On the Optimal Allocation of Resources in Stochastic Workflow Nets

Kees van Hee[*,†] Hajo Reijers[*,†] Eric Verbeek[ϒ] Loucif Zerguini[*]

Abstract

Stochastic workflow nets are used for modelling and analysing business processes. For a specific subclass, a marginal allocation strategy of resources is presented that minimises the mean sojourn time of individual cases. A popular alternative allocation strategy is shown to be sub-optimal.

1 Introduction

A business process can be seen as a set of logically related *tasks*. Each task can be performed by a number of *resources*, being either machines or humans. Individual *cases* (also known as *jobs*), e.g. insurance claims and credit applications, arrive more or less regularly at the process. They are routed through the business process along the tasks that should be performed for them. Typical tasks in a business process are, for example, calculating the validity of an insurance claim and checking the creditworthiness of an applicant. Within each business process a clear start and end can be distinguished. Also, any routing through a business process is allowable, but for each case the routing is uniquely defined. The time that is spent to perform a task for an individual case, the *service time*, can be of arbitrary length. If all resources that can work on a task are busy and new cases arrive for that task, the cases will have to line up (*queuing*). At each point during the processing of a case, a final state can be distinguished when all processing for this case is complete.

In this paper, we are interested in determining the optimal allocation of resources in a business process if there is just a limited set of resources that can be freely allocated. The optimisation criterion we apply is the *sojourn time*, informally defined as the mean time in steady-state that elapses between the moment a case becomes available for the process and the moment its processing ends. Other common names for this criterion are *traversal time*, *lead time* and *cycle time*. In business processes, a low or stable sojourn time is often a desirable or even necessary characteristic of a business process (see [HR00]). Obviously, a low sojourn time can give a business a competitive edge. The resource allocation problem is often encountered when the structure of a business process has been redesigned and the new process must be put to practice by assigning tasks to individual workers (see [AH01]).

[*] Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB, The Netherlands. Email: {k.m.v.hee, h.a.reijers,l.zerguini}@tue.nl

[†] Deloitte & Touche Bakkenist, Management Consultants, P.O. Box 23103, NL-1100 DP, Amsterdam, The Netherlands.

[ϒ] Department of Technology Management, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB, The Netherlands, Email: h.m.w.verbeek@tm.tue.nl

In this paper, an algorithm is given for the allocation of a fixed number of resources to minimise the mean sojourn time for a particular class of business processes. We will apply stochastic workflow nets to model and analyse these business processes. This type of Petri net model is derived from the formal model in [Aal98, Hee94]. More specific, optimality is proven for each stochastic workflow net that can be projected on a Jackson network as described in [Jac57, Jac63]. This result builds upon [Fox66, Rol71, DP77, Web80] in which the optimality of marginal allocation algorithms for various queuing networks is studied. In particular, it is shown that a allocation strategy popular in industry as advocated in [GC84] is not always optimal for the types of business processes we consider. Queuing theory and simulation will be applied to support our claims; the Petri net tool ExSpect (see [ACG00]) is used in the presentation of an example.

The paper is organised as follows. Section 2 explains the basic terminology and concepts used throughout this paper, including the performance criterion that is applied. In Section 3 we present Goldratt's approach and discuss its properties. Then, in Section 4, we introduce the marginal allocation strategy, its optimality and its limits of application. In the last section, we present our concluding remarks and directions for further research.

2 Stochastic Workflow Nets

2.1 Workflow

A workflow (see e.g. [AH01]) is a case-based business process, i.e., every action performed in a workflow corresponds to some case. Typically, cases are generated by the workflow's environment. The goal of the workflow is to handle cases as efficient and effective as possible.

The most important aspect of a workflow is its process definition: a partial ordering of the tasks that defines which tasks need to be executed in what order. A task that is to be executed for a specific case is called a work item. Most work items need a resource to be executed. A task that is being executed for a specific case is called an activity.

2.2 Petri Nets

Workflow process definitions can be modelled by workflow nets, which form a subclass of ordinary Petri nets. A Petri net is a directed graph with two kinds of nodes: places and transitions. An arc in a Petri net connects a place to a transition or a transition to a place. The input (output) places of a transition are those places from (to) which there is an arc to (from) that transition. A Petri net where no two arcs connect the same transition to the same place, or the same place to the same transition, is called ordinary.

Places in a Petri net may contain so-called tokens (i.e., may be marked). The distribution of these tokens over the places (also referred to as the marking) determines the state of the Petri net. A transition is enabled (i.e., can be executed) if and only if all its input places contain tokens. If the transition fires (i.e., executes), it removes a token from every input place, and adds a token to every output place, resulting in a new marking. A marking is reachable from another marking if and only if there exists a series of transition firings that transforms the latter marking to the first.

2.3 Workflow nets

A workflow net is an ordinary Petri net which satisfies the following three requirements:

1. There is one source place, i.e., a place without incoming arcs;
2. There is one sink place, i.e., a place without outgoing arcs;
3. For every node there exists a directed path from the source place to that node, and a directed path from that node to the sink place.

A token in the source place indicates the arrival of a new case; a token in the sink place indicates the completion of a case.

To enforce proper behaviour of a workflow, the following soundness requirement is imposed on the behaviour of the corresponding workflow net:

1. A marking of the sink place can be reached from every marking reachable from the marking with only the source place marked;
2. For every marking reachable from the marking with only the source place marked, all other places are unmarked if the sink place is marked;
3. Every transition can be enabled from the marking with only the source place marked.

The first requirement states that we can always complete a case; the second states that upon completion of a case no work is left behind in the workflow; the third states that every task can become an activity for some case.

2.4 Stochastic workflow nets

A task in a workflow can be modelled by a start transition, an end transition, a busy place, and an idle place:

- The start transition indicates that the task becomes an activity;
- The end transition indicates that the activity is completed;
- The busy place holds a combination of a case and a busy resource and indicates that the task is an activity;
- The idle place holds the idle resources.

The net in Figure 1 is a task. Because there is a token in the busy place, this task is an activity for the case associated with that token.

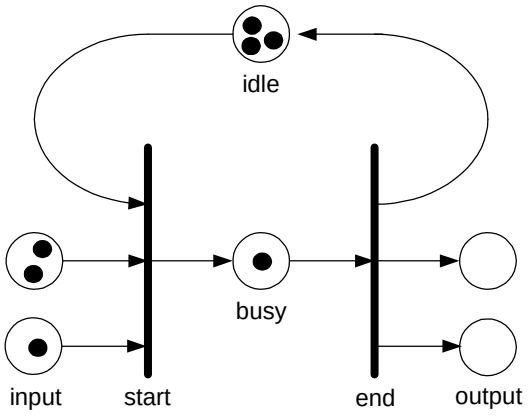

Figure 1: The task model

Note that cases are associated with tokens and that the start transition needs to be able to distinguish tokens from different cases: it should be enabled only if each input

place contains a token associated with the same case. For this reason, we assume some colour to exist in the stochastic workflow net: Each case has a unique identity, tokens associated with a case have the colour of that case, and transitions are only allowed to use case tokens (resource tokens have no identity) with the same identity during one execution. So, if one of the tokens in one input place has the same identity as the token in the second, the task is a work item for the case associated with both tokens. Otherwise, the task is not a work item for any case. Note also that the task contains four available resources: three idle and one busy. For ease of use, we will use boxes to visualise tasks.

The partial ordering of the workflow can be modelled by routing transitions and condition places:

- Routing transitions are additional transitions that do not correspond to tasks, but may be necessary to achieve a correct routing;
- Condition places define the partial order of all tasks and routing transitions; we can distinguish queuing places and routing places:
- Queuing places are places that directly precede tasks and correspond to FIFO queues in the workflow;
- Routing places are places that directly precede routing transitions, do not correspond to queues, but may be necessary to achieve a correct routing.

Note that we assume that tasks and routing transitions do not share input places, and that all queues in the workflow are First-In-First-Out (FIFO) queues.

To be able to say something about the workflow's sojourn time, we add several parameters to the workflow net:

- Each task has an arbitrary service time distribution that is independently sampled for each time it is executed;
- Each task has a number of available resources, which are initially idle;
- Each routing transition has a relative weight: if a choice is to be made between two (or more) enabled routing transitions then these weights are taken into account to determine which routing transition is executed first;
- The process has an arbitrary arrival time distribution, which specifies the arrival pattern of new cases; each new case is assumed to arrive at the source place of the workflow.

Figure 2 shows an example containing two conflicting routing transitions left and right. Transitions left has weight 2 and transition right weight 1. If only these transitions are enabled, then in 2 out of 2+1=3 cases transition left will fire first, and in 1 out of 3 cases transition right will fire first.

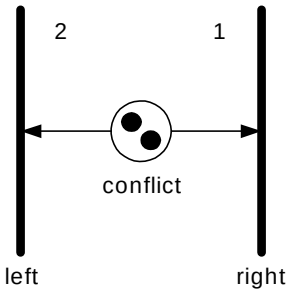

Figure 2: Transition weights

The sojourn time of a workflow can now be defined as the average time it takes the (sound) workflow net to move a case from the source place to the sink place. This sojourn time is a combination of:

- Service times: the time a case spends in a task's service place;

- Queue times: the time a case spends in places waiting for a resource token;
- Wait times: the time a case spends in places waiting for other case tokens.

Note that wait times can only occur at synchronising transitions, i.e., transitions with multiple input places. The relevance of distinguishing between *queue* and *wait* time will become clear in Section 4.

3 Goldratt's conjecture

3.1 Introduction

In 1984, Goldratt published [GC84], a textbook in the form of a novel. This best-seller quickly gained popularity with manufacturing and logistic industries. It describes various techniques to improve the performance of business processes. Because the nature of the descriptions is informal, the semantics of these techniques – to some extent – is up to the reader. One interesting, explicitly stated claim to improve the rate at which cases are being completed is to add resources at specific locations in the process. We present a genuine interpretation of this conjecture and apply it to our business process model. We will refer to it as the *Goldratt algorithm*. We think that it is instructive to see how an intuitive approach, presented in a way that appeals to management in industry, does not necessarily lead to an optimal allocation of scarce resources. In Section 4.1 we present an allocation strategy that is optimal for a large class of processes.

The Goldratt algorithm starts to identify the *bottleneck* within the process among the so-called *work centres*. A work centre is any group of *similar* resources: being capable of executing the same set of tasks. Typically, different groups of workgroups can be distinguished. For the sake of simplicity, we assume for what follows that each group of resources is dedicated to exactly one task. This by no means limits the application of Goldratt's algorithm. A bottleneck, then, is principally defined as that task of which its resources have a utilisation that exceeds 1. Obviously, in this situation there is no steady state for the process (queues grow infinitely large) and the mean sojourn time is meaningless. For these reasons, we turn this definition aside and focus on the secondary definition Goldratt provides: "work centres that have the largest amount of work-in-process sitting in front of it are the bottlenecks". It seems reasonable to deduce from this that the bottleneck is the task for which the mean queue time is maximal. After all, the physical size of a queue or even the number of cases in queue is no accurate indicator of the *time* that is involved with handling such a queue. Goldratt algorithm ends by stating that if an extra resource is available it should be placed at the bottleneck to increase the overall performance of the process.

3.2 Limits

The intuition behind the Goldratt algorithm is apparent: reduce the queue time at those places where it is maximal, to improve on the sojourn time. However, consider the following example. A telephone operator of a service organisation handles incoming complaints. The operator can handle 90% of all incoming complaints. On average, this takes him two and a half minutes. For the remaining 10%, one of the ten service men has to visit the complainant. On average, a visit takes three hours and fifty minutes. The service times of tasks O ("Operator") and S ("Service men") have a negative exponential distribution. On average, every two and a half minutes a new complaint arrives. The arrivals of complaints is Poisson-distributed. This process is depicted as a stochastic workflow net in Figure 3. Note that the small number of tasks

in this process is by no means indicative for the size of workflow nets as they are used in practice to model business processes; practical use commonly renders workflow nets with dozens or hundreds of tasks.

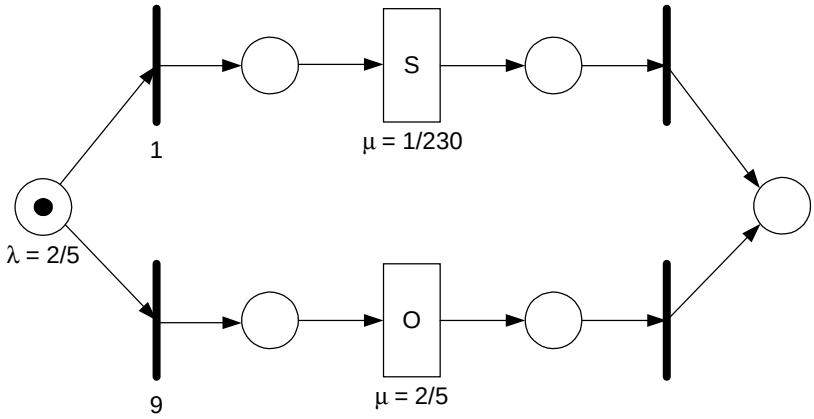

Figure 3 Example of a service organisation

As all probabilities in the depicted stochastic workflow net are independent, the performance of this process can be analytically determined by splitting it up into two well-known queuing systems: an M/M/10 and an M/M/1 queuing system, with respective Poisson arrival rates of 1/25 and 9/25 and negative exponential service time parameters of 1/230 and 2/5.

The mean queue time for task S, W_S, is approximately 209.7 minutes and the mean queue time for task O, W_O, equals 22.50 minutes. So, the mean sojourn time is about 41.22 minutes. Using Goldratt's algorithm, the bottleneck turns out to be task S, as its mean queue time exceeds that of task O. Hiring an eleventh service man yields a mean queue time W_S of 60.75 minutes. This is a local gain of almost 150 minutes. While the processing in the other part of the system does not change, this results in a mean sojourn time of approximately 26.33 minutes.

However, if we had hired a second operator instead of an eleventh service man the mean queue time W_O would be 0.635 minutes. This is a local gain of more than 20 minutes, resulting in a mean sojourn time of 21.54 minutes. This is a substantially lower sojourn time than the former scenario.

The results from this example are summarised in Table 1. The number of service men and operators are denoted with n_S and n_O respectively. It is clear that Goldratt's algorithm is not optimal. It is possible to gain more time at other places in the process than at the bottleneck. The effect is obtained by:

1. the higher marginal effect of one extra operator, where there is just one operator working, compared to the benefit of adding a service man to the ten service men already working, and
2. the higher execution frequency of the operators.

n_S	n_O	W_S	W_O	*mean sojourn time*
10	1	209.7	22.50	41.22
10	2	209.7	0.635	21.54
11	1	60.75	22.50	26.33

Table 1: Results service organisation

Note that if we had used the occupation rate as criterion to identify the bottleneck, task S would still have been the bottleneck: The occupation rate of the service men equals 92%, where the occupation rate of the operator equals 90%.

4 The Method of Marginal Allocation

4.1 Introduction

As an alternative to the Goldratt algorithm, we propose a method of marginal allocation. We consider a stochastic workflow net that consists of N tasks. The service times of the tasks $t_1, t_2,\ldots,t_N$ are arbitrarily distributed, characterised by $\Lambda = \langle \mu_1, \mu_2,\ldots \mu_N\rangle$ with μ_i for $1 \le i \le N$ containing all parameters to characterise the service time distribution of task t_i. The arrival process of new cases is arbitrarily distributed with characterisation λ. We suppose that initially a sufficiently large number of resources is allocated to each task to ensure that the mean queue time is finite for every queue in the system. M additional resources are available to be freely allocated amongst the tasks ($M \in \mathbb{N}\backslash 0$). Each resource can be allocated to any of the tasks. However, after its allocation, a resource can only work on work items that are to be executed for the task it has been allocated to. Let n_i for $1 \le i \le N$ be the number of resources that is dedicated to task i after M additional resources have been allocated to the process. We denote $\Gamma = \langle n_1, n_2,\ldots n_N\rangle$. The mean sojourn time of the process after adding M extra servers can then be expressed as a function $f(\lambda, \Lambda, \Gamma)$. Suppose that after marginally allocating M-1 resources the number of resources at task i is m_i. We denote for $1 \le k \le N$ with ϑ_k the allocation $\langle n_1, n_2,...,n_N\rangle$ where $n_k = m_k + 1$ and $n_j = m_j$ for $j \ne k$ and $1 \le j \le N$. The method of marginal allocation is to allocate the Mth resource to a task l such that

$$f(\lambda, \Lambda, \vartheta_l) = \left(\underline{\min}\, k : 1 \le k \le N : f\left(\lambda, \Lambda, \vartheta_k\right)\right). \quad \text{(i)}$$

In some cases the quantity $f(\lambda, \Lambda, \vartheta_k)$ can be calculated from an explicit formula; in other cases it can be estimated by simulation. If M additional servers are to be applied, the marginal allocation requires $N \cdot M$ evaluations of the sojourn time. If, instead of marginal allocation, all possible allocations were to be tried, $\binom{M+N-1}{N-1}$ evaluations are required.

4.2 Example

We use the service organisation from Figure 3 as an example to demonstrate the method of marginal allocation. Table 1 shows the mean sojourn times of three possible resource allocations: $(n_O, n_S) = (1, 10)$ and both possible successors of this allocation. From this table, we deduce that an additional operator should be hired, because an extra operator reduces the mean sojourn time most. Suppose we can hire six additional employees. Table 2 shows the mean sojourn times of the organisation for allocations up to 17 employees. From this table, we deduce that the first employee to hire would be an operator (because 21.54 < 26.33), the second would be a service man (21.04 > 6.65), the third a service man (6.15 > 3.02), etc. In the end, three operators and fourteen service men are employed. Note that this is the optimal solution with regard to the sojourn time for seventeen employees.

n / n_O	n_S = 10	11	12	13	14	15	16
1	41.22	26.33	22.70	21.33	20.74	20.47	20.35
2	21.54	6.65	3.02	1.66	1.07	0.80	
3	21.04	6.15	2.52	1.06	0.57		
4	20.98	6.09	2.46	1.09			
5	20.97	6.08	2.45				
6	20.97	6.08					
7	20.97						

Table 2: Mean sojourn times service organisation

Using Goldratt's algorithm, we first hire three service men, then an operator, and then two more service men, resulting in a non-optimal situation. If we use occupation rates as criterion for the bottleneck, we first hire a service man, then an operator, and then four service men, resulting in the same, non-optimal, situation.

4.3 Optimality

To discuss the optimality of the marginal allocation strategy we briefly discuss a result in [Rol71] in which networks are considered that are composed of an arbitrary number of tasks, where each task behaves like a G/GI/*m* queue. An important condition on these networks is that the arrival and queuing processes among the tasks are independent. Using the previously introduced notation and denoting the expected *queue time* at task *i* after allocating *M* servers to the total system by $W_i(\lambda_i, \mu_i, n_i)$, the allocation problem in [Rol71] can be formulated as:

$$\min_{n_1, n_2, \ldots n_N} \left\{ \sum_{i=1}^{N} W_i(\lambda_i, \mu_i, n_i) \right\} \text{ with } M = \sum_{i=1}^{N} n_i \,. \qquad \text{(ii)}$$

In [Fox66] it was already shown that for this kind of problem the marginal allocation algorithm is optimal if the function $W_i(\lambda_i, \mu_i, n_i)$ is non-increasing and convex in n_i. In other words, for any task *i*:

$$W_i(\lambda_i, \mu_i, n_i) - W_i(\lambda_i, \mu_i, n_i + 1) \geq W_i(\lambda_i, \mu_i, n_i + 1) - W_i(\lambda_i, \mu_i, n_i + 2) \geq 0 \,. \qquad \text{(iii)}$$

This condition ensures that it is never necessary to remove already placed resources, so that marginal allocation is optimal. This result cannot immediately be translated to the optimality of marginal allocation to minimise the sojourn time in stochastic workflow nets. In the first place, the arrival and queuing processes among the tasks in a stochastic workflow net are generally not independent. The structure of the net, as well as the service time distributions of the tasks may influence the arrival and queuing processes at other tasks. In the second place, a minimal queue time does not guarantee a minimal sojourn time (this is illustrated by the example in Section 4.4). In general, the sojourn time in a stochastic workflow net consists of service time, queue time, and wait time. Although the service time is independent of queue time in our model, the wait time cannot be neglected in minimising the sojourn time. Recall that wait time is related with synchronizations in concurrent parts of a workflow net (see Section 2.4).

First, we consider stochastic workflow nets of which the net structure is a state machine, i.e., for each transition *t* of the Petri net holds that it has exactly one input and exactly one output place. We will call such a workflow net a State-Machine-

Stochastic (SMS) workflow net. In an SMS workflow net, no concurrency can take place, so the wait time for each case is zero. As a result, the mean sojourn time equals the sum of the mean total service time and the mean total queue time. This implies that minimising the mean queue time for an SMS workflow net suffices to minimise its mean sojourn time. After all, the mean service time is constant.

Suppose we further narrow down the class of workflow nets under consideration to those SMS workflow nets where each case is directly routed to one of N alternative tasks after which the processing ends. An example of such a net is the one in Figure 3. An SMS workflow net like this can be divided into N independent G/GI/m queuing systems. In [Web80] it is proved that a G/GI/m queue is convex and non-increasing in the number of servers m. So, on basis of condition (iii), we can determine that marginal allocation is optimal for arbitrary arrival processes, arbitrary service time distributions, and arbitrarily chosen weights. Unfortunately, this is a rather small class of stochastic workflow nets.

Now consider the class of SMS workflow nets with a Poisson arrival process with parameter λ, negative-exponentially distributed service time distributions with intensities μ_1, μ_2,..., μ_N for tasks t_1, t_2, ..., t_N, arbitrarily chosen weights, and an initial allocation of resources such that the mean queue time at each task is finite. It is straightforward to see that such a net can be mapped onto a (single class) open Jackson queuing network. For Jackson networks it is a well known result that the equilibrium distribution can be determined analytically (see [Jac57, Jac63]). In steady state, each node in a Jackson network behaves as if it were in isolation and subject to Poisson arrivals, although the arrivals may in fact not be Poisson. Using the Jackson equilibrium formula and Little's law, it is possible to express the mean queue time at each task i as a function of the arrival rate of cases for this particular task (λ_i), its service rate (μ_i), and the number of resources working on that task (n_i) (see e.g. [CMP99]):

$$\frac{(1/n_i\mu_i)(\lambda_i/\mu_i)^{n_i}}{n_i!(1-\lambda_i/n_i\mu_i)^2}\left[\sum_{j=0}^{n_i-1}\frac{(\lambda_i/\mu_i)^j}{j!}+\frac{(\lambda_i/\mu_i)^{n_i}}{n_i!(1-\lambda_i/n_i\mu_i)}\right]^{-1}. \qquad \text{(iv)}$$

This is the standard expression for the expected queue time of an M/M/n_i queue. In steady state, λ_i can be determined on basis of the arrival rate of new cases and the weights applied; μ_i and n_i are given. Note that these parameters are independent of the other queues in the network. We will denote the expected queue time for a task i within this type of net with $Q_i(\lambda_i, \mu_i, n_i)$. If we denote with f_i the expected number of executions of task i for an arbitrary case, this allows us to express the problem of minimising the mean queue time of the stochastic workflow net over n_1, n_2, ..., n_N as:

$$\min_{n_1,n_2,\ldots,n_N}\left\{\sum_{i=1}^{N} f_i\cdot Q_i(\lambda_i,\mu_i,n_i)\right\} \text{ with } M=\sum_{i=1}^{N} n_i\ . \qquad \text{(v)}$$

The figure f_i can be derived for any Jackson network from the weights, the net structure, and traffic equations of the net in question. Note that formula (v) resembles the problem as formulated in [Rol71] as stated with formula (ii). The interesting thing is that it is proved in [DP77] that the expected queue time for an M/M/m queue (formula iv) is a non-increasing and convex function of m. So, marginal allocation is optimal to minimise the queue time in a network of independent M/M/m nodes. But if marginal allocation is optimal for formula (ii), it is also optimal for formula (v). After all, on basis of $Q_i(\lambda_i, \mu_i, n_i)$ being non-increasing and convex in n_i, we can conclude that $f_i \cdot Q_i(\lambda_i, \mu_i, n_i)$ is non-increasing and convex in n_i. Hence, marginal allocation is optimal to minimise the mean sojourn time in SMS workflow nets with a Poisson arrival, negative exponential service times, and arbitrary weights.

Practical experience and simulations of large models seem to suggest that the marginal allocation is optimal for any SMS workflow net with a Poisson arrival process, arbitrary service time distributions, arbitrary weights and sufficient numbers of resources available to ensure a steady state. However, an analysis of these nets is very complex from a queuing network perspective.

4.4 Limits

From the following example it becomes clear that the marginal allocation strategy is in general not optimal if concurrency is allowed in the process. Figure 4 shows a stochastic workflow net. Note that to prevent this system from getting clogged, we need initially at least three resources at each task.

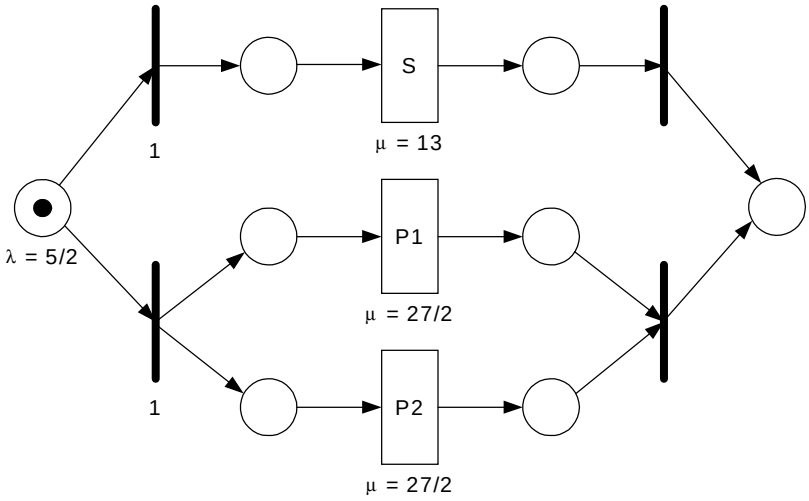

Figure 4: Counter example for both strategies

Because of the synchronisation following tasks P1 and P2, we cannot compute results for this example analytically using queuing theory. Furthermore, because cases can overtake each other, we also need to take the colour of the tokens into account when synchronising. We used simulation with the software package ExSpect to get approximated results.

To clearly show the characteristics of the example we used as service time distribution for each transition a beta distribution with modus (and, because the distribution is symmetric, mean) μ, minimum $9\mu/10$, maximum $11\mu/10$, and variance $\mu/900$. The beta distribution is often very well suited to express the variance in time behaviour for tasks that are executed by humans. Similar results can be obtained with e.g. negative exponential service times, but then the confidence intervals are wider.

Using ExSpect as the simulation tool, we simulated for several resource allocations 52 subruns of length 40000 each. Only the latter 50 subruns were taken into account for the simulation results, i.e., the first two were seen as start-up phase. We obtained the results as shown in Figure 5.

Note that these results clearly show that allocation (4,3,3) performs better than allocation (3,3,4). The simulation results indicate that the optimal allocations for 9, 10 or 11 resources are (3,3,3), (4,3,3) and (3,4,4). So, if we have one additional resource available, then the optimal allocation would be (4,3,3). If we would have two extra resources, then it would be (3,4,4). Apparently, for an optimal allocation strategy we

need to be able to *reallocate* resources. Both the Goldratt algorithm and the marginal allocation strategies lack this possibility and are hence not optimal in this case.

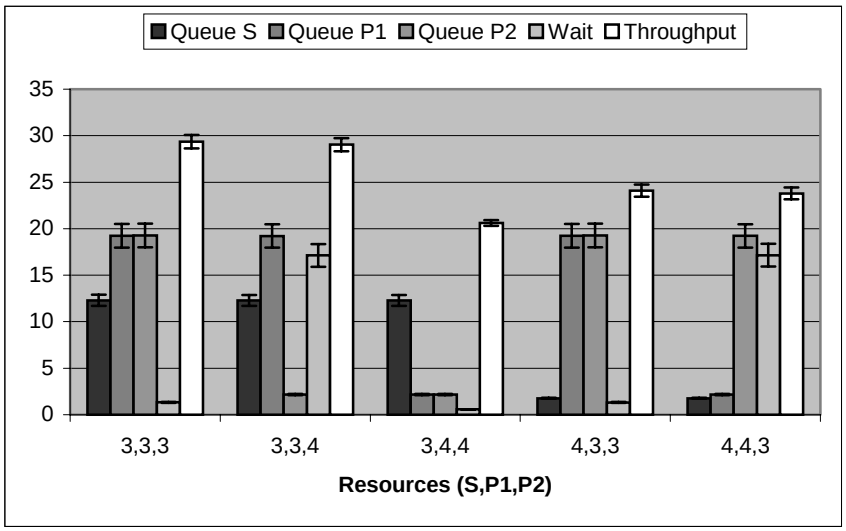

Figure 5: Simulation results counter example

5 Conclusion

We described a strategy of marginally allocating resources to a stochastic workflow net to minimise the mean sojourn time. The Goldratt algorithm was used to illustrate the superiority of the marginal allocation strategy. We derived two types of stochastic workflow nets for which a marginal allocation strategy is optimal. Both types of networks have an underlying ordinary Petri net that is a state machine. We conjectured the optimality of the algorithm for a wider class of stochastic workflow nets. Finally, we showed an example of a stochastic workflow net including concurrent behaviour for which neither strategy is optimal.

There are several directions to extend the applicability of the marginal allocation algorithm. For example, different cost functions for different types of resources may be incorporated in the algorithm. After all, the assumption that resources can be allocated to any task in a business process is rather theoretical. Concerning the class of stochastic workflow nets for which the strategy is optimal, it seems straightforward to extend the class of stochastic workflow nets of the second type to the class of BCMP networks in [BCM75], as independent equilibrium distributions can be determined for these networks. Also, the stochastic workflow net could easily be extended to allow for multiple classes of customers with their own routing probabilities. An interesting direction to formulate necessary and sufficient structural conditions for the marginal allocation strategy to be optimal is to apply the results of [BS94] and [DS92] on stochastic workflow nets.

Acknowledgement

The authors wish to thank prof. dr. Jan van der Wal of the Mathematics and Computing Science Department of the TU/e for his valuable ideas and suggestions.

References

[Aal92] W.M.P. van der Aalst. *Time Coloured Petri Nets and Their Application to Logistics*. Ph.D. Thesis, Eindhoven University of Technology, Eindhoven, 1992.

[Aal98] W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, vol. 8, pp. 21-66, 1998.

[AH01] W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT Press, Cambridge, 2001.

[ACG00] W.M.P. van der Aalst, P.J.N. de Crom, R.R.H.M.J. Goverde, K.M. van Hee, W.J. Hofman, H.A. Reijers, and R.A. van der Toorn. Ex*Spect* 6.4: An Executable Specification Tool for Hierarchical Colored Petri Nets, in Lecture Notes in Computer Science 1825, M. Nielsen and D. Simpson (eds.). Springer-Verlag, 2000.

[BCM75] F. Baskett, K. Chandy, R. Muntz, and F. Palacios. Open, Closed, and Mixed Networks of Queues with Different Classes of Customers. *Journal of the ACM*, vol. 22, pp. 248-260, 1975.

[BS94] R.J. Boucherie and M. Sereno. A Structural Characterisation of Product Form Stochastic Petri Nets. CWI Report BS-R9402, Amsterdam, 1994.

[CMP99] X. Chao, M. Miyazawa, and M. Pinedo. *Queing Networks: Customers, Signals, and Product Form Solutions*. Wiley & Sons, Chichester, 1999.

[DS92] S. Donatelli and M. Sereno. On the Product Form Solution for Stochastic Petri Nets, in Lecture Notes in Computer Science 616, K. Jensen (ed.). Springer-Verlag, pp. 154-172, 1992.

[DP77] M.E. Dyer and L.G. Proll. On the Validity of Marginal Analysis for Allocating Servers in M/M/c Queues. *Management Science*, vol. 21, pp. 1019-1022, 1977.

[Fox66] B. Fox. Discrete Optimization via Marginal Analysis. *Management Science*, vol. 13, pp. 210-216, 1966.

[GC84] E.M. Goldratt and J. Cox. *The Goal*. Gower, Aldershot, 1984.

[Hee94] K.M. van Hee. *Information Systems Engineering: a Formal Approach*. Cambridge University Press, 1994.

[HR00] K.M. van Hee and H.A. Reijers, Using Formal Analysis Techniques in Business Process Redesign, in Lecture Notes in Computer Science 1806, W. van der Aalst, J. Desel, and A. Oberweis (eds.). Springer-Verlag, pp. 142-160, 2000.

[Jac57] J.R. Jackson. Networks of Waiting Lines. *Operations Research*, vol. 5, pp. 518-521, 1957.

[Jac63] J.R. Jackson. Jobshop-like Queuing Systems. *Management Science*, vol. 10, pp. 131-142, 1963.

[Rol71] A. Rolfe. A Note on Marginal Allocation in Multiple-Server Service Systems. *Management Science*, vol. 17, pp. 656-659, 1971.

[Web80] R.R. Weber. On the Marginal Benefit of Adding Servers to G/GI/m Queues. *Management Science*, vol. 26, pp. 946-951, 1980.