Random Neural Network for Emergency Management

Erol Gelenbe and Stelios Timotheou
Dept. of Electrical & Electronic Eng.
Imperial College
London SW7 2BT, UK
Contact: e.gelenbe@imperial.ac.uk

David Nicholson
BAE Systems Ltd
Sowerby Building
Filton, Bristol, BS34 7QW, UK

***Abstract*—We consider decision problems in emergency management, such as simultaneously dispatching emergency teams to locations where incidents have occurred, and propose an algorithmic solution using the Random Neural Newtwork. This is an NP-hard optimisation problem, but the approach we suggest is solved in polynomial time, and is also distributed so that each of the teams can potentially decide where to go based on shared information about the location of the incidents and of the teams, without consulting the others concerning the decision. The proposed approach is evaluated on a large number of instances of the problem, and we observe that it comes within 10% of the cost achieved by the optimal solution.**

I. Introduction

Consider a set T of “emergencies” (tasks to execute) and a set W of emergency teams or “ambulances” (assets that can be used to execute the tasks). Our purpose is to investigate (a) simultaneous and distributed, (b) near optimal, and (c) very fast and necessarily polynomial time complexity assignments of the elements of W to the elements of T, where we wish to maximise the number of injured individuals that are collected, and minimise the response time. Thus with each of the emergencies we assume that there are a known number of injured individuals, and that each of the ambulances has a known travel time to each of the emergencies. Such problems arise in emergency management [7], [6] and are instances nonlinear combinatorial optimisation problems for assigning assets to tasks so as to minimise a desired cost function [34]. Several variations have been studied and have found widespread application in telecommunications, transportation systems and signal processing [4]. A detailed presentation of our work will appear in [23].

In this paper we consider that the outcome of any assignment is uncertain due to the possibility that certain ambulances may not be able to handle or reach some of the emergencies. All the tasks can be executed by any one of the assets, but there is a penalty $K(t) > 0$ for not executing task $t \in T$, and a cost $C(w,t)$ for executing task t with $w \in W$. A task execution may fail despite the fact that an asset has been assigned to it, represented by the probability $q(w,t)$ that w fails in executing t, and $q(w,t)$ is a characteristic of the pair (w,t) where the w is not perfectly effective or suited to deal with t. We also allow several assets to be assigned to the same task. We represent the decision of assigning the asset w to the task t by the probability $p(w,t)$, so that it may happen that some asset is not assigned to any task, $0 \leq p(w,t) \leq 1$, $\sum_{t\in T} p(w,t) = \pi(w) \leq 1$. If $\pi(w) < 1$, this means that we may decide not to assign the asset w to any task. When 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 also corresponds to real-time situations where, for the given time epoch considered, decisions are irrevocable.

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:

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

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, 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:

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

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.

Our formulation belongs to the general class of nonlinear assignment problems [33]. A related problem with a product of terms that also have the decision variables in their exponent is the Weapon Target Assignment (WTA) problem that is NP-complete [27] and hence exact algorithms have been proposed only for solving special cases of the problem, while solutions to the general WTA problem are discussed in [2] 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 metaheuristic techniques such as Hopfield neural networks [38], ant colony optimisation [39], [25], genetic algorithms [26] and very large scale neighbourhoods [2]. 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 [28]. However, in these problems not only each asset must be assigned once, but also each task must be associated to only one asset which is not in agreement with (2), 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 is the quadratic semi-assignment problem (QSAP) [35]. Practical applications of QSAP include clustering and partitioning in distributed computing [24], [32] and scheduling [5], [3]. In the general case, QSAP is NP-hard [36] and in practice optimal solutions cannot be obtained even for small cases [29]. As a result, exact algorithms have been developed only for special cases of QSAP that are of polynomial time complexity [30], [31].

II. A Distributed RNN Based Heuristic

We now develop a Random Neural Network (RNN) [12], [19] 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 [22], [15] without a learning phase [17], and they are therefore computationally fast.

The RNN is an open recurrent neural network inspired by the spiking behaviour of natural mammalian neuronal networks [10] and using paradigms from queueing theory [8]. 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, $\mathbf{k}(\tau) = [k_1(\tau), \dots, k_N(\tau)]$. 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 $\pi(\mathbf{k}, \tau) = Pr[k_1(\tau) = k_1, \dots, k_N(\tau) = k_N]$ where $\mathbf{k} = [k_1, \dots, k_N]$. 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 $d(i)$, and:

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

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) we have:

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

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

The values of the stationary excitation probabilities $Q_i = \lim_{\tau \to \infty} Q_i(\tau) \quad i = 1, \dots, N$ and the stationary probability distribution are obtained from [10]. The total arrival rates of positive and negative signals to each neuron $\lambda^+(i)$ *and* $\lambda^-(i)$,

TABLE I
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

$i = 1, ...N$ 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:

$$Q_i = \begin{cases} \frac{\lambda^+(i)}{r_i + \lambda^-(i)}, & \text{if } \lambda^+(i) < r_i + \lambda^-(i) \\ 1 & \text{if } \lambda^+(i) \geq r_i + \lambda^-(i) \end{cases} \tag{9}$$

The solution to the nonlinear system (7)-(9) always exists and is unique [11], [12].

The RNN model has been successful in the solution of optimisation problems [1], [17], [18], in modelling complex interactions between entities in queueing networks [21], natural neuronal networks [19] and gene regulatory networks [13], as well as in learning [14], [16], [20]. A survey of RNN can be found in [37].

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:

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

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

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

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

$\lvert W\rvert$	$\lvert T\rvert$	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 II
MEAN AND STANDARD DEVIATION OF THE RELATIVE PERCENTAGE DEVIATION FROM OPTIMALITY FOR DATA FAMILY 1

$\lvert W\rvert$	$\lvert T\rvert$	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 III
MEAN AND STANDARD DEVIATION OF THE RELATIVE PERCENTAGE DEVIATION FROM OPTIMALITY FOR DATA FAMILY 2

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 [9], when NI_{RNN} iterations are performed 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 $O(NI_{RNN} \cdot |W|^2 \cdot |T|)$ as $|W|$ assets are assigned at most.

III. Evaluation

The effectiveness of the proposed RNN algorithm is evaluated with respect to two generated data families. 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 (12) 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 $\overline{C}_{max} = 30$.

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

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:

$$\sigma_{opt} = \frac{C_{alg,i} - C_{opt,i}}{C_{opt,i}} \times 100\% \qquad (13)$$

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 II and III for the two data families, where σ_{opt}^{mean} and

$\|W\|$	$\|T\|$	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 IV
AVERAGE PERCENTAGE DEVIATION OF THE MMR APPROACH FROM THE RNN APPROACH FOR LARGE-SIZED PROBLEMS

$\|W\|$	$\|T\|$	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 V
EXECUTION TIMES FOR THE RNN AND MMR APPROACHES FOR DATA FAMILY 2

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 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}.

$$\sigma_{RNN} = \frac{C_{Greedy,i} - C_{RNN,i}}{C_{RNN,i}} \times 100\% \qquad (14)$$

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

Table IV 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 V. 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. We only present the execution times of data family 2 as both data family required similar execution times. As expected, the MMR approach is faster than the RNN one because it is based on greedily selecting the asset with the maximum marginal return. 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.

IV. CONCLUSIONS

The results of the proposed RNN-based algorithm indicate its usefulness for the solution of emergency management optimization problems such as the asset-task assignment problem addressed. We have shown that our approach can obtain results close to optimal, in a distributed manner and in polynomial time, so that solutions for large-scale problems can be obtained in real-time.

References

[1] J. Aguilar and E. Gelenbe, "Task assignment and transaction clustering heuristics for distributed systems," *Information Sciences – Informatics and Computer Science*, vol. 97, no. 1 & 2, pp. 199–221, 1997.

[2] R. K. Ahuja, A. Kumar, K. C. Jha, and J. B. Orlin, "Exact and Heuristic Algorithms for the Weapon-Target Assignment Problem," *OPERATIONS RESEARCH*, vol. 55, no. 6, pp. 1136–1146, 2007.

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

[4] R. E. Burkard and E. Cela, "Linear assignment problems and extensions," in *Handbook of Combinatorial Optimization - Supplement Volume A*, P. Pardalos and D.-Z. Du, Eds. Kluwer Academic Publishers, 1999, pp. 75–149.

[5] P. Chretienne, "A polynomial algorithm to optimally schedule tasks on a virtual distributed system under tree-like precedence constraints," *European Journal of Operational Research*, vol. 43, no. 2, pp. 225 – 230, 1989.

[6] N. Dimakis, A. Filippopoulitis, and E. Gelenbe, "Distributed building evacuation simulator for smart emergency management," The Computer Journal, 2010, doi:10.1093/comjnl/bxq012.

[7] A. Filippoupolitis and E. Gelenbe, "A distributed decision support system for building evacuation," in *Proceedings of the 2nd IEEE International Conference on Human System Interaction, Catania, Italy*. New York, NY, USA: IEEE Press, 2009, pp. 323–330.

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

[9] J. Fourneau, "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.

[10] E. Gelenbe, "Random Neural Networks with Negative and Positive Signals and Product Form Solution," *Neural Computation*, vol. 1, no. 4, pp. 502–510, 1989.

[11] ——, "Stability of the random neural network," *Neural Computation*, vol. 2, no. 2, pp. 239–247, 1990.

[12] ——, "Learning in the recurrent random network," *Neural Computation*, vol. 5, pp. 154–164, 1993.

[13] ——, "Steady-state solution of probabilistic gene regulatory networks," *Physical Review E*, vol. 76, no. 1, p. 031903, 2007.

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

[15] E. Gelenbe, A. Ghanwani, and V. Srinivasan, "Improved neural heuristics for multicast routing," *IEEE Journal of Selected Areas of Communications*, vol. 15, no. 2, pp. 147–155, 1997.

[16] E. Gelenbe and T. Kocak, "Area-based results for mine detection," *IEEE Trans. on Geoscience and Remote Sensing*, vol. 38, no. 1, pp. 1–14, 2000.

[17] E. Gelenbe, V. Koubi, and F. Pekergin, "Dynamical random neural network approach to the traveling salesman problem," *ELEKTRIK*, vol. 2, no. 2, pp. 1–10, 1994.

[18] E. Gelenbe and S. Timotheou, "Random Neural Networks with Synchronised Interactions," *Neural Computation*, vol. 20, pp. 2308–2324, 2008.

[19] ——, "Synchronised Interactions in Spiked Neuronal Networks," *The Computer Journal*, vol. 51, no. 4, pp. 723–730, 2008.

[20] E. Gelenbe, "Cognitive packet network," *US Patent*, vol. 6804201, 2004.

[21] ——, "Product-Form Queueing Networks with Negative and Positive Customers," *Journal of Applied Probability*, vol. 28, no. 3, pp. 656–663, Sep. 1991.

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

[23] E. Gelenbe, S. Timotheou, and D. Nicholson, "Fast distributed near optimum assignment of assets to tasks," The Computer Journal, 2010, doi:10.1093/comjnl/bxq010.

[24] P. Hansen and K. W. Lih, "Improved algorithms for partitioning problems in parallel, pipelined, and distributed computing," *IEEE Transactions on Computers*, vol. 41, no. 6, pp. 769–771, Jun 1992.

[25] Z.-J. Lee, C.-Y. Lee, and S.-F. Su, "An immunity-based ant colony optimization algorithm for solving weapon-target assignment problem," *Applied Soft Computing*, vol. 2, no. 1, pp. 39 – 47, 2002.

[26] Z.-J. Lee, S.-F. Su, and C.-Y. Lee, "Efficiently solving general weapon-target assignment problem by genetic algorithms with greedy eugenics," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 33, no. 1, pp. 113–121, Feb 2003.

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

[28] E. M. Loiola, N. M. M. de Abreu, P. O. Boaventura-Netto, P. Hahn, and T. Querido, "A survey for the quadratic assignment problem," *European Journal of Operational Research*, vol. 176, no. 2, pp. 657 – 690, 2007.

[29] V. F. Magirou and J. Z. Milis, "An algorithm for the multiprocessor assignment problem," *Operations research letters*, vol. 8, no. 6, pp. 351–356, 1989.

[30] F. Malucelli, "A polynomially solvable class of quadratic semi-assignment problems," *European Journal of Operational Research*, vol. 91, no. 3, pp. 619 – 622, 1996.

[31] F. Malucelli and D. Pretolani, "Lower bounds for the quadratic semi-assignment problem," *European Journal of Operational Research*, vol. 83, no. 2, pp. 365 – 375, 1995.

[32] I. Z. Milis and V. F. Magirou, "A lagrangian relaxation algorithm for sparse quadratic assignment problems," *Operations Research Letters*, vol. 17, no. 2, pp. 69 – 76, 1995.

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

[34] D. W. Pentico, "Assignment problems: A golden anniversary survey," *European Journal of Operational Research*, vol. 176, no. 2, pp. 774 – 793, 2007.

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

[36] S. Sahni and T. Gonzalez, "P-Complete Approximation Problems," *Journal of the ACM*, vol. 23, no. 3, pp. 555–565, 1976.

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

[38] E. Wacholder, "A Neural Network-Based Optimization Algorithm for the Static Weapon-Target Assignment Problem," *INFORMS JOURNAL ON COMPUTING*, vol. 1, no. 4, pp. 232–246, 1989.

[39] W. Yanxia, Q. Longjun, G. Zhi, and M. Lifeng, "Weapon target assignment problem satisfying expected damage probabilities based on ant colony algorithm," *Journal of Systems Engineering and Electronics*, vol. 19, no. 5, pp. 939 – 944, 2008.