International Journal of Energy, Information and Communications
Vol. 2, Issue 2, May 2011

A Belief-Based Multi-Agent Markov Decision Process for Staff Management

Philip H.P. Nguyen[1], Minh-Quang Nguyen[2], and Ken Kaneiwa[3]
[1] *Justice Technology Services, Department of Justice, Government of South Australia, 30 Wakefield St, Adelaide, SA 5000, Australia*
[2] *University of Quebec at Montreal, 405, rue Sainte-Catherine Est, Montréal (Québec) H2L 2C4, Canada*
[3] *Department of Electrical Engineering and Computer Science, Iwate University, 4-3-5 Ueda, Morioka, Iwate 020-8551, Japan*
philip.nguyen@sa.gov.au, nguyen.minh-quang@uqam.ca, kaneiwa@cis.iwate-u.ac.jp

Abstract

This paper presents a system designed for task allocation, staff management and decision support in a large enterprise, in which permanent staff and contractors work alongside under the overall management of a manager to handle tasks initiated by end-users. The process of allocating a new task to a worker is modeled under different situations, taking into account user requirements as well as the different goals of management, permanent staff and contractors. Their actions and strategies are formalized as autonomous decision-support sub-systems inside a multi-agent system, based on Contract Net Protocol, belief theory, multi-objective optimization theory and Markov Decision Process.

Keywords: *Task allocation, Multi-agent system, Belief theory, Multi-objective optimization, Markov decision process.*

1. Introduction

A multi-agent system (MAS) is a computational system consisting of multiple interacting software agents, each of which models the actions of an individual in the real world in the pursuit of his or her goals. The most important feature of a MAS is that it could be self-organized and dynamically adapt its behaviors to respond to new situations. MAS's could therefore be used for problem solving in dynamically changing environments and play an important role in Artificial Intelligence science [7]. This paper describes the design of a MAS that realistically models the processes involved in task or project allocation and staff management in a large enterprise, in which permanent staff and contractors work alongside to handle tasks and/or projects initiated by end-users. The word *task* will be used to designate a task or project in such a context, the word *worker*, a permanent staff member or a contractor, and the word *manager* or *management*, the enterprise management of those tasks and workers. In essence, the manager's objectives are to meet the task requirements as expressed by end-users on one hand and to efficiently manage its permanent staff and contractors on the other. A *task reward* is a measure of recompense credited to the worker who completes the task, and is proportional to the degree of difficulty of the task and the extent of effort required to complete it. It includes a bonus or a penalty depending on how well the task deadline is met. This is an attempt to take into account the idea of de-commitment penalties and financial option pricing [16][21][22]. Performance of a worker is then

assessed through the sum of all task rewards collected by that worker during the period considered (such as a financial year).

Dynamic MAS's have been proposed, in which teams of workers could be dynamically formed and dissolved in order to maximize their task rewards [2]. That approach is more relevant in environments in which all workers are contractors, who work independently and are free to self-organize and self-manage. This paper examines a different but more common environment in which permanent staff works alongside contractors under different management policies. Contract Net Protocol (CNP) is used to model the interactions between those various people [15][16]. In the original CNP theory, an agent is either a manager or a contractor, and all tasks are available for bidding by all contractors [5][19]. This paper adds the modeling of permanent staff and management policies concerning those staff. In a nutshell, management prefers permanent staff over contractors in task allocation as permanent staff receive fixed salary while contractors are paid on hourly rate and only for the tasks they perform. Management is also attentive to the need of permanent staff for professional improvement in compliance with common human resource policies in most enterprises. This reflects best management practice that minimizes the overall cost to the enterprise while providing opportunities for skill development to permanent staff in need. The latter is similar to the principle of "social justice" [14], which advocates more assistance to the under-privileged (or under-skilled in this context). From the workers' viewpoint, both contractors and permanent staff are *self-interested* agents whose main goal is to maximize the rewards associated with the tasks assigned [12][18]. Permanent staff also aim to meet a performance target for the period considered, usually to be in line with the staff's job specification. These decision support strategies are formalized as an application of the belief theory and the multi-objective optimization theory [8]. The contractor strategy is also further modeled as a Markov Decision Process (MDP) with reinforcement learning [13]. By incorporating different management criteria in task allocation, one of the shortcomings of CNP is addressed, that is, its inability to detect and resolve conflicts during task allocation [22].

The main contribution of the paper is to present a novel conceptual design of a multi-agent system for staff management with the view of ensuring that the theoretical and mathematical foundation of the system is valid before any implementation. One of the applications of the theory presented in this paper is to enable consultancy service providers to plan for staff allocation in future customer projects, during which the service provider management is often faced with the dilemma of whether to accept a new job offer from a customer, knowing that better offers may emerge at a later time from that and/or other customers. No research similar to that presented in this paper has been found in existing scientific literature, especially in the formulation of strategies for staff management with belief theory and their resolution using MDP with reinforcement learning and Linear Programming (LP)-like techniques.

This paper is organized as follows: Section 2 recalls the definition of an MDP and its main features that are relevant to this paper, Section 3 introduces the basic objects of the proposed multi-agent system, including belief-based expected rewards for completed tasks, and manager's and worker's strategies with regard to task bidding and allocation, Section 4 formalizes the worker's strategy as a multi-objective optimization problem as well as an MDP, together with an example showing how the mathematical

problem associated with the optimum strategy is resolved, and finally Section 5 concludes the paper together with some directions for future research.

2. Markov decision process

A Markov Decision Process [13] is a tuple (S,A,P,R) where:

(1) S is the state space.

(2) A is the action space.

(3) $P_{a,t}(s,s') = Pr(s_{t+1}=s' \mid s_t=s,\ a_t=a)$ is the probability that action a in state s at time t will lead to state s' at time $t+1$. To qualify as an MDP, these probabilities must be independent of prior state history, i.e., at each state, they are solely determined based on the current and future states (and associated actions), but not on past states.

(4) $R(s)$ is the immediate reward received in reaching state s. This value may depend on the environment surrounding s, the action that has just been performed in order to reach s, as well as other factors (such as the consequences of the action).

(5) The goal of an MDP is to maximize the *total cumulative reward CR*, defined as the sum of all rewards received during the period considered, discounted by a rate γ (with $0 \le \gamma \le 1$) for each move from one state to another. The rate γ could be interpreted as a rate of deflation or depreciation of the rewards, or the cost of investment to gain those rewards.

$$CR = \sum_{t=0}^{\infty} (\gamma^t * R(s_t))$$

An optimum *policy* for the MDP is a function $\pi: S \rightarrow A$ that specifies the best action to take at each state in order to achieve the MDP goal.

(6) At a state if the probabilities to move to other states and/or if the rewards in the other states are not known in advance, then the system must determine the best actions by *learning* from the environment. In this case the model is said to be an *MDP with reinforcement learning* [20][24]. In some cases an immediate reward may be low at a particular state but from that state the agent may expect to reach other states with higher rewards. Therefore an optimum policy for an MDP with reinforcement learning is the one that maximizes the *long-term value V(s)* at each state s (rather than its immediate reward $R(s)$ alone). $V(s)$ can be recursively determined by the Bellman equation [3]:

$$V(s) = R(s) + \gamma * \left(\sum_{\{s' \mid s' \neq s\}} (P_{\pi(s)}(s,s') * V(s')) \right)$$

where $P_{\pi(s)}(s,s')$ is the probability to move from s to s' as determined by policy π. The term on the right-hand side of the equation (after $R(s)$) is called the *discounted average future reward from state s*.

Thus the optimum policy derives from the determination of the values of $V(s)$ for all states s. Vice-versa, if the optimum policy is known (i.e., $P_{\pi(s)}(s,s')$ are known for all s and s') and if there is a number of finite states n, then there would be n Bellman equations with n variables $V(s)$ (assuming that the other variables $R(s)$ and γ are also known). Therefore a solution could be mathematically determined through a method similar to, but more complex than, LP [23]. A similar non-LP formulation and resolution has not been found in existing literature.

3. Belief-based rewards

3.1. Task and agents

3.1.1. Task: A *task* is a tuple $T = (S, Z, P, D, F)$ in which:

(1) S is the set of *technical requirements* of the task, also representing the technical areas where the skills of the task handler are required,
(2) Z is the task completion *deadline* as required by the end-user,
(3) P is the task *priority* in line with the impacts of the completion or non-completion of the task to the enterprise,
(4) D is the degree of *difficulty* of the task, indicating the level of skills required from the task handler, and
(5) F is the *total effort* necessary to complete the task, representing the *size* of the task and expressed as the total amount of actual time (not *elapsed* time) required to complete the task.

S and Z are usually specified by the end-user while P, D and F are assessed by the manager. The notation Z_0 denotes the task start date when it is assigned to a worker. A difference between two dates, e.g., $(Z - Z_0)$, represents the number of *business* (*working*) *days* between those dates.

3.1.2. Worker: A worker is a permanent staff member or a contractor. To simplify writing, workers and manager will be referred to as male persons. Each worker is represented by a *software agent,* defined at an instant t as a tuple $A_t = (Sk, Go, Sched_t)$ in which:

(1) Sk represents the worker's skill set,
(2) Go is the worker's goals, which are to maximize the rewards for tasks handled, and for permanent staff, to also meet a performance target, and
(3) $Sched_t$ is the worker's schedule, including *expected* dates of completion of current tasks and any planned unavailability of the worker (e.g., recreational leave, training, etc.).

3.1.3. Manager: The manager's main objective is to apply best management practices to task allocation and worker management, i.e., he must be:

(1) *Efficient*, by ensuring that all tasks are completed according to their priorities and deadlines. In addition, the manager should reward (resp. penalize) workers for increased (resp. decreased) productivity. This will be reflected in the definition of reward bonus and penalty that is based on the elapsed time taken to complete a task compared to its deadline (see Sect. 3.2.1).
(2) *Cost-effective*, by preferring permanent staff over contractors when both are suitable for a particular task, as permanent staff's remunerations are fixed while contractors are paid on a task-by-task basis.
(3) *Duly diligent*, that is, to assess and manage contractors according to the *due diligence process,* by allowing contractors to bid for a new task when suitable permanent staff are not available and by selecting the *best* contractor for the new task, i.e., the one with the highest cumulative reward in the technical category to which the task belongs.
(4) *Equal opportunist*, that is, to comply with the enterprise's equal opportunity policy, by enabling permanent staff to have equal opportunities to earn similar total reward

for the period, i.e., by attempting to allocate an equal workload to all permanent staff. This means that if there were more than one permanent staff suitable for a new task, then the person with *the least cumulative reward* over the period being considered (rather than the most skilled person) would be selected. This is similar to the principle of *social justice*, which aims to provide more opportunities for the disadvantaged in a society [14]. This also has the effect that a permanent staff member who has worked hard from the start of the period would have a more relaxed workload by the end of the period (or alternatively, would receive other financial compensation or professional development opportunities).

(5) *Promoting staff efficiency*, by allocating bonuses and penalties based on how well the task deadlines are met.

3.2. Rewards and belief

3.2.1. Task reward: The *task reward W* associated with a task $T = (S, Z, P, D, F)$ is defined as: $W = BW * (1 + b) = (D*F) * (1 + b)$ (Eq.1)

in which:

(1) $BW = (D*F)$ representing the baseline reward when the task is completed on schedule. D could be thought as the daily rate and F the number of person-days necessary to complete the task.

(2) b is the bonus or penalty when the task is completed ahead of, or behind, schedule. b is expressed as a (positive or negative) percentage of elapsed time gained or loss compared to the task deadline. For example, if the task deadline is 5 days from the start date and if it is completed in 4 days, then b=20%.

3.2.2. Basic belief assignment: For a worker, a task T and an instant t, the *belief mass* of the worker is the set $\{w_i\}$ with i being a real number. Each w_i represents the probability to complete the task ahead of schedule by i% (when i is positive), or behind schedule by $|i|$% (when i is negative) (with |x| representing the absolute value of x). For example, if the worker believes that there is a 50% chance to complete the task behind schedule by 10%, 30% chance to complete the task on schedule, and 20% chance to complete the task ahead of schedule by 10%, then $w_{-10} = 0.5$, $w_0 = 0.3$ and $w_{10} = 0.2$. The sum of all belief masses w_i must always be 100%. In addition, the *belief function*, or *basic belief assignment* in Dempster-Shafer theory, denoted as *Bel*, is a partial function that assigns a task and an instant to the corresponding set of belief masses of the worker. Formally, let $\{T\}$ be the set of all tasks being handled by the worker, $\boldsymbol{R}$ the set of real numbers representing the time chronology, and n the number of realistic levels of performance improvement and degradation that the worker might achieve. *Bel* is defined as: $Bel: \{T\} \times \boldsymbol{R} \rightarrow [0,1]^n$ with $Bel(T, t) = \{w_i \mid w_i \in \boldsymbol{R}, w_i \in [0,1] \text{ and } \Sigma_i w_i = 1\}$. To simplify, $Bel(T, t)$ is also written as Bel_T.

To facilitate understanding and without loss of generality, it is assumed that:

(1) $n = 21$

(2) $Bel_T = (w_{-100, \ldots} w_{-20}, w_{-10}, w_0, w_{10}, w_{20}, .., w_{100})$, that is, a worker may achieve from 100% degradation to 100% improvement on schedule, with 10% increments.

The set $I=\{i\}=\{ -100, -90, \ldots, 0, 10, \ldots, 100\}$ is called the *frame of discernment* in belief theory, where the cardinality of I is equal to n, the number of realistic levels of performance improvement and degradation for any worker.

3.2.3. Expected task reward: The *expected task reward EW*, of a task *T*, calculated at an instant *t*, is the reward that the worker *expects* to receive when the task is completed, based on his belief on the degree of timeliness that he could achieve. *EW* is the *utility* of the task in Dempster-Shafer terminology, i.e.,

$$EW = \Sigma_i \ (w_i * (D * F) * (1 + i/100))$$
$$= (D * F) * \Sigma_i w_i * (1 + i/100) \quad \text{(Eq.2a)}$$

in which (*D*F*) represents the *baseline reward* (when the task is completed on time) and *i*/100 represents the variation in reward when the belief in meeting the task completion date is varied by *i%*.

Let:

(1) $Bel_T = (w_{-100, \ ...} \ w_{-20}, \ w_{-10}, \ w_0 \ , \ w_{10}, \ w_{20}, \ .., \ w_{100})$ be a 1-column 21-row (vertical) vector with values ranging from w_{-100} to w_{100} .

(2) *Dis* = (0, 0.1, 0.2, …,1.9, 2) be a 1-row 21-column (horizontal) vector representing the series of values (1 + (*i*/100)). *Dis* is so-called because it is similar to the frame of *discernment* in belief theory.

The expected reward can then be written as:

$$EW = D * F * Bel_T * Dis \quad \text{(Eq.2b)}$$

The expected task reward *EW* could be interpreted as the reward seen from the worker's perspective, taking into account the state that he is in at that particular time while the (absolute) task reward *W* is the reward allocated by the manager who does not know, nor consider, the states that his workers are in.

3.2.4. Expected task completion date: Similarly, the *expected task completion date EZ* is defined as:

$$EZ = Z_0 + (\Sigma_i w_i * (Z - Z_0) * (1 - i/100))$$
$$= Z_0 + ((Z - Z_0) * \Sigma_i w_i * (1 - i/100)) \quad \text{(Eq.3a)}$$
$$= Z_0 + ((Z - Z_0) * Bel_T * Dis_2) \quad \text{(Eq.3b)}$$

with Dis_2 = (2, 1.9, 1.8, …, 0) representing the series of values: (1 – (*i*/100)). Dis_2 is the same vector as *Dis* but arranged in the descending order of its elements.

Note that the expected completion date is inversely proportional to the expected reward, and this is reflected in the factor (1-*i*/100) in Eq.3a, as opposed to (1+*i*/100) in Eq.2a.

3.2.5. Expected cumulative reward: The *expected cumulative reward CW* for a worker during a particular period (denoted as the time interval $[t_0, t_1]$) and calculated at a particular instant *t* is the sum of *cumulative past and future rewards* (denoted as *PW* and *FW*), that have been apportioned to the period considered, i.e.,

$$CW = PW + FW$$
$$PW = \Sigma_T PW_T$$
$$FW = \Sigma_{T'} FW_{T'}$$
$$PW_T = W * (Z - Max\ (Z_0, t_0)) / (Z - Z_0)$$
$$= D * F * (1 + b) * (Z - Max\ (Z_0, t_0)) / (Z - Z_0) \quad \text{(Eq.4a)}$$
$$FW_{T'} = EW * (Min\ (EZ, t_1) - Max(Z_0, t_0)) / (EZ - Z_0)$$
$$= D * F * Bel_{T'} * Dis * (Min(EZ, t_1) - Max(Z_0, t_0)) / (EZ - Z_0) \quad \text{(Eq.4b)}$$

The *past cumulative reward* PW_T (Eq.4a) is a generalization of Eq.1, in which:

(1) (Z - $Max(Z_0, t_0)$) represents the period from the task or period start date (Z_0 or t_0, whichever the later, determined by *Max*), to the task completion date (Z) and ($Z - Z_0$) represents the full period from start to finish of the task.

(2) $(Z - Max(Z_0, t_0)) / (Z - Z_0)$ represents the pro-rata percentage of the (completed) task reward for the period considered.

Similarly, the *future cumulative reward* $FW_{T'}$ (Eq.4b) is a generalization of Eq.2b, in which:

(1) $(Min(EZ, t_1) - Max(Z_0, t_0))$ represents the period from the task or period start date (Z_0 or t_0, whichever the later) to the expected task completion date (EZ) or the period end date (t_1), whichever the earlier (determined by *Min*). Note that for future tasks, $Max(Z_0, t_0) = Z_0$ is always true.

(2) $(Min(EZ, t_1) - Max(Z_0, t_0)) / (EZ - Z_0)$ represents the pro-rata percentage of the expected task reward for the period considered.

3.2.6. Expected working pace: For a worker, the *required working pace RP* on a task is defined as the quotient of the total effort required to complete the task by the total elapsed time required to complete the task on time, i.e., $RP = F/(Z - Z_0)$.

For example, if a task requires 1 person-month to complete and its deadline is 3 months away, then the required working pace is 33%. Similarly, the *expected working pace* (also called *working pace* for short) of a worker on a task is defined as:

$$WP = F / (EZ - Z_0) \quad \text{(Eq.5a)}$$
$$= F / ((Z - Z_0) * Bel_T * Dis_2) \quad \text{(Eq.5b)}$$
$$= RP / (Bel_T * Dis_2) \quad \text{(Eq.5c)}$$

3.2.7. Expected workload: At any instant t, the *expected workload WL* (also called *workload* for short) of a worker is defined as the sum of all working paces of all outstanding tasks that the worker handles at that instant, i.e.,

$$WL = \Sigma_T WL_T$$
$$WL_T = WP * Max(0, (EZ - t)/|EZ - t|)$$
$$= (F/(EZ - Z_0)) * Max(0, (EZ - t)/|EZ - t|) \quad \text{(Eq.6)}$$

The factor $Max(0, (EZ - t)/|EZ - t|)$ in Eq.6 expresses that the working pace on a task is nil when the expected task completion date occurs before t (i.e., when $(EZ - t) < 0$), or equal to 1 otherwise (i.e., $(EZ - t) = |EZ - t|$).

3.2.8. Worker schedule: The expected workload of a worker can be used to determine whether the worker can handle a new task. This could be implemented through a function called *verifySchedule,* which accepts as inputs:

(1) the worker's schedule $Sched_t$,

(2) the total effort F_T necessary to complete the new task T, and

(3) *MaxPace*, representing the maximum working pace of the worker (with *MaxPace* = 100% by default),

and returns as output:

(1) the expected date of completion of the new task EZ_T if it is undertaken by the worker, i.e., $verifySchedule(Sched_t, F_T, MaxPace) = EZ_T$

This result is then compared against the deadline of the new task (Z_T) to determine whether the task could be assigned to the worker.

4. Agent strategies

The overall process of management and handling of a task is as follows:

(1) For each new task, the system determines the list $\varphi(T)$ of all suitable workers based on the technical requirements S of the task and the known skills of the workers.
(2) If $\varphi(T)$ is empty, then the manager should hire additional permanent staff or contractors with skill sets matching the new task's technical requirements.
(3) If $\varphi(T)$ contains permanent staff, then for each of them, the system verifies whether the new task could fit into the worker's workload (with the function *verifySchedule*). Optionally, these permanent staff could be asked whether they are prepared to work overtime so that their *MaxPace* values could be increased in the calculation by *verifySchedule*. At first view, it seems that the task should be assigned to the worker with the lightest workload. However, the enterprise's *Equal Opportunity* policy (Sect. 2.1.3) may dictate that it should be assigned to the worker with the lowest expected cumulative reward instead, similarly to the principle of "social justice" discussed previously.
(4) If $\varphi(T)$ contains only contractors, then each of them should define his belief masses w_i concerning the new task, then make a decision based on a strategy detailed in Sect. 3.2.
(5) In the above case, each concerned contractor should first check the task's details, then express his belief on how well he can complete the task, by specifying the values of w_i for the task, and finally make the decision whether to bid for the new task. The contractor's decision may be one of the following options:
    - The contractor may decide not to bid for the new task, even when he can meet its deadline, in order to focus on the tasks at hand and to attempt to gain maximum bonuses from their early completion, and/or in order to prepare for some future tasks with higher rewards.
    - The contractor may realize that he may not be able to meet the deadline of the new task unless the working pace on his other existing tasks could be reduced. This may then result in delays, and hence penalties, for these existing tasks, although the total cumulative reward (including the new task) still improves for the contractor. However, *professional ethics* dictate that the contractor should avoid excessive penalty on any task by ensuring that each task is completed within an acceptable timeframe (i.e., a threshold not far beyond the task's desired completion date).
(6) The manager then selects a winner among those bidding contractors. A number of alternate strategies are possible: random selection, selection of the contractor with the lowest cumulative reward (as per the principle of "social justice", as used for permanent staff), or selection of the contractor with the highest cumulative reward, i.e., the "best person for the job". In general, the last strategy is preferred by the manager, although it may contribute to the strengthening of an *elitist* class of contractors [6], which may or may not be desirable in an enterprise.

4.1. Manager strategy

The manager's strategy in task allocation is as follows:
(1) If during the period considered, a worker handles a *small* number of tasks but receives a *high* level of rewards, then he is likely to be skilled in handling difficult and/or long-duration tasks (both imply high rewards). He is likely to be an expert on some technical area and the manager should prefer him for future tasks of that nature.

(2) If during the period considered, a worker handles a *large* number of tasks and receives a *high* level of rewards, then he is likely to be efficient in solving common (easy) problems, and should be preferred for common tasks on tight schedule.
(3) If during the period considered, a worker handles a *small* number of tasks and receives a *low* level of rewards, then he is likely not to be very productive. If he is a contractor, then he should only be considered when others are not available.
(4) If during the period considered, a worker handles a *large* number of tasks and receives a *low* level of rewards (possibly due to reward penalties), then he is likely to be unproductive and untimely. If he is permanent staff, he may need further training. If he is a contractor, then he should only be considered when others are not available.
(5) Finally, the analysis of bonuses and penalties received by each worker during the period considered could assist the manager in staff profiling. Further data mining exercises can produce lists of top contractors per technical area that the manager can use for his selection strategy during task allocation (point (1) above).

4.2. Contractor strategy

At any instant, the aim of a contractor is to optimize the belief masses w_i for all current and future tasks so that his total expected reward is maximized.

4.2.1. Multi-objective optimization formalization: The contractor strategy could be formalized as a multi-objective optimization problem within the belief theory. Let m be an arbitrary large number representing the largest number of tasks that could possibly exist within the period considered.

At any instant t, the contractor strategy is represented by a multi-objective partial function:

$f_t: X \rightarrow Y$

$f_t(\{Bel_T\}^m) = \{EW_T\}^m$

where:

(1) X is the *decision space* of the contractor in which each element is a m-dimensional vector of basic belief assignments corresponding to the tasks that the contractor is handling or expects to be awarded within the period considered (i.e., $X = B^m$).
(2) Y is the *objective space* of the contractor in which each element is a vector of real numbers representing the rewards for the tasks whose basic belief assignments are in the decision vector (i.e., $Y = R^m$).

The contractor strategy is thus formalized as an m-objective optimization problem. Furthermore, a total order u on the objective space Y could be defined by *aggregation* of the objectives (as opposed to other methods such as *Pareto optimization* [8]) by summing up all the task rewards of the objective vector, i.e.,

$u: Y \rightarrow \boldsymbol{R}$

$u(\{EW_T\}^m) = \Sigma_T EW_T$

Concatenating the two functions u and f_t yields a function g_t that transforms the problem into a single-objective optimization problem:

$g_t = u \,.\, f_t$

$g_t(\{Bel_T\}^m) = \Sigma_T EW_T = FW$ (Eq.7)

The objective then becomes to determine all basic belief assignments Bel_T for all current and expected future tasks T such that g_t ($\{Bel_T\}^m$) is maximized. This is also the value of FW (Eq.4b).

4.2.2. Markov decision process formalization: *MDP* has been used to model decision support systems in which a series of actions are to be determined, and at each action decision step, past actions are not taken into account [10][11]. In Eq.7, tasks that have been completed can be ignored since their rewards have been determined and cannot affect actions and rewards for current and future tasks. Thus, the optimization process for a particular contractor could be modeled as an MDP. Moreover, it is an *MDP with reinforcement learning* [20][24] since the arrival of new tasks and whether the contractor will bid for them are not known in advance. In this MDP, which concerns only one contractor:

(1) Each state s is a set of tasks being handled by the contractor, represented as an m-dimensional vector of binary values 0 and 1, i.e., $s \in \{0,1\}^m$. The first element of the vector represents the first task being available in the period considered, and so on. A value of 1 indicates that the contractor is working or has worked on the corresponding task and a value of 0 indicates otherwise. There are therefore 2^m *possible* states for a contractor.

(2) At each state, when a new task is available, the contractor must decide whether to bid for the new task and if successful, will accept the task. If the contractor decides not to bid for the new task or if his bid is unsuccessful, then he is situation in the new state is the same as in the current state, with regard to the tasks handled and the expected rewards. This strategy qualifies the model as an MDP since the decision to bid and accept a new task depends solely on the current state, i.e., is solely based on current and future tasks (and not on any previous states and/or previous rewards).

(3) The expected reward $V(s_t)$ in reaching state s (at time t) is the net expected total reward, defined as the difference between the future cumulative reward associated with state s (i.e., $FW(s)$) and the same reward at its previous state (at time t-1), i.e.,

$$V(s_t) = FW(s_t) - FW(s_{t-1}) \quad \text{(Eq.8)}$$

Thus, the value of $V(s)$ is mainly based on the set of beliefs Bel_T associated with current and future tasks (Eq.4b and Eq.7) at times t and t-1. Eq.8 is another way of expressing $V(s)$ similar in meaning to the Bellman equation of MDP [3], with the beliefs Bel_T in Eq.8 being similar to the probabilities to move to another state $P_{\pi(s)}(s,s')$ of the Bellman equation.

(4) If during the period considered, tasks occur randomly without following any probability distribution, then when faced with a new task the contractor would be unable to predict and reserve his effort for future tasks other than the new task being offered. The only logical strategy would then be to completely ignore other possible future tasks and take into account only existing tasks and the new task. The calculation of $V(s)$ helps the contractor determine whether to bid for the new task. If $V(s)$ is not positive, then the contractor should not bid for the new task. Otherwise, he should bid and if successful, should accept the new task. This is the *best conservative strategy*, which guarantees that the total expected reward would never decrease at any time (any state).

(5) However, if the occurrence of tasks and their reward levels follow some known probability distributions, then the contractor may prefer not to bid for a new task, so that he can devote more effort to later tasks with higher rewards. In an automated

system, statistical data on task distribution could be used to train the MDP in order to produce an optimum policy, e.g., by using a Q-Learning algorithm [20].

4.2.3. Problem formulation: The above single-objective optimization problem could be formalized as a problem to determine an optimum set of belief assignments that maximizes the value of g_t (Eq.7) or FW (Eq.4b), i.e.,

Maximize $FW = \Sigma_T FW_T$
with:
$FW_T = D^* F * Bel_T * Dis * (Min(EZ, t_1)\text{-}Max(Z_0, t_0)) / (E - Z_0)$
under the following constraints:
(1) $\forall T \ w_{iT} \in \boldsymbol{R}$, $w_{iT} \in [0,1]$ and $\Sigma_i w_{iT} = 1$
(2) $\Sigma_T WP_T \leq MaxPace$
with: $WP_T = F / ((Z - Z_0) * Bel_T * Dis_2)$
(3) $\forall T \ ((EZ - Z) / (Z - Z_0)) \leq MaxDelay$

In the above, the first constraint simply expresses the definition of belief masses. The second constraint expresses that at any time the sum of all working paces corresponding to all current and future tasks cannot exceed a threshold *MaxPace* determined by the contractor. For example, a *MaxPace* value of 1.2 means that the contractor is prepared to work overtime, up to a level of 20% above normal working hours. The last constraint expresses the contractor's *ethics*, which imposes a delay limit *MaxDelay* on any task. For example, a *MaxDelay* value of 0.1 means that on any task the contractor should not be more than 10% behind schedule (or should not accept a reward penalty of more than 10%).

4.2.4. Problem resolution: Since the number of variables w_i in the above problem is usually large, the problem resolution is generally *NP-hard* (as per the mathematical theory of *complexity*). This conclusion is acknowledged in other similar work [4]. In simple terms, this means that there is no computational algorithm that would provide a theoretical (and thus quick) solution, and the only way to solve the problem is a computationally intensive and exhaustive analysis of all possible outcomes, which may or may not meet the time constraint of the situation being envisaged or the resource constraint of the computer(s) on which the analysis is performed. In real life, this means that the contractor would specify a finite number of alternate sets of reasonable belief masses, and the system would calculate the value of g_t for each set and then compare them to determine the one that maximizes the value of g_t. However, in simple situations, the problem could be solved by computational algorithm, as demonstrated by the example below.

4.2.5. Scenario: Following is the example of a simple situation of a contractor:
(1) Based on his ethics, the contractor aims to complete any task on time or with a delay of 10% maximum beyond schedule. This means that *MaxDelay*=0.1 and for each task there are only two non-null belief masses w_{-10} and w_0 (with $w_{-10} + w_0 = 1$).
(2) The contractor is prepared to work up to 20% overtime to meet the above objective, i.e., *MaxPace*=120%.
(3) The contractor has a task T at hand and a new task T' is offered to him for bidding. Both tasks start, and are expected to finish, within the period considered.

(4) For the current task, the baseline reward BW_T is \$100 and the working pace RP_T required to complete it on time is 50%. For the new task, the baseline reward $BW_{T'}$ is \$200 and the required working pace $RP_{T'}$ is 80%.

The formalized problem becomes to determine the sets of belief masses $\{w_i\}$ and $\{w'_i\}$ such that the total expected future reward FW is maximized.

Let: $x=w_0$, $y=w'_0$, $z=BW_T$ and $p=MaxPace$

The function to be maximized becomes:

$FW = EW_T + EW_{T'}$

$= z\,(0.9\,w_{-10} + w_0) + 200\,(0.9\,w'_{-10} + w'_0)$ (by applying Eq.2a)

$= z\,(0.9\,(1-x) + x) + 200\,(0.9\,(1-y) + y)$

(by applying constraint (1) with $w_{-10}=(1-w_0)$ and $w'_{-10}=(1-w'_0)$)

$= z\,(0.9 + 0.1x) + 200\,(0.9 + 0.1y)$

$= 180 + 0.9z + 0.1zx + 20y$

Since $(180 + 0.9z)$ is a fixed amount, it can be ignored in the function to be maximized. So the problem becomes to maximize $(0.1zx + 20y)$ or to maximize:

$f = 0.01zx + 2y$ (Eq.9)

And constraint (2) is calculated as follows:

$WP_T + WP_{T'} \leq p$

$(RP_T / (Bel_T * Dis_2)) + (RP_{T'} / (Bel_{T'} * Dis_2)) \leq p$ (as per Eq.5c)

$(0.5 / (1.1\,w_{-10} + w_0)) + (0.8 / (1.1\,w'_{-10} + w'_0)) \leq p$

$(0.5 / (1.1\,(1-w_0) + w_0)) + (0.8 / (1.1(1-w'_0) + w'_0)) \leq p$ (as per constraint 1)

$(0.5 / (1.1 - 0.1x)) + (0.8 / (1.1 - 0.1y))) \leq p$

$0.5\,(1.1-0.1y) + 0.8\,(1.1-0.1x) \leq p\,(1.1-0.1x)(1.1-0.1y)$

$(0.11p-0.08)x + (0.11p-0.05)y - 0.01pxy \leq 1.21p - 1.43$ (Eq.10)

The remaining constraint (3) on $MaxDelay$ is automatically satisfied by considering only two non-null belief masses w_{-10} and w_0.

For z=100, the function to be maximized (Eq.9) becomes:

$f = x + 2y$ (Eq.9a)

For z=100 and p=120%, constraint (2) (Eq.10) becomes:

$0.052x + 0.082y - 0.012xy \leq 0.022$

$52x + 82y - 12xy \leq 22$ (Eq.10a)

In summary, the problem becomes to maximize:

$f = x + 2y$

under the constraints:

(1) $x,y \in [0,1]$

(2) $52x + 82y - 12xy \leq 22$

One way to resolve the problem is to use a method similar to (but more complex than) the simplex method of LP [23] by representing the function to be maximized and the 3 constraints on a graph with 2 orthogonal axes x and y. The graph would look like Fig. 1. An arbitrary value of the function f to be maximized is represented by a line with an arrow indicating the direction in which it should move (in parallel, i.e., while keeping the angle to the axis x or y constant) in order to increase its value.

The optimal solution is where that line representing f meets the boundary of the constraint area, which is delimited by the two axes x and y, and the (slightly convex) curve that represents the boundary of constraint (2). In that constraint, if the term $-12xy$

in the left side of the equation is ignored (since it is small compared with other terms, especially with x, $y \in [0,1]$), then the curve becomes a straight line (representing the equation: $52x + 82y = 22$) and the optimal solution would be: (x=0 and y=0.2683) or (w_0=0, w_{-10}=1, w'_0=0.2683 and w'_{-10}=0.7317). This means that that the contractor should bid for the new task, and if successful, *should focus his effort more on the new task*. By doing so, he would also certainly miss the deadline of the existing task by 10% and would have a 73% chance to miss the deadline of the new task by 10% too.

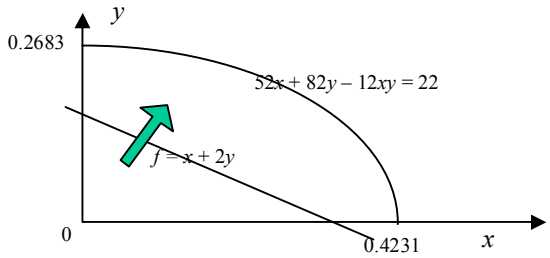

Figure 1. Graphical representation of objective function and constraints

4.2.6. Strategy variation: The contractor's strategy in the above example varies according to the values of the base reward BW_T and the maximum working pace *MaxPace*. Following are a few scenarios:

- **50% Increase in baseline reward**

In the above scenario, if the baseline reward for the existing task were \$150 (instead of \$100), i.e., $BW_T = z$ =150 (instead of 100), and $MaxPace = p$ = 120%, then the same calculation as above would yield:

$FW = 315 + 15w_0 + 20w'_0$

and the function to be optimized would become:

$f = 1.5x + 2y$ with the same constraints as above (Eq.10a).

The optimal solution would then be:

(x=0.4231 and y=0), or (w_0=0.4231, w_{-10}=0.5769, w'_0=0 and w'_{-10}=1)

This means that the contractor should still bid for the new task and accept it if successful. However, *he should now better focus on the existing task* rather than on the new task to the point of accepting a 10% reward penalty on the new task for delay on its deadline.

In addition, if constraint (2) is linear (i.e., $52x + 82y \leq 22$), then the simplex method would show that the recommendation to focus on the existing task would be true whenever the baseline reward z of the existing task is over \$126.83. However, if the original equation of constraint (2) is used (i.e., $52x + 82y - 12xy \leq 22$), then the optimal solution will correspond to a point on the curve representing constraint (2) in Fig.1. That point could be more accurately determined through a computationally intensive process, e.g., by examining all values of f corresponding to various values of z between 125 and 129. However, since the curve is only slightly convex (i.e., the term $-12xy$ is small compared to other terms in Eq.10a), it could be assumed that the optimal solution corresponds to a baseline reward for the existing task of about \$127. This example shows that the resolution of the problem is more complex than LP.

- **10% Overtime scenario**

Another scenario is that if the contractor is only prepared to put in 10% extra effort (instead of 20%), i.e., p=1.10 (and still with z=100), then the function to be optimized is still $f = x + 2y$ (Eq.9a) but constraint (2) (Eq.10) becomes:

$41x + 71y - 11xy \leq -99$ (Eq.10b)

It is not possible to satisfy this constraint as it can be easily proven that the left side of the equation is always positive (i.e., $41x + 71y \geq 11xy$ when $x,y \in [0,1]$). *This means that the contractor should not bid for the new task.*

- **Break-even point for working pace**

Furthermore, as with the previous calculation and with all other assumptions the same as initially, it could be also demonstrated that the decision to bid would be recommended whenever the contractor's working pace p is greater than a certain threshold such that with that threshold, there exists a solution (x,y) (with $x,y \in [0,1]$) that satisfies Eq.10. Again, that solution could be more accurately determined through a computationally intensive process, e.g., by examining the satisfaction of constraint (2) (Eq.10) with various values of p between 110% and 120%. In this case, it can be demonstrated that the recommendation to focus on the existing task is true whenever the baseline reward of the existing task is more than about $127.

- **Summary**

Table 1 summarizes the different decision options for the contractor.

Table 1. Contractor decision recommendation

Main Assumptions	Objective Function to be maximized	Constraints	Solution	Recommended Decision
BW_T = \$100 MaxPace =120%	f = x + 2y ($x=w_0$, $y=w'_0$)	$0 \leq x \leq 1$ $0 \leq y \leq 1$ $52x+82y-12xy \leq 22$	$w_0 = 0$ $w_{-10} = 1$ $w'_0 = 0.27$ $w'_{-10} = 0.73$	Should bid and if successful should focus more on new task
BW_T = \$150 MaxPace =120%	f = 1.5x + 2y ($x=w_0$, $y=w'_0$)	$0 \leq x \leq 1$ $0 \leq y \leq 1$ $52x+82y-12xy \leq 22$	$w_0 = 0.42$ $w_{-10} = 0.58$ $w'_0 = 0$ $w'_{-10} = 1$	Should bid and if successful should focus more on existing task
BW_T < \$127 MaxPace =120%	f = 0.01 z x + 2y ($x=w_0$, $y=w'_0$, $z=BW_T$)	$0 \leq x \leq 1$ $0 \leq y \leq 1$ $52x+82y-12xy \leq 22$ $z < 127$	$w_0 = 0$ $w_{-10} = 1$ $w'_0 = 0.27$ $w'_{-10} = 0.73$	Should bid and if successful should focus more on new task
$BW_T \geq$ \$127 MaxPace =120%	f = 0.01 z x + 2y ($x=w_0$, $y=w'_0$, $z=BW_T$)	$0 \leq x \leq 1$ $0 \leq y \leq 1$ $52x+82y-12xy \leq 22$ $z \geq 127$	$w_0 = 0.42$ $w_{-10} = 0.58$ $w'_0 = 0$ $w'_{-10} = 1$	Should bid and if successful should focus more on existing task
BW_T = \$100 MaxPace =110%	f = x + 2y ($x=w_0$, $y=w'_0$)	$0 \leq x \leq 1$ $0 \leq y \leq 1$ $41x+71y-11xy \leq -99$	No solution	Should not bid on new task

4.3. Permanent staff strategy

The goals of a permanent staff member are similar to those of a contractor. But, in addition, a permanent staff member is also required to achieve a minimum performance target during any period considered (as is often the case in most enterprises). This is expressed as a new constraint (4) that complements the 3 constraints listed in Sect. 4.2.3 (with *PW* and *FW* calculated as per Eq.4a/b):

(4) $CW = PW + FW \geq Quota$

This constraint also means that if a permanent staff member has already achieved the desired quota for the period then there is no need for him to bid for any new task during that period (unless increased remuneration or job prospect is considered). The manager should recompense him through other means (e.g., further training).

In this case, while the permanent staff strategy could still be modeled as a multi-objective optimization problem (with 4 constraints) and resolved in a way similar to what is proposed for the contractor, the model is not an MDP since the consideration of work quota implies that past actions should be taken into account in determining any new action, and this does not meet the definition of an MDP.

5. Conclusion

This paper proposed a multi-agent system for use in a large organization or business section, such as a customer support center or a project management department. The system is designed to assist management in its daily role of task allocation to permanent staff and contractors, as well as to assist the latter in their decision whether to bid for a new task and on which tasks they should focus their effort. As such, the system consists of two autonomous decision-support agents, one for the manager and one for the contractors and permanent staff. For the manager, his decision-support sub-system takes into account different management policies towards permanent staff vs. contractors, with the ultimate aim of minimizing the overall cost to the enterprise and of fairly treating all staff. The system enables management to spread the team workload evenly among permanent staff while also permits contractors with the best demonstrated performance to be offered new tasks. On the other hand, all workers are assisted in their decisions by their decision-support sub-systems, which help them decide whether to bid for a new task and on which tasks they should spend most effort. Their sub-systems are formalized as a belief-based multi-objective optimization problem, with different constraints representing the different goals between these two categories of worker. The main contribution of this paper is to propose a method for staff management combining belief theory with LP-like and MDP with reinforcement learning formulations. An example of the former case is presented in this paper. When formulated as an MDP, existing MDP resolution methods such as the one proposed in [1] or other Q-learning techniques [20] could be used to produce an optimum strategy for the contractor.

The proposed system could be further enhanced by examining other more complex scenarios, such as the possibility for workers to negotiate task swapping and/or team forming in order to better satisfy user requirements and/or to improve their individual gains [17][9][2]. For the same objectives, the proposed system could also be recursively applied, by allowing a worker to break a task into multiple sub-tasks and to sub-contract the latter to other contractors. In addition, the possibility of sub-tasks being performed concurrently could help meet the deadline of the overall task. Each worker's

bidding strategy could also be viewed as a multi-agent system in itself, in which each current or future task is an autonomous agent sharing a common goal, which is to maximize the total reward for the worker, even at the expense of the individual goal of each agent. Genetic Algorithms could be applied to individual agents in this case as suggested in Sect. 5 of [4]. Finally, performance data collected from the system could be further mined for a variety of other purposes, such as to assist in the annual performance review of each permanent staff member and contractor.

References

[1] D. Aberdeen, S. Thiébaux and L. Zhang, "Decision-Theoretic Military Operations Planning," 14th International Conference on Automated Planning & Scheduling (ICAPS), Whistler, Canada, pp. 402-411, June 2004.

[2] Q. Bai and M. Zhang, "Dynamic Team Forming in Self-Interested Multi-Agent Systems," Eighteenth Australian Joint Conference on AI, Sydney, Australia, 2005.

[3] R. E. Bellman, Dynamic Programming, Princeton University Press, Princeton, NJ, 1957.

[4] A. Cardon, T. Galinho and J.P. Vacher, "An Agent Based Architecture for Job-Shop Scheduling Problem Using the Spirit of Genetic Algorithm," *Evolutionary Algorithms in Engineering and Computer Science,* Jyväskylä, Finland, pp. 12-19, 1999.

[5] R. Davis and R.G. Smith, "Negotiation as a Metaphor for Distributed Problem Solving," Artificial Intelligence, pp. 63–100, 1983.

[6] U. Endriss, N. Maudet, F. Sadri and F. Toni, "Resource Allocation in Egalitarian Agent Societies," Secondes Journées Francophones sur les Modèles Formels d'Interaction, pp. 101-110, 2003.

[7] J. Ferber, Multi-agent Systems - An Introduction to Distributed Artificial Intelligence, Addison-Wesley, 1999.

[8] C. Fonseca and P. Fleming, "An Overview of Evolutionary Algorithms in Multiobjective Optimization," Evolutionary Computation Journal, pp. 1-16, 1995.

[9] M. Golfarelli, D. Maio and S. Rizzi, A Task-Swap Negotiation Protocol Based on the Contract Net Paradigm, Technical Report 005-97, CSITE, Universita di Bologna, 1997.

[10] M.Q. Nguyen, P.HP. Nguyen, T. HNguyen and D. O'Shaughnessy, "A Proposed AI Planning Approach for Staff Management in a Service Center," 5th Int. Conf. on Industrial Automation, Montreal, Canada, pp.11-13, June, 2007.

[11] M.Q. Nguyen, P.HP. Nguyen, D. O'Shaughnessy and J-G. Meunier, "A Quality-Focused Spoken Dialog System With Reinforcement Learning And Simulated User," 6th IEEE International Conference on Computer Science, International Conference on Research, Innovation & Vision for the Future (RIVF 2008), HoChiMinh City, Vietnam, July 2008.

[12] H.S. Nwana, "Software Agents: An Overview," Knowledge Engineering Review, II(3), Cambridge University Press, 1996.

[13] M.L. Puterman, Markov Decision Processes, Wiley, 1994.

[14] J. Rawls, A Theory of Justice, Oxford University Press, 1971.

[15] T. Sandholm, "An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations.", Eleventh National Conference on Artificial Intelligence, Menlo Park, Calif., USA, pp. 256–262, 1993.

[16] T. Sandhol and V. Lesser, "Issues in Automated Negotiation and Electronic Commerce: Extending the Contract Net Protocol.", Second International Conference on Multiagent Systems, Menlo Park, Calif. USA, pp. 328–335, 1995.

[17] T. Sandholm, "Contract Types for Satisficing Task Allocation: I - Theoretical Results," AAAI Spring Symposium Series: Satisficing Models, Sanford University, USA, 1998.

[18] T. Sandholm, Distributed Rational Decision Making. In Multiagent Systems, a Modern Approach to Distributed Artificial Intelligence, The MIT Press, London, 2000.

[19] R.G. Smith. "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver," IEEE Transactions on Computers, pp. 1104-1113, 1980.

[20] R. Sutton and A. Barto, Reinforcement Learning: An Introduction, Cambridge. MA: MIT Press, 1998.

[21] K. Sycara, Using Option Pricing to Value Commitment Flexibility in Multiagent Systems, Technical Report CMU-CSTR-97-169,Carnegie Mellon University, 1997.

[22] K. Sycara, "Multiagent Systems," American Association for Artificial Intelligence, AI Magazine, Summer ,1998.

[23] S. Waner and S. Costenoble , Finite Mathematics, Brooks/Cole Publishing Co., 2001.

[24] C. Watkins, Learning from Delayed Rewards, Ph.D. Thesis, King's College, UK, 1989.

Authors

Philip H.P. Nguyen received the MS degree in pure mathematics from University Joseph Fourier, Grenoble, France, the postgraduate degree in economic mathematics, the postgraduate degree in industry management, and the PhD degree in information technology, respectively, from University Pierre and Marie Curie, Paris, France. His research interests include knowledge representation, ontology and multi-agent system. He is currently a principal technical specialist for Department of Justice, Government of South Australia. Prior to this position, he worked for various universities, Thomson-CSF and Honeywell- Bull, all in Paris, France, and CSIRO, Canberra, Australia.

Minh-Quang Nguyen received his MS degree from University Paris Diderot, France, in 1987 and his PhD degree in cognitive science from University of Quebec at Montreal, Canada, in 2008. His research is focused on Artificial Intelligence, in particular, knowledge representation, spoken dialog system, human-computer interaction and multi-agent system. He currently works for the information technology service at the University of Quebec at Montreal, Canada.

Ken Kaneiwa is Associate Professor at Department of Electrical Engineering and Computer Science, Iwate University, Japan. He worked for Fujitsu Ltd. from 1993 to 1996 and received his MS and PhD degrees from Japan Advanced Institute of Science and Technology in 1998 and 2001, respectively, majoring in Information Science in both cases. His research interests include knowledge representation and reasoning, order-sorted logic and semantic web. He received Best Paper Award from Japanese Society for Artificial Intelligence in 2006 and is a member of Japan Society for Software Science and Technology, Institute of Electronics, Information and Communication Engineers, Information Processing Society of Japan, Japanese Society for Artificial Intelligence and Association for Logic Programming.