Approximating multi-skill blocking systems by HyperExponential Decomposition

Revised version

Geert Jan Franx, Ger Koole, Auke Pot

Vrije Universiteit, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands

franx@few.vu.nl   –   koole@few.vu.nl   –   SA.Pot@few.vu.nl

http://www.math.vu.nl/obp

23rd August 2005

Abstract

We consider multi-class blocking systems in which jobs require a single processing step. There are groups of servers that can each serve a different subset of all job classes. The assignment of jobs occurs according to some fixed overflow policy. We are interested in the blocking probabilities of each class. This model can be used for call centers, tele-communication and computer networks. An approximation method is presented that takes the burstiness of the overflow processes into account. This is achieved by assuming hyperexponential distributions of the inter-overflow times. The approximations are validated with simulation and we make a comparison to existing approximation methods. The overall blocking probability turns out to be approximated with high accuracy by several methods. However, the individual blocking probabilities per class are significantly more accurate for the method that is introduced in this paper.

keywords: multi-class blocking system, blocking probability, Equivalent Random Method, Hayward-Fredericks Method, Interrupted Poisson Process Method, call center, hyperexponential distribution, decomposition, overflow routing

correspondence to: A. Pot

1 Introduction

An open network is considered with several arrival streams. Each arrival stream is a Poisson process and contains its own type of jobs. The network consists of multiple many-server stations or agent groups. The routing of jobs to servers or agents occurs according to a fixed overflow routing policy. According to this type of policy a call is assigned to the server with the highest priority. These priorities are specified by the policy. Overflow routing implies that: (1) jobs are rejected when all servers with the required skill are busy and (2) jobs leave the network immediately after service completion. Inter-arrival and service times are exponentially distributed with different parameters for different job types. The main subject of this paper is the approximation of the blocking probabilities.

Possible applications can be found in the area of tele-communication, computer networks and call centers. A possible application in tele-communication are joint-ventures between mobile phone service providers to share regions of their bandwidth. At time intervals that a provider is out of bandwidth, additional resource is available in these shared regions. In the area of computer networks we could think of web-hosting. Consider a group of servers that handle the service request of some internet pages. Each server is dedicated to serve the requests of a subset of these pages. Due to economic arguments it may be desired to minimize the total number of servers that treat a certain page. On the other hand, due to service level requirements, it may be attractive that the network also contains servers that can handle multiple pages. This could be modeled similarily. With respect to call centers, an example is optimizing the performance by permutating the servers among the groups. This is desired because it enables call centers to take into account the skills of the available employees in the optimization of their service level. The idea is that the optimal solution for the blocking model will also perform very well if queueing is allowed. Additional information is given in Koole, Pot & Talim [7].

Our approach is to consider each server group separately, as a $(\sum G/M)/s/s$. The notation $(\sum G/M)/s/s$ denotes that there are multiple arrival streams with different inter-arrival and service time distributions. A difficulty is that an exact analysis of the overflow process of the $(\sum G/M)/s/s$ system is extremely complicated. An accurate description of this process is important because it determines the arrival processes of the next groups. Of course, it is desired that also the next groups are described as accurate as possible as a $(\sum G/M)/s/s$ system.

For the reason that not much is known about the $(\sum G/M)/s/s$ system we started thinking of an approximation algorithm. A first step was made by Koole & Talim [8], in which the overflow processes are approximated by Poisson processes. No distinction is made between the different types; weighted average service times per group are taken. This method will be refered to as the Exponential Decomposition (ED) method in this paper. From the research in Koole, Pot & Talim [7] we concluded that a better approximation method is desired. We accomplished this by approximating each group with a $(\sum H_2/M)/s/s$ model (where H_2 denotes order-2 hyperexponential inter-arrival times), and we will refer to this method as HyperExponential Decomposition. An advantage of independent hyperexponential arrival streams is that the first moment of the overflow pro-

cess can be calculated by an exact analysis. Still, obtaining higher moments is difficult, but we succeeded in finding a good approximation. The entire approximation method is treated in this paper. Moreover, the improvement with respect to alternative methods, such as the ED method, is shown.

Several other methods exist that also implicitly take higher moments of the overflow processes into account. These are discussed next.

Equivalent Random Method

The first method we introduce is a well-known method from the literature, called the Equivalent Random Method (ERM), see Cooper [3] pages 165-171. This method serves for approximating blocking probabilities of overflow networks. It is based on a formula for the peakedness of the overflow process of an $M/M/s/s$, developed by Wilkinson [21] and Bretschneider [1] and also given in Kosten [9]. Wilkinson [21] developed approximation tables and he gave examples of the ERM. Jagerman [6] presents an iterative method and generalizes Kosten's formula for systems with general service time distributions. Rapp [13] developed an approximation formula.

While in the literature the ERM is often described to solve networks with two layers, it can be generalized to multi-layer networks, see for instance Tabordon [17]. A shortcoming is that the ERM can only be applied to networks with a tree structure (splitting of overflow traffic is not allowed). Moreover, the ERM does not support different service times for different customer classes. However, there is literature available about this subject, see Schehrer [16].

To overcome the shortcomings in the context of call centers the ERM was generalized to more general overflow routings in Tabordon [17], chapter 4. The generalization was inspired by procedures from other methods, which are presented next.

We note that this extended ERM is not included in our numerical comparison, because the performance is dominated by most of the other methods. For further details, see Tabordon [17].

Hayward-Fredericks Method

The Hayward-Fredericks method (HF) is an extension of the ERM. Hence, we refer the reader to the same literature. This method is also called the Equivalent Congestion Method, see for example Sanders, Haemers & Wilcke [15]. For server groups with Poisson input the procedure of the HF is similar to the ERM. However, if groups in the system receive overflow streams from other server groups as input, the approach is different. While in the ERM multiple groups are replaced by one 'equivalent' group, the HF considers each server group seperately. This became possible due to Hayward and Fredericks [5]. We note that a reference to the work of Hayward cannot be given, because it has never been published on his name. (Therefore, we mention that it is given as reference in several books and papers, see for example Wolff [22] pages 354 and 355.) The contribution of Hayward was that he found a good approximation for the blocking probability of a multi-server group

with peaked input. This enabled approximations of the blocking probability of two-layer networks. Fredericks found an approximation for the peakedness factor of the overflow stream of systems with peaked input, which made it possible to analyze networks with more than two layers. These results, were applied to call centers in Chevalier & Tabordon [2]. But there are many other papers about the Hayward-Fredericks method available in the literature. We will compare our methods to the HF method.

Interrupted Poisson Process Method

The name Interrupted Poisson Process method (IPP) denotes the type of process by which the overflow streams are approximated. The overflow of a group with Poisson input was analyzed by Kuczura [10]. This result was extended by Chevalier and Van Muylder [20] to the overflow process of a group with an IPP as input. It was studied and applied to call centers in the same thesis. His contribution also contains an efficient method to approximate the superposition of n IPP processes and a way to dispatch the overflow process among the next groups. For further details about these methods and a comparison we refer the reader to Tabordon [17] and Van Muylder [20].

HyperExponential Decomposition

The approximation method presented in this paper was developed independently from the work of Chevalier, Van Muylder and Tabordon. It uses different methods for calculating both blocking probabilities and higher moments of the overflow streams. Also, the dispatching of an overflow stream to several destinations is handled in an alternative way.

This paper is structured as follows. In section 2 we introduce the model and notation. An exact analysis is only computable when the total number of states is moderate, e.g., one thousand states. For larger state spaces we suggest the heuristic algorithm as presented in this paper. The main idea behind the algorithm is a reduction of the state space. This is realized by decomposition. The problem size is reduced by considering each server group separately. The arrival processes to each group are approximated by multiple hyperexponential arrival streams of order 2, hence it is denoted as the HyperExponential Decomposition (HED) method. Section 3 presents relevant results about the overflow process of a $M/M/s/s$ system. This basic overflow process plays an important role in the algorithm that is presented in this paper. Section 4 describes the decomposition algorithm; an exact analysis is applied to each group separately. Section 5 shows the benefit of the HED method in comparison to simulation. In Section 6 we refer to Appendix B, which contains a number of numerical examples. Further, we will show that the HED method is more accurate than the ED method and the benefit is in certain cases huge. We also consider the other methods: the HF method and the IPP method. This is shown by means of several examples. Sections 7 gives the conclusions, possible extentions and other ideas.

4

2 Model description

The model is depicted by Figure 1, which is the graphical representation of the considered blocking system. This figure completely defines the system, within our model. Each arrow

$$\mu(1) = 1 \qquad \mu(2) = 1 \qquad \mu(3) = 1 \qquad \mu(4) = 1$$

s(7)=5

s(6)=4

s(5)=4

s(1)=3   s(2)=3   s(3)=3   s(4)=3

$$\lambda(1) = 5 \qquad \lambda(2) = 5 \qquad \lambda(3) = 5 \qquad \lambda(4) = 5$$
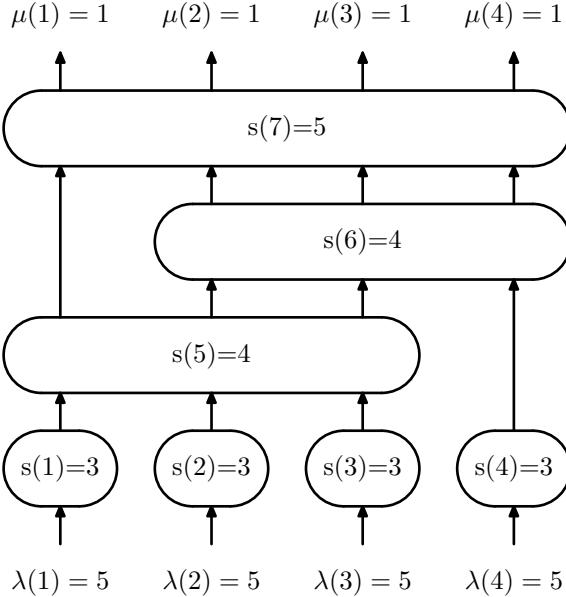
Figure 1: Illustration of the call center model

at the bottom represents the arrival process of a certain job type (or class). Process i is a Poisson process with rate $\lambda(i)$, where i denotes the job type. The service rate $\mu(i)$ of each job type is given at the top. These are the parameters of the exponential distributions. In addition, each circle represents an server group. The number $s(j)$ of servers in group j is given in the middle of each circle. The incoming Poisson streams, as well as the overflow streams, are depicted by arrows. Job assignment occurs according to a fixed overflow route. This means that for each job type there exists an ordering of groups such that a job is assigned to the first group with an available server that has the required skill. Jobs are rejected when all servers along the overflow route are busy. Served jobs leave the system immediately.

The following notation is used:

$\mathcal{C} \subset \mathbb{N}$ is the index set of job classes. All jobs from a certain class are generated by the same 'source'. Therefore, $\mathcal{C}$ will also be refered to as the set of sources.

$\mathcal{G} \subset \mathbb{N}$ is the index set of server groups.

$\mathcal{C}^g$ is the sub-set of job classes that server group g can handle, $g \in \mathcal{G}$ and $\mathcal{C}^g \subset \mathcal{C}$.

$\mu(c)$ specifies the service rate of class c, $c \in \mathcal{C}$. The service time distributions are assumed to be exponential. In this paper we assume that the service rates do not depend on the server or server group. However, this assumption is not essential. The hyperexponential decomposition method can also be applied to a system with different service rates for different groups.

$\lambda(c)$ is rate of the Poisson process by which jobs from class c arrive.

$s(g)$ specifies the size of server group g, $g \in \mathcal{G}$.

$g(c, j)$ denotes the j-th server group in the overflow routing policy for source c, $c \in \mathcal{C}$ and $g(c, j) \in \mathcal{G}$.

The blocking system in Figure 1 could be described as:

$\mathcal{C} = \{1, 2, 3, 4\}$.

$\mathcal{G} = \{1, 2, 3, 4, 5, 6, 7\}$.

$\mathcal{C}^i = \{i\} \quad \forall i \in \mathcal{C}, \quad \mathcal{C}^5 = \{1, 2, 3\}, \quad \mathcal{C}^6 = \{2, 3, 4\}, \quad \mathcal{C}^7 = \{1, 2, 3, 4\}$.

$\mu(c) = 1 \quad \forall c \in \mathcal{C}$.

$\lambda(c) = 5 \quad \forall c \in \mathcal{C}$.

$s(1) = s(2) = s(3) = s(4) = 3, \quad s(5) = s(6) = s(7) = 4$.

$g(1, \cdot) = (1, 5, 7), \quad g(2, \cdot) = (2, 5, 6, 7), \quad g(3, \cdot) = (3, 5, 6, 7), \quad g(4, \cdot) = (4, 6, 7)$

This model formulation is quite different from the one considered by Van Muylder [20] and Tabordon [17]. In our paper the service rates only depend on the job class, whereas in their work they are group dependent and job independent. However, including also group dependent service rates creates no fundamental problems in our method. Moreover, we consider overflow policies that are deterministic, instead of randomized. Deterministic overflow policies route each job through the network in a deterministic manner, while randomized policies choose the next group randomly with some fixed probabilities. In Tabordon [17] these probabilities are optimized such that the workload is balanced among the different agent groups. This load balancing is essential for their method to work. On the other hand, we also consider unbalanced systems, which often occur in reality. However, including randomization creates no fundamental problems in our method.

Model Limitation

Concerning the approximation algorithm that is discussed in Section 4 it should hold that server groups can be numbered in such a way that

$$g(c, i) < g(c, j) \Leftrightarrow i < j \qquad \forall c \in \mathcal{C} \quad \forall i, j \in \mathbb{N}.$$

This means that it must be possible to arrange Figure 1 in such a way that there are only upward pointing arrows.

3 Fitting the overflow process of the $M/M/s/s$ model

When the inter-arrival times of jobs to a group with identical servers are independent, it is possible to calculate the first moments of the overflow stream analytically. This holds for server groups in the first layer of the overflow routing network. In addition, we know that the independence of the inter-arrival times also holds if the arrival stream of a group is a renewal process originating from exactly one first-layer group. In general, if a group receives calls that destinate from several groups, the independence of the inter-arrival times does not hold anymore. In other words, the superposition of multiple arrival processes is no longer a renewal process. Since we are interested in an algorithm that supports this type of routing, an exact analysis is very difficult. Next, we elaborate on the calculation of the first moments of the $M/M/s/s$. The results are important for the algorithm that is introduced in Section 4.

The real overflow process of the $M/M/s/s$ system is a renewal process with hyperexponential distribution of order $s + 1$ with density (see Riordan [14] page 39)

$$f(t) = \sum_{i=0}^{s+1} p_i \gamma_i e^{-\gamma_i t}. \tag{1}$$

This section deals with fitting this hyperexponential distribution to a lower order hyperexponential distribution of order $\bar{s}, \bar{s} \leq s$. This reduces the size of the state space to such an extend that the model becomes tractable for blocking systems of realistic sizes.

The method is discussed from a numerical point of view, because analytical expressions of the exact distribution get too big and cannot be solved efficiently. In this calculation we distinguish three steps: (I) determine the coefficients of the real H_{s+1} distribution (given the arrival rate, service rate and the number of servers), (II) determine the first $2\bar{s} - 1$ moments and, finally, (III) determine the coefficients of the fitted $H_{\bar{s}}$ distribution. As an alternative, steps (I) and (II) could be replaced by estimating the moments with a simulation table of a $M/M/s/s$ system. This is explained later in this section. First, the three steps are discussed:

(I) We refer the reader to Riordan [14], pages 36-39. On these pages the relevant equations are given. The computation involves a partial fraction expansion of a division of two Poisson-Charlier polynomials. This can be solved numerically with standard methods, e.g. the Heaviside cover-up method, which includes a method for finding roots. Mathematical software packages like Maple or Matlab are also capable to handle them. In our implementation we obtained accurate solutions with groups up to ten servers.

(II) The n-th moment of a random variable X with the density function of Equation 1 is determined by

$$\mathbb{E}\bar{X}^n = \sum_{i=0}^{2\bar{s}-1} n! \frac{p_i}{\gamma_i^n}.$$

This is obtained by using the Laplace transform or moment generating function.

(III) The coefficients of the fitted $H_{\bar{s}}$ are in general hard to calculate, analytically as well as numerically. For H_2 we get (see Tijms [19], page 360)

$$\gamma_{1,2} = \frac{1}{2}\left\{a_1 \pm \sqrt{a_1^2 - 4a_2}\right\}, \quad p_1 = \frac{\gamma_1(1 - \gamma_2 m_1)}{\gamma_1 - \gamma_2}, \quad p_2 = 1 - p_1, \quad (2)$$

where

$$a_2 \equiv \frac{6m_1 - 3m_2/m_1}{3m_2^2/2m_1 - m_3}, \quad a_1 \equiv \frac{1}{m_1} + \frac{m_2}{2m_1}a_2$$

and

$$m_n \equiv \mathbb{E}\bar{X}^n.$$

This holds when $m_1 m_3 \geq \frac{3}{2}m_2^2$. There are no such explicit expressions for $\bar{s} > 2$.

Simulation is an alternative way with regard to the steps (I) and (II) of the analytical method. In first instance this seems to be unattractive, because a drawback of simulation (and even of both numerical methods) are long computation times. Therefore we suggest to do these calculations in advance.

We calculated the moments for different parameter combinations, a finite number of pairs (s, a), with a the workload and s the number of servers. We let s vary between 0 and 30 and λ from 0 to 50 with steps of $1/8$ and fixed $\mu = 1$. The moments for other values of μ are found by rescaling. The outcomes are stored on hard disk and are loaded before the algorithm is executed. During the algorithm interpolation is applied to find approximations for real values of a.

There exists a completely different approach to obtain an exact expression for the first $\bar{s}$ moments of the overflow process of a $M/M/s/s$ system. It is based on Takács [18] and an essential part of the IPP method. It is applied by Van Muylder [20] and also refered to and discussed by Tabordon [17]. Because it is also relevant to the algorithm that is presented in this paper it will be summarized next. The idea behind the approximation method is to compare two systems, denoted with A and B. System A is a multi-server group with Poisson input, and blocked calls are served by an infinite-server group. System B is an infinite-server group and its inter-arrival time distribution is hyperexponential. The arrival time distribution in system B is approximated by using the first $\bar{s}$ moments of the overflow distribution in system A. This is achieved by equating the first three moments of the number of jobs in the infinite-server group. In Van Muylder [20] it is assumed that the arrival process of system A is an IPP. In this paper we consider this approach to determine the overflow of a group with Poisson input. Since a Poisson process is a special case of an IPP we could use their results.

In our software implementation we have chosen to determine the moments with a simulation table for group sizes bigger than ten servers. For group sizes smaller than ten the exact method for the computation of the first three moments was used (based on Takács). Moreover, the simulation table was filled in advance, and only once. The reason for using these tables is shorter computation times. We remark that the outcomes with the three different methods are all very accurate.

4 HyperExponential Decomposition algorithm

In this section we give a description of the algorithm. During the algorithm the overflow streams are approximated by hyperexponential distributions. This is depicted in Figure 2 for an arbitrarily chosen instance.
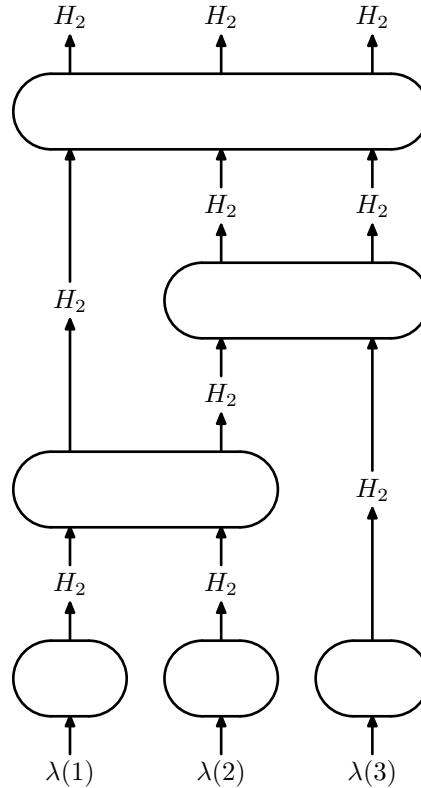
Figure 2: Illustration of the decomposition

First we consider the most basic case of a call center with 2 layers; layer 1 consists of specialists and layer 2 of generalists. The algorithm processes the server groups separately and successively, starting with the groups in the first layer. Each group of specialists is considered as a $M/M/s/s$ system. This is an exact representation within the model. We are interested in the overflow process of each group. These are approximated by 2nd-order

9

hyperexponential distributions. The approximation requires the first three moments of the overflow process. These moments determine the coefficients of the hyperexponential distribution, see Equation (2) in Section 3. The hyperexponential distributions are taken as input to the group of generalists in layer 2. Next, the group of generalists is considered separately as a system with hyperexponential arrival streams, which can be modeled as a birth-death process. By solving the balance equations the blocking probability is determined, see Section 4.3 and Appendix A.

The algorithm that holds for the general model from Section 2 is more extensive. With regard to the previous example two problems can arise and need to be solved. The first one concerns the splitting of overflow streams. The meaning of splitting is explained next, also by means of an example. Consider a call center with the generalists in the first layer and the specialists in the second layer. The essential difference with the previous example is that the overflow process of the generalists go to multiple groups. Since initially one overflow process is determined, this process must be split up to the different groups in the next layer. This type of construction is supported in the general algorithm, see Section 4.5.

Another problem becomes visible when we consider a call center with three layers. The question is what the overflow process from the groups in layer 2 looks like. In the description above only the calculation of the blocking probability of these groups is treated. But additional information is required to compute the parameters of the hyperexponential distributions that characterize the overflow processes to the groups in layer 3. More specifically, the first three moments are required. Therefore, an equivalent random process is taken, derived from the $M/M/s/s$ system. Also this will be treated in the remainder of this section, see Section 4.4.

The pseudo-code of the hyperexponential decomposition algorithm is presented next.

Heuristic algorithm()
```
 1    determine the level of each group, define L as the highest level
 2  for  level 0 up to L
 3      do for  each group in current level
 4            do - calculate the weighted average service rate, per arrival stream
 5                - calculate analytically the blocking probability,
 6                  assuming H_2 arrival processes
 7                - determine the second and third moment of the
 8                  overflow process.
 9                - approximate and dispatch the fitted overflow process
10                  to the next groups
```

The steps at lines 4-10 are discussed in the next sections. The number between the brackets refers to these lines.

4.1 Determining the level of each group (line 1)

The level of each group is calculated with an iterative method.

4.2 Weighted average service rate (line 4)

Consider some arbitrary server group g with skill set $\mathcal{C}^g$. If jobs from different classes arrive from the same preceding group, their arrival processes are no longer independent. An exact analysis of those arrivals will be hard. However, we do not wish to exclude these arrival streams, since they are abundant in real life overflow systems. Therefore we use an approximation in which this type of arrival stream is modeled as one hyperexponential renewal process with one exponential service time distribution. Here we use the weighted average service rate of the jobs originating from the same preceding group. The weighted average service rate of the overflow stream from group i to group g, denoted as $\bar{\mu}_{ig}$, is then determined by

$$\bar{\mu}_{ig}(g) = \sum_{c \in \mathcal{C}^g \cap \mathcal{C}^i} \lambda_{igc} \left(\sum_{c \in \mathcal{C}^g} \frac{\lambda_{igc}}{\mu(c)} \right)^{-1}, \tag{3}$$

where λ_{igc} is the overflow rate of jobs of class c from group i to group g. In case that an inflow stream of group g originates directly from outside, we replace λ_{igc} by $\lambda(c)$.

4.3 Calculating the overflow rate (line 5)

Let us consider server group g, of size $s(g)$. This section describes the calculation of the blocking probabilities and the overflow rate of this group, as illustrated in Figure 3. In the algorithm the inter-arrival times of the overflow streams to group g are fitted by hyperexponential distributions of order 2. If we assume the arrival streams originating from different groups to be independent from one another, we can model the sub-system of group g as a birth-death process, in which each arrival stream has two possible states. Owing to this, the number of states of the system that is relevant for group g becomes $m(g)2^{n(g)}$ with $n(g)$ the number of immediately preceding groups from which the arrival stream originate and $m(g)$ the number of states of group g. A more extensive explanation is given in Appendix A.

Inaccuracy is caused by the fact that we determine a weighted average service rate of the jobs that originate from the same group, as described in Section 4.2. This averaging of the service times is not fundamental to the method but reasonable. It is included for the sake of reducing the state space of decomposed group g and can therefore be left out when the number of servers in group g is relatively low.

To obtain the blocking probabilities, we solved the balance equations, see Appendix A. The overflow rates follow directly from the equilibrium state probabilities. P_{jig} denotes the total stationary probability of the source generator from i to g being in state j, while at the same time all servers in group g are busy. Now, $\lambda_{g,i}$, the total overflow rate at group
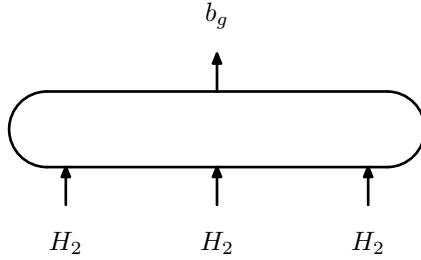
Figure 3: Blocking probability for group g

g of jobs originating from group i can be expressed as

$$\lambda_{g,i} = \sum_{j=1}^{2} P_{jig}\gamma_{jig},$$

where γ_{jig} is the rate of the generator from group i to group g when this generator is in state j. In case of combined input streams, the λ_{igc} as defined in Section 4.2 can be calculated by assuming that the fractions of different classes in combined streams remain equal. The blocking overall probability up to group g becomes

$$b_g = \frac{\displaystyle\sum_{c\in\mathcal{C}^g} \lambda_{igc}}{\displaystyle\sum_{c\in\mathcal{C}^g} \lambda(c)}$$

and will be used in Section 4.4.

4.4   Determining the second and third moment of the overflow process (line 7)

In this section a method is described for finding an approximation for the second and third moment of the inter-overflow times of group g. We assume that the blocking probability of group g has been determined according to the previous section. This probability will be denoted by b_g.

The second and third moment are determined next, by interpolation. See Figure 4 for an illustration. The general idea is to compare the complete overflow process up to group g to an $M/M/s/s$ system with the same blocking probability, under the assumption that the burstiness is then comparable as well. Firstly, by using $B(s,a)$, the Erlang loss formula, we determine a lower and upper bound for s as follows

$$s_1 := \max\{s \in \mathbb{N} : B(s,a) \geq b_g\}$$
$$s_2 := \min\{s \in \mathbb{N} : B(s,a) \leq b_g\}$$

12
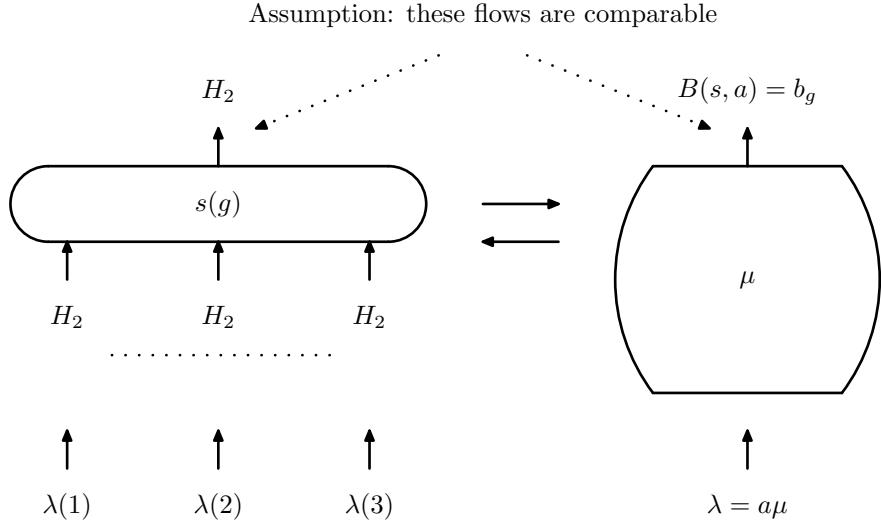
Figure 4: Second and third moment estimation by comparison

with

$$a := \sum_{c \in C^g} \frac{\lambda(c)}{\mu(c)}.$$

Secondly, for both s_1 and s_2 the first three moments of the overflow processes are determined according to the formula from the $M/M/s/s$ system, as described in Section 3. Finally, linear interpolation is applied to obtain the final values for the moments. This is done in such a way that the first moment matches with b_g.

We conclude from our experiments that the accuracy of the higher moments is comparable to the accuracy of b_g.

4.5 Determining overflow processes to the next groups, dispatching (line 9)

In line 7 for each group the first three moments of the inter-overflow times are calculated. These moments are used to fit a hyperexponential distribution of order two uniquely, of which the density function is given in Equation 1. In general this overflow stream will contain different type of jobs. We need to keep in mind that the overflow processes of these different job types are not independent. Therefore, dispatching of overflow streams will only take place between different destinations, not between job types. So, if the overflow stream is directed to only one higher level group, there is no need for dispatching, even if there are different job classes in the stream. On the other hand, if the overflow stream is directed to l different higher level groups, the stream needs to be split into l substreams. Each of these substreams may contain several job classes, see Figure 5 for the case $l = 3$. In general, an exact analysis of this dispatching problem is extremely

13
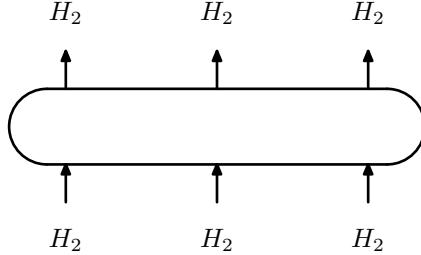
Figure 5: Dispatching

complicated. However, it turns out that excellent results can be obtained by assuming that the dispatched substreams are renewal processes with 2nd-order hyperexponential inter-event times. For the parameters of these hyperexponential distributions we use the same p_i's as found for the hyperexponential fit of the total overflow stream. For the γ_i's of the substream from group g to k, the γ_i's of the total overflow stream are multiplied by p_{gk}, the fraction of the total overflow stream from group g to k that is directed to group k. This fraction is determined by the following summation:

$$p_{gk} = \frac{\displaystyle\sum_{c \in \mathcal{C}^g \cap \mathcal{C}^k} \lambda_{gkc}}{\displaystyle\sum_{c \in \mathcal{C}^g} \lambda_{gkc}},$$

where λ_{gkc} is the overflow rate of jobs of class c from group g to group k. Here, we define 'rate' as the inverse of the expectation of the concerning inter-event times.

If the inflow of group g consists of a separate hyperexponential stream per skill type c, then λ_{gkc} follows directly from Appendix A. In case that an inflow stream of group g contains multiple job classes, then from the exact analysis, only the total overflow rate per input stream follows. Therefore, we assume that the fractions remain equal in the inflow and outflow. For example, consider the case that $\mathcal{C}^k$, the skill set of group k, cannot be composed by taking the union of job classes from inflow streams of group g. Then, the λ_{gkc} is obtained by taking the overflow rate of the input stream that contains class c and multiplying this rate by the probability that a job in the inflow stream is from class c.

There is also literature available about splitting streams of traffic, see for example de Boer [4] and Rajaratnam & Takawira [12]. The methods are not applicable to our model because their routing policies are very limited and multiple skills and different service rates are not supported.

5   Comparison to simulation (computation times)

The first thing that we would like to mention is that in this paper simulation is used to validate the approximation methods with respect to accuracy. It is considered to be the
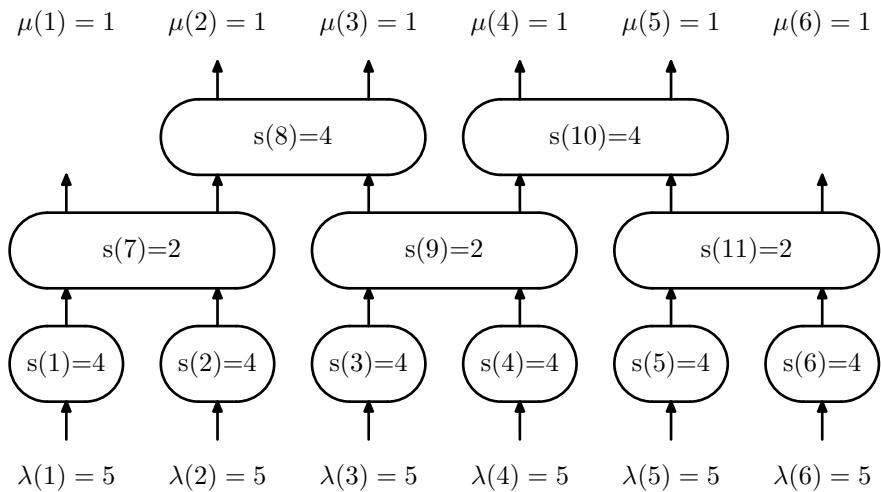
Figure 6: High dimensional case
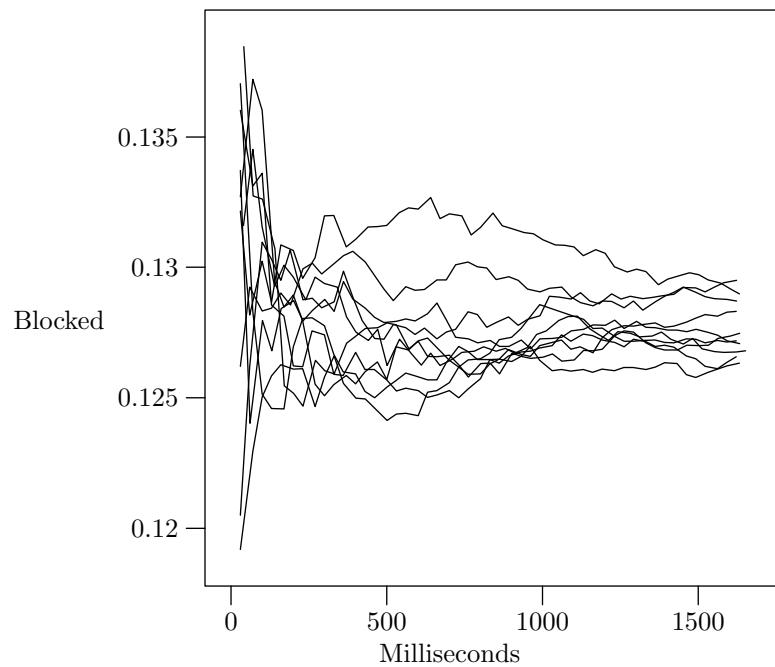
Figure 7: Sample paths of simulation runs

true value since simulation is the only way to carry out the calculations without un-proven assumptions.

In this section we focus on the computation times of the HED method and simulation, especially on the differences between both methods. The accuracy will be compared in Section 6.

The comparison is made by analyzing the call center from Figure 6. All computations were executed on a Asus laptop with an Intel Pentium III 1066 Mhz processor. Execution of the HED method took 1.7 millisecond. The estimation of the blocking probability by simulation is displayed against time in Figure 7. Totally, each run consists of 1.5×10^6 simulated events. We observe that the blocking probabilities converge slowly. Even after one second the two most extreme probabilities of ten sample paths differ around 0.005. The relative difference is about 4 %. In comparison to the HED method, within 1.7 millisecond a more accurate approximation is found. Consequently, this method is by far superior if fast computation times are required.

6 Numerical results

Numerical examples are presented in Appendix B. A total of 18 instances are treated by comparing the different evaluation methods. These methods are: simulation, the equivalent random method, the ED method, the HF method, the IPP method and the HyperExponential Decomposition method, as introduced in this paper. In the table below each figure the blocking probabilities per skill are given, as well as the weighted average. The instances are chosen in such a way that the diversity is high. The first 7 instances have equal service rates. Instances 8-14 have more variation in the service rates among the different classes. Finally, instances 15-18 contain more server groups and skills than the preceding ones.

The deviations in blocking probability with regard to simulation are presented in Table 1 for five different methods: the ERM, the HF method, the IPP method, the ED method and the HED method. The examples correspond to the instances from Appendix B. Concerning the ERM, in our implementation the method given by Rapp is not used, see Cooper [3] page 170. Instead s and a are determined from a table. As we mentioned in Section 1 the ERM could only be applied to a few instances, with a special routing network.

Table 1 shows that the IPP and the ED method are less accurate than the HF and HED method. The HF method is easier to implement and can handle larger server groups with many skills.

Finally, we make a short note about the weakness of the ED method that was introduced in an earlier paper of Koole & Talim [8]. The ED method performs poorly in call centers with many layers in the overflow routing policy and many servers. See Figure 8 for an example. According to the ED method the blocking probability is only 0.4%, while simulation yields 4.3%. (The HED, takes higher moments into account, gives an approximation of 4.2%.)

Method	Relative deviation (%)				
	ERM	HF	IPP	ED	HED
1	1.1	1.3	3.2	31.9	0.0
2	4.6	1.4	4.8	24.0	0.5
3	-	3.0	9.5	21.6	0.7
4	2.3	1.7	7.4	20.0	1.3
5	-	3.1	31.0	44.2	3.1
6	-	4.2	20.2	35.3	0.8
7	0.4	4.2	0.1	68.0	0.2
8	-	1.6	0.6	23.6	0.1
9	-	2.0	5.6	19.4	0.6
10	-	1.3	22.7	37.8	0.7
11	-	0.1	9.6	25.6	0.2
12	-	1.5	21.9	30.3	2.7
13	-	6.0	11.2	27.4	1.5
14	-	12.7	72.4	79.9	2.3
15	-	2.8	1.0	3.6	0.7
16	-	8.1	19.1	24.8	8.8
17	-	5.4	24.8	29.9	6.5
18	-	3.0	8.7	15.5	0.0

Table 1: Accuracy with regard to simulation

It is obvious that the inaccuracy of the ED method is caused by the fact that here the distribution of the interoverflow times is assumed to be exponential. Owing to this assumption, the burstiness is underestimated. This shortcoming was already discussed in Koole, Pot & Talim [7].

$$\mu(1) = 1$$

$$\lambda(1) = 30$$
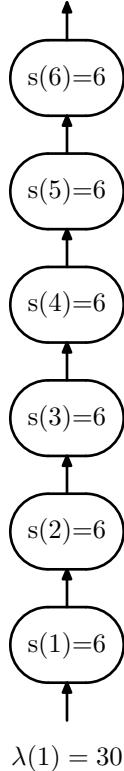
Figure 8: A bad case of the ED method

7 Conclusions and further research

The HED method as presented in this paper yields the most accurate approximations for the overall blocking probability as well as the individual blocking probabilities for each class.

Concerning the HED, taking weighted average service rates per group, instead of an exact analysis, has not much influence on the quality of the approximation. On average the quality gets a bit worse.

The approximation methods like the HF method and HED are in general by far superior to simulation because of the short computation time. This is discussed and illustrated in Section 5.

A shortcoming of the HED is that call centers with big groups and having many skills is computational hard to solve. In that case the HF method is a good alternative. This is related to the fact that an exact analysis of a group in the decomposition becomes intractable when the state space grows.

A possible extension is to support service rates that are group dependent into the model. This has already been implemented in our software, but is not considered in this paper. Service rates that are also group dependent is in some cases very useful. An example is multi-skill call centers. Agents with more skills are less efficient in comparison to specialists. The reason is that the work of generalists varies excessively.

Currently, the overflow processes are split in the HED method in such a way that only the first moment is approximated correctly, see Section 4. A possible improvement is to split the hyperexponential distributed streams in a more advanced and more accurate manner.

At several places in the HED method interpolation is applied. We suggest to investigate other interpolation techniques than linear interpolation because it is likely that improvement is possible.

When the performance is low (because group sizes are big or there are many arrival streams), improvement of performance is possible by splitting servers groups in multiple parts and putting them behind one another in the overflow routing network. On the other hand the accuracy decreases somewhat, see instances 7 and 8 from Appendix B, but it remains satisfying.

A possibility is to support randomized routing in the HED method. Randomized routing means that for a job type the overflow process from a group is next directed to multiple other groups in the routing policy, in a randomized manner. This extension is straightforward to implement.

References

[1] G. Bretschneider. Die Berechnung von Leitungsgruppen für überfliessenden Verkehr in Fernsprechwahlanlagen. *Nachrichtentechnische Zeitschrift*, 9:533–540, 1956.

[2] P. Chevalier and N. Tabordon. Overflow analysis and cross-trained servers. *International Journal of Production Economics*, 85:47–60, 2003.

[3] R.B. Cooper. *Introduction to Queueing Theory*. North Holland, 2nd edition, 1981.

[4] J. de Boer. Blocking of overflow traffic components. In *International Teletraffic Congress*, volume 11, 1985.

[5] A. Fredericks. Congestion in blocking systems-a simple approximation technique. *The Bell System Technical Journal*, 59(6):805–827, 1980.

[6] D.L. Jagerman. Methods in traffic calculations. *AT&T Bell Laboratories Technical Journal*, 63(7):1291–1309, 1984.

[7] G.M. Koole, S.A. Pot, and J. Talim. Routing heuristics for multi-skill call centers. In *Proceedings of the Winter Simulation Conference*, pages 1813–1816, 2003.

[8] G.M. Koole and J. Talim. Exponential approximation of multi-skill call centers architecture. In *Proceedings of QNETs 2000*, pages 23/1–10, 2000.

[9] L. Kosten. *Stochastic Theory of Service Systems*. Pergamon, Oxford, England, 1973.

[10] A. Kuczura. The interrupted poisson process as an overflow process. *Bell System Technicl Journal*, 52(3):437–448, 1973.

[11] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 2002.

[12] M. Rajaratnam and F. Takawira. A two-moment analysis of adaptive routing. case study: Dynamic control routing. In *International Teletraffic Congress*, volume 15, pages 199–210, 1997.

[13] L.Y. Rapp. Planning of junction networks in a multi-exchange area. *Ericcson Technics*, 20(1):77–130, 1964.

[14] J. Riordan. *Stochastic Service Systems*. Wiley, 1961.

[15] B. Sanders, W.H. Haemers, and R. Wilcke. Simple approximation techniques for congestion functions for smooth and peaked traffic. In *International Teletraffic Congress*, volume 10, 1983.

[16] R.G. Schehrer. A two moment method for overflow systems with different mean holding times. In *International Teletraffic Congress*, volume 15, pages 1303–1314, 1997.

[17] N. Tabordon. *Modeling and Optimizating the Management of Operator Training in a Call Center*. PhD thesis, Université catholique de Louvain, 2002.

[18] L. Takács. *Introduction to the Theory of Queues*. Oxford University Press, 1962.

[19] H.C. Tijms. *Stochastic Models. An Algorithmic Approach*. Wiley, 1986.

[20] N. van Muylder. Phénomènes de pertes et de temps d'attente dans un call-center. Master's thesis, Université catholique de Louvain, 2001.

[21] R. Wilkinson. Theories for toll traffic engineering in the U.S.A. *Bell System Technical Journal*, 35(2):421–514, 1956.

[22] R.W. Wolff. *Stochastic Modeling and the Theory of Queues*. Prentice Hall, Inc., New Jersey, 1989.

A Exact Analysis

The model describes a decomposed group $g \in \mathcal{G}$, resulting from the decomposition method. The arrival streams are modeled as independent from one another. For each stream the interarrival times are assumed to be hyperexponential. Concerning the groups in the first layer, note that the exponential distribution of Poisson processes fit in the family of hyperexponential distributions. An advantage of the hyperexponential distribution is that it can be modeled as a generator with two different states. The state of the generator is a stochastic variable and it changes when a call is generated, independently of the history and the current state. In each state the time until the next call is generated, is exponentially distributed. We introduce some additional variables:

- $\mathcal{A}_g$ is the set of indices of all preceding groups of group g, $\mathcal{A}_g \subset \mathcal{G}$.

- $n(g) := |\mathcal{A}_g|$, the number of elements in set $\mathcal{A}_g$.

- $p(i,j)$ is the coefficient of the j-th term of the hyperexponential distribution describing the overflow stream from group i to g, $i \in \mathcal{A}_g$ and $j \in \{1,2\}$. It is the probability that the generator goes to state j immediately after the epoch that a call flows over from group i.

- $\lambda(i,j)$ specifies the rate of the j-th order term of the hyperexponential distribution of the stream from group i to j, $i \in \mathcal{A}_g$ and $j \in \{1,2\}$. It is the rate of the exponential distribution when the generator is in state j.

- $\bar{\mu}_{ig}^{-1}$ is defined conform Equation 3.

The state of the decomposed subsystem of group g consists of the state of group g and each source $i \in \mathcal{A}_g$. The state of group g describes the number of busy servers per job class. If some classes have the same service rate, then the size of the state could be reduced. This is not worked out in this section, but it is implemented in our code. The number of different states for group g, denoted by m, is

$$m := \binom{n(g) + s(g)}{n(g)}.$$

The state dimension of a source is 1 and the number of different values it can take is 2, the order of the used hyperexponential distribution. Consequently, k, the total number of states is

$$k = m \times 2^{n(g)}.$$

We denote the state by (x,y) and it consists of the state vector of the arrival streams and the state vector of group g:

(I) $x(i)$ is the state (a positive integer) of source i. It should hold that

$$x(i) \in \{1,2\}.$$

(II) y is the state vector of group g and element $y(i)$ describes the number of jobs from preceding group i that are in service at group g. For a feasible state it holds that

$$\sum_{i \in \mathcal{A}_g} y(i) \leq s(g).$$

Transitions

We distinguish three different types of transitions:

(I) Assignment: a call is assigned to an idle server.

(II) Blocking: a call is lost because all relevant servers are busy.

(III) Completion: an server completes the service of call.

The first two types concern job arrivals. When the state of the group satisfies

$$s(g) > \sum_{i \in \mathcal{A}_g} y(i) \tag{4}$$

the job is assigned to an idle server. Otherwise, when

$$s(g) = \sum_{i \in \mathcal{A}_g} y(i), \tag{5}$$

the job is blocked. The three different transitions will be discussed next.

Assignment

Consider the transitions $(x, y) \to (\bar{x}, \bar{y})$ for arrival stream $i \in \mathcal{A}_g$ such that

$$\bar{y}(i) = y(i) + 1,$$

which is only possible when Equation 4 is satisfied. The values of $x(j)$ and $\bar{x}(j)$ may be different, because the state of source c may change when a call arrives. Further, with the exception of these two differences, both states are equal. The transition rate is

$$\lambda(i, x(i))p(i, \bar{x}(i)).$$

These transitions can be found for all $i \in \mathcal{A}_g$.

Blocking

If all servers that can handle a certain call type are busy, arriving calls are rejected. As a result only, the state of the source may change.

Consider the transitions $(x, y) \to (\bar{x}, y)$ for arrival stream $i \in \mathcal{A}_g$ such that Equation 5 is satisfied. The transition rate is

$$\lambda(i, x(i))p(i, \bar{x}(i)).$$

Completion

Consider the transitions $(x, y) \to (x, \bar{y})$ with the only difference

$$\bar{y}(i) = y(i) - 1$$

for $i \in \mathcal{A}_g$. This transition is possible whenever $y(i) > 0$. The transition rate is

$$y(i)\bar{\mu}_{ig}.$$

This holds for all $i \in \mathcal{A}_g$.

Equilibrium probabilities

The balance equations are composed by the transitions that are defined in the previous sections. All transition probabilities are put in a matrix. For efficiency, the inverse of this matrix is calculated by decomposing the matrix in a lower and upper triangular matrix, called a LU-decomposition. Next, the inverse of both matrices is calculated. This can be done very efficiently. Finally, multiplying the inverses of U and L yields the inverse of the original matrix. For details, see Numerical Recipes in C [11]. The calculation time can possibly be reduced by approximating the steady-state probabilities. Appropriate techniques are successive matrix multiplications and value iteration.

B Numerical comparison

Several approximation methods have been applied to a total of 18 blocking systems, which are represented by figures. Below each figure the blocking probabilities are given. These were calculated in six different ways: simulation (SIM), Exponential Decomposition (ED), HyperExponential Decomposition (HED), the Equivalent Random Method (ERM), Hayward-Fredericks method (HF) and the Interrupted Poisson Process method (IPP).

In order to make a fair comparison between the different approximation methods, a large variety of routing policies, arrival rates, group sizes and service rates are considered. The parameters are chosen such that the service level would be realistic in case of a delay system.

The sign '-' means that is was not possible to obtain the approximation. The ERM could not always be applied because it requires a special type of overflow routing policy.

Instance 1

$$\mu(1) = 1 \qquad \mu(2) = 1$$

s(3)=4

s(1)=5   s(2)=5
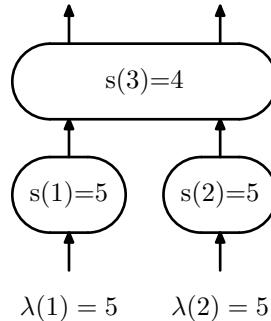
$$\lambda(1) = 5 \qquad \lambda(2) = 5$$

		skills	
method	1	2	
SIM	0.0792	0.0791	0.0791
ED	0.0539	0.0539	0.0539
HED	0.0791	0.0791	0.0791
ERM	0.0800	0.0800	0.0800
HF	0.0781	0.0781	0.0781
IPP	0.0766	0.0766	0.0766

Instance 2

$$\mu(1) = 1 \qquad \mu(2) = 1 \qquad \mu(3) = 1$$

s(4)=5

s(1)=5   s(2)=5   s(3)=5

$$\lambda(1) = 5 \qquad \lambda(2) = 5 \qquad \lambda(3) = 5$$
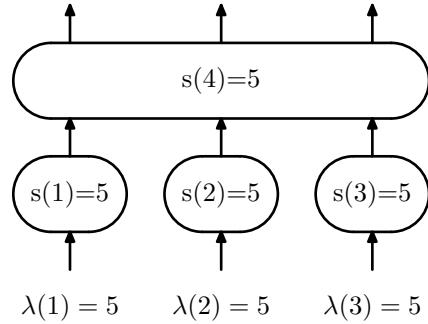
	skills			
method	1	2	3	
SIM	0.0836	0.0837	0.0839	0.0837
ED	0.0636	0.0636	0.0636	0.0636
HED	0.0833	0.0833	0.0833	0.0833
ERM	0.0876	0.0876	0.0876	0.0876
HF	0.0825	0.0825	0.0825	0.0825
IPP	0.0797	0.0797	0.0797	0.0797

Instance 3

$$\mu(1) = 1 \qquad \mu(2) = 1 \qquad \mu(3) = 1$$

s(6)=3

s(5)=4

s(4)=4

s(1)=2   s(2)=2   s(3)=2

$$\lambda(1) = 5 \qquad \lambda(2) = 5 \qquad \lambda(3) = 5$$
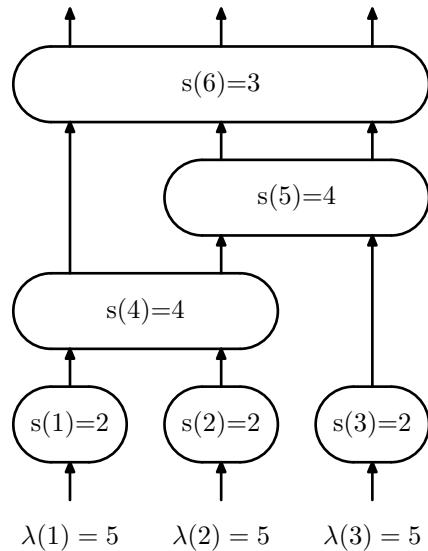
	skills			
method	1	2	3	
SIM	0.1737	0.0971	0.1527	0.1412
ED	0.1509	0.0615	0.1195	0.1106
HED	0.1744	0.0933	0.1586	0.1421
ERM	-	-	-	-
HF	0.1903	0.0852	0.1605	0.1453
IPP	0.1702	0.0734	0.1394	0.1277

Instance 4

$\mu(1) = 1$ $\mu(2) = 1$ $\mu(3) = 1$ $\mu(4) = 1$

s(7)=3

s(5)=4   s(6)=4

s(1)=3 s(2)=3 s(3)=3 s(4)=3

$\lambda(1) = 5$ $\lambda(2) = 5$ $\lambda(3) = 5$ $\lambda(4) = 5$
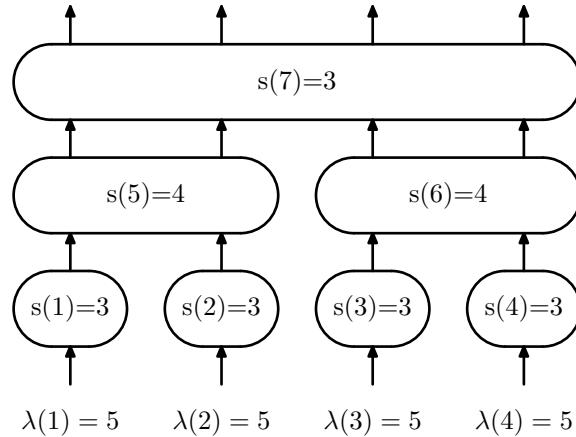
	skills				
method	1	2	3	4	
SIM	0.1368	0.1365	0.1363	0.1368	0.1366
ED	0.1092	0.1092	0.1092	0.1092	0.1092
HED	0.1383	0.1383	0.1383	0.1383	0.1383
ERM	0.1397	0.1397	0.1397	0.1397	0.1397
HF	0.1389	0.1389	0.1389	0.1389	0.1389
IPP	0.1264	0.1264	0.1264	0.1264	0.1264

Instance 5
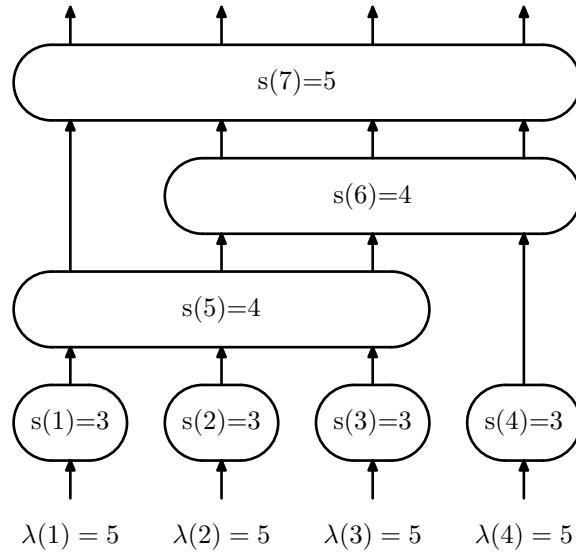
	skills				
method	1	2	3	4	
SIM	0.0915	0.0612	0.0621	0.0837	0.0746
ED	0.0620	0.0278	0.0278	0.0486	0.0415
HED	0.0889	0.0633	0.0633	0.0918	0.0768
ERM	-	-	-	-	-
HF	0.1076	0.0540	0.0540	0.0915	0.0768
IPP	0.0749	0.0352	0.0352	0.0602	0.0514

Instance 6
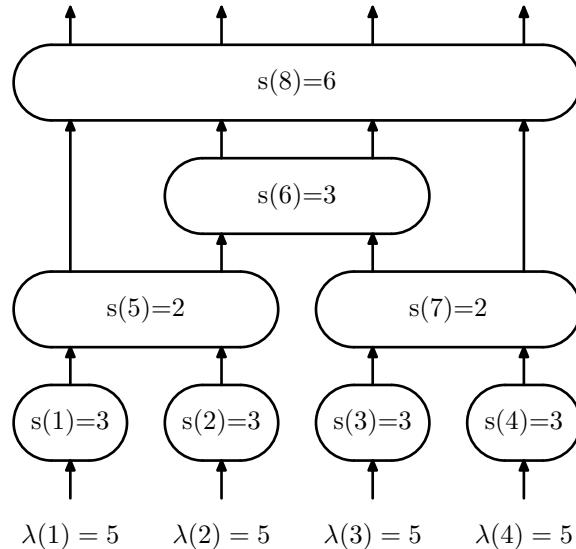
	skills				
method	1	2	3	4	
SIM	0.1033	0.0609	0.0615	0.1028	0.0821
ED	0.0752	0.0314	0.0314	0.0752	0.0533
HED	0.1048	0.0613	0.0613	0.1048	0.0830
ERM	-	-	-	-	-
HF	0.1156	0.0560	0.0560	0.1156	0.0858
IPP	0.0906	0.0408	0.0408	0.0906	0.0657

Instance 7

$$\mu(1) = 1$$

```
        ▲
    ┌─────────┐
    │ s(4)=6  │
    └─────────┘
        ▲
    ┌─────────┐
    │ s(3)=6  │
    └─────────┘
        ▲
    ┌─────────┐
    │ s(2)=6  │
    └─────────┘
        ▲
    ┌─────────┐
    │ s(1)=6  │
    └─────────┘
        ▲
```
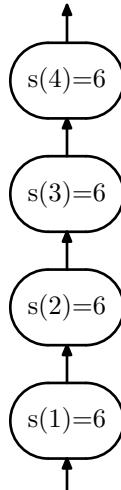
$$\lambda(1) = 20$$

	skills	
method	1	
SIM	0.0661	0.0661
ED	0.0211	0.0211
HED	0.0660	0.0660
ERM	0.0661	0.0661
HF	0.0630	0.0630
IPP	0.0658	0.0658

Instance 8

$$\mu(1) = 1.75 \quad \mu(2) = 0.6$$

```
    ▲             ▲
  ┌───────────────────┐
  │      s(3)=4       │
  └───────────────────┘
      ▲           ▲
  ┌───────┐   ┌───────┐
  │s(1)=4 │   │s(2)=6 │
  └───────┘   └───────┘
      ▲           ▲
```
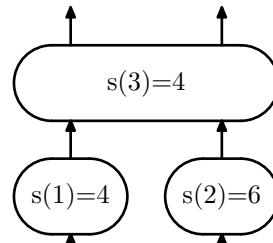
$$\lambda(1) = 3 \qquad \lambda(2) = 5$$

	skills		
method	1	2	
SIM	0.0201	0.1410	0.0957
ED	0.0175	0.1065	0.0731
HED	0.0201	0.1408	0.0955
ERM	-	-	-
HF	0.0225	0.1371	0.0941
IPP	0.0227	0.1385	0.0951

Instance 9

$\mu(1) = 0.5 \quad \mu(2) = 1.25 \quad \mu(3) = 1.5$

s(4)=6

s(1)=3   s(2)=6   s(3)=4
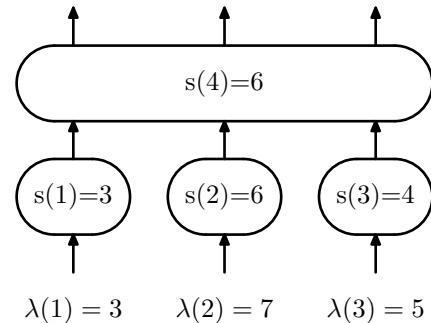
$\lambda(1) = 3 \quad \lambda(2) = 7 \quad \lambda(3) = 5$

	skills			
method	1	2	3	
SIM	0.1587	0.0777	0.0726	0.0922
ED	0.1426	0.0571	0.0586	0.0747
HED	0.1584	0.0774	0.0728	0.0921
ERM	-	-	-	-
HF	0.1733	0.0694	0.0712	0.0908
IPP	0.1669	0.0669	0.0686	0.0875

Instance 10

$\mu(1) = 0.5$ $\mu(2) = 1.5$ $\mu(3) = 1$

s(6)=5

s(5)=4

s(4)=3

s(1)=3 s(2)=2 s(3)=1
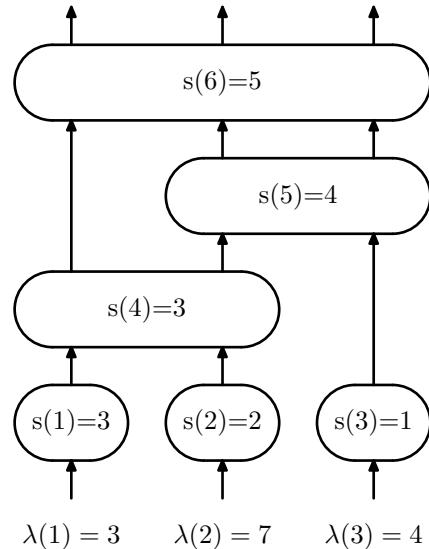
$\lambda(1) = 3$ $\lambda(2) = 7$ $\lambda(3) = 4$

	skills			
method	1	2	3	
SIM	0.1098	0.0741	0.1080	0.0914
ED	0.0817	0.0370	0.0726	0.0567
HED	0.1061	0.0702	0.1146	0.0906
ERM	-	-	-	-
HF	0.1234	0.0607	0.1164	0.0901
IPP	0.0989	0.0470	0.0905	0.0706

Instance 11

$\mu(1) = 0.5$ $\mu(2) = 2$ $\mu(3) = 0.5$ $\mu(4) = 2$

s(7)=3

s(5)=4 s(6)=5

s(1)=4 s(2)=3 s(3)=2 s(4)=3
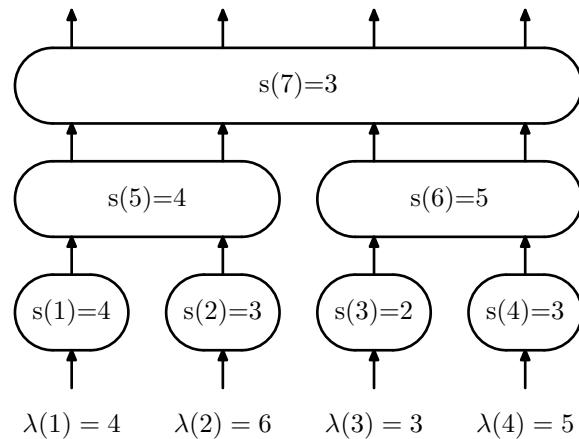
$\lambda(1) = 4$ $\lambda(2) = 6$ $\lambda(3) = 3$ $\lambda(4) = 5$

method	skills				
	1	2	3	4	
SIM	0.1481	0.0936	0.1178	0.0549	0.0990
ED	0.1142	0.0688	0.0922	0.0361	0.0737
HED	0.1524	0.0938	0.1178	0.0522	0.0993
ERM	-	-	-	-	-
HF	0.1513	0.0912	0.1268	0.0497	0.0990
IPP	0.1371	0.0826	0.1146	0.0449	0.0896

Instance 12

$\mu(1) = 0.5 \quad \mu(2) = 0.75 \quad \mu(3) = 1 \quad \mu(4) = 1.25$

s(7)=5

s(6)=4

s(5)=4

s(1)=4   s(2)=3   s(3)=3   s(4)=2

$\lambda(1) = 3 \quad \lambda(2) = 5 \quad \lambda(3) = 5 \quad \lambda(4) = 5$
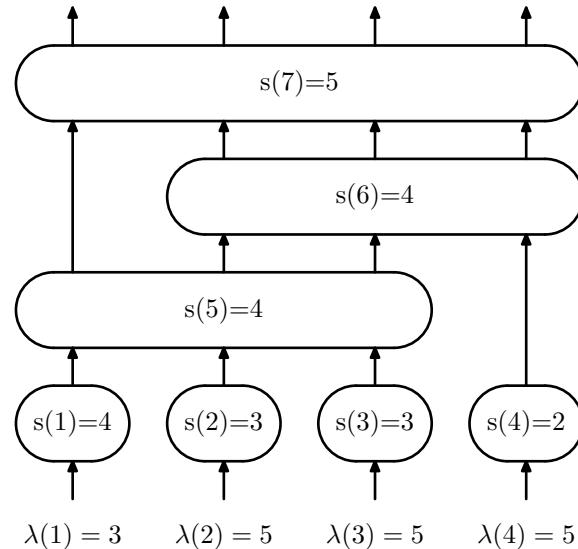
method	skills				
	1	2	3	4	
SIM	0.1115	0.1007	0.0885	0.1277	0.1066
ED	0.0916	0.0627	0.0533	0.0975	0.0746
HED	0.1040	0.1051	0.0901	0.1376	0.1098
ERM	-	-	-	-	-
HF	0.1269	0.0932	0.0793	0.1421	0.1085
IPP	0.1010	0.0709	0.0603	0.1090	0.0836

Instance 13
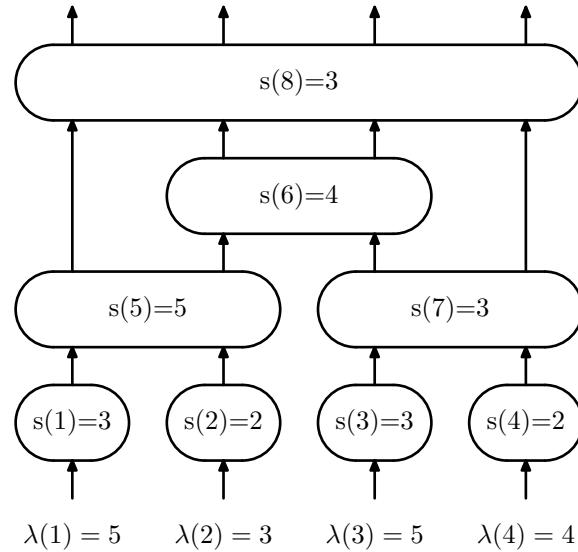
	skills				
method	1	2	3	4	
SIM	0.0960	0.0523	0.0633	0.2013	0.1035
ED	0.0692	0.0252	0.0306	0.1749	0.0750
HED	0.0931	0.0532	0.0591	0.2150	0.1047
ERM	-	-	-	-	-
HF	0.0962	0.0515	0.0595	0.2319	0.1094
IPP	0.0834	0.0377	0.0438	0.2025	0.0917

Instance 14

$\mu(1) = 0.5 \quad \mu(2) = 1 \quad \mu(3) = 1.25 \quad \mu(4) = 0.75$

s(4)=5

s(3)=5

s(2)=5

s(1)=5

$\lambda(1) = 3 \quad \lambda(2) = 3 \quad \lambda(3) = 3 \quad \lambda(4) = 3$
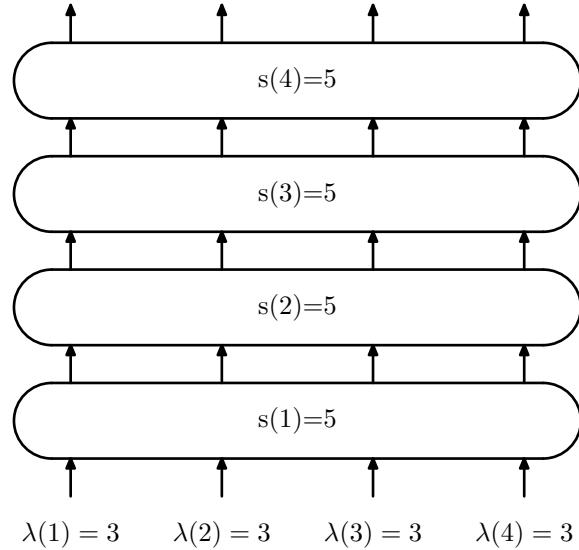
	skills				
method	1	2	3	4	
SIM	0.0525	0.0527	0.0524	0.0525	0.0525
ED	0.0106	0.0106	0.0106	0.0106	0.0106
HED	0.0515	0.0515	0.0515	0.0515	0.0515
ERM	-	-	-	-	-
HF	0.0461	0.0461	0.0461	0.0461	0.0461
IPP	0.0146	0.0146	0.0146	0.0146	0.0146

Instance 15

$\mu(1) = 1 \quad \mu(2) = 0.5 \quad \mu(3) = 1.1 \quad \mu(4) = 0.7 \quad \mu(5) = 0.6 \quad \mu(6) = 1$

s(7)=10

s(1)=5 s(2)=10 s(3)=5 s(4)=4 s(5)=4 s(6)=3

$\lambda(1) = 5 \quad \lambda(2) = 7 \quad \lambda(3) = 5 \quad \lambda(4) = 4 \quad \lambda(5) = 3 \quad \lambda(6) = 5$
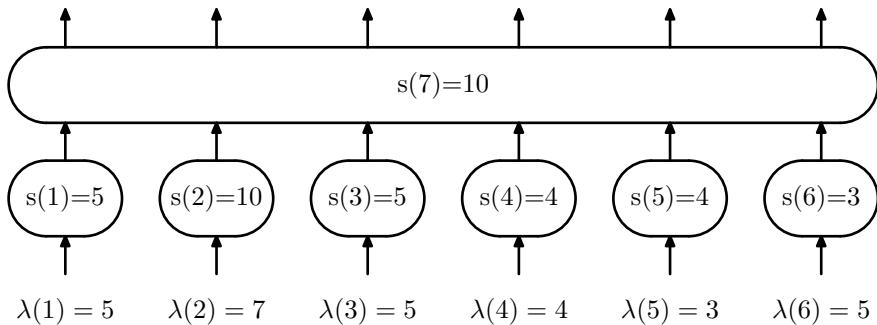
| method	skills						
	1	2	3	4	5	6	
SIM	0.1228	0.1707	0.1063	0.1874	0.1658	0.2179	0.1613
ED	0.1173	0.1553	0.1017	0.1856	0.1640	0.2181	0.1554
HED	0.1249	0.1688	0.1088	0.1891	0.1653	0.2216	0.1624
ERM	-	-	-	-	-	-	-
HF	0.1251	0.1657	0.1085	0.1980	0.1750	0.2327	0.1658
IPP	0.1205	0.1596	0.1044	0.1906	0.1685	0.2240	0.1596

Instance 16

$\mu(1) = 1 \quad \mu(2) = 0.5 \quad \mu(3) = 1.1 \quad \mu(4) = 0.7 \quad \mu(5) = 0.6 \quad \mu(6) = 1$

s(12)=5

s(10)=6   s(11)=5

s(9)=4   s(8)=3

s(7)=3

s(1)=3   s(2)=8   s(3)=3   s(4)=2   s(5)=2   s(6)=1

$\lambda(1) = 5 \quad \lambda(2) = 7 \quad \lambda(3) = 5 \quad \lambda(4) = 4 \quad \lambda(5) = 3 \quad \lambda(6) = 5$
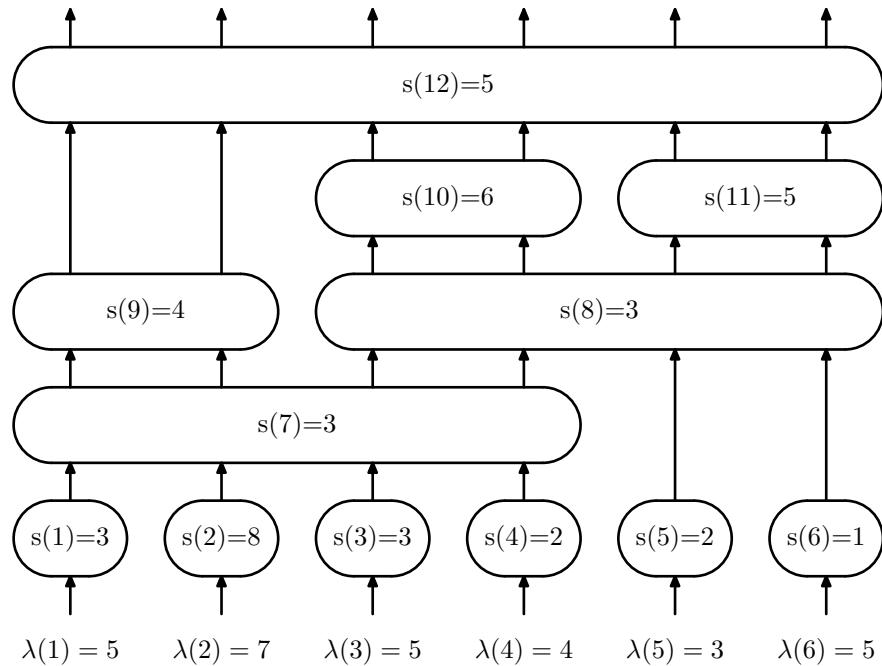
method	skills						
	1	2	3	4	5	6	
SIM	0.1299	0.1319	0.0347	0.0428	0.1033	0.1212	0.0977
ED	0.1055	0.0977	0.0167	0.0238	0.0801	0.0988	0.0733
HED	0.1465	0.1371	0.0357	0.0503	0.1097	0.1347	0.1060
ERM	-	-	-	-	-	-	-
HF	0.1414	0.1309	0.0334	0.0477	0.1170	0.1443	0.1053
IPP	0.1124	0.1040	0.0188	0.0268	0.0865	0.1067	0.0788

Instance 17

$\mu(1) = 1$ $\mu(2) = 0.5$ $\mu(3) = 1.1$ $\mu(4) = 0.7$ $\mu(5) = 0.6$ $\mu(6) = 1$

s(12)=10

s(10)=5   s(11)=4

s(9)=3   s(8)=2

s(7)=2

s(1)=3 s(2)=8 s(3)=3 s(4)=2 s(5)=2 s(6)=1
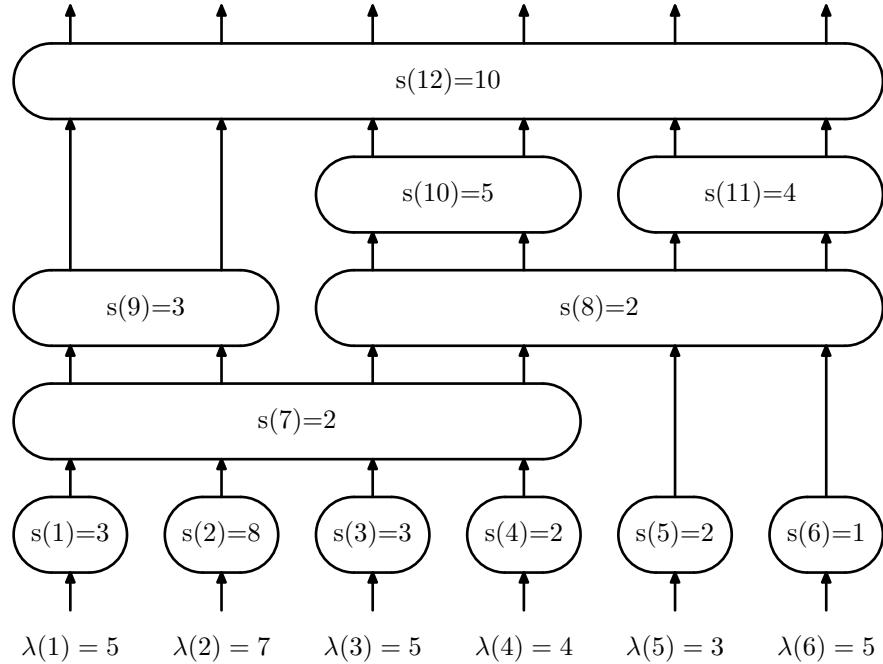
$\lambda(1) = 5$ $\lambda(2) = 7$ $\lambda(3) = 5$ $\lambda(4) = 4$ $\lambda(5) = 3$ $\lambda(6) = 5$

method	skills						
	1	2	3	4	5	6	
SIM	0.0965	0.0984	0.0418	0.0537	0.0894	0.1045	0.0823
ED	0.0730	0.0676	0.0227	0.0325	0.0658	0.0811	0.0581
HED	0.1101	0.1027	0.0418	0.0591	0.0924	0.1136	0.0883
ERM	-	-	-	-	-	-	-
HF	0.1057	0.0979	0.0387	0.0553	0.0988	0.1218	0.0874
IPP	0.0781	0.0723	0.0248	0.0353	0.0706	0.0871	0.0624

Instance 18

$\mu(1) = 1$ $\mu(2) = 0.5$ $\mu(3) = 1.1$ $\mu(4) = 0.7$ $\mu(5) = 0.6$

s(6)=20

s(1)=3 s(2)=7 s(3)=3 s(4)=2 s(5)=4
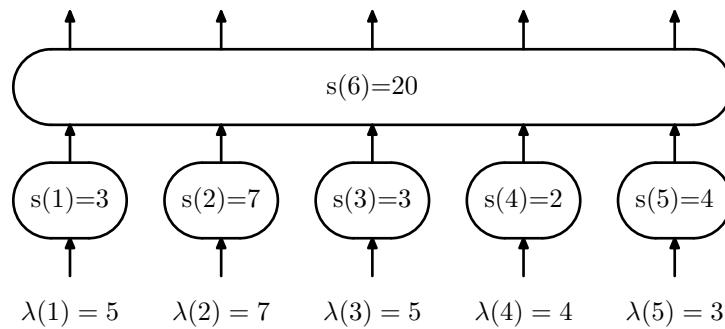
$\lambda(1) = 5$ $\lambda(2) = 7$ $\lambda(3) = 5$ $\lambda(4) = 4$ $\lambda(5) = 3$

method	skills					
	1	2	3	4	5	
SIM	0.0778	0.0852	0.0737	0.0979	0.0593	0.0802
ED	0.0662	0.0687	0.0620	0.0886	0.0498	0.0677
HED	0.0787	0.0841	0.0745	0.0981	0.0593	0.0802
ERM	-	-	-	-	-	-
HF	0.0807	0.0838	0.0756	0.1079	0.0607	0.0826
IPP	0.0715	0.0743	0.0670	0.0957	0.0538	0.0732