Optimizing a Business Process Model by Using Simulation

Farzad Kamrani
Rassul Ayani
Anvar Karimson
School of Information and Communication Technology
Royal Institute of Technology (KTH)
Kista, Stockholm, Sweden
Emails: last-name@kth.se

***Abstract*—In this paper we present the problem of optimizing a business process model with the objective of finding the most beneficial assignment of tasks to agents, without modifying the structure of the process itself. The task assignment problem for four types of processes are distinguished and algorithms for finding optimal solutions to them are presented: 1) a business process with a predetermined workflow, for which the optimal solution is conveniently found using the well-known Hungarian algorithm. 2) a Markovian process, for which we present an analytical method that reduces it to the first type. 3) a non-Markovian process, for which we employ a simulation method to obtain the optimal solution. 4) the most general case, i.e. a non-Markovian process containing critical tasks. In such processes, depending on the agents that perform critical tasks the workflow of the process may change. We introduce two algorithms for this type of processes. One that finds the optimal solution, but is feasible only when the number of critical tasks is few. The second algorithm is even applicable to large number of critical tasks but provides a near-optimal solution. In the second algorithm a hill-climbing heuristic method is combined with Hungarian algorithm and simulation to find an overall near-optimal solution for assignments of tasks to agents. The results of a series of tests that demonstrate the feasibility of the algorithms are included.**

I. Introduction

Business process modeling refers to describing business processes at a high abstraction level, by means of a formal notation to represent activities and their causal and temporal relationships as well as specific business rules that process executions have to comply with [1]. Over the years, several business modeling methodologies such as *flow chart*, *data flow diagram*, *control flow diagram*, *Gantt chart*, and *Unified Modeling Language (UML)* have been employed to model and analyze different aspects of business processes. In recent years *Business Process Modeling Notation (BPMN)* [2] has emerged as a de facto standard for modeling organizations and business processes [3]–[5].

BPMN is a graphical notation that provides organizations the capability to document, analyze and communicate their business procedures in a simple and standard manner. The graphical notation of BPMN is chosen to resemble the flow-chart format. One of the objectives in the development of BPMN has been to create a simple and understandable mechanism for creating business process models, while at the same time being able to handle the complexity inherent in business processes. For this reason the graphical notation is organized into a small set of specific notation categories. While the reader of a BPMN diagram can easily recognize the basic types of elements and understand the diagram, additional variations and information can be added within the basic elements to support the requirements for complexity without dramatically changing the basic look and feel of the diagram [2].

Even though one of the major goals of business process modeling and BPMN is to express the business process structure and provide the capability of understanding and analyzing the business procedure, it is possible to extend BPMN to cover other aspects of a process model. We suggest extending the BPMN constructs with a performance metric using characteristic of agents and the importance of these characteristics in different activities. The ultimate goal of this approach is to provide a means to estimate the quality of outcome of the process and improve this outcome without any major changes in the process model and only by reorganizing human resources within the existing organizational structure.

The task to measure performance of business processes is a challenging one and it is not possible to define a universal measure of performance, which is applicable to all different types of processes. Here, we are mainly interested in processes performed by humans resources or agents. Measuring human performance is generally more difficult than measuring performance of non-human systems because of the versatility of individual characteristics and unpredictable nature of human beings. Our point of departure is the assumption that the quality of the output of a business process is highly dependent on the degree of matching between the tasks and the capabilities of agents that perform these tasks. Individual characteristics like cognitive ability, motivation, mental models, expertise, experience, creativity and mood all have critical impacts on the performance of individuals.

In BPMN, the association of a particular action or set of

This work was supported by the FOI research project "Real-Time Simulation Supporting Effects-Based Planning", which is funded by the R&D program of the Swedish Armed Forces.

actions with a specific resource is illustrated through the use of the *Pool* and *Lane* constructs, commonly called *Swimlanes*. However, *Wohed* et al. show that the BPMN's support for the resource perspective is not sufficient [6]. To overcome these limitations an extension using UML constructs is introduced by *Siegeris* and *Grasl* and an example of process modeling for a large-scale modeling effort using BPMN is provided [7].

To the best of our knowledge there are no published studies on the evaluation of the performance of BPMN diagrams. However, a related approach has been proposed by *Magnani* and *Montesi* to evaluate the (monetary) cost of BPMN diagrams [8]. Two methods are suggested: *cost intervals*, in which the cost of a task is expressed by its lower and upper limits, and *average cost* in which, an average cost is assigned to each task together with the probability of alternative paths. In both methods costs are aggregated in an additive manner to the overall cost of the entire process.

In the same manner, we suggest extending the BPMN constructs such as *Task* and *Activity* with a performance metric and provide a model to estimate this value using characteristics of tasks and agents. In BPMN, activity is a generic term for work that company performs, which can be either atomic (task) or non-atomic (compound) [2]. Nevertheless, we use the terms task and activity interchangeably throughout this paper.

We also suggest how to aggregate values added by agents to tasks to obtain the overall performance of a business process model. Considering this metric as an objective function an optimization problem is defined and solution to the problem is provided.

The outline of the rest of this paper is as follows. Section II describes a model for estimating the performance of human agents. In Section III we briefly suggest how performance of agents should be aggregated to yield a single performance measure for the business process model and the optimization problem is formulated. In Section IV several algorithms for finding the optimal and near-optimal solutions are presented. Section V, discusses results from test scenarios. Section VI concludes the paper and discusses future work.

II. A MODEL FOR MEASURING PERFORMANCE OF HUMAN AGENTS

In a previous paper [9] we presented a model for estimating performance of human agents in a business process. In this model the performance or value added by an agent to a task is defined as a function of the capabilities of the agent and the importance of these capabilities for the task. Capabilities of an agent and corresponding importance for the task are assumed to be subjective assessments that are assigned by domain experts and decision makers to the agents and tasks. Furthermore, the influence of peripheral factors, such as working environment or access to adequate computer software and tools on the performance of agents are discussed.

The model suggests that the weighted sum of agents' contributions to tasks constitute an indicator of the quality of a business process. This is based on the assumption that a higher amount of qualified work yields an output that is more thoroughly worked out.

The performers of activities are modeled as a collection of agents. Each agent has a *role*, which defines the set of permissible positions in the organization, and several *attributes*, which define the characteristics of the agent and its behavior. Examples of agent attributes that influence the output are:

- Knowledge or expertise (competency and technical skills)
- Experience
- Cognitive ability (the ability to solve a problem)
- Creativity (the ability to find new and original problem solutions)
- Communication (the ability to express knowledge)
- Mood (the psychological mood)
- Motivation or willingness

The effect of these characteristics on the performance depends heavily on the nature of the work, e.g. creativity may have a huge impact on a problem solving task, while it is less important for work on a conveyor belt.

Considering m agents each having n attributes, the *capability* matrix $\mathcal{C} = [c_{ij}]_{m\times n}$, where c_{ij} is the attribute j of agent i is defined. In the same way, given s tasks the *weight* matrix $\mathcal{W} = [w_{jl}]_{n\times s}$, where w_{jl} is the weight of attribute j for task l is defined. The weighted sum of attributes v_{il}, where v_{il} is the value added by agent i to task l, is estimated by

$$v_{il} = \sum_{j=1}^{n} c_{ij} w_{jl}. \tag{1}$$

In this model, the numerical values of attributes c_{ij} and weights w_{jl} are (subjective) values assigned by domain experts and decision makers to individuals and tasks.

Other conditions such as *environmental factors* (e.g. high noise levels) and *technical aids* (e.g. computers) may also influence the performance of the agents.

We designate these factors by the common name *peripheral factors*, since they impact the performance indirectly by strengthening or weakening one or more specific capabilities. Defining the *peripheral factor* matrix $\mathcal{P} = [p_{kj}]_{r\times n}$, where p_{kj} is the effect of peripheral factor k on attribute j of agents, one can say that the peripheral factors modify the capability matrix, i.e. map the $\mathcal{C}$ to the *modified capability matrix* $\mathcal{G} = [g_{ij}]_{m\times n}$, where $g_{ij} = \prod_{k=1}^{r} p_{kj} c_{ij}$. If peripheral factors have no effects on the capabilities, i.e. $p_{kj} = 1, \forall p_{kj} \in \mathcal{P}$, then $\mathcal{G} = \mathcal{C}$. By incorporating the peripheral factors (1) is modified to

$$v_{il} = \sum_{j=1}^{n} (\prod_{k=1}^{r} p_{kj} c_{ij}) w_{jl}. \tag{2}$$

Expressing the above in matrix form, the *value-added* matrix $\mathcal{V} = [v_{il}]_{m\times s}$, where v_{il} is the value added by agent i to task l is calculated by

$$\mathcal{V} = \mathcal{G}\mathcal{W}. \tag{3}$$

III. Problem formulation

Matrix $\mathcal{V} = [v_{il}]_{m\times s}$, as calculated in (3), defines the values added by agents to a process for different tasks that compose the process. In a Business Process Model (BPM), each activity may be performed several times, something that may be predetermined, modeled as a random number, or may depend on how the tasks are performed (which agents have performed the tasks). Moreover, the importance of different tasks on the overall performance of the process may vary. In aggregating values added by agents to a process at least the following three factors should be considered.

1) How many times each activity is performed, we denote this number by x_l for activity l.
2) The impact factor of each activity on the overall performance of the process denoted by q_l.
3) The assignment of activities to agents. We denote this assignment scheme by matrix $\mathcal{Z} = [z_{il}]_{m\times s}$, where $z_{il} = 1$ if task l is assigned to agent i and 0 otherwise.

Hence, we define the total value added by agents to a process, denoted by $u(\mathcal{Z})$, as the sum of value added to each task ($v_{il}z_{il}$) weighted by the impact factor of the task (q_l) and number of times the task is performed (x_l). Generally number of times a task is performed, i.e. x_l is not a fix and predetermined number and it may be a random variable or depend on the assignment scheme. Assuming that the performance of a BPM is defined as above, the optimization problem can be formulated as the following.

Given the objective function

$$u(\mathcal{Z}) = \sum_{l=1}^{s}\sum_{i=1}^{m} x_l(\mathcal{Z})q_l v_{il} z_{il}, \tag{4}$$

find an assignment matrix $\mathcal{Z}$, i.e. all $z_{il} \in \{0, 1\}$, such that $u(\mathcal{Z})$ is maximized, subject to constraints

$$\sum_{i=1}^{m} z_{il} = 1,\ for\ all\ l \in \{1, \dots s\} \tag{5}$$

$$\sum_{l=1}^{s} z_{il} = 1,\ for\ all\ i \in \{1, \dots m\}. \tag{6}$$

Constraints 5 and 6 mean that each task is assigned to one agent and each agent performs one task, implying that numbers of tasks and agents are equal. Equality of the number of tasks and agents does not imply any loss of generality, since these numbers can always be balanced by adding dummy tasks or agents, which do not add any value to the process.

IV. Proposed solution

In its most general case the objective function defined in (4) incorporates a random element $x_l(\mathcal{Z})$ and there is no easy way to find the optimal assignment. However, depending on the type of the process, the problem may be reduced to simpler problems which require less efforts to be solved. In the following we discuss first less complicated cases and their solutions, and finally introduce an algorithm for more general cases.

A. Deterministic processes

When no randomness is involved in the process and the value of x_l for all l is predetermined and independent of $\mathcal{Z}$, the problem is reduced to a standard assignment problem. Equation (4) can be rewritten as $u = \sum_{l=1}^{s}\sum_{i=1}^{m} v'_{il}z_{il}$, where $v'_{il} = x_l q_l v_{il}$.

With constraints 5 and 6, number of feasible solutions to the task-assignment problem grows factorially in the number of tasks (agents), which makes an exhaustive search method practically impossible. However, the well-known Hungarian algorithm solves the task-assignment problem in polynomial time of the number of tasks. The Hungarian algorithm, also known as Kuhn-Munkres algorithm, was originally developed and published by *Harold Kuhn* [10] and had a fundamental influence on combinatorial optimization [11]. Since we employ this method as a part of our algorithms presented later, we outline the key steps of the method here.

The matrix interpretation of Hungarian algorithm assumes a non-negative matrix $\mathcal{C} = [cost_{il}]_{m\times m}$, where the element $cost_{il}$ represents the cost of assigning task l to agent i, and finds a minimum cost assignment of the tasks to the agents. The algorithm consists of the following steps [12], [13].

1) Subtract the minimum value of each row from every element of that row.
2) Subtract the minimum value of each column from every element in that column.
3) Use as few lines as possible to cover all zeroes in the matrix.
4) Add the minimum uncovered element to every covered element. If an element is covered twice, add the minimum element to it twice.
5) Subtract the minimum element from every element in the matrix.
6) Cover the zero elements with lines again. If the number of lines is not equal to the number of rows, return to step 4.
7) Select a matching by choosing a set of zeroes so that each row or column has only one selected.

Informally stated, the algorithm attempts to build a feasible solution by identifying agents with minimal cost for each task and tasks that are performed by minimal cost by each agent. If these assignments are infeasible the algorithm will select the second best mapping and tries to achieve a feasible solution, while keeping the cost to a minimum. A solution is feasible when every task has a unique agent assigned to it.

To find the maximum gain of assigning tasks to agents one can subtract all elements of the matrix $\mathcal{V}'$ from a value which is greater than all of them and apply the Hungarian algorithm to find the assignment scheme that gives the minimum cost for the obtained matrix. This solution gives the maximum gain for the original matrix.

B. Markovian processes

In many business processes the workflow is not predetermined and may take different paths, depending on what

happens during the performance of activities. In such conditions the value of the objective function u as defined by (4) is a random variable and a natural approach is to find an assignment scheme that maximizes the expected value $E[u]$. If the uncertainty in the business process is modeled by fixed probabilities assigned to alternative paths, then the business process model can be considered as a *Markov chain*. BPMN's *Activities* will constitute transient states and *End Events* will be absorbing states of the Markov chain. Using (4) the expected value of the objective function is calculated by

$$E([u]) = \sum_{l=1}^{s} \sum_{i=1}^{m} E[x_l] q_l v_{il} z_{il}. \tag{7}$$

It can be shown that $E[x_l]$ is the first row of the invertible matrix $(\mathcal{I} - \mathcal{Q})^{-1}$. Here $\mathcal{I}$ is the identity matrix and $\mathcal{Q}$ is obtained from the transition matrix of the Markov chain if we strike off all rows and columns containing absorbing states [9]. In other words the expected value of the number of times each task is performed can be calculated by

$$E[\mathcal{X}] = (1, 0, 0, \dots)(\mathcal{I} - \mathcal{Q})^{-1}, \tag{8}$$

where $\mathcal{X} = [x_l]_{1\times s} = [x_1, x_2, \dots, x_s]$ is the number of times each task is performed. Using (8) one can rewrite (7) as

$$E[u] = \sum_{l=1}^{s} \sum_{i=1}^{m} v'_{il} z_{il}, \tag{9}$$

where $v'_{il} = E[x_l] q_l v_{il}$. The optimal solution to the obtained objective function can be found by Hungarian algorithm. These steps are summarized in Algorithm 1.

Algorithm 1: *Optimal Assignment of a Markovian Process*

given: *agents A, tasks T, Business Process Model BPM*
return: *optimal assignment $\mathcal{Z}$*

1 $\mathcal{V} \leftarrow \mathcal{GW}$ *(Equation 3)*
2 *derive Markov chain M from BPM*
3 $E[\mathcal{X}] \leftarrow (1, 0, 0, \dots)(\mathcal{I} - \mathcal{Q})^{-1}$ *(Equation 8)*
4 *for each task $t_l \in T$*
5 *for each agent $a_i \in A$*
6 $v'_{il} \leftarrow E[x_l] q_l v_{il}$
7 *end for*
8 *end for*
9 $\mathcal{Z} \leftarrow$ *Hungarian algorithm*$(\sum_{l=1}^{s} \sum_{i=1}^{m} v'_{il} z_{il})$

C. Non-Markovian processes

Although the presented analytical method in Section IV-B is appealing, it can only be used for a Markovian process. In modeling a real-world business process, the Markovian constraint is usually too restrictive and not justified. The reason is that the probability that the workflow takes a path generally depends not only on the current state of the process but also on the history of the workflow. For non-Markovian processes there is no analytical solution and alternative approaches such as simulation must be considered.

The simulation method follows the intuitive structure of the BPMN diagram. Tokens are generated at *Start Events* and are propagated along *Sequence Flows*, across *Tasks*, *Activities*, and *Gateways*, being duplicated and merged when necessary, until they are consumed by an *End Event*. An element holding a token is considered to be active, which may result in updating the values of some parameters. The simulation is repeated a sufficient number of times so that the average values of the desired parameters are calculated. Details of the simulation are explained in a Master thesis [14] by one of the co-authors.

The aim of the simulation is to estimate $E[\mathcal{X}]$, i.e. to determine the average value of number of times each task is performed. Once $E[\mathcal{X}]$ is estimated, the objective function can be calculated as in (9) and the Hungarian algorithm can be employed to efficiently find the optimal solution. The approach is shown in Algorithm 2.

Algorithm 2: *Optimal Assignment of a Non-Markovian Process*

given: *agents A, tasks T, Business Process Model BPM*
return: *optimal assignment $\mathcal{Z}$*

1 $\mathcal{V} \leftarrow \mathcal{GW}$ *(Equation 3)*
2 *run simulation of the BPM*
3 $E[\mathcal{X}] \leftarrow$ *average token through each task*
4 *for each task $t_l \in T$*
5 *for each agent $a_i \in A$*
6 $v'_{il} \leftarrow E[x_l] q_l v_{il}$
7 *end for*
8 *end for*
9 $\mathcal{Z} \leftarrow$ *Hungarian algorithm*$(\sum_{l=1}^{s} \sum_{i=1}^{m} v'_{il} z_{il})$

D. Assignment depended Processes

In all three cases discussed above, i.e. deterministic, Markovian, and non-Markovian processes we have assumed that there is no correlation between the assignment scheme and the path of the workflow or on the probabilities of branches at decision points. However, in many situations this is not the case. For instance, assignment of a task to a less qualified agent may increase the probability that the task is repeated several times. In such scenarios some assignment combinations may change the probabilities that govern the path of the workflow. This means that x_l in (4) is a function of the assignment matrix $\mathcal{Z}$. The algorithms as presented above do not result in a unique value-added matrix anymore and are inadequate.

For this problem, we present two solutions shown in Algorithms 3 and 4. The first one finds the optimal assignment but is only feasible for scenarios where the number of tasks that affect probabilities of the workflow is small. The second algorithm finds near optimal solutions in the general case.

Both these algorithms are presented for the general non-Markovian process, which require the simulation of the business process model. Due to lack of space corresponding algorithms for deterministic and Markovian processes are not included here. They can be considered as special cases of the presented algorithms. However, it is possible to modify algorithms in Sections IV-A and IV-B to obtain simplified versions of Algorithms 3 and 4.

Algorithm 3: *Optimal Assignment of a Depended Process*

given: *agents A, critical tasks T_c, non-critical tasks T_{nc}, Business Process Model BPM*
return: *optimal assignment $\mathcal{Z}$*

1 $\mathcal{V} \leftarrow \mathcal{GW}$ *(Equation 3)*
2 $\tau \leftarrow$ *number of tasks in* T_c
3 $\Omega \leftarrow$ *all possible permutations of* τ *agents* $\in A$
4 *for each* $\omega \in \Omega$
5 $max\_gain \leftarrow 0$
6 $\mathcal{Z}_c \leftarrow$*assign tasks in* T_c *to agents in* ω
7 *run simulation of the BPM(*$\mathcal{Z}_c$*)*
8 $E[\mathcal{X}] \leftarrow$ *average token through each task*
9 *for each task* $t_l \in T$
10 *for each agent* $a_i \in A$
11 $v'_{il} \leftarrow E[x_l]q_l v_{il}$
12 *end for*
13 *end for*
14 $\mathcal{Z}_{nc} \leftarrow$ *Hungarian (*T_{nc}*, non-busy agents)*
15 $gain \leftarrow u(\mathcal{Z}_c \cup \mathcal{Z}_{nc})$
16 *if* $gain > max\_gain$
17 $max\_gain \leftarrow gain$
18 $\mathcal{Z} \leftarrow \mathcal{Z}_c \cup \mathcal{Z}_{nc}$
19 *end if*
20 *end for*

In Algorithm 3 the steps of finding the optimal solution for the case that assignment of just a few tasks affect the workflow are shown. From now on, we call such tasks critical tasks, and denote the set of these tasks by T_c. If the set of all other non-critical tasks is denoted by T_{nc}, we have $T_c \cup T_{nc} = T$ and $T_c \cap T_{nc} = \emptyset$. The basic idea of the algorithm is that by partitioning the tasks in two parts and employing the Hungarian algorithm to find the optimal assignment for non-critical tasks, the run-time complexity of the algorithm is kept relatively low. The complexity of the algorithm is $O((\tau_c + \tau_{nc})!\tau_{nc}^4/\tau_{nc}!)$, where τ_{nc} and τ_c are the cardinalities of T_{nc} and T_c, respectively. Clearly, this algorithm is feasible only for small values of τ_c.

A polynomial algorithm that finds near-optimal solutions is presented in Algorithm 4. Like the one presented in Algorithm 3, this algorithm relies on partitioning tasks in two sets of critical tasks (T_c) and non-critical tasks (T_{nc}). The optimal solution for non-critical tasks is found by the Hungarian algorithm, while a heuristic optimization technique is employed to find a near-optimal solution for critical tasks. The heuristic method, which we use and is shown in Algorithm 4 is quite similar to *hill-climbing*. The critical tasks T_c are assigned to a random feasible set of agents A_r in line 2 of the algorithm. This solution is improved in the main while-loop of the algorithm between lines $5 - 31$. The loop is stopped if it reaches a local maximum and the result does not improve anymore or the loop has iterated a sufficient number of times and the *stop_condition* has been set to true. In each iteration of the outer for-loop between lines $7 - 30$ an assignment of a critical task to an agent is removed and in the inner for-loop between lines $10 - 28$ this task is assigned to all non-busy agents. This assignment is used to run a simulation of the BPM in line 13 to estimate the average number of times each task is performed (line 14). Having the expectation of these numbers a standard task-assignment problem is formulated in lines $15 - 19$ and Hungarian algorithm is used to find the optimal solution for the non-critical tasks and so far non-busy agents (line 20). The gain is calculated for the current assignments for critical and non-critical tasks (line 21). The best assignment for the task is distinguished in lines $22 - 27$ and added to the assignments of critical tasks (line 29).

Algorithm 4: *Near-Optimal Assignment of a Depended Process*

given: *agents A, critical tasks T_c, non-critical tasks T_{nc}, Business Process Model BPM*
return: *near-optimal assignment $\mathcal{Z}$*

1 $\mathcal{V} \leftarrow \mathcal{GW}$ *(Equation 3)*
2 $\mathcal{Z}_c \leftarrow$ *assign* T_c *to a random feasible set of agents* A_b *and mark all agents* $a_b \in A_b$ *as busy*
3 $max\_gain \leftarrow 0$
4 $progressing \leftarrow true$
5 *while progressing and not stop_condition do*
6 $progressing \leftarrow false$
7 *for each task* $t_c \in T_c$
8 *release agent* a_b *performing task* t_c
9 $\mathcal{Z}_c \leftarrow \mathcal{Z}_c \setminus \{(t_c, a_b)\}$
10 *for each agent* $a_f \in A$ *which is free*
11 *assign* t_c *to* a_f *and set* a_f *as busy*
12 $\mathcal{Z}_{tmp} \leftarrow \mathcal{Z}_c \cup \{(t_c, a_f)\}$
13 *run simulation of the BPM(*$\mathcal{Z}_{tmp}$*)*
14 $E[\mathcal{X}] \leftarrow$ *average token through each task*
15 *for each task* $t_l \in T$
16 *for each agent* $a_i \in A$
17 $v'_{il} \leftarrow E[x_l]q_l v_{il}$
18 *end for*
19 *end for*
20 $\mathcal{Z}_{nc} \leftarrow$ *Hungarian (*T_{nc}*, non-busy agents)*
21 $gain \leftarrow u(\mathcal{Z}_{tmp} \cup \mathcal{Z}_{nc})$
22 *if* $gain > max\_gain$
23 $max\_gain \leftarrow gain$
24 $z \leftarrow (t_c, a_f)$
25 $\mathcal{Z} \leftarrow \mathcal{Z}_{tmp} \cup \mathcal{Z}_{nc}$
26 $progressing \leftarrow true$
27 *end if*
28 *end for*
29 $\mathcal{Z}_c \leftarrow \mathcal{Z}_{tmp} \cup z$
30 *end for*
31 *end while*

Our tests of the algorithm indicate that regardless of the choice of the initial random assignment, the quality of the near-optimal solutions are generally very high when $T_c \leq T_{nc}$, i.e. they have a value above 90% of the value of the optimal solution. However, having a smaller number of critical tasks improves the performance of the algorithm. This is due to the fact that the Hungarian algorithm always provides the optimal solution for non-critical tasks. Algorithm 4 can easily be modified to a *random restart hill-climbing* algorithm in order to improve the result.

V. Implementation and experiment results

In order to test the algorithms suggested above we have developed an application program in $C\#$, which takes advantage of calling models developed in the simulation software Arena [15]. We omit the details here and refer the reader

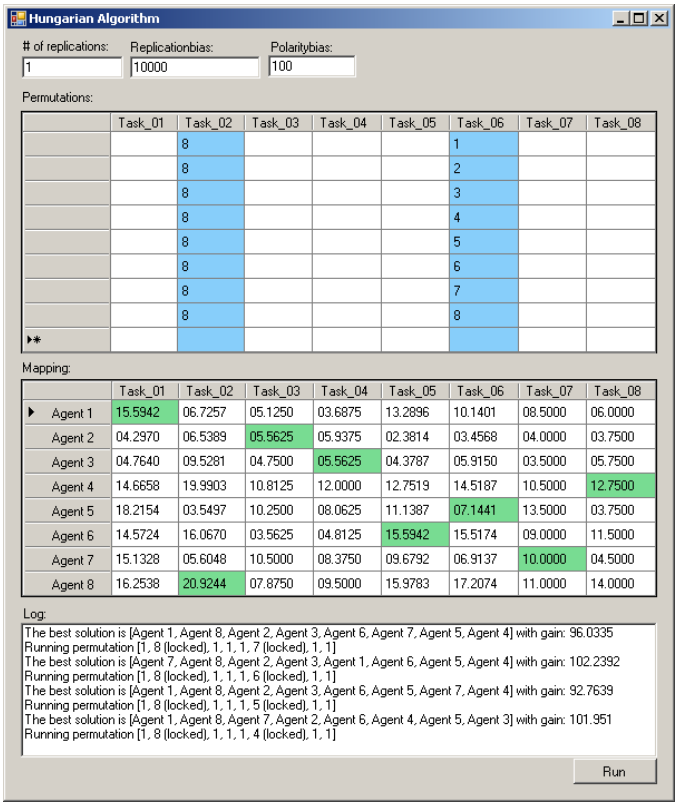

Fig. 1. The application program running for 8 tasks and agents.

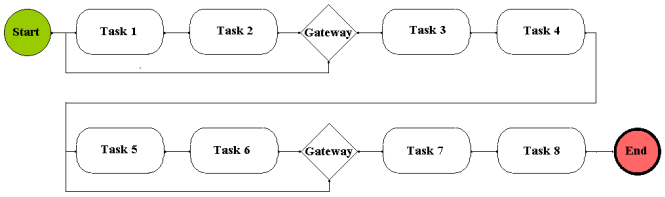

Fig. 2. A process with 8 tasks. Gateways after tasks 2 and 6 may change the workflow.

to the Master thesis by one of the co-authors [14]. The aim of this application originally has been to provide a tool that combines the strengths of simulation for constructing the relation between agents and tasks, and the efficiency of Hungarian algorithm to find out the optimal value. However, with little modifications it has been used to test the above algorithms. A screen shot of the application searching for the optimal solution for a BPM consisting of 8 tasks is shown in Fig. 1. The BPM developed in Arena is also shown in Fig. 2. One of the features of the application developed in [14] is a BPMN template developed for Arena, which provides a convenient tool for modeling business processes. To run the optimization problem one should first develop the BPM in Arena and then start the application. The application reads required data from the Arena model and runs the Arena simulation when necessary.

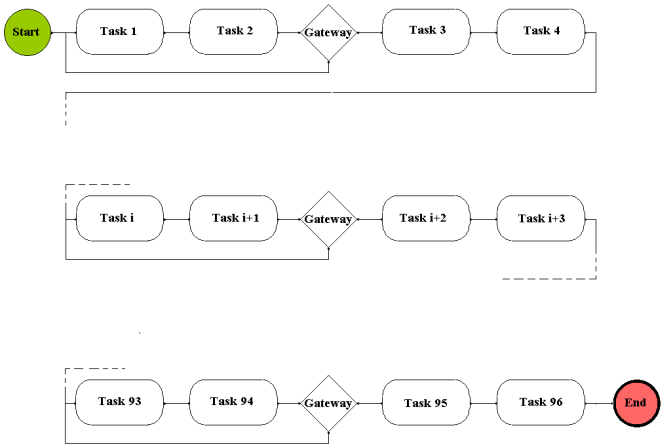

Fig. 3. A process with 96 tasks. Gateways after tasks 4i+2; i = 0, 1, ..., 23 may change the workflow.

A. *Test of Algorithm 2*

To test the performance of Algorithm 2 a series of experiments with different problem sizes is conducted. In all these simulations number of tasks is a multiple of 4, starting with 8 and up to 96 tasks. The workflow goes through all tasks sequentially, however, Gateways after tasks $4i+2$, $i = 0, 1, \ldots$ may change the workflow and send the token upstream to task $4i+1$. The probability of this event is chosen to be a decreasing function of the number of times the token has already passed tasks $4i + 1$ and $4i + 2$. This assumption means that the process is not Markovian and has some kind of memory, i.e. each failure to perform tasks $4i + 1$ and $4i + 2$ increases the probability of succeeding in future trials.

Fig. 2 and Fig. 3 show the simulation models for 8 and 96 tasks respectively. All tasks are not shown in Fig. 3, due to the lack of space. The values for agent attributes are randomly chosen from a uniform distribution between 0 and 4. The values for task attributes are chosen randomly from a uniform distribution between 0 and 4 such that the constraint for the sum of activity attributes is satisfied. The results of the test for Algorithm 2 are summarized in TABLE I. For each problem size, the gain for the optimal solution and minimum gain are presented. The number of feasible solutions for each size is also given. Execution time for the largest problem size with 96 tasks on a modest computer (Celeron 1.73 GHz processor and 2GB RAM) is under 100 seconds.

B. *Test of Algorithm 3*

For test of Algorithm 3, we use models of the type shown in Fig. 3. However we assume that task numbers $4i + 2, i = 0, 1, \ldots$ are critical, i.e. depending on the agent performing these tasks the probability of alternative paths in Gateways may change. The largest problem that can be solved in a reasonable time by Algorithm 3 is one with 16 tasks. The algorithm runs $16!/12! = 43680$ simulations and the

TABLE I
RESULTS FOR ALGORITHM 2

Size	Min Gain	Max Gain	Nr. of Combinations
8	46.80	103.71	4.03×10^4
16	93.70	213.53	2.09×10^{13}
24	124.34	323.13	6.20×10^{23}
32	162.25	435.38	2.63×10^{35}
40	203.30	554.92	8.16×10^{47}
48	249.52	655.53	1.24×10^{61}
56	298.37	775.47	7.11×10^{74}
64	327.25	882.18	1.27×10^{89}
72	393.78	1016.79	6.12×10^{103}
80	451.29	1137.24	7.16×10^{118}
88	506.57	1256.00	1.85×10^{134}
96	556.60	1385.57	9.92×10^{149}

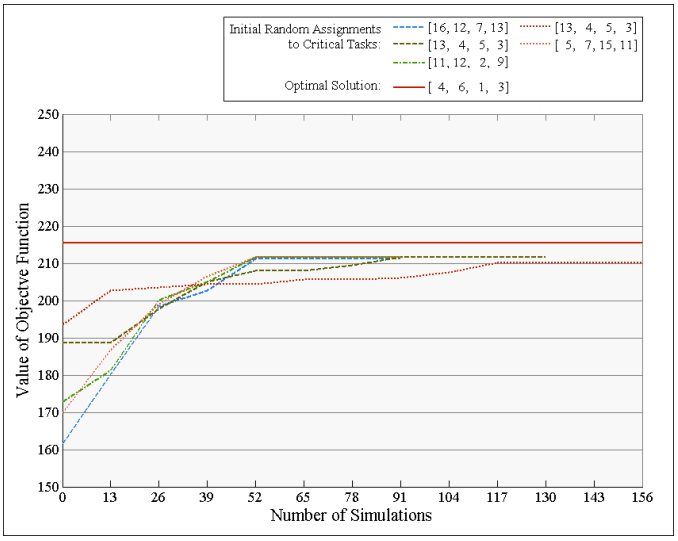

Fig. 4. Result of Algorithm 4 on a model having 16 tasks showing convergence of five initial random solutions toward the optimal one.

same number of instances of the Hungarian algorithms, which can be compared with the number of required simulations in an exhaustive search (2.09×10^{13}). An experiment on a modest computer (Celeron 1.73 GHz processor and 2GB RAM), which was not especially dedicated to this task was completed in about one week and the value 215.37 for the optimal assignment was obtained.

C. Test of Algorithm 4

The above result (215.37), obtained for 4 critical and 12 non-critical tasks is compared with the results of Algorithm 4 for the same problem. As shown in Fig. 4 all five initial random solutions after a small (different) number of simulations reach a local maximum, all near the optimal value 215.37. The rather high values of the initial random solutions are due to the fact that they are combined with the results of the Hungarian algorithm for 12 non-critical tasks.

To evaluate the efficiency of Algorithm 4 on larger models a series of tests on a model with 48 tasks is conducted and

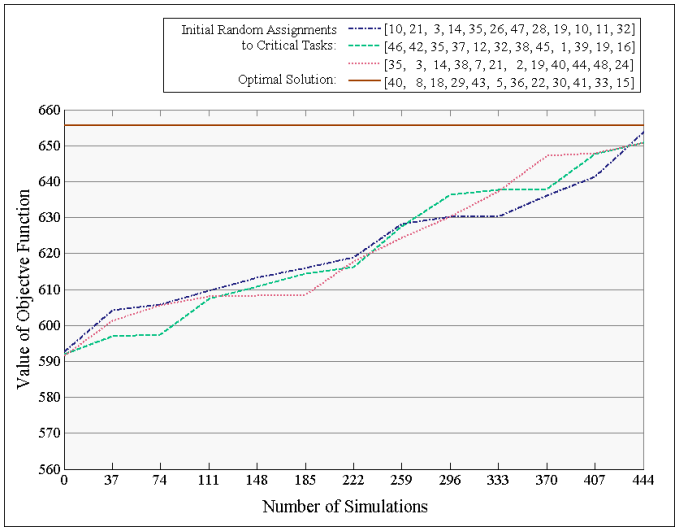

Fig. 5. Result of Algorithm 4 on a model having 48 tasks showing convergence of three initial random solutions toward the optimal one.

the results are compared with the optimal solution. Obviously, it is not feasible to find the optimal solution by Algorithm 3 for such a large model. Therefor, the model is deliberately chosen such that it is possible to find the optimal solution by Algorithm 2, i.e. it has no "real" critical tasks. The model has the same configuration as the models in Fig. 3, i.e. the number of "critical" tasks is 12 and the total number of tasks is 48. As shown in Fig. 5 the three initial random solutions evolve rather rapidly towards the value of the optimal solution. The figure includes only the results for one execution of the main loop of the algorithm. However, the results do not reach a local maximum in this stage and continue to improve.

VI. CONCLUSION AND FUTURE WORK

In this paper we employed a model of human agents' performance in a business process to estimate an overall measure for performance of a BPM. This model is based on the agents capabilities, the weight (importance) of these capabilities for each task, and the effect of peripheral factors on these capabilities. This performance metric is used to find the optimal assignment of tasks to agents. Four types of processes are distinguished.

1) Deterministic process with a predetermined workflow. The optimal solution for this type of processes is found by using the Hungarian algorithm in polynomial time.
2) Markovian process. We presented an analytical method to calculate the number of times each task is performed and reduced the problem to type one, which can be solved using the Hungarian algorithm.
3) non-Markovian process for which we use a simulation method to estimate the expected values of number of times each task is performed. These values are used to find the optimal solution.

4) non-Markovian process for which the probabilities of alternative paths of the workflow are a function of assignment of critical tasks to agents. Critical tasks are those tasks that may affect the workflow. We introduced two algorithms for this type of processes. The first one finds the optimal solution, but is feasible only when the number of critical tasks is few. The second algorithm that is even applicable to rather large number of critical tasks, provides a near-optimal solution. In this algorithm a hill-climbing heuristic method is combined with Hungarian algorithm and simulation to find an overall near-optimal solution for assignments of tasks to agents. The Hungarian algorithm always finds the optimal assignment for non-critical tasks and the heuristic method tries to find near-optimal assignments for critical tasks. Both these methods employ simulation in order to deal with the uncertainty in the system.

A series of tests that demonstrate the feasibility of the discussed algorithms is conducted and the results are included. These tests confirm that the algorithms are well-performing for at least medium sized assignment problems.

One of the shortcoming of the method introduced in this paper is the model of performance of human agents, which does not take into account different aspects of team working and interaction between agents. Incorporating various aspects of team working in the model is our next step of the work, which we have already initiated.

ACKNOWLEDGMENT

This paper is part of a project funded by the R&D program of the Swedish Armed Forces and conducted by the Swedish Defence Research Agency (FOI). The authors would like to thank all members of the project group, especially Johan Schubert and Farshad Moradi for their comments, suggestions and constructive critiques.

REFERENCES

[1] W. van der Aalst, A. H. M. T. Hofstede, and M. Weske, "Business process management: A survey," in *Proceedings of the 1st International Conference on Business Process Management, volume 2678 of LNCS*. Springer-Verlag, 2003, pp. 1–12.

[2] Object Management Group (OMG), "Business process modeling notation (BPMN) version 1.2," 2009, available via http://www.bpmn.org [accessed January 11, 2010].

[3] T. Takemura, "Formal semantics and verification of BPMN transaction and compensation," *Asia-Pacific Conference on Services Computing. 2006 IEEE*, vol. 0, pp. 284–290, 2008.

[4] O. Nicolae, M. Cosulschi, A. Giurca, and G. Wagner, "Towards a BPMN semantics using UML models," in *Business Process Management Workshops*, 2008, pp. 585–596.

[5] S. Smirnov, "Structural aspects of business process diagram abstraction," in *CEC '09: Proceedings of the 2009 IEEE Conference on Commerce and Enterprise Computing*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 375–382.

[6] P. Wohed, W. M. P. van der Aalst, M. Dumas, A. H. M. ter Hofstede, and N. Russell, "On the suitability of BPMN for business process modelling," in *Business Process Management*, ser. LNCS, S. Dustdar, J. L. Fiadeiro, and A. P. Sheth, Eds., vol. 4102. Springer, 2006, pp. 161–176.

[7] J. Siegeris and O. Grasl, "Model driven business transformation – an experience report," in *BPM*, ser. LNCS, M. Dumas, M. Reichert, and M.-C. Shan, Eds., vol. 5240. Springer, 2008, pp. 36–50.

[8] M. Magnani and D. Montesi, "Computing the cost of BPMN diagrams," Department of Computer Science, University of Bologna, Mura Anteo Zamboni 7, 40127 Bologna, Italy, Tech. Rep., May 2007.

[9] F. Kamrani, R. Ayani, F. Moradi, and G. Holm, "Estimating performance of a business process model," in *Proceedings of the 2009 Winter Simulation Conference*, M. D. Rossetti, B. Hill, R. R.and Johansson, A. Dunkin, and R. G. Ingalls, Eds., Austin, TX, December 2009.

[10] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, pp. 83–97, 1955.

[11] A. Frank, "On Kuhns Hungarian method – a tribute from Hungary," *Naval Research Logistics,*, vol. 52, pp. 2–5, 2005.

[12] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, Mars 1957.

[13] M. Sasieni, A. Yaspan, and L. Friedman, *Operations Research – Methods and Problems*. New York: John Wiley, 1959.

[14] A. Karimson, "Optimizing business processes with Arena simulation," Master's thesis, Royal Institute of Technology (KTH), Stockholm, Sweden, December 2009.

[15] W. D. Kelton, R. P. Sadowski, and N. B. Swets, *Simulation with Arena*, 5th ed. McGraw-Hill, 2009.