CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2013; **25**:427–442
Published online 12 June 2012 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.2855

Incorporating structural improvement into resource allocation for business process execution planning‡

Jiajie Xu [1,2,*,†], Chengfei Liu [1], Xiaohui Zhao [3] and Zhiming Ding [2]

[1] *Faculty of ICT, Swinburne University of Technology, Melbourne, Australia*
[2] *NFS, Institute of Software, Chinese Academy of Sciences, Beijing, China*
[3] *Faculty of Information Sciences and Engineering, University of Canberra, Canberra, Australia*

SUMMARY

Resource management has been recognised as an important topic for business process execution for a long time. Most existing works on resource allocation for business processes simply assume that the structure of a business process is always fixed, and therefore do not discuss the possibility of optimising resource allocation by adapting process structures to actual resource situations. To fill this gap, we propose a resource optimisation approach of improving process structures according to resource situations and thereby pursuing the best resource utilisation efficiency. This approach comprises a role-based business process model for resource allocation and the strategies for optimising resource allocation in conjunction with a business process improvement. A set of heuristic rules are established to guide the resource allocation for the purposes of preventing resource conflicts, shortening the total execution time, minimising the total cost, etc. Particular algorithms are also developed to implement the resource allocation according to these rules. In addition, an experimental study is conducted to discuss the incorporation of business process improvement into resource allocation for optimal process execution. Copyright © 2012 John Wiley & Sons, Ltd.

Received 18 May 2010; Revised 5 February 2012; Accepted 4 April 2012

KEY WORDS: business process management; resource allocation; process change management

1. INTRODUCTION

Nowadays, business process performance has become a key issue for enterprises to thrive and survive in fierce market competition. Enterprises expect to ensure their business processes achieve business goals with lower cost, shorter time, better quality, etc. Because business process execution relies on the necessary resources, its performance is directly subject to the matching extent between available resources and the given business process. To guarantee the matching quality and the business process fulfilment, it is essential to plan the process execution and resource allocation at build time.

We reckon that resource allocation and business process structural improvement impact on each other. On one side, business process structure confines resource allocation because of the task dependencies. So far, many works [1–3] have proposed to investigate this impact from business process structure to resource allocation and planning. On the other side, the business process structure may not fit into the ideal resource utilisation scheme, and thereby may result in sacrifice in cost or time. Therefore, as illustrated in Figure 1, we combine these issues together for optimal process execution pattern planning, when the process requirements are changed, or resource

*Correspondence to: Jiajie Xu, Faculty of ICT, Swinburne University of Technology, Melbourne, Australia.
†E-mail: jiajie@nfs.iscas.ac.cn
‡The initial work was published in Proc. of 6th International Conference on Business Process Management (BPM 2008), pp. 228–243, September 2008, Milan, Italy. Lecture Notes in Computer Science, Vol. 5240, Springer.

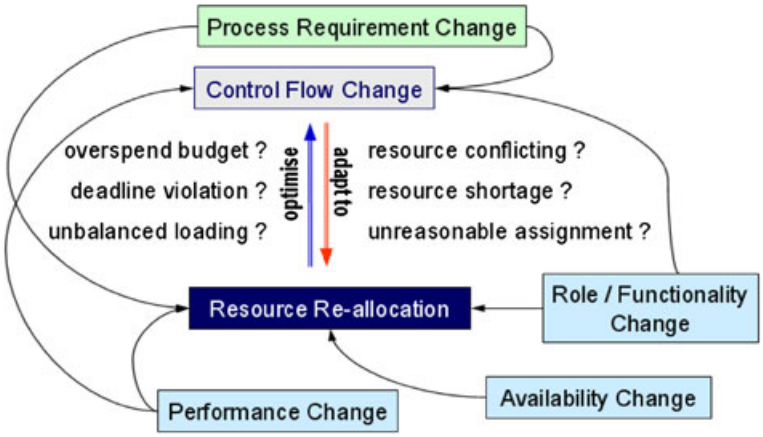


Figure 1. Resource allocation and process improvement.

environment is changed because of the resource availability, role/functionality, and performance changes. For example, if a resource scheduled for a job suddenly becomes unavailable (availability change), or loses the permission (functionality change) to execute some tasks, the original planning schema may become invalid. Also, the given process structure may not be optimal after the environment change, and hence prevents resources from being effectively used. In such cases, it is necessary to conduct both resource reallocation and structural improvement. By adapting business process structure to available resources, the resource utilisation can be further improved, and so does the total business performance. However, to our best knowledge, few efforts have been put to discuss this issue. In this paper, we particularly target at the interinfluence between resource allocation optimisation and process structure improvements, and discuss how to optimise process execution planning by incorporating them together.

To incorporate resource allocation into business process improvement, this paper proposes a role-based business process model to specify the dependencies between business processes, tasks, roles and resources. On the basis of this model, a comprehensive framework is established to preanalyse and optimise resource allocation together with business process improvement, and thereby adjust the two to the best match. The contribution of this paper to current state of the art lies in the following aspects:

- Optimise the resource allocation for business processes at build time, and therefore enable to precheck if some resource allocation requirements can be satisfied before execution;
- Enable business process structure change to better optimise resource allocation;
- Develop algorithms for allocating resources to a business process to seek minimal cost under certain time constraint.

The remainder of this paper is organised as follows: Section 2 reviews the related works, and discusses the advantages of our approach; Section 3 introduces a role-based business process model, which defines the related notions for resource allocation, and the relationship among these notions; Section 4 introduces a role-based business process model for resource allocation, and based on the model the problem we intend to address is defined. The strategy for resource allocation optimisation and resource-oriented business process improvement are presented in Section 5. The experimental study of our strategies is conducted in Section 6. Lastly, concluding remarks are given in Section 7.

2. RELATED WORK

The resource scheduling problem has been discussed at task level and workflow level in previous works. Task level scheduling is based on independent tasks. Many algorithms have been proposed to schedule tasks within homogenous systems in previous literatures such as in [4–6]. Our paper investigates the resource allocation in a heterogeneous environment, and for heterogeneous

DOI: 10.1002/cpe

scheduling many works have been carried out. Some classical heuristic methods such as genetic algorithm [5,7], simulated annealing [8], duplex [9], and colony simulation [10] have been proposed to address this issue. Also, Topcuoglu *et al*. have proposed an earliest-finished-time algorithm in [11]. The work in [12] targets to schedule tasks with optimisation criteria similar to our problem, that is, minimise transition cost within a rigid deadline for completion. However, all the works on task level scheduling have their limit in effective resource management because they do not follow the structure of a business process.

Some recent progress of the processes scheduling and planning problems are generalised in the survey [13]. Compared with task scheduling, workflow scheduling is based on tasks of compulsory execution order constraint. Works in [14, 15] modelled the workflow with resource constraints. In addition, how time constraints impact on the run time resource scheduling for workflows have been investigated in many works like Ref. [16]. Also, many techniques have been developed for the temporal verification on workflows at the build time, and thereby enforcing these constraints at run-time. A number of Grid workflow management systems such as Refs. [17, 18] have proposed scheduling algorithms to facilitate the workflow execution and minimise the execution time. However, those works only considered single optimisation criterion, while in many applications we have to balance different goals and constraints. Yu and Buyya [19] considered both execution time and execution cost in scheduling scientific workflows for utility Grid applications, and the optimization is made to minimize the overall cost while meeting user's deadline constraint. In addition, a workflow execution planning approach in [20] was proposed to generate a set of trade-off schedules according to two user specified QoS requirements. Targeting to maximize the overall throughput, Liu *et al*. [21] proposed a throughput maximization strategy to schedule the transaction-intensive workflows. Eckert *et al*. [22] investigated resource planning for service oriented workflows. Also, in the area of Web service composition, several approaches [23–25] have been proposed to deal with the planning problem based on artificial intelligence planning methods in a bottom–up manner. These works are based on an assumption that the given process structure is optimal, and resource allocation is based on this fixed structure. However, this assumption may not be always true, and the nonoptimal process structure could become a bottleneck for the performance of planning.

In contrast to the discussed works, the work discussed in this paper is intended to improve business process performance by integrating resource allocation with business process structural improvement. Compared with existing approaches, our approach has the following features:

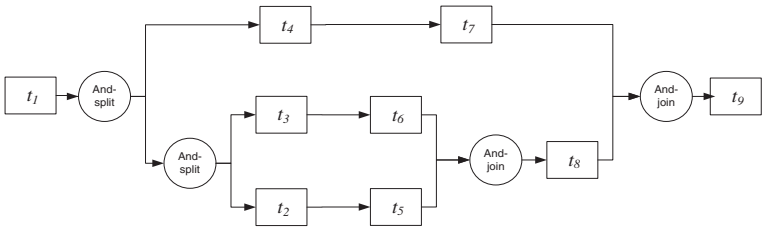
(1) Business process improvement for bettering resource allocation. In our approach, business process structure is allowed to be tuned to better adapt to available resources when necessary.
(2) Resource optimisation based on business process characteristics. In our approach, business process characteristics, including the constraints and dependencies predefined by the process structure, are preserved in the resource allocation.
(3) Requirement-oriented resource allocation. Our approach fully guarantees the predefined conditions for resource allocation, for example, the time constraint and cost requirements.

3. MOTIVATING EXAMPLE

We use an example to illustrate the problem that we are tackling in this paper. Figure 2 shows a business process with nine tasks and four gateways. Assume that the set of resources used for this business process are given in Table I and are classified according to roles. The cost for each role is also shown in the table. A role describes the capability of its resources for performing certain tasks. For each task, the roles that are capable of performing it are shown in Table II. The time for a role to perform a task is also indicated in the table. For example, Resources s_{31} and s_{32} can perform role r_3 at the cost of \$40 per hour. Task t_5 can be performed by resources of role r_1 and role r_4 in 2 h and 2.5 h, respectively.

Time and cost are two major criteria for business process performance evaluation. Assume resources are allocated as Figure 3(a). The time required is 9 h, and meanwhile the expense is \$637.5. In the situation of allocation as Figure 3(b), the cost is reduced to \$565, while the execution time is increased to 11.5 h. In reality, an enterprise always has a time constraint on a production business

DOI: 10.1002/cpe



t_1: Overall Design t_2: Function Verification t_3: Database Verification
t_4: UI Develop t_5: Function Develop t_6: Database Develop
t_7: UI Testing t_8: FD Testing t_9: Overall Testing

Figure 2. Business process structure.

Table I. Resource classification.

Role	Name	Cost ($/h)	Resource
r_1	Expert developer	50	s_{11}, s_{12}
r_2	Senior developer	25	s_2
r_3	DB developer	40	s_{31}, s_{32}
r_4	Junior developer	20	s_4
r_5	Tester	25	s_{51}, s_{52}

Table II. Capabilities of roles.

	Task								
Resource	t_1,	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
r_1	2 h	2 h		2 h	2 h		1 h	1 h	
r_2		3 h	1.5 h						
r_3			1 h			2 h		2 h	
r_4				2 h	2.5 h				
r_5							2 h		3 h

Task	Resource
t_1	s_2
t_2	s_{12}
t_3	s_2
t_4	s_4
t_5	s_{12}
t_6	s_{31}
t_7	s_{51}
t_8	s_{11}
t_9	s_{52}

(a)

Task	Resource
t_1	s_2
t_2	s_2
t_3	s_{31}
t_4	s_4
t_5	s_4
t_6	s_{32}
t_7	s_{52}
t_8	s_{11}
t_9	s_{51}

(b)

Task	Resource
t_1	s_2
t_2	s_{12}
t_3	s_2
t_4	s_4
t_5	s_4
t_6	s_{32}
t_7	s_{51}
t_8	s_{11}
t_9	s_{52}

(c)

Task	Resource
t_1	s_2
t_2	s_2
t_3	s_2
t_4	s_4
t_5	s_4
t_6	s_{32}
t_7	s_{52}
t_8	s_{11}
t_9	s_{51}

(d)

Figure 3. Resource allocation.

process such that the processing time is no more than a deadline. Therefore, the resource allocation is considered to be optimised when the expense is low but the time constraint can be satisfied. In this example, we assume the deadline is 9.5 h. An optimised resource allocation for this scenario is shown in Figure 3(c) where the expense is $587.5 and time is 9.5 h, which just satisfies the time constraint. Compared with Figure 3(c), the allocation in Figure 3(a) is worse because it is more

DOI: 10.1002/cpe

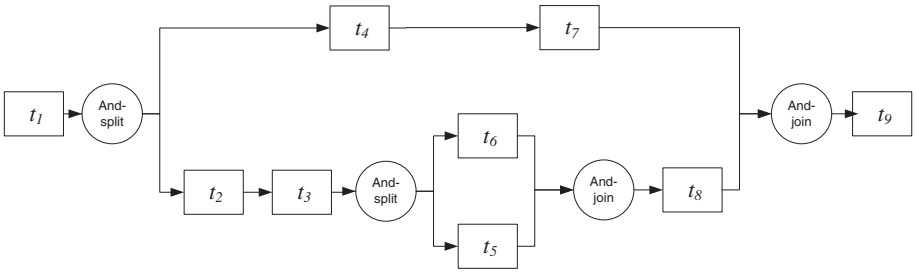


Figure 4. Changed business process structure.

expensive, even though both of them can satisfy time constraint; the allocation in Figure 3(b) is less expensive, however, it violates the time constraint and hence not usable. Therefore, to improve the performance of this business process, resources are expected to be allocated as Figure 3(c) under the time constraint.

However, sometimes the structure of business process may prevent resources from being allocated in the optimised way. For instance, if the time constraint is 13.5 h, Figure 3(b) is the optimised allocation pattern under the business process structure. However, because the limit of time is rather long, t_1 and t_2 can be carried out in sequential order rather than parallel order. In other words, the business process can be changed to a new business process as shown in Figure 4. If we choose to allocate resources as shown in Figure 3(d), we can achieve an expense of \$557.5, which is less than that in Figure 3(b), and the time is 13.5 h thereby satisfy the time constraint. Therefore, based on the time requirement and available resources, business process redesign may contribute to improve the performance of business process through enabling resource to be allocated in a more optimised way.

From this example, we know that given a set of available resources, optimised resource allocation is based on the structure of business process and the requirements on the business process. Furthermore, a business process can be improved for the purpose of optimising resource allocation. In summary, we expect that the following requirements will be met in our resource allocation scheme:

- It should take into account the structural characteristics of a business process. The structural constraints and dependencies defined in a business process must be reflected in resource allocation.
- It should guarantee resources be allocated with minimal expense within a given period.
- When necessary, a business process may be improved to better optimise resource allocation.

4. PROBLEM DEFINITION AND MODEL

In this section, a resource allocation model has been proposed with notions of resources, roles, tasks and business processes. This model is aimed to characterise the relationships among these notions, and thereby use this knowledge in resource allocation and business process improvement.

Definition 1 (Resource)
A resource s denotes an available unit for executing a task. In real cases, a resource can be a human, a machine or a computer. In this model, a resource has an attribute of role:

- *Role* indicates the group that the resource is belonged to according to its position. In this model a resource can have only one role to perform, yet a role may include multiple resources. Function $role(s)$ returns the role that resource s belongs to.

Definition 2 (Role)
A role r denotes a class of resources that own the same capability. A role has an attribute of cost.

- *Cost* denotes the monetary expense for using a resource of role r. Cost in this model is valued hourly for the role.

DOI: 10.1002/cpe

A role is an abstraction to define the classification of a set of resources according to their capabilities and positions. Given a role r and a task t, function $capable(r, t)$ returns true if r is capable of performing t, or false otherwise. A resource s that belongs to r is allocatable to execute t only when the above function $capable(r, t)$ is true.

Definition 3 (Task)
A task t is a logical unit of work that is carried out by a resource. A task has an attribute of *role*:

- *role* defines what kind of resources can perform this task. In other words, a task can be performed by resources that can match the role attribute of task. In this model, one task can be performed by multiple roles, thus its role attribute refers to a nonempty set of roles, that is, $R{:}\{r\}$.
- *time* is associated with a role r and a task t, specifying the duration required for r to execute t. We denote this by function $time(r, t)$.

Definition 4 (Business process)
A business process represents a series of linked tasks, which collectively describe the procedure how a business goal is achieved. The structure of a business process p can be modelled as an directed acyclic graph in the form of $P(T, E, G, type, v_s, v_t)$, where

(1) $T = \{t_1, t_2, \ldots, t_n\}$, $t_i \in \mathrm{T}$ $(1 \leqslant i \leqslant n)$ represents a task in the business process fragment;
(2) $G = \{g_1, g_2, \ldots, g_m\}$, $g_i \in \mathrm{G}$ $(1 \leqslant i \leqslant m)$ represents a gateway in the business process fragment;
(3) E is a set of directed edges. Each edge $e = (v_1, v_2) \in E$ corresponds to the control dependency between *vertex* v_1 and v_2, where $v_1, v_2 \in T \cup G$;
(4) For each $v \in T \cup G$, $ind(v)$ and $outd(v)$ define the number of edges, which take v as terminating and starting nodes respectively;
(5) $type$: $G \rightarrow Type$ is a mapping function, where $Type = \{$*And-Joint*, *And-Split*, *Or-Joint*, *Or-Split*$\}$. Therefore,
    If $type(g) =$ '*And-Split*' or '*Or-Split*' then $ind(g) = 1$, $outd(g) > 1$;
    If $type(g) =$ '*And-Joint*' or '*Or-Joint*' then $ind(g) > 1$, $outd(g) = 1$;
(6) v_s is the starting node of the business process fragment, which satisfies that $v_s \in T \cup G$ and $ind(v_s) = 0$;
(7) v_t is the terminating node of the business process p, which satisfies that $v_t \in T \cup G$ and $outd(v_t) = 0$;
(8) $\forall v \in N \backslash \{v_s, v_t\}$, $ind(v) = outd(v) = 1$.

Figure 5 illustrates the relationships among these definitions for resource allocation purposes. A business process consists of a set of tasks and gateways. Each resource performs as one role, and one role may represent multiple resources. Any task can be performed by a set of roles, and a role may be capable of executing multiple tasks. Cost is an attribute of a role, and it denotes the money that the enterprise has to pay for using the resources of that role. Time for completing a task is determined by the role of the resource assigned to perform this task. We assume that a single resource is allocated to execute each task, while each resource may perform multiple tasks in different time durations. When allocating a resource to a task, the allocation is subject to the role of this resource and temporal consistency between executing duration and the schedule of allocated resource.

On the basis of this model, we seek the best execution plan by adapting resource allocation optimisation and process structural improvement. Given a business process p with its task set T and deadline $t_{\max}$, a resource S, role set R, the mappings from S to R, that is, $S \rightarrow R$, and from R to T, that is, $R \rightarrow T$, the problem is to find a resource allocation mapping $T \rightarrow S$ on the improved business process p' of p, such that p' can finish in $t_{\max}$ and the total cost is minimal.

DOI: 10.1002/cpe

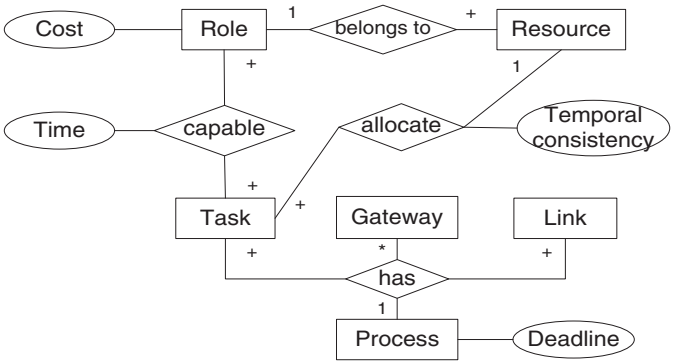


Figure 5. Resource allocation model for business processes.

5. RESOURCE ALLOCATION AND BUSINESS PROCESS IMPROVEMENT

To handle the defined problem, we first discuss a set of basic rules to specify the resource allocation criteria. Then we describe main data structures for resource allocation. The main steps of our optimisation algorithms are introduced afterwards. Finally, we present two strategies and corresponding algorithms for resource allocation and business process improvement.

5.1. Basic rules

To find a resource allocation scheme that meets the requirements on cost and time mentioned above, the searching process has to comply with the following rules:

Rule 1. One resource can only serve one task at one time. If this rule is violated, it indicates a resource allocation *conflict* at the task.
Rule 2. The overall execution time of a business process is not allowed to exceed the time limit. If this rule is violated, resources will be required to be reallocated for shortening the execution time.
Rule 3. Whenever possible, the expense for executing a business process should be minimal.

In fact, Rule 3 is our optimisation objective while Rule 1 and Rule 2 are the constraints for achieving the objective. In other words, Rule 3 is applicable under the condition that Rule 1 and Rule 2 hold.

5.2. Data structure

For describing our resource allocation algorithms, we import two specialised structures, viz., an *allocation table* and a *path table*.

1. For a business process, an *allocation table* $<$ *task* (t), *role* (r), *resource* (s), *start time* (st), *end time* (et) $>$ is used to record the allocation information for each task in the business process. When a task is allocated with a resource, the allocation table will be appended with a new record, where (1) 'Task' t denotes the name of task; (2) 'Role' r denotes the role that is selected; (3) 'Resource' s of role r is the specific resource that is assigned to execute t; (4) 'Start Time' st is the starting time of t; (5) 'End Time' et is the time that t finishes. It is computed as $et = st + time(role(s), t)$, where function $time(r, t)$ is defined in Section 3.
2. For business process p, a *path table* $<$*path* (i), *taskSet* (ts), *time* (tm)$>$ records the information of all paths from start node v_s to end node v_t of p. In the path table, '*Path*' i denotes the ID of this path. '*TaskSet*' ts records the set of tasks on path i. '*Time*' t denotes the total time to execute all the tasks in ts. Note, a task in p may appear in more than one path.

DOI: 10.1002/cpe

5.3. Resource allocation steps

Because the cost and time for task execution are unknown until it is allocated with actual resource, the analysis on business process performance inevitably involves resource allocation. According to the rules discussed in Section 5.1, resource allocation seeks the minimal cost while satisfying time constraint. Correspondingly, the allocations proceed in the following three steps:

(1) A basic allocation strategy will be applied to search for a resource allocation satisfying Rule 3 and Rule 1. This strategy aims at the minimal expense for executing a business process with balanced allocation for all paths. To achieve this, a basic allocation is conducted first to assign most economical resources without considering time conflicts. After that, we handle the existing temporal conflicts. However, after the first step, the given deadline may not be satisfied.
(2) In case that the allocation scheme in Step (1) violates Rule 2, an adjustment strategy will be applied to shorten the execution time by reallocating resources until time constraint is satisfied. However, it may not be able to reach the optimum because the process structure may not be optimal.
(3) If the total execution time is less than the deadline after from Step (1) or Step (2), according to Rule 3, an adjustment strategy will be applied to do the resource-oriented business process improvement for the purpose of lowering the cost by narrowing the gap between the total execution time and the deadline.

Step (1) will be discussed in Section 5.4, and Section 5.5 introduces how Step (2) and Step (3) are carried out. Given two factors (i.e., time and cost) to balance, our approach seeks to consider only one factor first in Step (1), and then further optimises the result by Step (2) and Step (3) for overall optimisation. In particular, we select cost as initial standard because it is nonrelevant to the whole process structure (time is relevant to the structure).

5.4. Basic allocation strategy

The goal of this basic strategy is to minimise the overall cost without considering the time limit. This strategy is achieved by two steps. First, each task is allocated with a role which results in the lowest cost, and the allocation table is updated accordingly. However, because of the characteristic of business process structure, it is possible that a role is over-allocated, that is, the resource of this role is allocated to perform more than one task at one time. In this case, Rule 1 is violated, and therefore the second step is initiated to handle allocation conflicts through reallocation. In this procedure, the resources with the next lowest cost may be selected to replace the over-allocated resources. Also, for the average efficiency, resources are allocated to symmetrise the execution times of different branches of parallel or selective process structures.

The basic resource allocation strategy is detailed in Algorithm 1. Lines 1–3 initialise several variables *ntbp* for nodes to be processed, *pd* for processed nodes and *pathT* for storing all paths of business process p. Function *genPathTable*(p) generates the path table for p with time for each path set to 0. Lines 4–17 are the loop for processing one node of the graph for p, starting from v_s to v_t. Function *getNextNode*(*ntbp*) (Line 5) finds the next node v in *ntbp* such that v cannot be processed before its predecessor nodes. For a task node v(Line 6), function *bestRole* (v) (Line 7) returns the role of minimal expense to execute v from the role set $R = \{r|capable(r, v) \text{ is true}\}$. A heuristic rule is used here: if two roles are capable to execute task v at same expense and one can only be assigned to v, then this role is selected. Function *allocRes*(r) (Line 8) assigns a resource s for r. When a resource of role r is allocated v, the one that is available to perform v is selected. Lines 9–10 calculate the maximum ending time for all paths involving v as a node.

When all the required information is ready, function *alloc*(t, r, *s, st, et*) adds a new record into the allocation table *allocT*. Line 11 resets the ending time for all paths for v. After that, for either a task node or a gateway node, the successor nodes of v will be added to *ntbp*, and v is removed from *ntbp* and added to *pd* (Lines 14–16). To this point in Algorithm 1, each task has been allocated with a resource that is the least expensive for executing the task. However, we need to check if Rule 1 is violated. If so, we have to change resources for the task at which the allocation conflict occurs. This

DOI: 10.1002/cpe

Input:	p – a business process; $roleT$ – the associated role table for p; $capaT$ – the capability table for p.
Output:	$allocT$ – the result allocation table.

```
1   ntbp = { v_s }; // initial value of nodes to be processed
2   pd = ∅; // set for processed nodes
3   pathT = genPathTable(p);
4   while (ntbp ≠ ∅)
5     v = getNextNode(ntbp);
6     if (v∈P.T) then
7       r = bestRole(v);
8       s = allocRes(r);
9       pts = paths(v); // find all paths that involve v
10      tm = max{pts[i].time}; // max time for paths in pts
11      alloc(v, r, s, tm, tm+time(r, v))→allocT;
12      for each pt in pts do pt.time += time(r, v) end for;
13    end if
14    ntbp = ntbp ∪ succ(v);
15    ntbp = ntbp \ {v};
16    pd = pd ∪ {v};
17  end while
18  call conflictProc(allocT, 0, p);
19  return allocT;
```

Algorithm 1. Basic allocation

```
function conflictProc(allocT, startTime, p)
for each v_0: first node in conflict starting after startTime
    V_c={v'|v'.st< v_0.et & v'∈V}
    while (|V_c| ≠ 0)
        select v_1 ∈V with minimal st
        t_1 : time of re-allocation on v_0;
        t_2 : time of re-allocation on v_1;
        t_3 : change process structure p on v_0 and v_1;
        if t_3≤min(t_1, t_2) then structure change on v_0, v_1;
        else if(t_2<t_1) then re-allocate on v_1;
        else re-allocate on v_0; break;
        end if
        V_c = V_c / {v_1};
    end while
end for
return allocT;
```

Algorithm 2. Resource allocation conflict resolution

is achieved by calling function *conflictProc*(*allocT, 0, p*) to check the allocation from the beginning of the business process (Line 18).

Algorithm 2 is a function for resolving resource allocation conflicts. *ConflictProc*(*allocT, t, p*) is to check conflict for tasks started after time t in the order they appear in p. Among all tasks starting after time t, we select the first (starting) node v_0 with resource conflicts, and all conflicts involving this task are handled. Among the task set V_c conflicting with v_0, the one v_1 with minimal start time is selected for treatment. In general, three approaches can be applied to resolve the conflict: reallocate a resource for v_0, reallocate a resource for v_1, or change the process structure. The processing time of the longest path in these three cases are computed respectively and compared. We conduct the process structural change when its processing time is minimal among the three cases with total cost remains unchanged. Otherwise, resource is reallocated at the task, which leads to the minimal overall processing time. The resource that increases the minimal expense but does not increase execution time is preferred. This procedure continues iteratively, until all conflicts are handled.

Consider the example introduced in Section 3. The basic allocation algorithm first comes up with an initial allocation as shown in Figure 6(a). After that, allocation conflicts are checked. Starting

DOI: 10.1002/cpe

Task	Resource
t_1	s_{11}
t_2	s_2
t_3	s_2
t_4	s_4
t_5	s_4
t_6	s_{31}
t_7	s_{52}
t_8	s_{11}
t_9	s_{51}

(a)

Task	Resource
t_1	s_{11}
t_2	s_{12}
t_3	s_2
t_4	s_4
t_5	s_4
t_6	s_{31}
t_7	s_{52}
t_8	s_{11}
t_9	s_{52}

(b)

Figure 6. Allocation conflict and handling.

from t_1, each task will be checked. When t_2 is examined, we can easily detect that task t_2 conflicts with t_3 because they use the same resource s_2 in an intervening duration. At this stage, reallocation will be applied on t_2 because it is on the longest path, and hence the overall time can be reduced and times of different paths become more balanced. This results in the change shown in Figure 6(b). Process structure is changed to Figure 4 to resolve the conflicts between tasks t_4 and t_5.

5.5. Adjustment strategy

The basic strategy does not consider time constraints. As stated in *Rule* 2, the overall execution time of a business process is not allowed to exceed the time limit. In case Rule 2 is violated, that is, the ending time of v_t in the allocation table is greater than time limit $t_{\max}$, we have to shorten the time by reallocating resources until it is within $t_{\max}$. However, if the overall execution time is less than $t_{\max}$, we may relax the time and to reduce the expense based through business process change.

First we discuss the adjustment strategy for the case that time constraint is violated. We have several heuristic rules. Because the overall time depends on the longest path of the business process, it is more effective to adjust those tasks belonging to the longest path. When task t currently assigned with resource s is reallocated with s', both the time and cost may change accordingly. Assume $\Delta time$ denotes the time reduction and $\Delta expense$ is expense increase. The value of $\Delta time/\Delta expense$, called *compensation ratio*, can be used to measure the effectiveness of an adjustment. Obviously, a higher *compensation ratio* is preferable because it implies higher time reduction with less expense. If this adjustment is on a task that belongs to the longest path, the overall time will be reduced accordingly.

Algorithm 3 is designed for handling time constraint violation. The reallocation process is carried out until time constraint is satisfied. First, the longest path *pt* is selected. In Lines 4–19, we select a task on *pt* to reallocate. *mcr*, with 0 as initial value, is used to record the maximal compensation ratio for each reallocation, and *mallocT* is the allocation table after such a reallocation. For each task v in *pt*, the role r_v of maximal compensation ratio (calculated by *maxRatioRole*(v)) for v is selected, and resource s_v of r_v is reallocated, and *allocT*′ is the allocation table to record this reallocation in Lines 5–10. If v is in *And* block, this reallocation may cause allocation conflict, hence function *conflictProc*(*allocT*′*, v.st, p*) is called to handle potential resource allocation conflicts from the starting time of v. Function *repRow*(v, r, s, $v.st$, $v.st.et$) returns an allocation table that replaces the row for v in *allocT* with the specified new row. Lines 11–14 compute the *compensation ratio cr* of this reallocation from an overall perspective. If *cr* is greater than *mcr*, *mcr* is updated to *cr* and *mallocT* is changed to *allocT*′ in Lines 16–17. After compensation ratio on all the tasks in *pt* has been computed, *allocT* is updated to *mallocT* and returned if time constraint is satisfied.

However, when the current processing time exceeds the deadline, adjustment strategy must be applied to guarantee time constraint is satisfied. Assume the longest path is computed as path l, reallocation will be carried out on tasks on path l to reduce the total time. It is easy to calculate and

DOI: 10.1002/cpe

Input:	p	– the business process;
	allocT	– the old allocation table;
	pathT	– a path table for p;
	t_{max}	– time limit.
Output:	*allocT*	– new allocation table.

```
1   while(v_t.et> t_max)
2       pt = longestPath(pathT);
3       mcr=0;
4       for each v ∈ pt.Tasks
5           r_v = maxRatioRole(v);
6           s_v = allocRes(r);
7           allocT'=repRow(v, r_v, s_v, v.st, v.st + time(r_v, v));
8           if(v is in And block)
9               call conflictProc(allocT', v.st, p)
10          end if;
11          et= allocT[v_t].et; exp=expense(allocT[v].role, v);
12          et'= adjustTime(allocT', v.st, p);
13          exp'=expense(r_v, v);  //new
14          cr=(et-et') / (exp'-exp);
15          if(cr>mcr) then
16            mcr=cr;
17            mallocT=allocT';
18          end if
19      end for
20      allocT=mallocT;
21  end while
22  return allocT;
```

Algorithm 3. Time constraint violation handling approach

compare the compensation ratio for replacing each task in this path and finds that reallocation for any task t_s among them, will achieve the maximal compensation ratio. Therefore, reallocation on t_s is applied and the new allocation is generated

Now we discuss the adjustment strategies for the case that the overall time is less than the time limit $t_{\max}$. We also have some heuristic rules to reduce cost while relaxing time. In Algorithm 1, the resource allocation conflict caused by two parallel executing tasks that require the same role with minimal cost but there was no sufficient resource for allocating them was resolved by reallocating one of them with a more expensive resource. In this scenario, we could change the process structure to support both of them to be assigned with the original cheapest resources. The reduced cost through process change is usually achieved with increased time. Therefore, it is wise to change the process structure of those tasks that do not belong to a long path, because a task in a long path has less room for time increasing.

Algorithm 4 is designed for relaxing time for maximal cost reduction. *allocT*′ and p' record the allocation table and the changed process structure, and they are initialised with *allocT* and p, respectively (Lines 1–2). Although the end time of *allocT*′ is less than $t_{\max}$ (Line 3), the process change is accepted (Line 22) and further process improvement can be made based on *allocT*′ and p'. Process change is realised as follows: In process $p$the shortest path *pt* is first selected (Line 4), and the change is focused on task in *pt*. For each task v in *pt* (Line 5), we select a role r by function *minRatioRole*(v) that looks for the maximal expense deduction with minimal time increase (Line 6). If such a role exists and it is not used by v (Line 7), function *getTasks*(r, *v.st*, *v.et*) returns the set of tasks that are within the same nearest *And* block, are overlapped with v, and are assigned resources with the same role (Line 8), then Lines 9–16 find task *mv* with the minimal time in its involved longest path. Lines 17–21 change the process structure and replace the resource.

For the example in Section 3, if the time constraint is changed from 10 h to 13.5 h, and the allocation after Algorithm 3 is the same as shown in Figure 6(b). We observe that t_3 is not using resource of the best role because of a conflict with t_2. Therefore, we examine if a process change can result in reduction of expenses. In the new process shown as Figure 3, resource allocation adjustment can be made as shown in Figure 3(d), which reduces the expense. Therefore, in this case, we adopt the changed business process structure in Figure 4 and resource allocation in Figure 3(d).

DOI: 10.1002/cpe

Input:	p – the business process; $allocT$ – the old allocation table; $pathT$ – a bath table for p; t_{max} – time limit.
Output:	$allocT$ – new allocation table; p – new process structure.

```
1   allocT'=allocT; //allocation table after re-allocation
2   p'=p;     // process structure after re-allocation
3   while(allocT'[v_t.et]< t_max)
4    pt=shortestpath(PathT);
5    for each v∈pt
6     r = minRatioRole(v);
7     if(r≠null and allocT[v].role≠r)
8      ts=getTasks(r, v.st, v.et);
9      mtm= v_t.et;
10     for each v'∈ts
11      ts'=longestPath(paths(v'));
12      if (ts'.time<mtm)
13       mv=v'; mtm=ts'.time;
14      end if
15     end for
16    end for
17    p'=changeP(p, v, mv);
18    s=allocRes(r);
19    allocT'= repRow (v, r, s, v.st, v.st+time(r, v));
20    call conflictProc(allocT', v.st, p');
21    adjustTime(allocT', v.st, p');
22    if(v_t.et< t_max) then allocT=allocT'; p=p'; end if;
23   end while
24   return allocT;
```

Algorithm 4. Relaxation approach

For each task in Algorithm 1, we pick up a resource with the minimal cost, therefore this algorithm runs in $o(n)$ time, where n stands for the number of tasks. Algorithm 2 handles the allocation conflicts. All tasks are treated one by one in the time order. Any selected task may conflict with all remaining tasks. For each of the conflicts, we calculate the reallocation on both tasks by scanning the resources, and compute the total time for structure change with $o(mn)$, where m stands for the number of resources. Therefore, the Algorithm 2 runs in $o(mn^2)$ time. If the time exceeds the deadline, we find the longest path and check the reallocate plan with the minimal compensation of each task on this path with $o(mn^3)$ in Lines 4–19, and such loop may continue n times at most. Therefore, Algorithm 3 runs in $o(mn^4)$ time. In the time relaxation algorithm, we calculate the compensation of each task of the longest path in Lines 5–16, and the complexity is$o(mn)$ time. Reallocation is applied and Algorithm 2 is called in Lines 17–22 with complexity $o(mn^2)$. The loop Lines 4–22 in Algorithm 3 may continue n times in worst cases, hence Algorithm 4 runs in $o(mn^3)$ time. Therefore, the worst time complexity for Algorithms 1–4 is $o(mn^4)$.

6. EXPERIMENTAL STUDY

We conduct an experimental study on the proposed strategy to illustrate the incorporation of process structural improvement into resource allocation and the consequent benefits for process execution performance. First, we introduce the test settings of the experiment. Second, the experimental results and test case classification are given. Then based on the test cases, we discuss how process change management contributes to the time optimization. After that, we discuss how the helpful information derived from our strategy can be used for assisting enterprise managers to conduct negotiations with customers. Lastly, we use evaluation data to compare the performance of our strategy with the traditional approach that does not consider process structural change.

DOI: 10.1002/cpe

6.1. Test settings

Our implementations are coded in Java on a personal computer with a 2 GHz CPU, 2 GB RAM, and Windows XP (Jiajie Xu, Melbourne, Australia) operating system. We generate 50 processes with the scale of tasks in scope from 10 to 30 inclusive, by machine, and assign them with different execution deadlines. In addition, 50 resources are initiated for allocation. The capability mapping, role of resources, and time required from role to task is also specified by machine.

6.2. Experimental results

Table III records the time and cost variation of those test cases whose performance is affected by process structural change. In this table, we focus on the major structure changes and list five structural types in maximal from P_1 to P_5 for each test case. Take the Case 1 for example, the best allocation on structure P_1 requires 53 h for execution and the cost is \$783. If we change the process structure to P_2, the process would be benefited with a lower cost at \$711 and compensated with longer time at 56 h. Step by step, when the process structure of this test case is changed to P_5, cost is further reduced to \$508 while time increased to 65 h. However, in some test cases whose process structure is rather simple, such as Case 6, the number of structural types may be less than five.

We observe that all test cases listed in Table III can be classified into three types according to their features of time and cost variation. Figure 7 is given to show typical cases of these three categories,

Table III. Time and cost optimisation by structural improvement.

	Structure						Structure				
	P_1	P_2	P_3	P_4	P_5		P_1	P_2	P_3	P_4	P_5
Case	Time cost					Case	Time cost				
Case 1	53	56	58	63	65	Case 17	53	55	58	63	66
	783	711	607	552	508		897	887	833	798	761
Case 2	43	47	49	51		Case 18	64	65	68		
	952	881	849	797			1078	943	842		
Case 3	29	30	32			Case 19	39	40	42	43	45
	478	456	430				633	613	578	569	545
Case 4	51	54	58	61		Case 20	33	34	36	37	
	837	805	788	758			527	507	468	402	
Case 5	62	67	71	73		Case 21	51	53	54	57	58
	1032	989	955	902			977	923	903	841	820
Case 6	39	41	44			Case 22	52	57	61		
	752	742	727				798	732	683		
Case 7	45	48	51	53		Case 23	57	60	61	64	
	633	602	589	576			810	780	762	740	
Case 8	68	70	73	75		Case 24	64	69	71	75	
	1063	976	937	902			1103	1043	987	901	
Case 9	62	66	69	72		Case 25	43	46	47	49	51
	866	840	822	805			802	772	760	745	689
Case 10	39	44	47			Case 26	33	36	38	39	
	585	542	512				639	601	582	568	
Case 11	55	56	58	62	65	Case 27	50	51	53	57	59
	562	541	527	511	498		779	772	742	709	698
Case 12	43	45	46	47		Case 28	55	60	63		
	495	466	437	403			789	701	555		
Case 13	40	43	44	46		Case 29	60	64	68	70	73
	566	547	518	505			895	842	798	779	747
Case 14	58	59	62	64	67	Case 30	36	39	40	42	44
	883	842	811	772	766		587	570	552	536	508
Case 15	53	56	57	61		Case 31	47	52			
	682	652	639	613			722	669			
Case 16	38	41	46	48		Case 32	58	62			
	433	412	378	372			976	866			

DOI: 10.1002/cpe

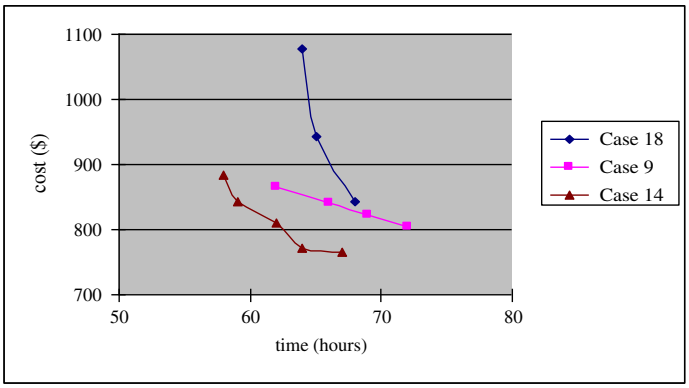


Figure 7. Comparison of three categories.

where points on a curve correspond to a structural type P_i ($1 \leqslant i \leqslant 5$). (1) Cases in the first category are skewed in time, which means cost changes are more significant than the time changes during the structural improvement. For example when the process structure of Case 18 is changed from P_1 to P_3, the execution time increases from 64 to 68 h, while the cost is significantly reduced from \$1078 to \$842, and the ratio ($\Delta cost/\Delta time$) of cost change to time change is 59. From Figure 7 we can see that the curve of Case 18 drops sharply reflecting the sharp cost decrease with slight time increase. (2) The second category includes test cases skewed in cost. That means, the cost change is in a less rate than the time change from structural change. For instance, as structure of Case 9 changes from type P_1 to P_4, time increases slightly from 62 to 72 h while cost is slightly reduced from \$866 to \$805. The value of $\Delta cost/\Delta time$ ratio is 6.1, much less than that of Case 18. The curve of Case 9 is shown in Figure 7, which is flat, in contrast to the curve of Case 18. (3) The third category is between the previous two categories: cases in this category are nonskewed in both time and cost. Take Case 14 for example, the ratio of cost change to time change is 10.7 from structural type from P_1 to P_5. However, part of the structural change of this category may be in the mode of skewed in time or cost. For example, the structural change from P_1 to P_2 is skewed in time with a $\Delta cost/\Delta time$ ratio at 41, while the change from P_4 to P_5 is skewed in cost with a $\Delta cost/\Delta time$ ratio at 2. As a comparison with the above two types, the curve of Case 14 is also shown in Figure 7.

6.3. Time optimisation

According to Figure 7, incorporating process structural improvement into process planning may contribute to reduce execution time. For example, a cost-efficient allocation scheme on the original process of Case 9 requires 72 h to complete, and it cannot meet the deadline of 65 h. However, a valid execution plan on an improved structure can be found to satisfy time constraint by our approach, with reduced execution time to 62 h and slight cost increase from \$805 to \$866. Therefore, our strategy supports the time-targeted process improvements to meet the deadlines so that enterprises are able to successfully save the business. Furthermore, for some extreme cases where deadline is still unable to be reached, our strategy contributes to shorten the gap, so that enterprises can use the new result to negotiate with the customer for deadline extension. In particular, shortening the time gap by structural change is especially worthy for the cases in the second category because of the least compensation of cost increase.

6.4. Guideline for business intelligence

Our strategy provides useful guidelines for enterprise managers to negotiate with customers. Apart from detecting the best process execution plan, the approach can also suggest possible cost reduction opportunities. For example, if the deadline of Case 18 is slightly extended from 64 to 68 h, our strategy can find a much cheaper execution plan (cost reduced from \$1078 to \$842) with an improved process structure. Such deadline extension is an ideal choice because this would deliver

DOI: 10.1002/cpe

more profits to the enterprise and lower prices to the customers. According to Figure 7, such suggestion for negotiation on cost reduction is most suitable for the cases in the first category because of high $\Delta cost/\Delta time$ ratio. In contrast, it is not worthy to the second category because of low $\Delta cost/\Delta time$ ratio. As for the third category, if needed, we may partition the curves to detect those structural changes skewed in cost for suggesting cost reduction opportunities, such as the structural change from P_1 to P_2 of Case 14. By predicting the resulted time and cost changes because of process structure improvements, our approach helps managers guide the negotiation with customers to achieve a win–win collaboration.

6.5. Comparison

Incorporating process structural improvement into resource allocation can benefit the enterprise with shorter time and less cost for business process execution. In this experiment, 42% of the test cases can reach its deadline without incorporating process change. In comparison, the successful rate rises to 66% when structural change is applied. Also, our strategy can improve resource utilisation by reducing 12% of total cost for the 50 test cases while keeping their deadlines. Furthermore, the cost of some cases can be considerably reduced by deadline extension. In the experiment, 10% of the 50 test cases can be greatly reduced by more than 12% in cost when the deadline is extended no more than 5% using our suggested negotiation for deadline extension with customers.

7. CONCLUSION AND FUTURE WORK

This paper has discussed the problem of resource allocation for business processes. An approach has been proposed to allocate resources to tasks to choose the minimal total expense under the execution time requirement. In this approach, a basic strategy has been applied first to minimise the total expense. Then, an adjustment strategy has been applied to adjust the allocation scheme to ensure the process finishes in the given time. In comparison to previous research, our approach is better in addressing the relationship between effective resource allocation and business process improvement. Particularly, we allow process structure to adapt to resource allocation requirements and available resources of an enterprise for better business process performance and resource utilisation.

In the future, we will apply our resource allocation approach in e-health applications. The availability of resources in hospitals tends to follow some constraints, for example, the timetable and workload limit for doctors and downtime of medical equipments for maintenance. Also, task dependencies in health care processes are always complex. It would be desirable to work out an efficient execution plan of complex health care processes under resource availability constraints.

ACKNOWLEDGEMENTS

The research work reported in this paper was partly supported by the Australian Research Council under Linkage Grant LP0990393 and the Chinese NSFC project under grand numbers 91124001 and 61073033.

REFERENCES

1. Du W, Eddy G, Shan M-C. Distributed resource management in workflow environments. *Proceedings of the 5th Database Systems for Advanced Applications*, Melbourne, Australia, 1997; 521–530.
2. Huang Y-N, Shan M-C. Policies in a resource manager of workflow systems: modeling, enforcement and management. In *Proceedings of the 15th International Conference on Data Engineering*. IEEE Computer Society: Sydney, Austrialia, 1999; 104.
3. R-Moreno MD, Borrajo D, Cesta A, Oddi A. Integrating planning and scheduling in workflow domains. *Expert Systems with Applications* 2006; **33**:389–406.
4. Vermeulen IB, Bohte SM, Elkhuizen SG, Lameris H, Bakker PJM, Poutré HL. Adaptive resource allocation for efficient patient scheduling. *Artificial Intelligence in Medicine* 2009; **46**(1):67–80.
5. Wu AS, Yu H, Jin S, Lin K-C, Schiavone GA. An incremental genetic algorithm approach to multiprocessor scheduling. *IEEE Transactions on Parallel and Distributed Systems* 2004; **15**:824–834.
6. Zomaya AY, Teh Y-H. Observations on using genetic algorithms for dynamic load-balancing. *IEEE Transactions on Parallel and Distributed Systems* 2001; **12**:899–911.

DOI: 10.1002/cpe

7. Yoo M. Real-time task scheduling by multiobjective genetic algorithm. *Journal of Systems and Software* 2009; **82**(4):619–628.
8. Jin F, Song S, Wu C. A simulated annealing algorithm for single machine scheduling problems with family setups. *Computers & OR (COR)* 2009; **36**(7):2133–2138.
9. Jose J, Ashikhmin AE, Whiting P, Vishwanath S. Scheduling and Precoding in Multi-User Multiple Antenna Time Division Duplex Systems, 2008. CoRR abs/0812.0621.
10. Heinonen J, Pettersson F. Hybrid ant colony optimization and visibility studies applied to a job-shop scheduling problem. *Applied Mathematics and Computation (AMC)* 2007; **187**(2):989–998.
11. Topcuoglu H, Hariri S, Wu M-Y. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transactions on Parallel and Distributed Systems* 2002; **13**(4):260–274.
12. Lee Y-J, Lee D-W, Chang D-J. Optimal task scheduling algorithm for non-preemptive processing system. In *Proceedings of the 8th Asia-Pacific Web Conference*, Vol. 3841. Springer–Verlag: Harbin, China, 2006; 905–910.
13. Hartmann S, Briskorn D. A survey of variants and extensions of the resource constrained project scheduling problem. *European Journal of Operational Research* 2009; **1**:1–14.
14. Etoundi RA, Ndjodo MF. Feature-oriented workflow modelling based on enterprise human resource planning. *Business Process Management Journal* 2006; **12**:608–621.
15. Senkul P, Toroslu IH. An architecture for workflow scheduling under resource allocation constraints. *Information System* 2004; **30**:399–422.
16. Johann E, Euthimios P, Michael R. Time constraints in workflow systems. In *Proceedings of the 11th International Conference on Advanced Information Systems Engineering*. Springer-Verlag: Heidelberg, Germany, 1999.
17. Blythe J, Jain S, Deelman E, Gil Y, Vahi K, Mandal A, Kennedy K. Task scheduling strategies for workflow-based applications in grids. *Proceedings of the 5th International Symposium on Cluster Computing and the Grid*, Cardiff, UK, 2005; 759–767.
18. Wieczorek M, Prodan R, Fahringer T. Scheduling of scientific workflows in the ASKALON grid environment. *SIGMOD Record* 2005; **34**:56–62.
19. Yu J, Buyya R. Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. *Scientific Programming* 2006; **14**:217-230.
20. Talukder A, Kirley M, Buyya R. Multiobjective differential evolution for scheduling workflow applications on global Grids. *Concurrency and Computation: Practice and Experience Vision* 2009; **21**(13):1742–1756.
21. Liu K, Chen J, Yang Y. A throughput maximization strategy for scheduling transaction-intensive workflows on SwinDeW-G. *Concurrency and Computation: Practice and Experience Vision* 2008; **20**(15):1807–1820.
22. Eckert J, Ertogrul D, Miede A, Repp N, Steinmetz R. Resource planning heuristics for service-oriented workflows. *Web Intelligence* 2008; **1**:591–597.
23. Pathak J, Basu S, Honavar V. Modeling web service composition using symbolic transition systems. *Proceedings of the AAAI Workshop on AI-Driven Technologies for Service-Oriented Computing (AAAI)*, Boston, USA, 2006; 44–51.
24. Giordano L, Martelli A. Web service composition in a temporal action logic. *Proceedings of 4th International Workshop on AI for Service Composition (AISC 2006)*, Marseille, France, 2006; 52–59.
25. Ermolayev A, Keberle N, Kononenko O, Terziyan V. Proactively composing web services as tasks by semantic web agents. In *Modern Technologies in Web Services Research*. IGI Publishing: Hershey New-York, 2006; 217–246. Modern technologies in web services research edition.

DOI: 10.1002/cpe