The price of coordination in resource management

Kees van Hee, Alexander Serebrenik, Natalia Sidorova,
Marc Voorhoeve, and Jan van der Wal

Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{*K.M.v.Hee, A.Serebrenik, N.Sidorova, M.Voorhoeve, Jan.v.d.Wal*}*@tue.nl*

Abstract. We propose a resource management policy that grants or refuses requests for resources based only on the request made and the number of free resources. Computations at runtime are independent of the number of active cases. The policy requires little coordination and is therefore easy to implement in workflow management systems. This policy has been shown to be successful in avoiding deadlocks. In this paper we investigate its performance characteristics.

1 Introduction

Workflow nets [11–14], a special class of Petri nets, are frequently used to model business processes. In business processes, three elements are essential: *cases* to be processed, *tasks* to be performed on the cases and *resources* needed to perform these tasks. Typical examples of such resources are money, machinery and manpower. Traditionally, models of workflow nets emphasise the partial ordering of activities (i.e. executing a task for a case) in the process and abstract from the resources needed for them. The resources, however, cannot be ignored in many practical applications [1, 3, 5, 6, 10]. Resource-constrained workflow nets have been introduced in [15] for resources that are *durable* instead of *consumable*. Bad resource management may cause deadlocks, even if the workflow net is well-designed, i.e. the workflow net is *sound* (cf. [14]).

Assessment of business process models involves *correctness* and *efficiency*. Correctness requirements include *proper termination*, i.e. that given some minimal initial number of resources, in each reachable state of the business process it is possible to release all claimed resources and complete all open cases. Efficiency criteria are *quality of service* (e.g. the time between the arrival and the completion of cases) and *costs of operation* (e.g. the number of resource-hours needed).

We consider business processes where an arbitrary number of cases is handled and the resources belong to a single class. Cases are independent, i.e. they only communicate with the resource manager by claiming and releasing resources in various quantities. A resource manager, a human being or a software component in a workflow management system, may either grant or refuse these resource claims. In principle the resource manager may base its decisions on the *global state* of the process, i.e. the number of cases, the state of each case and the number of available resources. The first task of resource management is to ensure correctness of the resulting business process. This involves *scheduling*. Note that we are not dealing with a *static* scheduling problem in which all cases to be handled are known. Instead we are dealing with a *dynamic* scheduling problem, in which new cases arrive according to a random process and the routing and resource consumption of cases are largely unknown to the resource manager. The banker's algorithm of E.W. Dijkstra [4] ensures correctness in this way. This algorithm considers for each case only the maximal number of resources needed by it (credit limit) and the number of resources claimed and not yet returned so far (debt), plus the number of available resources.

The original paper [4] does not consider efficiency, but it leaves room for prioritising between cases that need resources, when they become available. Such a choice can be based on heuristics like FIFO (first in first out), SPT (shortest rest processing time) or EDD (earliest due date).

In [15], a property for cases is defined that we call *solidity*. When all cases are solid, the business process is guaranteed to terminate properly. Cases that are not solid can be *solidified* by setting a *threshold* for the number of resources that need to be available for each claim to be successful. In practise, solidification amounts to the following policy: *before committing resources to a case, make sure that there are enough resources available to allow its completion independently of other cases*. This is akin to a well known approach in production control [2]. The thresholds can be chosen in advance, when the business process is defined.

In this paper, we investigate resource scheduling based on solidification. Resources are granted to a task when the number of available resources exceeds a given threshold. To determine thresholds that perform well, an iterative, simulation-based approach is proposed. We illustrate our approach with a small example inspired by the building industry.

The sketched approach leads to a robust "uncoordinated" resource management. Note that the computation time needed at runtime by the resource manager is independent of the number of active cases. It is interesting to compare the performance of these robust resource managers w.r.t. more sophisticated ones, thus investigating the price of coordination (cost of robustness). For a very small (tandem queue) example, it is possible to compute an optimal global scheduler by Markov decision theory [9]. We compare the performance of the robust and optimal approaches.

The remainder of the paper is organised as follows. We describe our motivating example in Section 2. Basic notions from Petri net theory are defined in Section 3. Correctness and solidification are introduced in Section 4. Then, efficiency criteria are proposed and assessed by means of two examples in Section 5. Finally, we discuss the results presented.

2 Motivating example

The business processes investigated are modelled as workflow nets. Our example model describes the business process of a building contractor, who constructs buildings of a similar nature in large numbers for various clients. Each building under construction represents a case. Each case is divided into tasks, that are performed by subcontractors and require a certain amount of time to complete. The resource considered is money; the contractor has an account at his bank with a fixed credit limit. At the onset of a task, an amount of money has to be paid to the subcontractor. At termination of a task, the client pays some amount of money (not necessarily the same as paid to the subcontractor). All payments pass through the contractor's bank account. So each construction task can be characterised by three parameters: payment to a subcontractor, duration and payment from the client. The net model for each task is shown in Figure 1. Horizontally, the control flow is depicted; the remaining edges describe the resource flow. Initially money is transferred to the account of the subcontractor and money is received at the end.

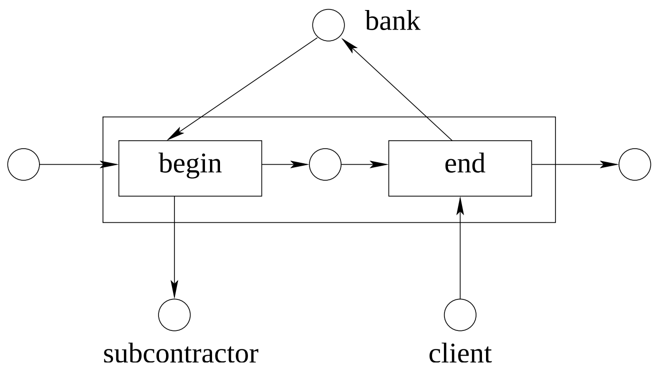

Fig. 1. A typical construction task

Construction starts by paying a subcontractor 2 mu (units of money, e.g. 20.000 euro) for the groundwork. This process takes twenty days and upon completion the client pays 1 mu. The next task is known as framing and includes building walls, windows and a roof. For this stage 4 mu is required by a subcontractor, and after thirty days the work is finished and 2 mu is paid by the client. Next, additional commissions (add-ons) of ten days long may be requested by the client. Each add-on requires 1 mu as initial payment to the subcontractor. The same amount is charged from the client when the task is finished. The independence of add-ons is modelled as a loop with 45% exit chance. Then, the internals of the building are installed (plumbing, heating, electricity). This task takes thirty days, requiring an initial sum of 1 mu for the subcontractor. This sum is payed by the client at termination of this task. Finally, when the construction is approved by the customer, she pays back the remaining 3 mu. The workflow net corresponding to this process description is presented in Figure 2. When considering correctness, we treat the tasks as transitions. However, when treating performance, the indicated timing delays must be observed and tasks are treated as subnets defined by Figure 1. For the sake of simplicity we abstract from the communication with subcontractors and clients, including money transfers.

The process described can deadlock and thus is not guaranteed to terminate properly. Suppose the credit limit equals 16 mu and the groundwork is started for 8 clients on days 1 to 8. On days 9 to 15, applications of 7 more clients are received, but these cases cannot start due to a lack of resources. On day 22, the groundwork for the first two clients has terminated, 2 mu is available and the groundwork for client 9 is started.

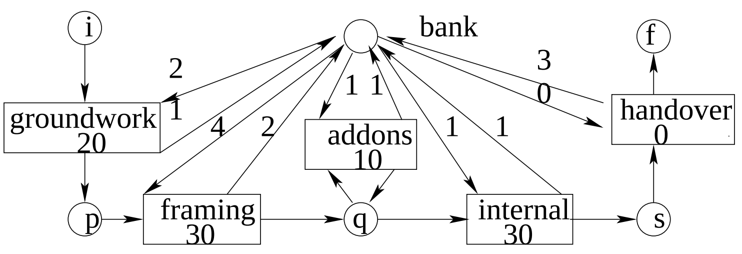

Fig. 2. Workflow of a building construction

By immediately starting groundwork for new clients as soon a 2 mu becomes available rather than waiting for 4 mu and starting framing activities, we will arrive at a state where only 1 mu is available and 15 clients are in state p, which is a deadlock state. Such a deadlock can be achieved for larger credit limits as well.

As explained in the introduction, several deadlock avoidance policies can be implemented. Dijkstra's scheduling algorithm [4] will prevent groundwork for the 13-th client to start in the above scenario. This will allow framing to start for at least one client, eventually freeing the resources invested. Deadlock is likewise avoided by solidifying the process, setting a threshold of 3 mu for the groundwork task. Groundwork is started for a new client, investing 2 mu, only if 5 mu (the required 2 plus the threshold amount) are available.

However, deadlock avoidance alone does not guarantee a good performance. Assume that all cases have one add-on option. With the threshold 3 and 16 mu, it is possible to reach a global state where 12 cases are waiting in state p and for one case framing has started. After 70 days, 5 mu becomes available, which allows either to start groundwork for a new case or to start framing for a waiting case. If the groundwork option is chosen, 3 mu are left, so framing has to wait 20 more days, upon which framing can start for one case and we are back in the initial state. Thus, only one case in 90 days is completed. It is not difficult to find a different resource management scenario allowing the completion of 3 cases every 70 days. The reason for the bad performance of the sketched scenario is that the number (13) of active cases is too high. By increasing the minimum threshold of 3 for starting the groundwork of a new case, the number of active cases can be reduced.

3 Preliminaries

We adopt standard notation for sets, bags and transition systems. A *Petri net* is a tuple $N = \langle P, T, F^+, F^- \rangle$, where:

- P and T are two disjoint non-empty finite sets of *places* and *transitions* respectively, the set $P \cup T$ are the *nodes* of N;
- F^+ and F^- are mappings $(P \times T) \to \mathbb{N}$ that are *flow functions* from transitions to places and from places to transitions respectively.

We present nets with the usual graphical notation.

Markings are states (configurations) of a net. We denote the set of all markings reachable in net N from marking m as $\mathcal{R}(N,m)$. We will drop N and write $\mathcal{R}(m)$ when no ambiguities can arise. Given a transition $t \in T$, the *preset* $^\bullet t$ and the *postset* $t^\bullet$ of t are the *bags* of places where every $p \in P$ occurs $F^-(p,t)$ times in $^\bullet t$ and $F^+(p,t)$ times in $t^\bullet$. Analogously we write $^\bullet p, p^\bullet$ for pre- and postsets of places. We will say that a node n is a *source* node if and only if $^\bullet n = \emptyset$ and n is a *sink* node if and only if $n^\bullet = \emptyset$.

A transition $t \in T$ is *enabled* in marking m if and only if $^\bullet t \leq m$. An enabled transition t may *fire*. This results in a new marking m' defined by $m' \overset{\text{def}}{=} m - {}^\bullet t + t^\bullet$. We interpret a Petri net N as a transition system/process where markings play the role of states and firings of the enabled transitions define the transition relation, namely $m + {}^\bullet t \xrightarrow{t} m + t^\bullet$. The notion of reachability for Petri nets is inherited from the transition systems. A net $N = \langle P, T, F^+, F^- \rangle$ is called a *state machine* if $^\bullet t$ and $t^\bullet$ are singleton bags for all $t \in T$.

Given a Petri net, a *place invariant* is a row vector $I : P \to \mathbb{Q}$ such that $I \cdot F = 0$.

In this paper we primarily focus upon the *Workflow Petri nets (WF-nets)* [11]. As the name suggests, WF-nets are used to model the processing of tasks in workflow processes. The initial and final nodes indicate respectively the initial and final states of processed cases.

Definition 1. *A Petri net N is a* Workflow net (WF-net) *if and only if:*

1. *N has two special places: i and f. The initial place i is a source place, i.e. $^\bullet i = \emptyset$, and the final place f is a sink place, i.e. $f^\bullet = \emptyset$.*
2. *For any node $n \in (P \cup T)$ there exists a path from i to n and a path from n to f.*

We extend the notion of WF-nets in order to include information about the use of resources into the model. A resource belongs to a type; we have one place per resource type in the net where the resources are located when they are free. We assume that resources are durable, i.e. they can neither be created nor destroyed, they are claimed during the handling procedure and then released again. By abstracting from the resource places we obtain the WF-net that we call *production net*.

Definition 2. *We will say that a WF-net $N = \langle P_p \cup P_r, T, F_p^+ \cup F_r^+, F_p^- \cup F_r^- \rangle$ with initial and final places $i, f \in P_p$ is a* Resource-Constrained Workflow net (RCWF-net) *with the set P_p of* production places *and the set P_r of* resource places *if and only if*

- *$P_p \cap P_r = \emptyset$,*
- *F_p^+ and F_p^- are mappings $(P_p \times T) \to \mathbb{N}$,*
- *F_r^+ and F_r^- are mappings $(P_r \times T) \to \mathbb{N}$, and*
- *$N_p = \langle P_p, T, F_p^+, F_p^- \rangle$ is a WF-net, which we call the* production net *of N.*

The processes that we consider can be modelled as WF nets with only one resource place, where the production net is a state machine (SM1WF-nets). In [15] it is shown that a business process modelled by an arbitrary workflow net can be converted to a state machine workflow net, provided that cases are independent.

Definition 3. *An RCWF-net $N = \langle P_p \cup P_r, T, F_p^+ \cup F_r^+, F_p^- \cup F_r^- \rangle$ is called a* state machine workflow net with one resource type (SM1WF-net) *if $P_r = \{r\}$ and the production net N_p of N is a state machine.*

Observe that the net in Figure 2 is indeed an SM1WF-net.

4 Correctness

As explained in the introduction the correctness criterion we consider is *proper termination*, also known as *soundness* in WF-nets. Proper termination is the property that every marking reachable from an initial marking can reach the corresponding final marking. Initial markings of the net have some tokens (say k) in the initial place and a number of resource tokens on each resource places. The corresponding final marking has k tokens in the final place; the resource places must contain the same number of tokens as initially. We assume that the number of resource available initially is sufficient.

Another correctness requirement that should be reflected by the definition is that resource tokens cannot be created during the processing, i.e. at any moment of time the number of available resources does not exceed the number of initially given resources. The definition of proper termination reads thus as follows:

Definition 4. *Let N be an RCWF-net.*
*N is (k,r)-*sound *for some $k \in \mathbb{N}, r \in \mathbb{N}^{P_r}$ if and only if for all $m \in \mathcal{R}(k[i]+r)$ holds: $m_r \leq r$ and $m \xrightarrow{*} (k[f]+r)$.*
N is sound *if and only if there exists $r \in \mathbb{N}^{P_r}$ such that it is (k,r')-sound for all $k, r' \in \mathbb{N}, r' \geq r$.*

In [15], it is proved that for any sound SM1WF-net there exists a unique place invariant W such that $W(i) = W(f) = 0$, $W(r) = 1$ and for all $p \in P_p$, $W(p) \geq 0$. Given a place p we call $W(p)$ the *weight* of p.

Example 1. Recall the construction Petri net presented in Figure 2. Then, $W(i) = W(f) = 0$, $W(p) = 1$, $W(q) = W(s) = 3$.

The discussion in Section 2 shows that the existence of a place invariant is necessary but not sufficient.

4.1 Solidity

The key ingredient in determining proper termination of SM1WF nets, called *solidity*, is the possibility that all resources claimed are eventually released. Important in the algorithm is the *path* concept. A path is a sequence of transitions such that the output state of a transition is the input state of the next transition. A path has an input and output state, resp. the input state of the first transition and the output state of the last transition. A path p is a *successor* of a path q if the input state of p equals the output state of q. If the weight of the input state of a path is less than the weight of its output state, the path is called a *consumption* path, if it is more than the weight of the output state the path is called a *production* path. Finally, we define the *resource need* of a path. This is the minimum number of resources needed for the execution of the path.

In our example net, the sequence $(\mathit{framing}, \mathit{addons}, \mathit{internal})$ is a path with input place p and output place s. Since p has weight 1 and s has weight 3, it is a consumption path. Its resource need is 5, since 4 free resources plus 1 resource occupied in the input place p are sufficient to fire the sequence, leading to s occupying 3 resources plus 2 free resource.

The above definition allows to formulate the necessary and sufficient condition for solidity: *Each consumption path produces enough resources to fulfil the resource need of at least one successive production path.*

Our example net does not satisfy this condition: the path $\sigma = (groundwork)$ (consisting of only one transition) is a consumption path, since its input place i has weight 0 and its output place p has weight 1. Its resource need equals 2. Any production path succeeding σ has input place p and thus must start with transition *framing* needing 4 free resources, so the resource need of such a production path is at least 5 (4 free ones and one occupied by p). So σ has no production successor with a resource need not exceeding 2.

In order to verify solidity, given an SM1WF-net with P_p production places, we introduce a matrix M, such that $M(p,p)$ is defined to be $W(p)$ for all $p \in P_p$ and $M(p,q)$ is defined as the sum of $W(q)$ and the minimal resource production of transitions from p to q. If there are no such transitions $M(p,q)$ is defined to be ω (denoting infinity). The condition above has considered paths rather than individual transitions. Therefore, we need to extend the definition of M to include paths of arbitrary length. To do so, we have introduced a binary operation $\circ$ such that for any $A,B : P_p \times P_p \to \mathbb{N}$, the product $A \circ B$ is defined as $C : P_p \times P_p \to \mathbb{N}$ where $C(p,q) = \min\{\max(A(p,r),B(r,q)) \mid r \in P_p\}$. We denote $A \circ A$ by A^2 etc. One can show that $M, M^2, M^4, \ldots$ converges to a fixpoint, which we call μ. Then, the intuitive condition for solidity stated above can be expressed as follows:

Corollary 1. *([15]) The SM1WF-net N is solid if and only if*

$$\forall x \in P_p : \min_y\{\mu(y,x) \mid W(y) < W(x)\} \geq \min_z\{\mu(x,z) \mid W(x) > W(z)\}.$$

In our running example, the following holds:

$$M = \begin{array}{c|ccccc} & i & p & q & s & f \\ \hline i & 0 & 2 & \omega & \omega & \omega \\ p & \omega & 1 & 5 & \omega & \omega \\ q & \omega & \omega & 3 & 4 & \omega \\ s & \omega & \omega & \omega & 3 & 3 \\ f & \omega & \omega & \omega & \omega & 0 \end{array} \quad M^2 = \begin{array}{c|ccccc} & i & p & q & s & f \\ \hline i & 0 & 2 & 5 & \omega & \omega \\ p & \omega & 1 & 5 & 5 & \omega \\ q & \omega & \omega & 3 & 4 & 4 \\ s & \omega & \omega & \omega & 3 & 3 \\ f & \omega & \omega & \omega & \omega & 0 \end{array} \quad M^4 = \begin{array}{c|ccccc} & i & p & q & s & f \\ \hline i & 0 & 2 & 5 & 5 & 5 \\ p & \omega & 1 & 5 & 5 & 5 \\ q & \omega & \omega & 3 & 4 & 4 \\ s & \omega & \omega & \omega & 3 & 3 \\ f & \omega & \omega & \omega & \omega & 0 \end{array}$$

M^4 is the fixpoint. Soundness condition is violated by p since

$$\min\{\mu(y,p) \mid W(y) < W(p)\} = 2 < 5 = \min\{\mu(p,z) \mid W(p) > W(z)\}.$$

However, it is possible to *solidify* SM1WF nets with a resource invariant. This is done by *thresholding* some of its transitions. A transition is thresholded by not firing it when the resources it needs are available *unless* some additional resources are available too. The amount of required extra resources is called the *threshold*. Thresholding replaces scheduling as by Dijkstra's algorithm ([4]; it is similar to the order acceptance strategy of ([2]). We have developed a method to find algorithmically minimal thresholds for a net to become solid. It should also be noted that any threshold exceeding the minimal one makes the net solid too.

The threshold solution proposed at the end of Section 2 has been obtained by the solidification technique.

5 Efficiency

5.1 Defining efficiency

In order to estimate the quality of resource management, we need to define efficiency criteria. As mentioned in the introduction, we distinguish two kinds of criteria: those considering the quality of service, and those considering the cost of operation. Quality of service is focused towards minimising the throughput or cycle time of a case, i.e. the time between the arrival and the completion of the case. In our case minimising the throughput is equivalent to minimising the waiting time of a case, since the processing times of tasks are independent of the resource allocation. For a random stream of cases it is natural to consider as quality of service the average cycle time or waiting time:

$$\lim_{n\to\infty} \frac{\sum_{i=1}^{n} w(i)}{n},$$

where $w(i)$ is a waiting time of case i. An alternative for the minimising the average expected time is minimising the probability that a case has a waiting time larger than some given bound.

For the cost of operation we consider as criterion the average expected use of resources. However we need an additional condition for this criterion to eliminate the situation when the cost of operation is zero but no cases are handled. There are at least two approaches to deal with this: one can assume some minimum level of service as a boundary condition (like the average expected waiting time) or one can assign cost to resources and rewards for handling cases. Then the cost of operation is transformed into the value of the operation by subtracting the the reward of handled cases from the average cost of resource usage. We choose this last option; a negative cost of operation corresponds to making profit.

To evaluate the solidification technique, we computed an optimal scheduler described in Section 5.2, comparing it to the solidification approach. Unfortunately, optimisation is only feasible for very small examples, like the tandem queue described in the next subsection. To tackle our building example, only simulation techniques have been used (Section 5.3).

5.2 Tandem queue example

In order to determine what extra cost we incur in case of the solidification approach, we need to find an optimal resource allocation strategy. Such a strategy defines for each possible global state a decision for resource assignment. Finding an optimal strategy is, in general, an extremely difficult task. Therefore, we consider a *tandem queue example*.

The tandem queue example has two sequential tasks as shown on Figure 3. Like in Figure 2, numbers inside the task boxes denote the duration of the task. We assume exponential service times with mean service times equal to 1 for both tasks. [1] The arrival

[1] In order that the system satisfies the Markovian property, we need at least phase type service time distributions. However, since each extra phase adds an extra dimension to the state space, the number of phases has to be limited so that the Markov decision approach is still feasible. For random service times with a squared coefficient of variation larger than 0.5, two phases suffice to mimic the first two moments. Here we have chosen for exponential service times.

process is Poisson with 4 arrivals per time unit on average. For this example the optimal strategy can be calculated by means of a Markov decision process theory [9, 7], using techniques such as successive approximation.

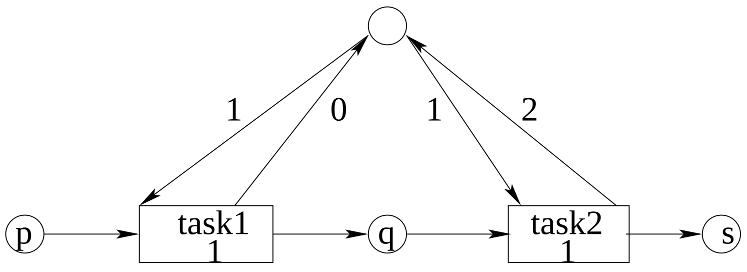

Fig. 3. Tandem queue example

To model this workflow system as a Markov decision process with continuous time we consider the two tasks as multi-server service stations, where the servers are the resources. The state of the Markov decision process is a quadruple $\langle q_1, t_1, q_2, t_2 \rangle$ where q_1 and q_2 are the number of waiting cases for the first and the second tasks and t_1 and t_2 are the number of cases treated in these service stations respectively. The possible actions are: admission of a newly arrived case, assigning resources to a waiting case in the queue q_1 and assigning resources to a waiting case in the queue q_2. Using the well-known uniformisation technique [7] we translate the model with continuous time into a Markovian decision process with discrete time. In order to determine the optimal strategy we need a finite state space, so we need to add some more restrictions. We assume $q_1 \leq 3$ and $q_2 \leq 3$. These assumptions imply that when a new case arrives to a queue but the queue is full, the case is lost. In order to achieve finiteness of the state space, t_1 and t_2 should be bounded as well. We set $t_1 \leq 4$ and $t_2 \leq 7$. Under these restrictions, the number of states space of the system is bounded.

The boundedness restrictions are not too severe if the loss of cases lost is penalised. High penalties imply that queue overflow is avoided as much as possible. Case loss in q_1 is punished by 5 cost units, case loss in q_2 by 10 cost units. We consider these penalties as opportunity costs.

Finally, to complete the specification of the Markov decision process, the performance measures have to be chosen. We consider two different measures, namely quality of service (QoS) and cost of operation (CoO). For the QoS measure, the waiting times at $q1$ and $q2$ plus the opportunity costs for queue overflow are added. For the CoO measure, we consider the average resource occupation of a case, i.e. the processing time of the first station plus the waiting time at $q2$ (where one resource is occupied by the case) plus twice the processing time of the second station (where two resources are occupied). As explained earlier, we subtract the reward for completed cases, which equals 7. After the subtraction, the cost of operation becomes negative for successful executions. Hence, minimising the cost of operation reflects the best possible scenario from “a case’s point of view”.

Using the standard successive approximation techniques (c.f. [7]), we derive an optimal strategy: it defines an action for each possible state. Since the state space consists of 640 states, the strategy becomes rather complex. Recall that the Bellman equation describes successive approximations $v_0, \ldots$ as follows:

$$\begin{aligned} v_0(s) &= 0 \\ v_{n+1}(s) &= \min_{a \in A}\{c(s,a) + \sum_{s' \in S} P(s' \mid s,a) v_n(s')\}, \end{aligned}$$

where $s \in S$.
In the equations above, S is the set of all possible states, i.e. the set of quadruples $\langle q_1, t_1, q_2, t_2 \rangle$, A is the set of all possible actions: $\{\mathit{reject}, \mathit{assign\ to}\ q_1, \mathit{assign\ to}\ q_2\}$, $P(s' \mid s,a)$ is the probability to move from the state s to state s' by executing an action a, and $c(s,a)$ is the cost per time unit when the system is in state s and action a is taken. The value $v_n(s)$ is the total cost for running the system for n time units from state s under an optimal strategy. We consider the two cost functions QoS and CoO as described above.

We note that the average cost g per time unit under an optimal strategy satisfies:

$$\min_{s \in S}(v_{n+1}(s) - v_n(s)) \leq g \leq \max_{s \in S}(v_{n+1}(s) - v_n(s))$$

In case the process is recurrent, which means that from every state one can reach every other state, these two bounds converge to the same value. In this way one can compute the exact values of quality of service and cost of operation for the tandem queue. The optimal values found are

$$\begin{aligned} \text{QoS}: &\ 1.73 \\ \text{CoO}: &\ -2.88. \end{aligned}$$

Next, we apply our solidification approach. We start by observing that the place invariant W exists and satisfies $W(p) = W(s) = 0$, $W(q) = 1$. Therefore, the solidification approach is applicable. The powers of the matrix M are defined as follows:

$$M = \begin{array}{c|ccc} & p & q & s \\ \hline p & 0 & 1 & \omega \\ q & \omega & 1 & 2 \\ s & \omega & \omega & 0 \end{array} \qquad M^2 = \begin{array}{c|ccc} & p & q & s \\ \hline p & 0 & 1 & 2 \\ q & \omega & 1 & 2 \\ s & \omega & \omega & 0 \end{array}$$

The fixpoint μ is reached with M^2. This net is unsound since

$$\min\{\mu(y,q) \mid W(y) < W(q)\} = 1 < 2 = \min\{\mu(q,y) \mid W(y) < W(q)\}.$$

Solidification requires that the resource request of the first task is granted only if there is at least one more resource available. In this way one guarantees that at least one resource is available when a case arrives at place q in the net, so that the second transition can fire. Observe that if more than one additional resource is available, proper termination is guaranteed as well. Therefore, we have performed a number of simulation runs for different values of the parameter *k*—the number of additional resources—ranging from one to three. For each one of the values of *k*, two priority configurations were considered. The first uses FCFS (first come first served) priority for all tasks of all cases. The second uses SPT priority for tasks (i.e. the second task has priority over the first one) and FCFS for the same task of different cases. In both cases, "greedy" resource allocation takes place: if the set of tasks that can be started (i.e. for which the number of resources is at least the requested number plus the threshold) is nonempty, some task will start immediately.

Finally, for each one of the cases two values have been measured: cost of operation (CoO) and quality of service (QoS).

Strategy	Threshold	QoS	CoO
Optimal		**1.73**	**-2.88**
Solidification; FCFS	0	3.02	-2.13
	1	2.14	**-2.67**
	2	**2.09**	-2.72
	3	2.53	-2.44
Solidification; SPT+FCFS	0	2.23	-2.60
	1	**1.94**	**-2.80**
	2	2.09	-2.72
	3	2.53	-2.44

Table 1. QoS and CoO for the tandem queue example

Table 1 represents our results for quality of service and cost of operation. The performance w.r.t. the optimal strategy is given at the top. Then come solidification strategies with various threshold values. The thresholds indicated represent the extra number of available resources required for entering the queue of the first task or station. Since the mathematical model has queues with finite capacity, no deadlocks are possible, so solidification is not required. The cases that are waiting in the queues are treated in FCFS order; when resources become available, the longest waiting case that can be served is selected. The third group of results stem from the solidification extended with the SPT priority rule. When resources become available, the second station has priority. Cases of that station are treated in FCFS order.

We observe that the simulation minima are obtained for QoS for a threshold value of 1 for the prioritised configuration and 2 without SPT priorities, and for CoO for threshold value 2, either with or without priorities. The exact values are 1.94 for QoS and -2.80 for CoO.

While comparing the theoretical results with the results of simulation, we observe that the relative error is quite small for CoO and somewhat larger for QoS. Probably, this difference is caused by the penalty for lost opportunities.

By examining scenarios where the solidification approach makes suboptimal decisions, it is possible to arrive at heuristics that improve upon the decisions made. It seems that information predicting the *future* availability of resources can be of some use here. Of course, improving performance in this way decreases the robustness, i.e. more information is needed and a less straightforward computation.

5.3 Simulation results for the construction example

A simulation study in Arena [8] has been conducted to determine the optimal solidification of the example net from Figure 2. We assumed that the credit limit is 50 mu and that the arrival of new customers is Poisson with the average time between arrivals being 8.5 days.

Strategy	Threshold	QoS	CoO
Solidification;	1	deadlock	
FCFS	2	deadlock	
	3	106.05	17.59
	4	**43.48**	6.93
	5	51.19	**6.89**
Solidification;	1	deadlock	
SPT+FCFS	2	50.93	7.71
	3	45.66	6.77
	4	**38.79**	**5.97**
	5	51.15	6.49

Table 2. QoS and CoO for the construction example

In Table 2, the simulation results for resource management in our building example are given. In the first series of simulations, resources are assigned to transitions based on the FCFS principle. This means that when the resources become available, the longest waiting case that has become enabled can continue. In the second series, this FCFS strategy is extended with the SPT priority rule, like in the tandem queue example. The first task (groundwork) gets lowest priority, next comes the framing task and the other two tasks have highest priority. Parameter of the simulation is again the threshold value for the groundwork task, which ranges from one to five. Thresholds one and zero yield deadlock in both cases. With a threshold of two, the simulation does not deadlock in combination with the SPT priority rule, although this is theoretically possible. Thresholds greater than three do solidify the net. For thresholds above five, both the CoO and QoS performance measure increase rapidly, so they have not been listed.

Summarising these results, we observe that the best quality of service is always achieved for threshold four: 43.48 in the FCFS case and 38.79 for SPT extension. Unlike this, the optimal threshold for the lowest cost of operation depends on the resource assignment policy. For FCFS, the minimum is obtained for threshold five (6.89), while for the SPT extension it happens for threshold four (5.97).

6 Conclusion

In this paper, we have presented a way, called *solidification*, to obtain a deadlock free scheduler that requires minimal coordination. The computation of this scheduler needed at runtime are independent of the number of active cases. This can be of importance in the implementation of workflow management systems.

We have studied the price of coordination in resource management, i.e. the difference in performance between the optimal (global) and the robust (local) scheduler based on the solidification approach. The performance criteria studied did correspond to quality of service and cost of operation respectively. Our experiments indicate that the performance loss due to a minimal coordination scheduler is not too high, but more convincing realistic case studies are sorely needed. The solidification scheduler can be significantly improved by extending it with an SPT priority rule.

For further research, it is interesting to apply our method to real-life resource scheduling problems. As it is computationally infeasible to compute an optimal scheduler for such processes, comparisons have to be made with heuristic schedulers used in practice. A second line of investigation is the improvement of our resource allocation strategy by adding more information without compromising robustness too much.

References

1. K. Barkaoui and L. Petrucci. Structural analysis of workflow nets with shared resources. In *Workflow management: Net-based Concepts, Models, Techniques and Tools (WFM'98)*, volume 98/7 of *Computing science reports*, pages 82–95. Eindhoven University of Technology, 1998.
2. J. Bertrand, J.C.Wortmann, and J.Wijngaard. *Production Control, A Structural and Design Oriented Approach*. Educatieve Partners, 1998. Second revised edition.
3. J. Colom. The resource allocation problem in flexible manufacturing systems. In W. van der Aalst and E. Best, editors, *Application and Theory of Petri Nets 2003, ICATPN'2003*, volume 2679 of *Lecture Notes in Computer Science*, pages 23–35. Springer-Verlag, 2003.
4. E. W. Dijkstra. *Selected Writings on Computing: A personal Perspective*. Texts and Monographs in Computer Science. Springer Verlag, 1982.
5. J. Ezpeleta. Flexible manufacturing systems. In C. Girault and R. Valk, editors, *Petri nets for systems engineering*. Springer-Verlag, 2003.
6. J. Ezpeleta, J. M. Colom, and J. Martínez. A Petri net based deadlock prevention policy for flexible manufacturing systems. *IEEE Transactions on Robotics and Automation*, 11(2):173–184, 1995.
7. E. Feinberg and A. Shwartz. *Handbook of Markov Decision Processes: Methods and Algorithms*. Kluwer, 2002.
8. W. Kelton, R. Sadowski, and D. Sadowski. *Simulation with Arena*. McGraw-Hill, 1998.
9. M. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. Wiley, New York, 1994.
10. M. Silva and E. Turuel. Petri nets for the design and operation of manufacturing systems. *European Journal of Control*, 3(3):182–199, 1997.
11. W. M. P. van der Aalst. Verification of workflow nets. In P. Azéma and G. Balbo, editors, *Application and Theory of Petri Nets 1997, ICATPN'1997*, volume 1248 of *Lecture Notes in Computer Science*. Springer Verlag, 1997.

12. W. M. P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.
13. W. M. P. van der Aalst. Workflow verification: Finding control-flow errors using Petri-net-based techniques. In W. M. P. van der Aalst, J. Desel, and A. Oberweis, editors, *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, pages 161–183. Springer-Verlag, 1999.
14. W. M. P. van der Aalst and K. M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT Press, 2002.
15. K. van Hee, A. Serebrenik, N. Sidorova, and M. Voorhoeve. Soundness of resource-constrained workflow nets. In G. Ciardo and P. Darondeau, editors, *Application and Theory of Petri Nets 2005, ICATPN'2005*, Lecture Notes in Computer Science. Springer Verlag, 2005. accepted.