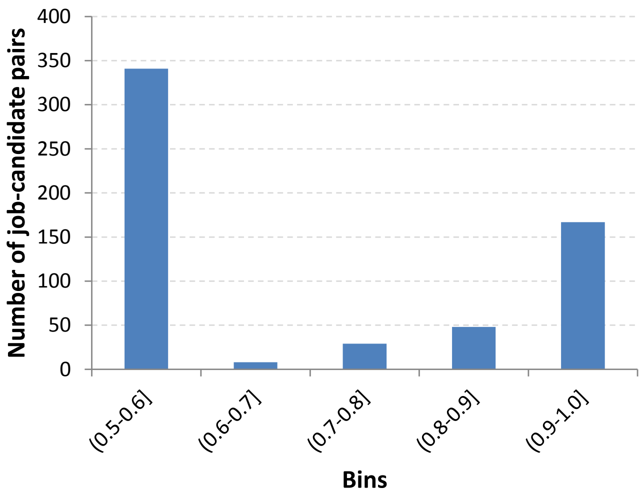


Figure 3: Offer probability distribution for business scenario.

that are matched to only one candidate, 33 matched to 2 candidates and so on. Note that the first bar does not imply that these 63 candidates will definitely get placed. The simulation was executed for both the job acceptance strategies as mentioned earlier. Figure 5a shows the convergence of norms for the SS strategy and Figure 5b shows the same for MS strategy. As can be seen, both strategies converge in almost the same number of iterations. Note that the simulation is run 20 times in each iteration.

In Figures 6a and 6b, we show the relationship between offer probability and *OP* for both the strategies. In these figures, size of bubble corresponds to the number of matched offers for the candidates. As can be seen, for similar offer probability, *OP* is higher for candidates that have more matching jobs. In general *OP* for candidates with good matches against some jobs are also high.

Figure 7 shows the scattersite of *OP* by the two approaches, and it is very clear that for most candidates, irrespective of the strategy *OP* depends on offer probability and number of offers. We have not shown any results for *OT* here because it depends not only only offer probability, but also on time of arrival of job. For example, a candidate might have a very high *OP*, but since her best suited jobs arrive late, her

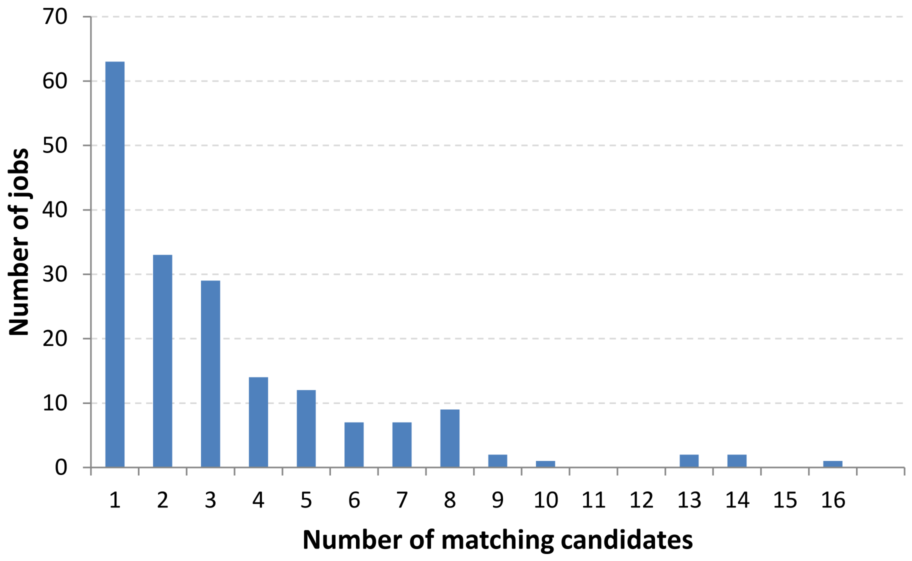
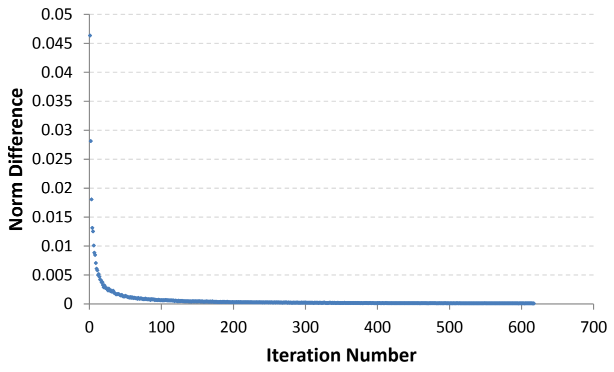
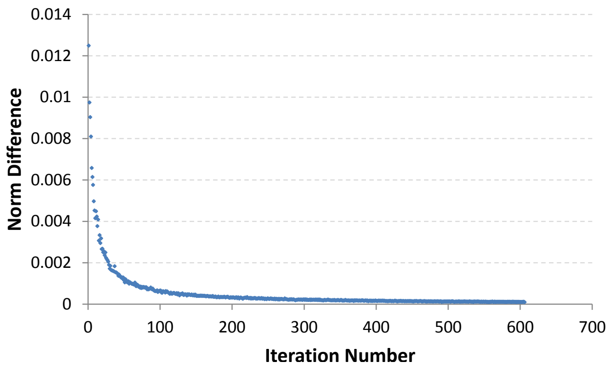


Figure 4: Job competition distribution for business scenario.

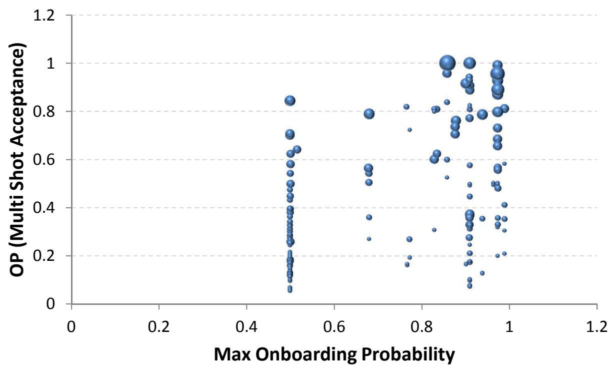


(a) Multi Shot Acceptance.

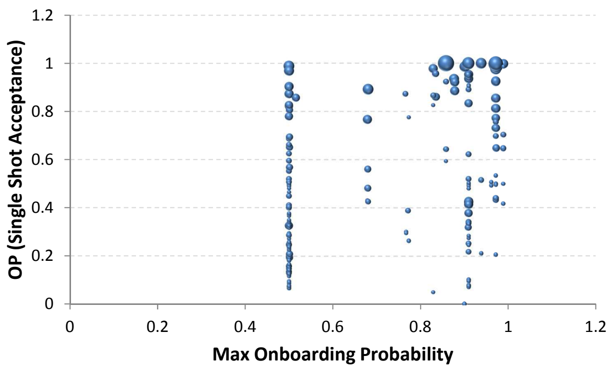


(b) Single Shot Acceptance.

Figure 5: Norm difference vs iterations for business scenario.



(a) Multi Shot Acceptance.



(b) Single Shot Acceptance.

Figure 6: Offer probability vs OP, heat by Offers.

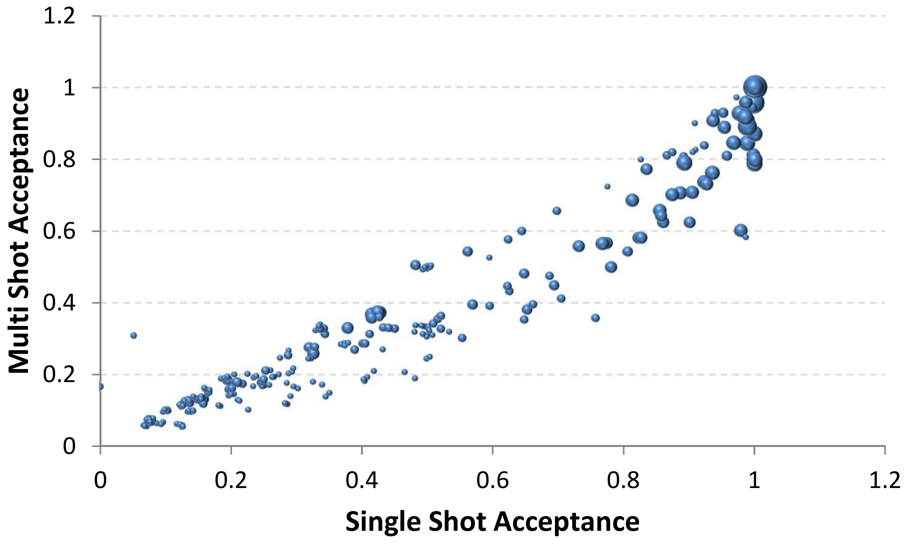


Figure 7: Multi Shot vs Single Shot Acceptance.

outplacement time will be high. Nonetheless, this simulation model does give the outplacement time as output as well since it is required by the business problem.

5 CONCLUSIONS AND FUTURE DIRECTIONS

Based on the simulation experiment results on the business case, we see that candidates that have higher offer probabilities or are matching to more jobs have higher chances of getting placed. There does not seem to be a strong correlation between offer probability and *OP* though. Since the problem of finding *OP* and *OT* is analytically intractable, the simulation models give a quick and easy way of estimating those. As shown by norm convergence curves, the *OP*s converge as number of iterations increases. We have used the simulation in context of outplacement, but any placement agency can use such simulation to estimate placement time and probability of their clients.

In this paper we analyzed two strategies of acceptance by candidates:

1. Single shot acceptance: Candidate makes a final decision at the time of first deadline.
2. Multi shot acceptance: Candidate decides about jobs as and when their deadlines arrive.

Real life decisions are somewhere in the middle of these two. That is candidate uses strategy 2 until a self defined deadline, and once the deadline arrives, she makes a final decision. Also, jobs may not stay in the system for the entire duration of simulation horizon. The need for a particular job may die out, or it might go to external competition. Hence the jobs have an expiry date, which has to be derived from historical data. These are some natural extensions to the problem.

Apart from the aforementioned variations, some technical details of simulation termination criteria can also be experimented with. Since the *OP* vector consists of only zeros and ones for each iteration, we could take the change in norm over a moving average instead of all iterations. The number of points to take moving average can be a function of number of candidates, or that of number of job candidate combinations. Also, the stopping criteria could be changed to *OP* confidence intervals being within a pre-specified range.

Though the problem mentioned in this paper is important to address from business point of view, we couldn't find any mention of such problem in literature. In the absence of any ground truth, it is difficult to validate our solution. We hope to gather some real time data and results, when our solution is operational. It will help us to set the accuracy of our approach.

REFERENCES

Baldwin, B., and G. Baldwin. 2009. "Job Matching System and Method". *U.S. Patent, Publication Number US7502748 B1*.

Chapman, D. S., and J. Webster. 2003. "The Use of Technologies in the Recruiting, Screening, and Selection Processes for Job Candidates". *International Journal of Selection and Assesment* 11:113–120.

Defoor, W. 2001. "Method for Matching Job Candidates with Employers". *U.S. Patent, Publication Number US20010042000 A1*.

Gross, D., J. F. Shortle, J. M. Thompson, and C. M. Harris. 2008. *Fundamentals of Queueing Theory*. 4th ed. Wiley Series in Probability and Statistics.

Malinowski, J., T. Keim, O. Wendt, and T. Weitzel. 2006. "Matching People and Jobs: A Bilateral Recommendation Approach". *System Sciences, 2006. HICSS* 6:137c.

Puram, K., and G. Sadagopal. 2001. "Consultant Matching System and Method for Selecting Candidates from A Candidate Pool by Adjusting Skill Values". *U.S. Patent, Publication Number US6289340 B1*.

Yi, X., J. Allan, and W. B. Croft. 2007. "Matching Resumes and Jobs Based on Relevance Models". In *SIGIR '07 Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 809–810.

AUTHOR BIOGRAPHIES

SUDHANSHU S SINGH works as an advisory Research Staff Member at IBM Research India. He has done his PhD in Operations Research from UNC Chapel Hill. His research interests are Deterministic Optimization, Simulation, and Machine Learning. His email address is sudsing3@in.ibm.com.

RAKESH R PIMPLIKAR is an Advisory Software Engineer at IBM Research India. He has done his MTech in Computer Science from IIT Bombay. His research interests are Machine Learning, Data Mining, Text Analytics. His email address is rakesh.pimplikar@gmail.com.

RITWIK CHAUDHURI is a postdoctoral researcher at IBM Research India. He has done his PhD in Statistics from UNC Chapel Hill. His research interests are Machine Learning, Statistical Analysis and Data Sciences, Stochastic Process and analysis of complex social models. His email address is ritwicha@in.ibm.com.

GYANA PARIJA is the manager of Analytics & Optimization research group in IBM Research India. He is the global lead for IBM Smarter Workforce initiative. His research interests are in the area of developing Cognitive Workforce solutions. His email address is gyana.parija@in.ibm.com.