A random neural network approach to an assets to tasks assignment problem

Erol Gelenbe[a], Stelios Timotheou[a] and David Nicholson[b]

[a]Dept. of Electrical & Electronic Engineering, Imperial College, London SW7 2BT, UK;
[b] BAE Systems Ltd, Sowerby Building Filton, Bristol, BS34 7QW, UK

ABSTRACT

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 propose an approach based on the Random Neural Network (RNN) which is fast and of low polynomial complexity. The evaluation indicates that the proposed RNN approach comes at most within 10% of the cost obtained by the optimal solution in all cases.

Keywords: optimum asset assignment; distributed decisions; nonlinear combinatorial optimization; random neural network

1. INTRODUCTION

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 the 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 when it is assigned to it. 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. 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: $0 \leq p(w,t) \leq 1$ and $\sum_{t \in T} p(w,t) = \pi(w) \leq 1$, $w \in W$. Here $\pi(w) \leq 1$ represents the fact that we may decide not to assign the asset w to some task, or that the assignment, once it is decided, may actually fail to occur, or that the allocated asset may fail to function. If all the $p(w,t)$ are either 0 or 1 then we have a deterministic formulation. We also assume that after an asset is allocated to some task, it cannot be re-assigned again to some other task; this corresponds to cases where the assets are expendable or to real-time situations where, for the given time epoch considered, decisions are irrevocable.

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",
- T is a set of "emergencies" and W is the set of "ambulances" or emergency personnel,
- T is a set of "entities that need to communicate" and W is the set of "communication channels or frequencies", etc.

Further author information: (Send correspondence to E.G.)
E.G.: E-mail: e.gelenbe@imperial.ac.uk
S.T.: E-mail: stelios.timotheou05@imperial.ac.uk
D.N.: E-mail: David.Nicholson2@baesystems.com

We develop an approach for making such allocations 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. The cost function that we minimise is:

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

where the first term is the total cost of the assets that are used, and the second term is the cumulative expected cost of not successfully executing each of the tasks, and 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 failure probabilities, $\{1-(1-q(w,t))p(w,t)\}$. The problem is then to choose the $\{p(w,t)\}$ for all $(w,t) \in W \times T$ such that (1) is minimised. In the deterministic case $p(w,t) \in \{0,1\}$, the cost can also be written with decision variables in the exponents of the $q(w,t)$ as:

$$\begin{aligned} \min C \quad &= \quad \sum_{t\in T}\sum_{w\in W} C(w,t)p(w,t) + \sum_{t\in T} K(t) \prod_{w\in W} q(w,t)^{p(w,t)} \\ s.t. \quad & \quad \sum_{t\in T} p(w,t) \leq 1, w \in W \\ & \quad p(w,t) \in \{0,1\},\ w \in W,\ t \in T \end{aligned} \tag{2}$$

In (2), 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. Whenever possible all decisions should be made separately so that assets are allocated independently of each other allowing for distributed decision making. Also, the algorithm for making the decision should be fast so that it can be used in real-time execution constrained environments. This implies that it 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.

2. RELATED PROBLEMS

The problem under investigation belongs to the general class of nonlinear assignment problems[1]. A related problem with a product of terms having the decision variables in their exponent is the Weapon Target Assignment (WTA) problem. WTA is NP-complete[2] and hence exact algorithms have been proposed only for solving special cases of the problem. Solution to the general WTA problem has been achieved for medium size problems in[3], by combining lower bounding schemes based on general network flow approximations with a branch and bound algorithm. Because the time required for the exact solution of the general WTA problem is large, much research has focused on metaheuristic techniques such as Hopfield neural networks[4], ant colony optimisation[5,6], genetic algorithms[7] and very large scale neighbourhoods[3].

Quadratic and biquadratic assignment problems are also related to problem (2), as their objective function includes products of two or more variables[8]. However, in these problems not only each asset must be assigned once, but also each task must be associated to only one asset, contrary to our problem where more than one assets can be assigned to one task. Problems without this particular constraint are called semi-assignment problems, of which the most widely studied is the quadratic semi-assignment problem (QSAP)[9]. In the general case, QSAP is NP-hard[10] and in practice optimal solutions cannot be obtained even for small size problems[11]. As a result, exact algorithms of polynomial time complexity have been developed only for special cases of QSAP[12,13].

3. A DISTRIBUTED RNN BASED HEURISTIC

We now develop a Random Neural Network (RNN)[14,15] based formulation of the asset-to-task assignment problem. The solution uses a RNN whose parameters are selected from the parameters of the optimisation problem. Similar approaches have been used successfully in other optimisation problems[16,17] without a learning phase[18], and they are therefore computationally fast.

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

3.1 The random neural network model

The RNN is an open recurrent neural network inspired by the spiking behaviour of natural mammalian neuronal networks[19] and using paradigms from queueing theory[20]. 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. Neuron i is said to be excited at time τ when its potential, $k_i(\tau)$, is positive and it is quiescent when $k_i(\tau) = 0$. Moreover, the excitation probability of the neuron is defined as $Q_i(\tau) = Pr[k_i(\tau) > 0]$. 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 $d(i)$, so that:

$$\sum_{j=1}^{N} \left[p^+(i,j) + p^-(i,j)\right] + d(i) = 1, \quad \forall i \tag{3}$$

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

$$\omega^+(i,j) = r_i p^+(i,j) \ \geq \ 0 \tag{4}$$

$$\omega^-(i,j) = r_i p^-(i,j) \ \geq \ 0 \tag{5}$$

Combining Eqs. (3), (4) and (5) yields:

$$r_i = (1 - d(i))^{-1} \sum_{j=1}^{N} [\omega^+(i,j) + \omega^-(i,j)] \tag{6}$$

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

The state of the network is described by the vector of potentials at time τ, $\mathbf{k}(\tau) = [k_1(\tau), \ ... \ , k_N(\tau)]$, while the probability distribution of the state at time τ is defined as $\pi(\mathbf{k}, \tau) = Pr[k_1(\tau) = k_1, \ ... \ , k_N(\tau) = k_N]$ where $\mathbf{k} = [k_1, \ ... \ , k_N]$.

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 are obtained from Theorem 1 in[19]. The total arrival rates of positive and negative

signals to each neuron $\lambda^+(i)$ *and* $\lambda^-(i)$, $i = 1, ...N$ are given by the equations:

$$\lambda^+(i) = \Lambda_i + \sum_{j=1}^{N} Q_j \omega^+(j,i) \tag{7}$$

$$\lambda^-(i) = \lambda_i + \sum_{j=1}^{N} Q_j \omega^-(j,i) \tag{8}$$

leading to:

$$Q_i = \min\{1, \lambda^+(i)/(r_i + \lambda^-(i))\} \tag{9}$$

The solution to the nonlinear system (7)-(9) always exists and is unique[14, 21]. The RNN model has been successful in the solution of optimisation problems [18, 22, 23], in modelling complex interactions between entities in queueing networks[24], natural neuronal networks[15] and gene regulatory networks[25], as well as in learning[26–28]. A survey of RNN can be found in[29].

3.2 The RNN solution 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. The external positive and negative arrival rates 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:

$$r_{(w,t)} = \sum_{w',t'} \omega^-(w,t;w',t') \tag{10}$$

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

$$Q_{(w,t)} = \Lambda_{(w,t)}/[\lambda_{(w,t)} + r_{(w,t)} + \sum_{w' \neq w} Q_{(w',t)} \omega^-(w',t;w,t) + \sum_{t' \neq t} Q_{(w,t')} \omega^-(w,t';w,t)] \tag{11}$$

and the system of equations (11) is then solved iteratively in the following manner to obtain the assignments of assets to tasks:

1. Initialisation: $W_{rem} \leftarrow W$, $S \leftarrow \emptyset$ and $K_{cur}(t) \leftarrow K(t),\ t \in T$.
2. Compute the RNN parameters based on $K_{cur}(t), \forall t$ and construct the neural network for $w \in W_{rem}$ and $t \in T$
3. Solve the system of Eqs. (11) for $w \in W$ and $t \in T$ to obtain $Q_{(w,t)}$.
4. 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 $t \in T$ stop: there is no assignment that reduces the cost of the objective function
5. Set $S \leftarrow S \cup (w^*, t^*)$
6. Set $W_{rem} \leftarrow W_{rem} \backslash \{w^*\}$
7. Set $K_{cur}(t^*) \leftarrow K_{cur}(t^*) q(w^*, t^*)$
8. 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.

The problem parameters can be shared among all agents prior to decision making. Once the assets have acquired the parameters, 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 solution to the RNN signal-flow equations is unique.

To derive the complexity of our algorithm we need first to examine the complexity of assigning one asset. This procedure is governed by the solution of the system of equations (7)-(9) which can be solved using an iterative algorithm of complexity $O(NI_{RNN}N^2)$ proposed in[30], when NI_{RNN} iterations are required to achieve convergence. In our case, the special structure of the constructed neural networks can be exploited to reduce the complexity to $O(NI_{RNN}|W||T|)$, so that the total complexity of our algorithm is $O(NI_{RNN}|W|^2|T|)$ as $|W|$ assets are assigned at most.

4. EVALUATION

To test the effectiveness of the algorithm that we have proposed, we tested it on different asset and task sets and compared our results to the optimal solution which was obtained by enumeration. Our algorithm was tested for the following cases:

1. The number of tasks is constant ($|T| = 5$) and the number of assets varies as follows $|W| = [4, 8, 12]$ (Figure 1).
2. The number of tasks is constant ($|T| = 8$) and the number of assets varies as follows $|W| = [4, 8, 12]$ (Figure 2).
3. The number of assets is constant ($|W| = 9$) and the number of tasks varies as follows $|T| = [3, 5, 8, 12]$ (Figure 3).

For each of the three sets of cases we generated $N_{PI} = 300$ problem instances by random selection of the problem parameters, as indicated below, and then obtained the optimal solution using exhaustive search. We also solved each of the randomly generated cases with the RNN heuristic.

Concerning the problem generation, parameters $K(t)$ for each task in T are generated from the uniform distribution in the interval $[10, 100]$. The asset execution failure probabilities are also drawn from the uniform distribution in the interval $[0.05, 0.3]$, while they are taken to be independent from the tasks, i.e. $q(w,t) = q(w), \forall t$. The associated asset costs $C(w)$ are drawn from the normal distribution with mean $\overline{C}(w)$ and variance

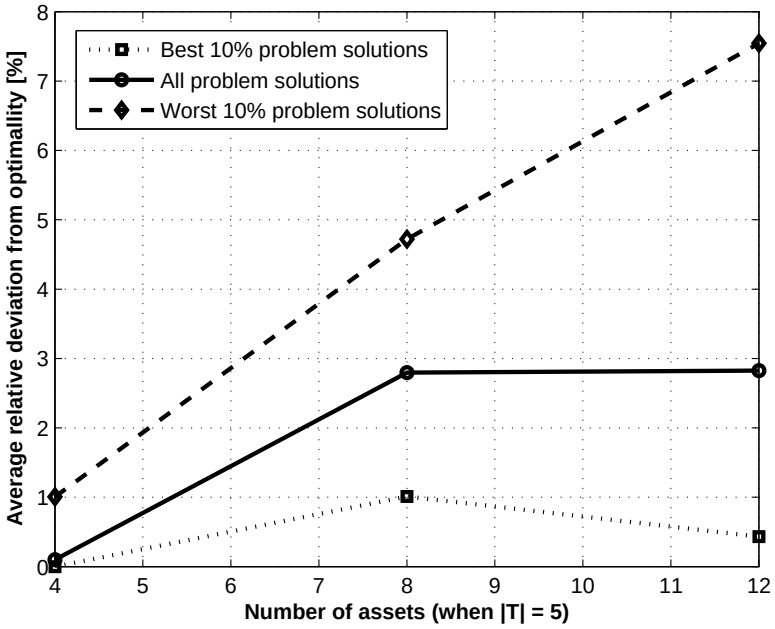

Figure 1. Average relative percentage deviation from the optimal solution for the worst 10%, all, and the best 10% problem solutions, when the number of tasks is 5 and the number of assets varies.

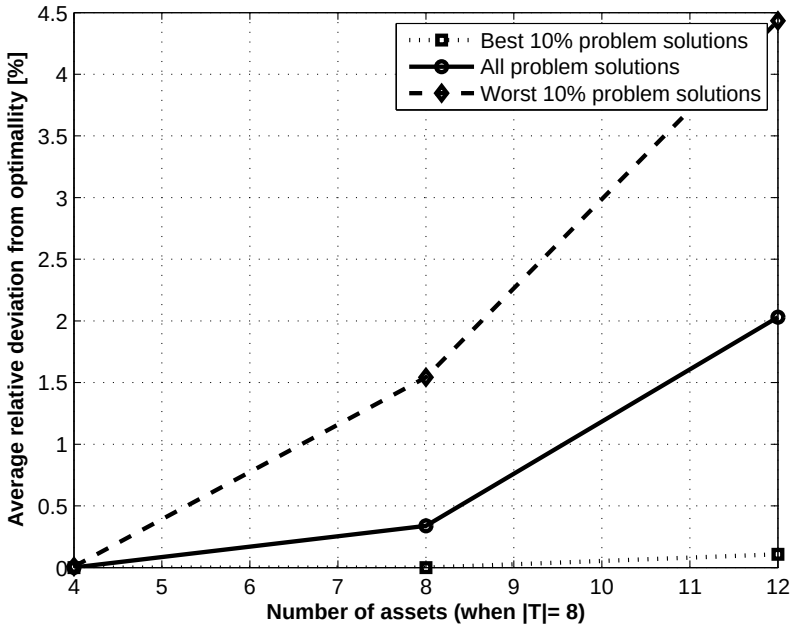

Figure 2. Average relative percentage deviation from the optimal solution for the worst 10%, all, and the best 10% problem solutions, when the number of tasks is 8 and the number of assets varies.

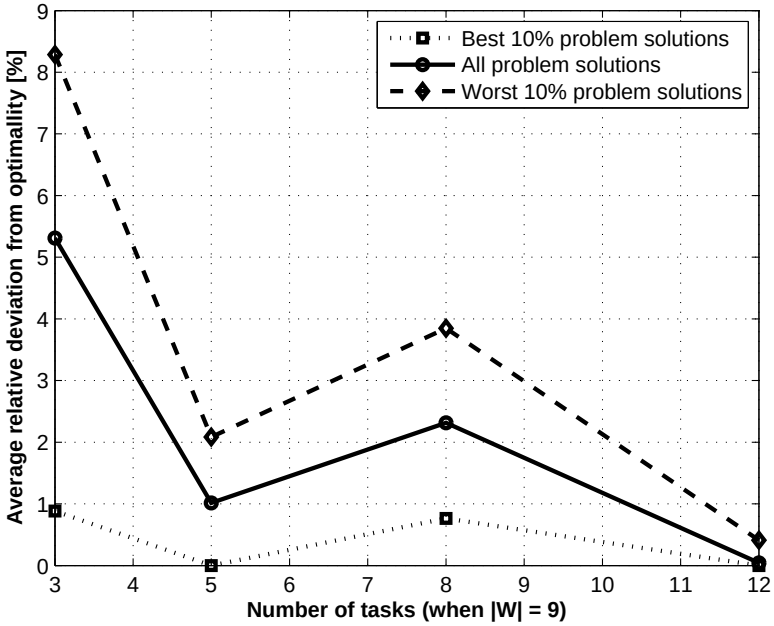

Figure 3. Average relative percentage deviation from the optimal solution for the worst 10%, all, and the best 10% problem solutions, when the number of assets is 9 and the number of tasks varies.

$0.1\overline{C}(w)$. The parameter $\overline{C}(w)$ is calculated from the linear equation (12) that connects points $(q_{max}, \overline{C}_{min})$ and $(q_{min}, \overline{C}_{max})$, where $q_{min} = 0.05$, $q_{max} = 0.3$, $\overline{C}_{min} = 5$ and $\overline{C}_{max} = 30$.

$$\overline{C}(w) = \frac{(\overline{C}_{max} - \overline{C}_{min})}{(q_{min} - q_{max})}(q(w) - q_{max}) + \overline{C}_{min} \tag{12}$$

The particular selection of parameters creates a positive correlation between the cost of an asset assignment and its associated execution success probabilities, so that "better" assets are more expensive.

The performance criterion that we use for comparing the RNN heuristic to the optimal solution is the average relative percentage deviation from the optimal, defined as:

$$\sigma = \frac{1}{N_{PI}} \sum_{i=1}^{N_{PI}} \frac{C_{RNN,i} - C_{opt,i}}{C_{opt,i}} \times 100 \tag{13}$$

In Figures 1, 2 and 3 we report the average relative percentage deviation from the optimal solution for the various $(|W|, |T|)$ pairs examined. We also show the percentage deviation σ for the worst 10% and the best 10% problem solutions. Thus we are able to report on the robustness of the proposed approach and evaluate whether the results obtained deviate significantly from the optimal minimum cost solution. As can be seen from these figures, the average relative percentage deviation from the optimal solution for all cases examined is within 5%. In addition, for the worst 10% performing instances, σ is no larger than 9%, which shows that the distance to the optimal solution is small even in the worst results that we have obtained.

5. CONCLUSIONS

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 (2). Now that these results have been obtained, the problem we consider, as well as the RNN based solution, can be extended in various directions. For

example, similar problems could be examined using the RNN approach. One such problem is the dynamic assets to tasks assignment problem, where new tasks are generated continually and we need to allocate the assets in a way that we minimise the total cost over time. Another interesting problem involves the allocation of assets in stages where assets may be maintained in reserve and re-allocated during the course of the assignment process. Furthermore, real-world applications involving assignments with uncertain execution could be tackled, especially when fast(even real-time) and close to optimal solution is required. We expect that some of these and other generalisations will be examined using an RNN based approach in future work.

6. ACKNOWLEDGEMENTS

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 (UK Engineering and Physical Research Council) strategic partnership under grant no. EP/C548051/1.

REFERENCES

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

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

[3] Ahuja, R. K., Kumar, A., Jha, K. C., and Orlin, J. B., "Exact and Heuristic Algorithms for the Weapon-Target Assignment Problem," *OPERATIONS RESEARCH* **55**(6), 1136–1146 (2007).

[4] Wacholder, E., "A Neural Network-Based Optimization Algorithm for the Static Weapon-Target Assignment Problem," *INFORMS JOURNAL ON COMPUTING* **1**(4), 232–246 (1989).

[5] Yanxia, W., Longjun, Q., Zhi, G., and Lifeng, M., "Weapon target assignment problem satisfying expected damage probabilities based on ant colony algorithm," *Journal of Systems Engineering and Electronics* **19**(5), 939 – 944 (2008).

[6] Lee, Z.-J., Lee, C.-Y., and Su, S.-F., "An immunity-based ant colony optimization algorithm for solving weapon-target assignment problem," *Applied Soft Computing* **2**(1), 39 – 47 (2002).

[7] Lee, Z.-J., Su, S.-F., and Lee, C.-Y., "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 (Feb 2003).

[8] Loiola, E. M., de Abreu, N. M. M., Boaventura-Netto, P. O., Hahn, P., and Querido, T., "A survey for the quadratic assignment problem," *European Journal of Operational Research* **176**(2), 657 – 690 (2007).

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

[10] Sahni, S. and Gonzalez, T., "P-Complete Approximation Problems," *Journal of the ACM* **23**(3), 555–565 (1976).

[11] Magirou, V. F. and Milis, J. Z., "An algorithm for the multiprocessor assignment problem," *Operations research letters* **8**(6), 351–356 (1989).

[12] Malucelli, F., "A polynomially solvable class of quadratic semi-assignment problems," *European Journal of Operational Research* **91**(3), 619 – 622 (1996).

[13] Malucelli, F. and Pretolani, D., "Lower bounds for the quadratic semi-assignment problem," *European Journal of Operational Research* **83**(2), 365 – 375 (1995).

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

[15] Gelenbe, E. and Timotheou, S., "Synchronised Interactions in Spiked Neuronal Networks," *The Computer Journal* **51**(4), 723–730 (2008).

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

[17] Gelenbe, E., Ghanwani, A., and Srinivasan, V., "Improved neural heuristics for multicast routing," *IEEE Journal of Selected Areas of Communications* **15**(2), 147–155 (1997).

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

[19] Gelenbe, E., "Random Neural Networks with Negative and Positive Signals and Product Form Solution," *Neural Computation* **1**(4), 502–510 (1989).

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

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

[22] Aguilar, J. and Gelenbe, E., "Task assignment and transaction clustering heuristics for distributed systems," *Information Sciences – Informatics and Computer Science* **97**(1 & 2), 199–221 (1997).

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

[24] Gelenbe, E., "Product-Form Queueing Networks with Negative and Positive Customers," *Journal of Applied Probability* **28**(3), 656–663 (Sep. 1991).

[25] Gelenbe, E., "Steady-state solution of probabilistic gene regulatory networks," *Physical Review E* **76**(1), 031903 (2007).

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

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

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

[29] Timotheou, S., "The Random Neural Network: A Survey," *The Computer Journal* **53**(3), 251–267 (2010).

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