© The Author 2005. Published by Oxford University Press on behalf of The British Computer Society. All rights reserved.
For Permissions, please email: journals.permissions@oupjournals.org
doi:10.1093/comjnl/bxh000

Fast distributed near-optimum assignment of assets to tasks

Erol Gelenbe[1], Stelios Timotheou[1] David Nicholson[2]

[1]*Dept. Electrical & Electronic Eng.*
Imperial College
London SW7 2BT, UK
[2]*BAE Systems*
Sowerby Building
Filton, Bristol BS34 7QW, UK
Email: e.gelenbe@imperial.ac.uk

We investigate the assignment of assets to tasks where each asset can potentially execute any of the tasks, but assets execute tasks with a probabilistic outcome of success. There is a cost associated with each possible assignment of an asset to a task, and if a task is not executed there is also a cost associated with the non-execution of the task. Thus any assignment of assets to tasks will result in an expected overall cost which we wish to minimise. We formulate the allocation of assets to tasks in order to minimise this expected cost, as a nonlinear combinatorial optimisation problem. A neural network approach for its approximate solution is proposed based on selecting parameters of a Random Neural Network (RNN), solving the network in equilibrium, and then identifying the assignment by selecting the neurons whose probability of being active is highest. Evaluations of the proposed approach are conducted by comparison with the optimum (enumerative) solution as well as with a greedy approach over a large number of randomly generated test cases. The evaluation indicates that the proposed RNN based algorithm is better in terms of performance than the greedy heuristic, consistently achieving on average results within 5% of the cost obtained by the optimal solution for all problem cases considered. The RNN based approach is fast and is of low polynomial complexity in the size of the problem, while it can be used for decentralised decision making.

Keywords: Optimum Asset Assignment, Distributed Decisions, Nonlinear Combinatorial Optimisation, Random Neural Network

Received 00 Month 2009; revised 00 Month 2009

1. INTRODUCTION

Assignment problems is a fundamental class of combinatorial optimisation problems which involve assigning assets to tasks to minimise a desired cost function [1]. Several variations of these problems have been studied over the years finding widespread application in diverse fields such as telecommunications, transportation systems and signal processing [2]. Nevertheless, an important assumption made in such problems is that the desired result of an assignment always occurs, e.g. a job assigned to a particular machine is executed successfully. In this paper we investigate an assignment problem where the result of an assignment is uncertain.

Consider a set of tasks T and a set of assets W. All of the tasks in the set T need to be executed, any of the tasks can be executed by any one of the assets, and there is a penalty denoted $K(t)$ for not executing the task t. Any one of the assets can potentially suffice to execute any one of the tasks, but there is a cost $C(w,t)$ for executing task t using asset w. It is also possible that a task execution may fail despite the fact that an asset has been assigned to it, and this will be represented by the probability $q(w,t)$ that asset w will fail in executing task t. Thus $q(w,t)$ is the probability that despite the fact that the asset w has been assigned to task t, w is unable to execute task t , where $0 \leq q(w,t) \leq 1$. Note that $q(w,t)$ may also be viewed as a characteristic of the pair (w,t) where the asset w is not perfectly effective or suited to deal with the needs of task t. Additionally, to reduce the probability of task execution failure more than one assets may be assigned to the same task.

Our objective is to be able to assign assets to tasks so that some "best" choice is made with respect to a cost function that will be defined below. We represent the decision of assigning the asset w to the task t by the probability $p(w,t)$, allowing for the fact that an asset may not be assigned to any of the tasks, so that:

$$0 \le p(w,t) \le 1, \text{ and } \sum_{t\in T} p(w,t) = \pi(w) \le 1, w \in W, \tag{1}$$

Here $\pi(w) \le 1$ represents the fact that we may decide not to assign the asset w to some task. Note that if all the $p(w,t)$ are either 0 or 1 then we have a deterministic formulation. In fact, given the probabilistic formulation, we can also consider a deterministic counterpart of this problem by seeking $\{0,1\}$ solutions. We also assume that after an asset is allocated to some task, it cannot be re-assigned again to some other task. Thus we are considering cases where the assets are expendable and cannot be re-used. This also corresponds to real-time situations where, for the given time epoch considered, decisions are irrevocable: e.g. once an ambulance has been sent out to some emergency, we are not allowed to also send it somewhere else. This probabilistic representation of the decision to allocate the asset w to the task t is also a convenient way of formulating a distributed decision algorithm. In a distributed approach, some decision agent may only handle decisions that concern a single asset, or a few assets, and this agent decides about this allocation without considering the other allocation decisions being made about other assets. Similarly, the task t may decide to select asset w without consulting or coordinating with other decision makers. Application areas that are covered by this abstract representation include examples where:

- T is a set of "jobs", and W is the set of "resources" and the goal is to find an assignment matrix in order to minimise the expected cost of not executing successfully the "jobs" as well as the assignment cost,
- T is a set of "emergencies" and W is the set of "ambulances" or emergency personnel with the combined goal of maximising the number of injured collected and minimising their response time when it is uncertain if an emergency unit will reach the targeted emergency,
- T is a set of "entities that need to communicate" and W is the set of "communication channels or frequencies" and the objective is to maximise the expected number of entities that will successfully communicate when communication in each of the channels is uncertain, etc.

The purpose of this work is to find ways of making these assignments so that:

- We minimise a relevant cost function, which includes both the cost of allocating assets to tasks and the penalty incurred due to the fact that a task is not executed either because an asset has not been allocated to it, or because assets were allocated but they were not sufficient or able to execute the task.
- Whenever possible all decisions are made separately for all the assets; i.e. the assets are allocated independently of each other; this would allow for distributed decision making.
- The algorithm for making the decision should, in all cases, be fast so that it can be used in real-time execution constrained environments. This implies that the decision algorithm cannot be based on enumerating all possible solutions, computing the cost for each solution, and then selecting the one that has the least cost among all of the enumerated solutions.

The paper is organised as follows: firstly, we formulate the described problem in rigorous mathematical terms before discussing other related problems in section 3. Then, we describe the random neural network, as well as the associated solution approach. In section 5, we evaluate the proposed method for the solution of the described problem by comparing its performance to the optimal as well as to the performance of a greedy approach. Finally, we conclude and discuss directions for future work.

2. THE COST FUNCTION

Let us denote the size of the set of tasks and assets as $|T|$ and $|W|$, respectively. Since we would like all tasks to be executed as a result of an assignment, we are interested in the case where the number of assets is at least as large as the number of tasks. However, we will not assume any specific relationship between $|T|$ and $|W|$, nor that all assets need to be assigned to some task.

Since the cost of using asset w for task t is $C(w,t)$, while the penalty of not carrying out the task t is $K(t)$, the overall cost function that we try to minimise can be written as:

$$\begin{aligned} \min C &= \sum_{t\in T}\sum_{w\in W} C(w,t)p(w,t) \\ &\quad + \sum_{t\in T} K(t) \prod_{w\in W} \{1-(1-q(w,t))p(w,t)\} \end{aligned} \tag{2}$$

where the first term is the average cost of the assets that are used, and the second term is the cumulative average cost of not successfully executing each of the tasks. C includes the fact that multiple assets may be assigned to a given task. It also implicitly assumes that the decision to assign some asset to some task can be made independently of the decision to assign another asset to the same task. This also embodies the premise that we are interested in distributed or uncoordinated decision making, where each asset can be assigned without coordination among distinct local decision elements. The cost also includes the important assumption that the allocation of different assets to the same task t has a cumulative independent effect as to the overall success, expressed by the product of the expected failure probabilities, $\{1-(1-q(w,t))p(w,t)\}$. This is reasonable to assume since one asset suffices to execute a task while combining the assets does not increase the probability of success.

The optimisation problem is then to choose the $\{p(w,t)\}$ for all $(w,t) \in W \times T$ such that (2) is minimised subject to the constraint (1). Since $p(w,t)$ is the probability of assigning asset w to task t, we can also obtain a deterministic formulation where an asset is allocated or not to a task, i.e. $p(w,t) \in \{0,1\}$, and in this case the optimisation problem becomes:

$$\begin{aligned}
\min C &= \sum_{t\in T}\sum_{w\in W} C(w,t)p(w,t) \qquad (3)\\
&\quad + \sum_{t\in T} K(t) \prod_{w\in W} \{1-(1-q(w,t))p(w,t)\}\\
s.t. &\quad \sum_{t\in T} p(w,t) \le 1, w \in W\\
&\quad p(w,t) \in \{0,1\},\ w\in W,\ t\in T
\end{aligned}$$

In the deterministic case, the problem can also be written with decision variables in the exponents of the $q(w,t)$ as:

$$\begin{aligned}
\min C &= \sum_{t\in T}\sum_{w\in W} C(w,t)p(w,t) \qquad (4)\\
&\quad + \sum_{t\in T} K(t) \prod_{w\in W} q(w,t)^{p(w,t)}\\
s.t. &\quad \sum_{t\in T} p(w,t) \le 1, w \in W\\
&\quad p(w,t) \in \{0,1\},\ w\in W,\ t\in T
\end{aligned}$$

In formulation (4), each of the product terms corresponds to the total expected failure probability of executing a particular task, while the first constraint represents the fact that we can assign one particular asset to at most one task. As a result, the total number of assigned assets can be less than $|W|$. Furthermore, there is no restriction on the number of assets that can be assigned to one task; it is therefore plausible to assign all or no assets to a particular task.

Since, in principle, it is possible to enumerate all possible allocations of assets to tasks, one can find the optimal or minimum cost solution just by enumeration. However, in addition to being of high computational cost, an enumerative solution to the optimisation problem would result in a fully centralised decision. On the other hand, if one attempts to allocate the assets separately or individually, there is a chance that the decision will be far from optimal since it can result in an over or under allocation of assets to tasks.

3. RELATED PROBLEMS

In our formulations, the decision variables appear either in the exponent of parameters or as a product and hence the examined problem is a nonlinear combinatorial optimisation problem which belongs to the general class of nonlinear assignment problems [3]. A related problem with a product of terms that also have the decision variables in their exponent is the Weapon Target Assignment (WTA) problem. In the WTA problem, we have a set of weapons W and a set of targets T and the goal is to find an optimal allocation of the weapons to the targets so that the expected damage on the targets is maximised or equivalently the expected leakage of the targets is minimised. It is assumed that weapon w has a probability of successfully intercepting target t equal to $(1-q(w,t))$, while each target has a cost equal to $K(t)$. Additionally, it is required that all weapons are assigned to some target. Therefore, the mathematical formulation of the WTA problem is as follows:

$$\begin{aligned}
\min C_{WTA} &= \sum_{t\in T} K(t) \prod_{w\in W} q(w,t)^{p(w,t)} \qquad (5)\\
s.t. &\quad \sum_{t\in T} p(w,t) = 1, \quad w \in W\\
&\quad p(w,t) \in \{0,1\}, \quad w\in W,\ t\in T
\end{aligned}$$

There are two main differences between formulations (4) and (5). The first is that in (4) each asset-task pair has an associated cost $C(w,t)$, while in (5) the weapons carry no cost. The second is that in our formulation not all assets need to be assigned to tasks. This stems from the fact that each asset-task pair has an associated cost and hence a particular assignment may not be beneficial.

In the general case, the WTA-problem is NP-complete [4] and hence exact algorithms have been proposed mostly for solving special cases of the problem. One such special case is when the probabilities $q(w,t)$ are independent of the weapon, i.e. $(q(w,t) = q(t),\ \forall w)$, while a second one is when we want to assign at most one weapon to each target. The first case can be solved either using the Maximum Marginal Return (MMR) algorithm [5, 6], where the weapons are assigned in a greedy fashion to the targets, resulting in the greatest decrease of the cost function, or using a local search algorithm to identify and swap any weapon-target pairs that reduce the overall cost [7]. The second special case results in a linear assignment problem that can be efficiently solved using a network flow algorithm [8].

Exact solutions of the general WTA problem have been recently discussed in [9] where several lower bounding schemes based on general network flow approximations are developed with a branch and bound algorithm that achieves the exact solution of medium sized problems (80 weapons and 80 targets). However, the time required for the exact solution of the general WTA problem is very large and much research has focused on developing heuristic algorithms based on metaheuristic techniques such as Hopfield neural networks [10], ant colony optimisation [11, 12], genetic algorithms [13] and very large scale neighbourhoods [9].

Apart from the WTA problem, other problems related to our formulation include assignment problems where the objective function involves the product of two or more variables such as the quadratic and biquadratic assignment problems [14]. However, each task is associated to only one asset contrary to formulation (4), where more than one assets can be assigned to one task. Such problems without this particular constraint are called semi-assignment problems, of which the most widely studied nonlinear problem is the quadratic semi-assignment problem (QSAP) [15] defined as:

$$\begin{aligned}\min C_{QSAP} &= \sum_{t\in T}\sum_{w\in W} C(w,t)p(w,t) \quad (6)\\ &+ \sum_{t\in T}\sum_{t'\in T}\sum_{w'\in W}\sum_{w\in W} b_{wt,w't'}p(w,t)p(w',t')\\ s.t. & \quad \sum_{t\in T} p(w,t) = 1,\ \forall w\\ & \quad p(w,t)\in\{0,1\} \quad w\in W,\ t\in T\end{aligned}$$

where $b_{wt,w't'}$ is the cost of assignment pairs (w,t) and (w',t'). Practical applications of QSAP include clustering and partitioning in distributed computing [16, 17] and scheduling [18, 19]. Note that in this problem all assets must be assigned to tasks and the products in the cost function involve only two decision variables, contrary to our formulations where we have $|W|$ product terms. Nevertheless, as in our formulation (4), the cost function in (6) has also a linear term associated with the cost of the assets. In the general case, QSAP is NP-hard [20] and in practice optimal solutions cannot be obtained even for small cases [21]. As a result, exact algorithms have been developed only for special cases of QSAP that result in algorithms that have polynomial time complexity [22, 23].

4. A DISTRIBUTED RNN BASED HEURISTIC

In this section we will develop and then evaluate a Random Neural Network (RNN) [24, 25] related formulation of the asset-to-task assignment problem (4). The solution is an algorithm that uses a RNN whose parameters, including the weights associated with the connections between neurons, are selected directly by from the parameters of the optimisation problem. The resulting RNN is then solved numerically, and a sequence of subsequently smaller RNNs are then solved. Each subsequent network is constructed by appropriately reducing the previous network, after selecting the most excited neuron. A similar approach was previously used successfully in other optimisation problems [26, 27]. This approach does not make use of a learning or adaptation phase as in [28] and is therefore computationally fast. Tests with a number of instances of the asset assignment problem show that this heuristic yields results which are very close to the known optima, outperforming a maximum marginal return greedy heuristic. Before we describe the proposed neural network approach for the solution of problem (4), we will introduce the mathematical model and the most important properties of the RNN.

4.1. The random neural network model

The RNN is an open recurrent neural network inspired by the spiking behaviour of natural mammalian neuronal networks [29] and using paradigms from queueing theory [30]. The network is composed of N fully connected neurons. Each neuron's internal state is represented by a non-negative integer, its potential. Each neuron can receive positive and negative unit amplitude signals (spikes) either from other neurons or from the outside world. Positive signals have an excitatory effect in the sense that they increase the signal potential of the receiving neuron by one unit. Negative arriving signals have an inhibitory effect reducing the potential of the receiving neuron by one unit, if the receiving neuron's potential is positive, while if the potential is zero an inhibitory signal has no effect on the receiving neuron. Also, we assume that positive and negative signals can arrive to neuron i from the outside world according to independent Poisson streams of rates Λ_i and λ_i respectively. A non-negative integer $k_i(\tau)$ represents the signal potential of neuron i at time τ. Neuron i is said to be excited when $k_i(\tau) > 0$, and it is quiescent when $k_i(\tau) = 0$. The state of the network is then described by the vector of potentials at time τ, $\mathbf{k}(\tau) = [k_1(\tau), \ldots, k_N(\tau)]$. The excitation probability of the neuron is defined as $Q_i(\tau) = Pr[k_i(\tau) > 0]$, and the probability distribution of the state at time τ is defined as $\pi(\mathbf{k}, \tau) = Pr[k_1(\tau) = k_1, \ldots, k_N(\tau) = k_N]$ where $\mathbf{k} = [k_1, \ldots, k_N]$.

If neuron i is excited, it can fire after a random and exponentially distributed delay of parameter (or firing rate) r_i; each time a neuron fires its potential is reduced by one. The spike that is sent out by the neuron when it fires can either reach neuron j as an excitatory spike with probability $p^+(i,j)$ or as an inhibitory (negative) spike or signal with probability $p^-(i,j)$, or it may depart from the network with probability $d(i)$. These probabilities sum up to one:

$$\sum_{j=1}^{N}\left[p^+(i,j) + p^-(i,j)\right] + d(i) = 1, \quad \forall i \qquad (7)$$

As a consequence, when neuron i is excited, it will fire excitatory and inhibitory spikes at random to neuron j with rates:

$$\omega^+(i,j) = r_i p^+(i,j) \ \geq\ 0 \qquad (8)$$
$$\omega^-(i,j) = r_i p^-(i,j) \ \geq\ 0 \qquad (9)$$

TABLE 1. List of RNN symbols

Notation	Definition
$k_i(\tau)$	Potential of neuron i at time τ
$Q_i(\tau)$	Probability neuron i is excited at time τ
Λ_i $[\lambda_i]$	External arrival rate of positive [negative] signals to neuron i
$\lambda^+(i)$ $[\lambda^-(i)]$	Average arrival rate of positive [negative] signals to neuron i
$p^+(i,j)$ $[p^-(i,j)]$	Probability neuron j receives a positive [negative] signal from firing neuron i
$\omega^+(i,j)$ $[\omega^-(i,j)]$	Rate of positive [negative] signals to neuron j from firing neuron i
$d(i)$	Probability a signal from firing neuron i departs from the network
r_i	Firing rate of neuron i

Combining Eqs. (7), (8) and (9) we have:

$$r_i = (1 - d(i))^{-1} \sum_{j=1}^{N} [\omega^+(i,j) + \omega^-(i,j)] \tag{10}$$

The main symbols that we use for the model are summarised in Table 1, to facilitate the reader's understanding of the paper.

The long-term behaviour of the network is described by the steady state probability distribution of the neuron potentials:

$$\pi(\mathbf{k}) = \lim_{\tau\to\infty} \pi(\mathbf{k},\tau)$$

The values of the stationary excitation probabilities $Q_i = \lim_{\tau\to\infty} Q_i(\tau) \quad i = 1,...,N$ and the stationary probability distribution $\pi(\mathbf{k})$ are obtained using the following Theorem.

Theorem [29]: *Let the total arrival rates of positive and negative signals $\lambda^+(i)$ and $\lambda^-(i)$, $i = 1,...N$ be given by the following system of equations*

$$\lambda^+(i) = \Lambda_i + \sum_{j=1}^{N} Q_j \omega^+(j,i) \tag{11}$$

$$\lambda^-(i) = \lambda_i + \sum_{j=1}^{N} Q_j \omega^-(j,i) \tag{12}$$

where

$$Q_i = \begin{cases} \frac{\lambda^+(i)}{r_i + \lambda^-(i)}, & \text{if } \lambda^+(i) < r_i + \lambda^-(i) \\ 1 & \text{if } \lambda^+(i) \geq r_i + \lambda^-(i) \end{cases} \tag{13}$$

If a unique non-negative solution $\{\lambda^-(i),\ \lambda^+(i)\}$ exists for the non-linear system of Eqs. (11)-(13) such that $Q_i < 1\ \forall i$, then:

$$\pi(\mathbf{k}) = \prod_{i=1}^{N} Q_i^{k_i} [1 - Q_i] \tag{14}$$

The theorem states that whenever a solution to the signal flow Eqs. (11)-(13) can be found such that $Q_i < 1, \forall i$, then the stationary joint probability distribution of the network has the simple product form (14) and can be expressed via the excitation probabilities of each neuron, Q_i. Product form implies independence of the neurons despite the fact that the neurons are coupled through the exchanged signals. It has also been proven that a solution to the nonlinear system of Eqs. (11)-(13) always exists and it is unique [31, 24].

The RNN model has been successfully exploited in many applications as a learning tool, for instance in pattern recognition [32] and image/video compression [33, 34]. It has also been highly successful as a modelling tool that can capture complex interactions between entities in various contexts such as queueing networks [35, 36], natural neuronal networks [25] and gene regulatory networks [37]. The RNN has also been successfully exploited for the solution of optimisation problems, including the travelling salesman problem [28], task assignment in distributed systems [38], ambulance allocation in emergency response [39]. For a comprehensive survey of the RNN and its applications see [40].

4.2. RNN parameter association approach

In the approach that we propose, each allocation decision (w,t) is represented by a neuron $N(w,t)$ of a RNN, so that $p(w,t)$ corresponds to the probability $Q_{(w,t)}$ that this particular neuron is excited. Thus the computational size of the problem to be considered will depend on $|W| \times |T|$ as indicated below. To specify the RNN which is used for the heuristic solution to the optimisation problem, we must specify the arrival rates of excitation and inhibition signals to each of the neurons $N(w,t)$, and the excitatory and inhibitory weights between neurons. These parameters are chosen as follows:

$$\Lambda_{(w,t)} = \max\{0, b(w,t)\}$$
$$\lambda_{(w,t)} = \max\{0, -b(w,t)\}$$

where

$$b(w,t) = K(t)(1 - q(w,t)) - C(w,t)$$

so that $b(w,t)$ represents the net expected reduction in the objective function when asset w is allocated to task t, since $K(t)(1 - q(w,t))$ is the expected reduction in the cost of task t if this allocation is made and $C(w,t)$ is the cost of allocating this asset to the given task. To discourage the allocation of distinct assets to the same task, we also set the inhibitory weights:

$$\omega^-(w,t;w',t) = \max\{0, b(w,t)\},\ \text{if } w \neq w'$$

Similarly we wish to avoid that the same asset be assigned to distinct tasks :

$$\omega^-(w,t;w,t') = \max\{0, b(w,t)\},\ \text{if } t \neq t'$$

To keep matters as simple as possible, we choose not to reinforce or weaken any of the assignments other than what is already done via the incoming excitatory signals, so that we choose $\omega^+(w,t;w,t') = 0$ and $\omega^-(w,t;w',t') = 0$ for all other w, w' and t, t', and we end with:

$$r_{(w,t)} = \sum_{w',t'} \omega^-(w,t;w',t') \quad (15)$$

Based on the above parameters the excitation level of each neuron satisfies:

$$\begin{aligned} Q_{(w,t)} &= \Lambda_{(w,t)}/[\lambda_{(w,t)} + r_{(w,t)} \\ &\quad + \sum_{w' \neq w} Q_{(w',t)}\omega^-(w',t;w,t) \\ &\quad + \sum_{t' \neq t} Q_{(w,t')}\omega^-(w,t';w,t)] \end{aligned} \quad (16)$$

and the system of equations (16) is then solved iteratively in the following manner to obtain the assignments of assets to tasks:

(i) Initialisation: $W_{rem} \leftarrow W$, $S \leftarrow \emptyset$ and $K_{cur}(t) \leftarrow K(t),\ t \in T$.
(ii) Compute the RNN parameters based on $K_{cur}(t), \forall t$ and construct the neural network for $w \in W_{rem}$ and $t \in T$
(iii) Solve the system of Equations (16) for $w \in W$ and $t \in T$ to obtain $Q_{(w,t)}$.
(iv) Select asset-task pair (w^*, t^*) that corresponds to the neuron with the largest positive $Q_{(w,t)}$; if all $Q_{(w,t)} = 0$, $w \in W_{rem}$ and $t \in T$ then stop: there is no assignment that reduces the cost of the objective function
(v) Set $S \leftarrow S \cup (w^*, t^*)$
(vi) Set $W_{rem} \leftarrow W_{rem} \backslash \{w^*\}$
(vii) Set $K_{cur}(t^*) \leftarrow K_{cur}(t^*)q(w^*, t^*)$
(viii) If $W_{rem} \neq \emptyset$ go to step (ii) otherwise stop: all assets has been assigned

In the algorithm, W_{rem} represents the assets remaining to be assigned, while S is the solution set where the assigned asset-task pairs are stored. $K_{cur}(t)$ is the current expected cost of task t given any previous assignments. Note that using this algorithm, the assignment of some asset w^* to a task t^* always results in reducing the cost of the objective function; otherwise if $b(w,t) < 0$ then $Q_{(w,t)} = \Lambda_{(w,t)} = 0$ and the neuron is not selected.

One interesting feature of the algorithm is that once all assets have acquired the parameters of the problem, then they can decide in a decentralised manner and arrive at a non-conflicting decision even though their actions are not coordinated. This is possible because the RNN algorithm is not stochastic, while the solution to the RNN signal-flow equations is unique. Knowledge of the problem parameters can be accomplished in an initial phase prior to decision making, in which each asset exchange with other assets any information associated with it.

4.3. Algorithm complexity

According to an iterative algorithm proposed in [41], the solution of the system of equations (11)-(13) is of complexity $O(N^2)$ per iteration while convergence is achieved at a rate better than a geometric sequence, so that the overall complexity is $O(NI_{RNN}N^2)$, where NI_{RNN} is the total number of iterations required. Note that each iteration involves the calculation of new Q_i values based on the Q_i's obtained at the previous iteration.

Because the system of equations (16) involves $N = |W||T|$ neurons, its solution is of computational complexity $O(NI_{RNN}|W|^2|T|^2)$. To improve the computational complexity for the solution of (16), we can take advantage of the special structure of the system to reduce the complexity of each iteration. Note that Eq. (16) can be written as:

$$\begin{aligned} Q_{(w,t)} &= \Lambda_{(w,t)}/[\lambda_{(w,t)} + r_{(w,t)} \\ &\quad + \sum_{w' \neq w} Q_{(w',t)} \max\{0, b(w',t)\} \\ &\quad + \sum_{t' \neq t} Q_{(w,t')} \max\{0, b(w,t')\}] \end{aligned} \quad (17)$$

The calculation of $Q_{(w,t)}$ in equation (17) for each neuron, is of complexity $O(|W| + |T|)$ so that the complexity of one iteration is $O(|W||T|(|W| + |T|))$.

We can further reduce the complexity of one iteration by rewriting equation (17) as:

$$\begin{aligned} Q_{(w,t)} &= \Lambda_{(w,t)}/[\lambda_{(w,t)} + r_{(w,t)} + \theta_1(t) \\ &\quad + \theta_2(w) - 2Q_{(w,t)} \max\{0, b(w,t)\}] \end{aligned} \quad (18)$$

where

$$\theta_1(t) = \sum_{w' \in W} Q_{(w',t)} \max\{0, b(w',t)\},\ \forall t$$

$$\theta_2(w) = \sum_{t' \in T} Q_{(w,t')} \max\{0, b(w,t')\},\ \forall w$$

Based on equation (18), the computation of each $Q_{(w,t)}$ term requires $O(1)$ operations so that the computation of all $Q_{(w,t)}$ values is of complexity $O(|W| \cdot |T|)$. Moreover, the terms $\theta_1(t), t \in T$ and $\theta_2(w), w \in W$ are also of complexity $O(|W| \cdot |T|)$ so that the complexity of solving the nonlinear system (16) is $O(NI_{RNN} \cdot |W| \cdot |T|)$.

Because each iteration of our algorithm is dominated by the solution of the signal flow equations of RNN and we need to perform at most $|W|$ iterations, the complexity of our RNN algorithm is $O(NI_{RNN} \cdot |W|^2 \cdot |T|)$.

5. EVALUATION

The effectiveness of the proposed RNN algorithm is evaluated with respect to two generated data families.

$\|W\|$	$\|T\|$	RNN σ_{opt}^{mean}	RNN σ_{opt}^{std}	MMR σ_{opt}^{mean}	MMR σ_{opt}^{std}
4	5	4.7034	7.069	6.0267	5.9397
8	5	0.7771	2.4631	3.5169	4.945
12	5	0.6904	1.6169	1.8547	3.4373
4	8	0.5988	2.0259	2.4002	2.9591
8	8	3.8079	4.9211	5.9078	4.7767
12	8	0.6245	1.1415	3.3973	3.7429
9	3	0.9027	2.5532	1.1752	3.1257
9	5	0.7643	1.7723	2.5461	4.0173
9	8	2.3081	3.7447	5.0732	4.2584
9	12	1.287	2.4339	4.8481	3.4261

TABLE 2. Mean and standard deviation of the relative percentage deviation from optimality for data family 1

$\|W\|$	$\|T\|$	RNN σ_{opt}^{mean}	RNN σ_{opt}^{std}	MMR σ_{opt}^{mean}	MMR σ_{opt}^{std}
4	5	2.0829	4.727	3.1862	3.0674
8	5	1.6644	2.8745	2.2584	2.5079
12	5	2.5007	3.5051	2.4829	3.1049
4	8	0.0167	0.2327	0.8051	1.1688
8	8	1.9708	3.1181	3.4331	2.5877
12	8	1.2554	1.9664	2.2629	1.9486
9	3	2.9186	4.574	2.5868	3.9413
9	5	1.5003	2.2913	1.8709	2.2523
9	8	1.8154	2.495	3.1534	2.4611
9	12	0.2507	1.0812	2.3452	2.0432

TABLE 3. Mean and standard deviation of the relative percentage deviation from optimality for data family 2

In data family 1, the parameters of the problem are all independently generated, while in data family 2 there is positive correlation between the cost of an asset assignment and its associated execution success probabilities, so that "better" assets are more expensive. In both data families, parameters $K(t)$ for each task in T are generated from the uniform distribution in the interval [10,200]. In data family 1 the other two problem parameters also follow the uniform distribution. The cost of assignment $C(w)$ for each asset in W, is taken to be independent from its assigned task and belongs in the interval $[5, 30]$. The execution failure probabilities $q(w, t)$ are randomly generated in the interval [0.05,0.4]. In data family 2, the asset execution failure probabilities are taken to be independent from the tasks, i.e. $q(w, t) = q(w), \forall t$, while the associated asset costs $C(w)$ are drawn from the normal distribution with mean $\overline{C}(w)$ and variance $0.1\overline{C}(w)$. The parameter $\overline{C}(w)$ is calculated from the linear equation (19) that connects points $(q_{max}, \overline{C}_{min})$ and $(q_{min}, \overline{C}_{max})$, where $q_{min} = 0.05$, $q_{max} = 0.4$, $\overline{C}_{min} = 5$ and $\overline{C}_{max} = 30$.

$$\overline{C}(w) = \frac{(\overline{C}_{max} - \overline{C}_{min})}{(q_{min} - q_{max})}(q(w) - q_{max}) + \overline{C}_{min} \quad (19)$$

The proposed algorithm is compared against a maximum marginal return (MMR) greedy heuristic in both small and large size problems for various asset-task pairs $\langle |W|, |T| \rangle$. In the MMR heuristic, an iterative procedure is followed where in each iteration we select the assignment corresponding to the maximum reduction in the cost of the objective function, represented by the term $\max\{0, b(w, t)\}$. In the small-sized problems considered, the solutions obtained from the two algorithms are compared against the optimal ones obtained by enumeration, while in large-sized problems the results of the two algorithms are directly compared because enumeration is infeasible.

In the small-sized problems, the performance criterion used for comparison between the different approaches and the optimal, is the relative percentage deviation from optimality, σ_{opt}, defined as:

$$\sigma_{opt} = \frac{C_{alg,i} - C_{opt,i}}{C_{opt,i}} \times 100\% \quad (20)$$

where $C_{alg,i}$ is the cost obtained from the solution of problem instance i using a particular algorithm and $C_{opt,i}$ is the corresponding optimal cost. The results are summarised in Tables 2 and 3 for the two data families, where σ_{opt}^{mean} and σ_{opt}^{std} correspond to the mean and standard deviation values of σ_{opt} derived from the solution of 300 problem instances for each asset-task pair. For data family 1, the RNN algorithm clearly outperforms the MMR approach for all asset-task pairs by up to 3.5% in terms of mean performance achieving in all cases results within 5% of optimality on average. In terms of standard deviation, the RNN approach is also better in most of the cases. In data family 2, the performance of the RNN approach is also better

$\|W\|$	$\|T\|$	Data Family 1 σ_{RNN}^{mean}	Data Family 2 σ_{RNN}^{mean}
10	5	1.8569	0.402
10	10	4.1559	1.7296
10	20	2.2053	0.0513
20	10	3.7143	0.4631
20	20	5.5449	2.309
20	40	2.0747	0.0537
20	80	0.476	0.0049
40	10	1.5104	-0.0611
40	20	3.9566	0.7944
40	40	6.0592	2.6939
40	80	1.9293	0.053
40	120	0.6976	0.0126
80	20	2.1806	-0.3092
80	40	4.2456	0.6978
80	80	5.116	2.9449
80	160	1.4703	0.0531
100	50	4.1462	0.6761
100	100	4.8068	2.9795
100	200	1.337	0.0534
200	100	3.4757	0.7187
200	200	3.7179	3.1527

TABLE 4. Average percentage deviation of the MMR approach from the RNN approach for large-sized problems

achieving results within 3% deviation from optimality on average in all cases. The mean performance of the MMR approach is only better when the assets to tasks ratio is large (pairs $\langle 12, 5\rangle$ and $\langle 9, 3\rangle$), while it also has 0.2% less standard deviation of σ_{opt} on average.

We have performed a second set of experiments for large problem instances with up to 200 assets and 200 tasks. Due to the large size of the problems optimal solution via enumeration is not possible; hence, we have compared the algorithms directly to each other. The evaluation criterion used in this case is the relative percentage deviation of the MMR from the RNN approach, σ_{RNN}.

$$\sigma_{RNN} = \frac{C_{Greedy,i} - C_{RNN,i}}{C_{RNN,i}} \times 100\% \qquad (21)$$

Note that σ_{RNN} is positive when the solution obtained from the RNN solution is better than the MMR one and negative otherwise.

Table 4 represents the σ_{RNN}^{mean} for both data families used in the first set of experiments. It is clear that the RNN approach outperforms the MMR approach for almost all asset-task pairs considered. In data family 1, where all parameters of the problems are independently generated, the performance of the RNN approach is in all cases better than the MMR one, achieving better performance by more than 5% for several cases. In data family 2, the RNN approach is still better, but the improved performance is not as good as the first data family. In fact, for $\langle 40, 10\rangle$ and $\langle 80, 20\rangle$, where the number of assets is quadruple the number of tasks, the MMR approach is on average slightly better than the RNN one.

Overall, an interesting point that arises from the results is the relationship between the observed performance and the ratio $|W|/|T|$. The best performing cases for the RNN approach are those with ratio $|W|/|T| = 1$ for both data families, while as the $|W|/|T|$ ratio diverges from 1, σ_{RNN}^{mean} is reduced. In addition, the observed performance is roughly constant for specific $|W|/|T|$ ratio and does not significantly depend on the size of the problem.

For the second set of experiments we have also examined the execution time of the two algorithms as presented on table 5. Note that the algorithms were implemented in Matlab, and the experiments were conducted on a Pentium IV 3.6 GHz PC with 1 GB of RAM. Notice also that because the execution times observed for the two data families were similar, we have only presented the execution times for data family 2 which were slightly larger. As expected, the MMR approach is faster than the RNN one because it is based on greedily selecting the asset with the MMR. However, the RNN approach is quite fast as well since it can solve large problems of up to 200 assets and 200 tasks in one second.

6. CONCLUSIONS AND FURTHER WORK

The results that we have obtained confirm the usefulness of a simple RNN based heuristic to the asset allocation problem that has been considered, which aims at efficiently solving problem (4). We have shown that the RNN can obtain solutions within 5% of optimality for small problems on average, while it has better performance compared to a greedy maximum marginal return approach for both small and large size problems. Moreover, the observation that in large problems the performance primarily depends on the ratio of the number of assets to the number of tasks, indicates that the performance of our approach will be as good as the one observed for small problems with respect to optimality. Furthermore, although the RNN approach is slower than the greedy approach, it is still fast and can solve large problems of up to 200 assets and 200 tasks in one second on a conventional machine.

Now that these results have been obtained, the problem we consider, as well as the RNN based solution, can be extended in various directions. One such direction involves the investigation of real-world problems such as the ones mentioned in the introduction where assignments have an uncertain result, especially when the problem requires a fast(even real-time) and close to optimal solution, as well as decentralised decision making.

Using the RNN based solution we can also solve problems closely related to the problem examined in this paper. One such extension is the dynamic assets to tasks assignment problem, where new tasks are

$\lvert W\rvert$	$\lvert T\rvert$	Time RNN(s)	Time MMR(s)	Ratio RNN/MMR
10	5	0.0023	0.0007	3.2394
10	10	0.0025	0.0007	3.3784
10	20	0.0027	0.0009	3.1765
20	10	0.0036	0.0013	2.7692
20	20	0.0057	0.0016	3.5625
20	40	0.0069	0.0018	3.7912
20	80	0.008	0.0024	3.2787
40	10	0.0049	0.0015	3.3793
40	20	0.0098	0.0026	3.7838
40	40	0.0173	0.0043	4.0139
40	80	0.0264	0.0067	3.9699
40	120	0.0369	0.0084	4.3981
80	20	0.0148	0.0045	3.2599
80	40	0.0447	0.0100	4.4835
80	80	0.0875	0.0193	4.5455
80	160	0.1617	0.0320	5.0500
100	50	0.0811	0.0160	5.0783
100	100	0.1658	0.0319	5.1926
100	200	0.2844	0.0575	4.9504
200	100	0.4823	0.0956	5.0450
200	200	1.0564	0.2345	4.5041

TABLE 5. Execution times for the RNN and MMR approaches for data family 2

generated continually and we need to allocate the assets in a way that we minimise the total cost over time. An additional interesting problem involves the allocation of assets in stages. First some assets can be allocated to tasks, and then the resulting outcome in terms of the set of tasks which are actually executed is observed. The remaining assets can then be used in the next stage, based on the work that was not completed in the first one. This would correspond to an approach where assets are maintained in reserve and re-allocated during the course of the process that one is studying.

We expect that some of these and other generalisations will be examined using an RNN based approach in future work.

FUNDING

This research was undertaken as part of the ALADDIN (Autonomous Learning Agents for Decentralised Data and Information Networks) project and is jointly funded by a BAE Systems and EPSRC strategic partnership (EP/C548051/1).

REFERENCES

[1] Pentico, D. W. (2007) Assignment problems: A golden anniversary survey. *European Journal of Operational Research*, **176**, 774 – 793.

[2] Burkard, R. E. and Cela, E. (1999) Linear assignment problems and extensions. In Pardalos, P. and Du, D.-Z. (eds.), *Handbook of Combinatorial Optimization - Supplement Volume A*, pp. 75–149. Kluwer Academic Publishers.

[3] Pardalos, P. and Pitsoulis, L. S. (2000) *Nonlinear Assignment Problems: Algorithms and Applications.* Kluwer Academic Publishers, Dordrecht, Netherlands.

[4] Lloyd, S. P. and Witsenhausen, H. S. (1986) Weapons allocation is NP-complete. *Proceedings of the 1986 Summer Computer Simulation Conference, Reno, Nevada, USA, 28-30 July*, pp. 1054–1058. Society for Computer Simulation.

[5] den Broeder, G. G., Ellison, R. E., and Emerling, L. (1959) On Optimum Target Assignments. *OPERATIONS RESEARCH*, **7**, 322–326.

[6] Kolitz, S. E. (1988) Analysis of a maximum marginal return assignment algorithm. *Proceedings of the 27th IEEE Conference on Decision and Control, Austin, Texas, USA, 7-9 December*, pp. 2431–2436. IEEE, New York, NY.

[7] Hosein, P. A. (1989) A Class of Dynamic Nonlinear Resource Allocation Problems. PhD thesis Massachusetts Institute of Technology, Cambridge, MA.,.

[8] Chang, S.-C., James, R. M., and Shaw, J. J. (1987) Assignment algorithm for kinetic energy weapons in boost phase defence. *Proceedings of the 26th IEEE Conference on Decision and Control, Los Angeles, CA, 9-11 December*, pp. 1678–1683. IEEE, New York, NY.

[9] Ahuja, R. K., Kumar, A., Jha, K. C., and Orlin, J. B. (2007) Exact and Heuristic Algorithms for the Weapon-Target Assignment Problem. *OPERATIONS RESEARCH*, **55**, 1136–1146.

[10] Wacholder, E. (1989) A Neural Network-Based Optimization Algorithm for the Static Weapon-Target Assignment Problem. *INFORMS JOURNAL ON COMPUTING*, **1**, 232–246.

[11] Yanxia, W., Longjun, Q., Zhi, G., and Lifeng, M. (2008) Weapon target assignment problem satisfying expected damage probabilities based on ant colony

algorithm. *Journal of Systems Engineering and Electronics*, **19**, 939 – 944.

[12] Lee, Z.-J., Lee, C.-Y., and Su, S.-F. (2002) An immunity-based ant colony optimization algorithm for solving weapon-target assignment problem. *Applied Soft Computing*, **2**, 39 – 47.

[13] Lee, Z.-J., Su, S.-F., and Lee, C.-Y. (2003) Efficiently solving general weapon-target assignment problem by genetic algorithms with greedy eugenics. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, **33**, 113–121.

[14] Loiola, E. M., de Abreu, N. M. M., Boaventura-Netto, P. O., Hahn, P., and Querido, T. (2007) A survey for the quadratic assignment problem. *European Journal of Operational Research*, **176**, 657 – 690.

[15] Pitsoulis, L. S. (2009) Quadratic semi-assignment problem. In Floudas, C. A. and Pardalos, P. M. (eds.), *Encyclopedia of Optimization*, pp. 3170–3171. Springer.

[16] Hansen, P. and Lih, K. W. (1992) Improved algorithms for partitioning problems in parallel, pipelined, and distributed computing. *IEEE Transactions on Computers*, **41**, 769–771.

[17] Milis, I. Z. and Magirou, V. F. (1995) A lagrangian relaxation algorithm for sparse quadratic assignment problems. *Operations Research Letters*, **17**, 69 – 76.

[18] Chretienne, P. (1989) A polynomial algorithm to optimally schedule tasks on a virtual distributed system under tree-like precedence constraints. *European Journal of Operational Research*, **43**, 225 – 230.

[19] Bullnheimer, B. (1998) An examination scheduling model to maximize students' study time. *Selected papers from the Second International Conference on Practice and Theory of Automated Timetabling II, Toronto, Canada, 20-22 August*, London, UK, pp. 78–91. Springer-Verlag.

[20] Sahni, S. and Gonzalez, T. (1976) P-Complete Approximation Problems. *Journal of the ACM*, **23**, 555–565.

[21] Magirou, V. F. and Milis, J. Z. (1989) An algorithm for the multiprocessor assignment problem. *Operations research letters*, **8**, 351–356.

[22] Malucelli, F. (1996) A polynomially solvable class of quadratic semi-assignment problems. *European Journal of Operational Research*, **91**, 619 – 622.

[23] Malucelli, F. and Pretolani, D. (1995) Lower bounds for the quadratic semi-assignment problem. *European Journal of Operational Research*, **83**, 365 – 375.

[24] Gelenbe, E. (1993) Learning in the recurrent random network. *Neural Computation*, **5**, 154–164.

[25] Gelenbe, E. and Timotheou, S. (2008) Synchronised Interactions in Spiked Neuronal Networks. *The Computer Journal*, **51**, 723–730.

[26] Gelenbe, E. and Batty, F. (1992) Minimum Graph Covering with the Random Neural Network Model. In Gelenbe, E. (ed.), *NEURAL NETWORKS: Advances and Applications, 2*, pp. 215–222. Elsevier Science Publishers B.V.

[27] Gelenbe, E., Ghanwani, A., and Srinivasan, V. (1997) Improved neural heuristics for multicast routing. *IEEE Journal of Selected Areas of Communications*, **15**, 147–155.

[28] Gelenbe, E., Koubi, V., and Pekergin, F. (1994) Dynamical random neural network approach to the traveling salesman problem. *ELEKTRIK*, **2**, 1–10.

[29] Gelenbe, E. (1989) Random Neural Networks with Negative and Positive Signals and Product Form Solution. *Neural Computation*, **1**, 502–510.

[30] Fourneau, J.-M., Gelenbe, E., and Suros, R. (1996) G-networks with multiple classes of positive and negative customers. *Theoretical Computer Science*, **155**, 141–156.

[31] Gelenbe, E. (1990) Stability of the random neural network. *Neural Computation*, **2**, 239–247.

[32] Gelenbe, E. and Kocak, T. (2000) Area-based results for mine detection. *IEEE Trans. on Geoscience and Remote Sensing*, **38**, 1–14.

[33] Gelenbe, E., Cramer, C., Sungur, M., and Gelenbe, P. (1996) Traffic and video quality in adaptive neural compression. *Multimedia Systems*, **4**, 357–369.

[34] Gelenbe, E. (2004) Cognitive packet network. *US Patent*, **6804201**.

[35] Gelenbe, E. (Sep. 1991) Product-Form Queueing Networks with Negative and Positive Customers. *Journal of Applied Probability*, **28**, 656–663.

[36] Artalejo, J. R. (2000) G-networks: A versatile approach for work removal in queueing networks. *European Journal of Operational Research*, **126**, 233–249.

[37] Gelenbe, E. (2007) Steady-state solution of probabilistic gene regulatory networks. *Physical Review E*, **76**, 031903.

[38] Aguilar, J. and Gelenbe, E. (1997) Task assignment and transaction clustering heuristics for distributed systems. *Information Sciences – Informatics and Computer Science*, **97**, 199–221.

[39] Gelenbe, E. and Timotheou, S. (2008) Random Neural Networks with Synchronised Interactions. *Neural Computation*, **20**, 2308–2324.

[40] Timotheou, S. (2009). The Random Neural Network: A Survey. The Computer Journal. doi:10.1093/comjnl/bxp032.

[41] Fourneau, J. (1991) Computing the Steady State Distribution of Networks with Positive and Negative Customers. *Proceedings of the 13-th IMACS World Congress on Computational and Applied Mathematics, Dublin, Ireland, July.* North-Holland, Amsterdam.