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. As we proposed in [1], we formulate the allocation of assets to tasks in order to minimise the overall expected cost, as a nonlinear combinatorial optimisation problem. We propose the use of network flow algorithms which are based on solving a sequence of minimum cost flow problems on appropriately constructed networks with estimated arc costs. We introduce three different scheme for the estimation of the arc costs and we investigate their performance compared to a random neural network algorithm and a greedy algorithm. We also develop an approach for obtaining tight lower bounds to the optimal solution based on a piecewise linear approximation of the considered problem.

Keywords: optimum asset assignment; execution uncertainty; nonlinear combinatorial optimisation; random neural networks; network flow problems; piecewise linear approximation; lower bounds

Received 00 Month 2010; revised 00 Month 2010

1. INTRODUCTION

Assignment problems is a fundamental class of combinatorial optimisation problems which involve assigning assets to tasks to minimise a desired cost function. 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.

Clearly, in real-world applications assigning an asset to a task does not necessarily imply successful execution. The outcome of an assignment can depend on several factors such as the surrounding environment, an asset’s ability in achieving its task, and also the status of a task. For example, in a sensor coverage problem where a set of sensors monitors an area to detect targets, a sensor may fail in detecting a target in its assigned monitoring sector for several reasons. Firstly, the environment may obstruct the visual contact between the sensor and the target. Secondly, the sensor may be malfunctioning to a certain extend so that detection may not be achieved even if there is line-of-sight. Finally, the target may camouflage itself to decrease the possibility of being detected. For this problem, a good strategy to increase the possibility of detecting a target would be to have more than one sensors monitoring a particular sector [3].

Additionally, uncertainty in successful completing a task can even be inherent. For instance in cancer therapy, the possibility of destroying a targeted tumour with a certain therapeutic tool (chemotherapy, radiotherapy, immunotherapy) can only be expressed probabilistically. Therefore, in such cases it is beneficial to apply more than one therapy tools to targeted areas to minimise not only the possibility of failure but also the cost and the side effects of the overall therapy [4].

In this paper we investigate a general assignment problem where the outcome of any assignment is uncertain, that we first examined in [1]. We model uncertainty by assuming that one asset has a certain probability in executing a particular task. In the examined problem, one asset suffices to execute one task, while more than one assignments can be made for the same purpose to increase the probability of success. We further assume that the assignments made to the
same task have an independent overall effect so that the total failure probability of the particular task is given by the product of the individual failure probabilities. The objective is to minimise the overall expected cost, given that there is a cost for each asset-task assignment, as well as a cost for the non-execution of each task.

Apart from the sensor coverage and cancer therapy problems discussed above other application areas that are covered by this abstract representation include examples where:

- Tasks represent “jobs”, and assets represent “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,
- Tasks are “emergencies” and assets are “ambulances or emergency personnel” that have the composite goal of maximising the number of injured collected and minimising their response time to the emergencies when it is uncertain if an “ambulance” will reach the targeted emergency,
- Tasks are “entities that need to communicate” and assets are “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 contribution of this paper is twofold:

1. We develop network flow algorithms for the solution of the considered asset-task assignment problem. These algorithms are based on solving a sequence of minimum cost flow problems on appropriately constructed networks with estimated arc costs. Specifically, we consider three different estimation schemes MCFmax, MCFmin and MCFmnn as described in section 5.
2. We propose an approach for obtaining tight lower bounds to the optimal solution by transforming the problem to an equivalent one and approximating the latter using piecewise linear functions. We derive analytical expressions for the upper and lower bounds of the approximation intervals and introduce a adaptive scheme for the selection of the piecewise linear segments that restricts the maximum approximation error to a desired value, as explained in section 6.

The structure of the paper is as follows. We start with the description and the mathematical formulation of the asset-task assignment problem with execution uncertainty, followed by a brief discussion of related problems in section 3. Then, we describe a random neural network approach for the solution of examined problem, proposed in [1], which will be useful in section 5 where the network flow approaches are examined. In section 6 we describe an algorithm for obtaining tight lower bounds to the studied problem, while in the last two sections we examine the performance of the proposed approaches and conclude.

2. PROBLEM DESCRIPTION AND MATHEMATICAL FORMULATION

Consider a set of tasks $\mathcal{T}$ that need to be executed by a set of assets $\mathcal{A}$. Task $t$ carries a penalty $U(t)$ if it is not executed, while there is also a cost $C_a(a,t)$ for assigning asset $a$ to task $t$. We assume that any one of the tasks can be executed by any one of the assets and that one asset suffices to execute one task. 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 $0 \leq p_f(a,t) < 1$ that asset $a$ will fail in executing task $t$ when it is assigned to it.

To compensate task execution failures more than one asset can be assigned to one task to increase the probability of successful execution. It is assumed that the assets assigned to the same task $t$ have an independent overall effect so that the overall failure probability for the particular task, $p_f^t(t)$, is given by the product of the failure probabilities of the assets assigned to it. For example, if a particular task is associated with three assignment with failure probabilities 0.4, 0.2 and 0.1, then the total failure probability for the particular task will be equal to $p_f^t(t) = 0.4 \times 0.2 \times 0.1 = 0.008$. We also assume that after an asset is allocated to some task, it cannot be re-assigned 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. It is also possible for one asset not to be assigned to any of the tasks, as the incurred assignment cost can increase the overall cost instead of decreasing it. Our objective if to find an allocation matrix $X$ with elements $X(a,t) \in \{0,1\}$, representing whether asset $a$ is assigned to task $t$, that minimises our cost function defined below:

$$C = \sum_{t \in \mathcal{T}} \sum_{a \in \mathcal{A}} C_a(a,t)X(a,t) + \sum_{t \in \mathcal{T}} U(t)p_f^t(t) \quad (1)$$

where $p_f^t(t) = \prod_{a \in \mathcal{A}} \{1 - p_s(a,t)X(a,t)\}$ is the overall failure probability of task $t$ and $p_s(a,t) = 1 - p_f(a,t)$ is the probability that asset $a$ will successfully execute task $t$ if it is assigned to it.

In equation (1) the first term is the total cost of the assignments made, while the second term expressed the expected remaining cost of task $t$. Expression $\{1 - p_s(a,t)X(a,t)\}$ in $p_f^t(t)$ denotes the failure probability from assigning asset $a$ to task $t$ without knowing if the assignment will take place. If the assignment is made, $X(a,t) = 1$, the failure probability is equal to $p_f(a,t) = 1 - p_s(a,t)$, otherwise if $X(a,t) = 0$ then the failure probability for that assignment is equal to 1. Note that $p_f^t(t)$ is given by the product of the aforementioned failure probabilities as we have assumed.
that assignments to the same task have an independent overall effect.

Moreover, when \(X(a, t) \in \{0, 1\}\), expressions \(\{1 - p_r(a, t)X(a, t)\}\) are equivalent to \(p_f(a, t)X(a, t)\) so that the cost function can be written with decision variables in the exponents of assignment failure probabilities, resulting in the following problem formulation which will be used throughout the paper:

\[
\min C = \sum_{t \in T} \sum_{a \in A} C_a(a, t)X(a, t)
+ \sum_{t \in T} U(t) \prod_{a \in A} p_f(a, t)^{X(a, t)}
\tag{2}
\]

\[s.t. \quad \sum_{t \in T} X(a, t) \leq 1, \quad a \in A\]

\[X(a, t) \in \{0, 1\}\]

Note that the constraint shows that one asset can be assigned to at most one task.

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 [5]. A related problem with a product of terms having the decision variables in the exponents of parameters is the Weapon Target Assignment (WTA) problem. In the WTA problem, each weapon has a probability of successfully intercepting a target and the objective is to assign all the weapons to targets so that the expected damage on the targets is maximised or equivalently the expected leakage of the targets is minimised.

There are two main difference between our problem and the WTA problem. The first difference is that in our problem each asset has an associated cost, while in the WTA the weapons carry no cost. The second difference, is that in our formulation not all assets need to be assigned to tasks. This stems from the fact that each asset has an associated cost and hence a particular assignment might not be beneficial.

In the general case, the WTA-problem is NP-complete [6] and hence exact algorithms have been mostly proposed for special optimally solvable cases of the problem. One such special case is when the interception probabilities are independent of the weapon, while a second one is when we want to assign at most one weapon to each target. The first special case can be solved either using the Maximum Marginal Return (MMR) algorithm [7, 8], where the weapons are assigned in a greedy fashion to the targets that result in the maximum decrease of the cost function, or using a local search algorithm to identify and swap any weapon-target pair that reduces the overall cost [9]. The second special case results in a linear assignment problem that can be efficiently solved using for example a network flow algorithm [10].

Exact solution to the general WTA problem has been achieved for medium size problems in [11], 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 [12], ant colony optimisation, [13, 14], genetic algorithms [15] and very large scale neighbourhoods [11].

Quadratic and biquadratic assignment problems are also related to problem (2), as their objective function includes products of two or more variables [16]. 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) [17]. Contrary to the WTA problem, in QSAP assignments carry a cost, while there is an additional cost for each pair of assignments. Hence, the associated cost term in the objective function of QSAP is comprised of products of two decision variables, contrary to our formulations where we have \(|T|\) variables in each product. Note also that all assets need to be assigned in QSAP.

In the general case, QSAP is NP-hard [18] and in practice optimal solutions cannot be obtained even for small size problems [19]. As a result exact algorithms have been developed only for special cases of QSAP that result in algorithms with polynomial time complexity [20, 21].

4. A RANDOM NEURAL NETWORK BASED ALGORITHM

In this section we will briefly describe a Random Neural Network (RNN) [22] based approach for the solution of the asset-task assignment problem (2) that we proposed in [1], which will be useful in Section 5 for constructing an RNN-based network flow approach and for performance evaluation.

4.1. The random neural network model

The RNN is an biologically inspired, open, recurrent neural network composed of \(N\) fully connected neurons [22]. 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 $\Lambda_i$ and $\lambda_i$ respectively. When neuron $i$ is excited (its potential is positive), it can fire spikes according to the exponential distribution; the spikes that are sent out can either reach neuron $j$ as excitatory spikes with rate $w^+(i, j)$ or as inhibitory (negative) spikes or signals with rate $w^-(i, j)$, or they may depart from the network with rate $r_id(i)$, where $r_i$ is the total firing rate of the neuron given by:

$$r_i = (1 - d(i))^{-1} \sum_{j=1}^{N} [\omega^+(i, j) + \omega^-(i, j)]$$  \hspace{1cm} (3)

The values of the stationary excitation probabilities $Q_{ij}, \; i = 1, ..., N$ and the stationary probability distribution are obtained from Theorem 1 in [22]. 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:

$$\lambda^+(i) = \Lambda_i + \sum_{j=1}^{N} Q_{ij} \omega^+(j, i)$$  \hspace{1cm} (4)

$$\lambda^-(i) = \lambda_i + \sum_{j=1}^{N} Q_{ij} \omega^-(j, i)$$  \hspace{1cm} (5)

where:

$$Q_{ij} = \min\{1, \frac{\lambda^+(i)}{(r_i + \lambda^-(i))}\}$$  \hspace{1cm} (6)

The solution to the nonlinear system (4)-(6) always exists and is unique [23, 24]. The RNN model has been successful in many applications as a learning or a modelling tool, as well as for the solution of optimisation problems [25, 26, 27]. A survey of RNN can be found in [28].

4.2. The solution approach

The solution approach is based on an algorithm that uses an RNN whose parameters, including the weights associated with the connections between neurons, are selected directly by translation of the parameters of the optimisation problem. Specifically, each allocation decision $(a, t)$ is represented by a neuron $N(a, t)$ of a RNN, so that $X(a, t)$ corresponds to the probability $q(a, t)$ that this particular neuron is excited. As a result, our neural network is composed of $|A||T|$ neurons, where $| \cdot |$ denotes the number of elements in the particular set. To construct the RNN to be used for the heuristic solution of problem (2), we must specify the arrival rates of excitation and inhibition signals to each of the neurons $N(a, t)$, and the excitatory and inhibitory weights between neurons. The external positive and negative arrival rates are chosen as follows:

$$\Lambda(a, t) = \max \{0, b(a, t)\}, \quad \lambda(a, t) = \max \{0, -b(a, t)\}$$

where

$$b(a, t) = U(t)p_a(a, t) - C_a(a, t)$$

so that $b(a, t)$ represents the net expected reduction in the objective function when asset $a$ is allocated to task $t$, since $U(t)p_a(a, t)$ is expected remaining cost of task $t$ if this allocation is made and $C_a(a, t)$ is the cost of allocating this asset to the given task. To discourage the allocation of distinct assets to the same task and to avoid the assignment of the same asset to distinct tasks we also set the inhibitory weights:

$$w^-(a, t; a', t) = \max \{0, b(a, t)\}, \quad a \neq a'$$

$$w^-(a, t; a', t') = \max \{0, b(a, t)\}, \quad t \neq t'$$

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 $w^+(a, t; a', t') = 0$ and $w^-(a, t; a', t') = 0$ for all other $a, a'$ and $t, t'$, and we end with $r(a, t) = \sum_{a', t'} w^-(a, t; a', t')$. Based on the above parameters the excitation level of each neuron satisfies:

$$Q(a, t) = \Lambda(a, t)/[\lambda(a, t) + r(a, t) + \sum_{a' \neq a} Q(a', t)w^-(a', t; a, t)] + \sum_{t' \neq t} Q(a, t')w^-(a, t'; a, t)$$  \hspace{1cm} (7)

Let us now summarise the RNN approach for the solution of problem (2). After initialisation of the RNN parameters and construction of the network, the system of equations (7) is solved numerically to obtain the most excited neuron $(a^*, t^*)$, make the corresponding assignment and update the penalty $U(t^*)$ to the value $U(t^*)p_f(a^*, t^*)$. Then, the neurons $N(a^*, t^*), \forall t$ are removed from the network and the procedure is repeated again until either all assets have been assigned or until there is no beneficial assignment. The described RNN algorithm is of complexity $O(|A|^2|T|)$. For more details see [1].

5. NETWORK FLOW ALGORITHMS

Network flow problems are an important class of linear programming problems. They can be utilised for the solution of many optimisation problems such as the maximum flow, assignment, transportation and shortest path problems. Due to their special structure, network flow problems can be solved tens of times faster than linear programming problems while there are strong polynomial algorithms that put bounds on their worst case performance. In addition, when a network
Asset-task assignment with execution uncertainty

The most fundamental problem in network flows is the minimum cost flow (MCF); most other network flow problems are either special cases or generalisations of the MCF. The MCF problem considers a directed graph or network \( G = (N, E) \) which consists of a set of vertices or nodes \( N \) and a set of directed edges or arcs \( E \) connecting the nodes. Each arc \((i, j)\) ∈ \(E\) is characterised by two parameters: the capacity \(u(i, j)\) of the particular arc which is the upper bound of flow \(X_f(i, j)\) allowed through \((i, j)\) and an associated cost per unit of flow \(C_f(i, j)\). Each node \(i ∈ N\) has a supply \(s(i)\) that is interpreted as the amount of flow that enters the node from the outside. Node \(i\) is a source or supply node if \(s(i) > 0\), a sink or demand node if \(s(i) < 0\) and a transshipment node if \(s(i) = 0\). Flow networks are governed by the flow conservation constraint which states that at each node the incoming and outgoing flows are equal. Note that the conservation constraint can hold only if \(\sum_i s(i) = 0\). In the MCF problem, the objective is to find the cheapest flows that satisfy the nodes’ supply, under the flow conservation constraint and the capacity constraint. The mathematical formulation of MCF is given by Eq. (8).

\[
\begin{align*}
\text{min} & \quad \sum_{(i,j) \in E} C_f(i,j)X_f(i,j) \\
\text{s.t.} & \quad s(i) + \sum_{j : (j,i) \in E} X_f(j,i) = \sum_{j : (i,j) \in E} X_f(i,j), \forall (i,j) \in E \\
& \quad 0 \leq X_f(i,j) \leq u(i,j), (i,j) \in E
\end{align*}
\]

The approach that we take is to construct a flow network with flow costs associated with the net expected reduction in the objective function when assigning assets to tasks.

Fig. 1 depicts the network used for the solution of problem (2). The network is comprised of three layers of nodes: the first layer contains the source nodes, the second layer contains the transshipment nodes and the third layer the demand node that aggregates the flows send by the source nodes. Each source node \(a\) has supply \(s(a) = 1\) and corresponds to asset \(a\). Each transshipment node \(f^{(m)}\) denotes the \(m\)th asset assignment to task \(t\), while node 0 corresponds to the case that an asset is not assigned to any task. At most \(M_t\) assets can be assigned to task \(t\). The role of the demand node \(d\) is to aggregate the flows send in the network and its demand is equal to the total supply of
the assets, \(s(d) = -|A|\).

A source node \(a\) is connected to all transshipment nodes \(t^{(m)}\) and the capacity of all arcs is equal to 1, so that the associated flows \(X_f(a,t^{(m)})\) represent the fact that asset \(a\) is the \(m\)th assignment to task \(t\). Even though there are \(|A|\) arcs arriving at each transshipment node there is only one arc leaving each such node towards the demand node \(d\). These arcs also have capacity 1 except from the arc \((0,d)\) whose capacity is equal to \(|A|\) so that even if no assignments are made the source nodes' supply reaches the demand node via node 0. Thus, \(X_f(t^{(m)}), t \in T\) denotes whether the \(m\)th assignment for task \(t\) has been made.

The resulting configuration guarantees that at most one asset can be assigned to a particular transshipment node. Moreover, as all arc capacities and supplies/demands of the nodes are integers, the integrality property guarantees that in the MCF solution all flows \(X_f(i,j)\) will be integer (see, e.g., p.318 of [29]). Actually, \(X_f(a,t^{(m)})\) flows have unit capacity so that the final value of the particular flows will be 0 or 1. We also need to ensure that the assignment of assets to a particular task \(t\) is contiguous in the sense that if there are already \(m_t - 1\) assignments, then the next one should be the \(m_t\)th one. The contiguous property will be established after we discuss about the costs of the arcs.

The costs of the arcs represent the net reduction in the cost function from assigning a particular asset to a task so our aim is to maximise the net reduction in the objective function. Thus, to solve the problem as a minimum cost flow problem we need to negate all the costs associated with the network.

The cost of the arcs associated with a first assignment \((a,t^{(1)})\) is given by:

\[
C_f(a,t^{(1)}) = \max\{0,U(t)p_a(a,t) - C_a(a,t)\}, \forall a, t \quad (9)
\]

In order to be able to correctly determine the arc costs associated with the second assignment \(C_f(a,t^{(2)})\) we need to know the expected cost of the task after the first assignment, \(U^{(1)}(t)\). If we assume that an oracle provides the first asset allocated to task \(t\), \(a^{(1)}\), then \(U^{(1)}(t) = U(t)p_f(a^{(1)},t)\). Similarly, if the oracle provides the first \(m_t\) assigned assets to task \(t\), \(a^{(1)}, \ldots, a^{(m_t)}\), then the arc costs for the next assignment of task \(t\) will be given by:

\[
C_f(a,t^{(m_t+1)}) = \max\{0,U^{(m_t)}(t)p_a(a,t) - C_a(a,t)\} = \max\{0,U(t)\prod_{m=1}^{m_t} p_f(a^{(m)},t)p_a(a,t) - C_a(a,t)\} \quad (10)
\]

Note that each assigned asset carries a cost so it is possible that a particular assignment will result in a negative net expected reduction; in that case we assign a zero cost to that arc. The maximum number of transshipment nodes associated with a task \(t\), \(M_t\) can be determined from the fact that \(C_f(a,t^{(M_t+1)}) = 0, \forall a\). Alternatively, we can assign a maximum value to \(M_t\) to limit the network size.

Concerning the cost of the arcs towards node 0 we take \(C_f(a,0) = \epsilon > 0, \forall a\), where \(\epsilon\) is a sufficiently small positive value. The use of a positive value for \(C_f(a,0)\) is important for the avoidance of unbounded solutions due to the zero arc costs present in the network. Additionally, \(\epsilon\) should be sufficiently small in order not to be considered as a beneficial assignment.

The arc costs from the transshipment nodes to the demand node are not important so \(C_f(t^{(m)},d) = 0, \forall t, m_t = 1, \ldots, M_t\); their role is to ensure that at most one asset is related to one task.

In practice, the asset assignments are not known beforehand and hence we cannot determine the cost values \(C_f(a,t^{(m)}), m_t > 1\). As a result, we need to develop approximation schemes. A conservative approach, which we call MCFmax, is to always assume that the previously assigned asset to a particular task is the least effective one i.e. the one with the largest execution failure probability \(p_f;max(t) = \max_{a \in A} p_f(a,t)\). Hence, every term \(p_f(a_t^{(m)}, t)\), \(m = 1, \ldots, m_t\) in Eq. (10) will be replaced by \(p_f;max(t)\). An optimistic approach, called MCFmin, is to always consider the most effective asset for previous assignments. If \(p_f;min(t) = \min_{a \in A} p_f(a,t)\) then we set \(p_f(a_t^{(m)}, t) \equiv p_f;min(t)\). A third approximation scheme, called MFRR, is to solve the problem using the RNN association approach and then use the derived allocations to obtain the arc costs for the MCF network. Hence, the terms \(p_f(a_t^{(m)}, t)\) are changed to \(p_f;ran(t^{(m)})\) which denote the probability of execution failure for the \(m\)th asset assigned to task \(t\) according to the RNN approach. Because the complexity of the RNN algorithm is small compared to the solution of an MCF problem the overall execution time is not significantly affected.

An important property of the described flow network is that because \(0 < p_f(a,t) < 1\), the following relationship holds for all cost determination approaches described above:

\[
C_f(a,t^{(1)}) > \ldots > C_f(a,t^{(m_t)}) > C_f(a,0) > 0 \quad (11)
\]

The fact that inequality (11), does not include \(C_f(a,t^{(M_t)})\) implies that \(C_f(a,t^{(m)}) = 0, m = m_t + 1, \ldots, M_t\), so that after the \(m\)th assignment asset \(a\) cannot be assigned to task \(t\). Inequality (11) guarantees the contiguous property as the most beneficial assignment for every asset-task pair is always the first available. For example if for task \(\hat{t}\) \(X_f(a^{(1)}, t^{(1)}) = 1\) and \(X_f(a^{(2)}, t^{(2)}) = 1\), then the next assignment to be made with \(X_f(a^{(3)}, \hat{t}^{(3)}) = 1\). That is because \(C_f(a, \hat{t}^{(3)}) > C_f(a, t^{(m)})\) for all assets \(a\).

The procedure for the solution of problem (2) using the proposed MCF approach is outlined below:
1. Initialise $A_{rem} \leftarrow A$, $S \leftarrow \emptyset$ and $U_{cur}(t) \leftarrow U(t)$, $t \in T$.
2. Compute $C_f(a, t^{(m)})$, $a \in A_{rem}$, $t \in T$ and $m_t = 1, ..., M_t$ according to Eq. (10) and the desired cost approximation scheme.
3. Construct the flow network for $a \in A_{rem}$ and $t \in T$ as in Fig. 1.
4. Solve the MCF problem with negated arc costs to obtain the optimal flows $X_f(a, t^{(k)})$.
5. Set $A_{ass} \leftarrow \{a : X_f(a, t^{(1)}) = 1, \ a \in A_{rem}, \ t \in T\}$.
6. Set $S_{cur} \leftarrow \{(a, t) : X_f(a, t^{(1)}) = 1, \ a \in A_{rem}, \ t \in T\}$ and $S \leftarrow S \cup S_{cur}$.
7. Set $A_{rem} \leftarrow A_{rem} \setminus A_{ass}$.
8. Set $U_{cur}(t) \leftarrow U_{cur}(t) \prod_{a \in \{a, t\} \in S_{cur}} p_f(a, t)$, $t \in T$.
9. If $A_{ass} \neq \emptyset$ and $A_{rem} \neq \emptyset$ go to step (2) otherwise stop: the objective function cannot be reduced further.

The procedure involves the solution of a sequence of MCF problems. At the beginning, we initialise the assignment pairs stored in set $S$ to zero as well as the remaining assets set $A_{rem}$ to $A$. At each iteration, we first compute the arc costs for the remaining asset nodes according to our desired cost estimation scheme and the current task costs $U_{cur}(t)$ and then we construct the network. After solving the MCF problem with negated arc costs, we select the optimal flows with value one that indicate first assignments, store the associated asset-task pairs in the solution set $S$ and remove the assigned assets from the set $A_{rem}$ so that in subsequent iterations they are not considered again. Finally, the task costs are reduced accordingly and the solution procedure is repeated until either we have assigned all assets or no beneficial assignment exist.

The approach proposed in this section is a modified version of the MCF construction based heuristic that was proposed for the solution of the WTA problem in [11]. However, our approach is different in several ways. Firstly, the arc costs in the network are different due to the incurred asset assignment costs $C_a(a, t)$ which are not present in the WTA problem. Secondly, we have modified the network structure to address the possibility of not assigning a particular asset to any task. Thirdly, instead of using a predefined constant number of transshipment nodes for each task, we have chosen to use a different number of transshipment nodes for each task according to the maximum possible number of assignments for a specific task. Finally, we have introduced the MCFrnn approach to efficiently estimate the network arc costs. As will be shown in section 7, this approach leads to the best results overall without adding on the time complexity of the MCF method.

6. Obtaining Tight Lower Bounds

To assess the performance of the discussed algorithms we have developed a tight lower bounding scheme which is based on deriving a piecewise linear approximation of the cost function of alternative formulation (12) of our problem:

$$\min C = \sum_{t \in T} U(t)2^{-z_t}$$

$$\sum_{a \in A} \sum_{t \in T} C_a(a, t)X(a, t) \quad (12)$$

$$s.t. \quad \sum_{a \in A} h(a, t)X(a, t) = z_t, \ \forall t$$

$$X(a, t) \in \{0, 1\}, \forall a, t \text{ and } z_t \geq 0, \forall t$$

To derive formulation (12) let us assume that:

$$2^{-z_t} = \prod_{a \in A} p_f(a, t)^{X(a, t)} \quad (13)$$

Taking the logarithm in both sides of Eq. (13) gives:

$$z_t = -\sum_{a \in A} X(a, t) \log_2 p_f(a, t) = \sum_{a \in A} X(a, t)h(a, t) \quad (14)$$

where $h(a, t) = -\log_2 p_f(a, t)$. Because $0 < p_f(a, t) \leq 1$, it is true that $h(a, t) \geq 0$ and $z_t \geq 0$. Substitution of Eq. (14) into (2) yields formulation (12).

Although the objective function remains nonlinear we can approximate the terms $2^{-z_t}$, $\forall t$ by a piecewise linear approximation function $\phi(z_t)$. In fact, we can obtain a lower bound to the problem’s cost $CLB$, if we make sure that $2^{-z_t} \geq \phi(z_t)$, $z_t \geq 0$. This can be achieved by taking the upper envelop of a number of lines tangent to $2^{-z_t}$. The piecewise linear approximation approach described above, was firstly proposed in [30] for the solution of the WTA problem.

Next, we will discuss how we can reduce the approximation interval for the $2^{-z_t}$ terms by a piecewise linear approximation scheme that restricts the maximum error to a desired value. Then, we will show how we can even remove $\phi(z_t)$, $\forall t$ from the formulation transforming it into a mixed integer optimisation problem with a linear objective function [31].

One problem with the piecewise linear approximation is the large approximation range which in our case is $z_t \geq 0$. We show that we only need to approximate the function for a specific range of values. Clearly, the lower bound for variable $z_t$ is 0, attained when no asset is assigned to task $t$. If there is at least one allocation for task $t$ then $z_t \geq z_t^{\text{min}} > 0$, where $z_t^{\text{min}}$ is the smallest positive value of $z_t$. This value is acquired by assigning the asset with the largest execution failure probability to task $t$,

$$z_t^{\text{min}} = -\log_2(\max_{a} \{p_f(a, t)\}) \quad (15)$$

We can also limit the approximation range by deriving an upper bound for $z_t$. Due to the asset
cost $C_a(a,t)$, assignments are made only when the net expected reduction in the objective function is positive. In other words, a new asset $a$ can be assigned to task $t$ if $U_{cur}(t)p_a(a,t) - C_a(a,t) > 0$, where $U_{cur}(t)$ is the expected cost of the task due to other assignments. If we consider the marginal case $U_{mar}(t) = C_a(a,t)/p_a(a,t)$, then the expected task cost after the assignment is made will be equal to $U_{mar}(t)p_a(a,t) = C_a(a,t)p_a(a,t)/p_a(a,t)$. Hence the smallest expected cost that we can obtain for task $t$ is $U_{min}(t) = \min_{a \in A} \{C_a(a,t)p_a(a,t)/p_a(a,t)\}$ so that:

$$z_{t}^{\max} = -\log_2 \left( \frac{\min_{a \in A} \{C_a(a,t)p_a(a,t)\}}{U(t)} \right)$$

(16)

One approach for obtaining the piecewise linear approximation of a function is to take the upper envelop of the lines tangent to it at integer multiples of a parameter $\kappa$. Nonetheless, using this approach we do not have any knowledge of the approximation error, while the maximum error for different segments varies. Our approach is to create lines whenever needed so that the approximation error does not exceed a predefined value $e_{\text{max}}$. Although this approach may lead to a large number of segments we can adjust $e_{\text{max}}$ to achieve the desirable number of segments and at the same time maintain the error less than a constant known value.

Fig. 2 depicts an example of piecewise linear approximation of term $2^{-z_t}$ when $e_{\text{max}} = 0.02$, $z_t^{\min} = 1.94$ and $z_t^{\max} = 8.62$. The solid thin line represents $2^{-z_t}$ while the dashed lines represent the various approximating lines. The thick line, which is the upper envelop of the approximating lines, corresponds to $\phi(z_t)$. On Fig. 2 the piecewise linear approximation function is tangent to $2^{-z_t}$ at points $P_1$, $P_2$, and $P_3$, whereas $P_2$, $P_3$ and $P_5$ are points of maximum approximation error.

In order to create $\phi(z_t)$ we follow an iterative procedure which involves the numerical solution of two equations. The first equation corresponds to the case that one point of maximum approximation error, $(z_t^{\max}, 2^{-z_t^{\max}} - e_{\text{max}})$, of a particular line segment is known and we want to obtain the point $z_t^{\max} > z_t^{\max}$ that is tangent to $2^{-z_t}$. This point is determined from the numerical solution of Eq. (18):

$$2^{-z_t^{\max}} - e_{\text{max}} = 2^{-z_t^{\max} - \ln 2}$$

(17)

The second equation corresponds to the case that one point of maximum approximation error, $(z_t^{\max}, 2^{-z_t^{\max}} - e_{\text{max}})$, of a particular line segment is known and we want to obtain the point $z_t^{\max} > z_t^{\max}$ that is tangent to $2^{-z_t}$. This point is determined from the numerical solution of Eq. (18):

$$2^{-z_t^{\max}} - (2^{-z_t^{\max} - \ln 2}) = 2^{-z_t^{\max} - \ln 2}$$

(18)

We now outline the approach followed to derive $\phi(z_t)$. Starting from point $(0,1)$ we solve Eq. (17) to obtain the point $z_t^{P_2}$, where the approximation error is equal to $e_{\text{max}}$. Since no approximation is needed in the interval $0 < z_t < z_t^{\min}$, we set the start of the new segment to $z_t^{P_2}$ only when $z_t^{P_2} > z_t^{\min}$ otherwise we impose point $(z_t^{\min}, 2^{-z_t^{\min}} - e_{\text{max}})$ to be a point on the second line segment (point $P_3$). Then the procedure alternates between the solution of Eq. (18) and (17). The second segment is fully determined by solving Eq. (18) to obtain point $P_5$. Then, point $P_5$ is used to find $P_6$ by solving Eq. (17); point $P_6$ denotes the end of the particular line segment and the start of the next one. The described procedure is repeated until an attained point is larger than $z_t^{\max}$.

Having obtained an approximating function $\phi(z_t), t \in T$, we now describe how to obtain a problem formulation with a linear cost function. Let us assume that the term $2^{-z_t}$ is approximated by $L_1$ linear segments with slopes $\alpha^{(1)}_1, \ldots, \alpha^{(L_1)}_1$ and start-points $z^{(1)}_t, \ldots, z^{(L_1)}_t$. Let us also assume that $z_t^{(L_1+1)} = z_t^{\max}$. Because $2^{-z_t}$ is convex, the envelop approximation $\phi(z_t)$ will also be convex and the slopes will have monotone increasing values: $\alpha^{(1)}_1 < \alpha^{(2)}_1 < \ldots < \alpha^{(L_1)}_1$.

Let $\xi^{(l)}_t, l = 1, \ldots, L_t$ be the value of $z_t$ corresponding to the $l$th linear segment so that $0 \leq \xi^{(l)}_t \leq z^{(l)}_t - z^{(l-1)}_t$, $l = 1, \ldots, L_t$. Under the assumption that $z^{(l)}_t = z^{(l+1)}_t - z^{(l)}_t$, $i = 1, \ldots, L_t - 1$ when $\xi^{(l)}_t > 0$, it is true that $z_t = \sum_{i=1}^{L_t} \xi^{(l)}_t$ and also that $\phi(z_t) = \sum_{i=1}^{L_t} \xi^{(l)}_t$. In other words, $z_t$ can be replaced by the sum of variables $\xi^{(l)}_t, l = 1, \ldots, L_t$ if we can ensure that the solution of the optimisation problem will always be such that each $\xi^{(l)}_t$ is nonzero only when the variables $\xi^{(l)}_t, i = 1, \ldots, L_t - 1$ have obtained their maximum value. As mentioned earlier, $\alpha^{(1)}_1$ has the smallest slope value and hence $\xi^{(1)}_t$ will be the first variable associated with $z_t$ to be assigned a nonzero value. Only when $\xi^{(l)}_t$ has been assigned its maximum value variable $\xi^{(2)}_t$ will be assigned a nonzero value and this procedure will continue until $z_t$ becomes equal to the sum of the nonzero variables. Thus, the assumption stated above is satisfied and formulation (12) becomes:

$$\min \sum_{t \in T} U(t) \sum_{l=1}^{L_t} \alpha^{(l)}_1 \xi^{(l)}_t + \sum_{t \in T} \sum_{a \in A} C_a(a,t)X(a,t)$$

(19)

s.t. $\sum_{t \in T} X(a,t) \leq 1, \forall a$

$$\sum_{a \in A} h(a,t)X(a,t) = \sum_{l=1}^{L_t} \xi^{(l)}_t, \forall t$$

$0 \leq \xi^{(l)}_t \leq z^{(l+1)}_t - z^{(l)}_t, l = 1, \ldots, L_t, \forall t$

$$X(a,t) \in \{0,1\}, \forall a, t$$

Formulation (19) is a linear mixed integer program that can be solved using a standard combinatorial

The Computer Journal, Vol. ??, No. ??, ????
optimisation solver.

We are now ready to outline the steps of the lower bounding algorithm (LBA) for problem (2):

1. For each task compute $z_{t}^{\text{min}}$ and $z_{t}^{\text{max}}$ using Eqs. (15) and (16) respectively.
2. Follow the proposed piecewise linear approximation scheme to compute the $L_t$ linear segments with slopes $\alpha(t)^{(1)}, \ldots, \alpha(t)^{(L_t)}$, $z_{t}^{(L_t+1)} = z_{t}^{\text{max}}$ based on the desired value of $e_{\text{max}}$.
3. Use an integer programming solver to solve problem (19). The obtained objective function cost corresponds to a lower bound of the original problem.

7. EVALUATION

The effectiveness of the proposed algorithms was tested with respect to two generated data families. In data family 1, the problem parameters are independently generated, while in data family 2 there is positive correlation between the cost of an asset and its associated execution success probabilities, so that “better” assets are more expensive.

In both data families, parameters $U(t)$ 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 $C_{a}(a) \in [4,30]$, $a \in A$ is taken to be independent from its assigned task, while for the execution failure probabilities we have that $p_f(a,t) \in [0.05,0.4]$. In data family 2, the asset execution failure probabilities are taken to be independent from the tasks, i.e. $p_f(a) \in [0.05,0.4]$, while the associated asset costs $C_{a}(a)$ are drawn from the normal distribution with mean $\overline{C}_a(a)$ and variance $0.1\overline{C}_a(a)$. The parameters $\overline{C}_a(a)$ are calculated from linear equation (20) that connects points $(p_f, \overline{C}_a) = (0.4, \overline{C}_a, 30)$. The parameters $\overline{C}_a(a)$ are calculated from linear equation (20) that connects points $(p_f, \overline{C}_a) = (0.4, \overline{C}_a, 30)$.

$$C_{a}(a) = \frac{(\overline{C}_a \cdot \overline{C}_a - \overline{C}_a \cdot \overline{C}_a)}{(p_f - p_f, \overline{C}_a - \overline{C}_a)} \quad (pf, \overline{C}_a) = (0.4, \overline{C}_a, 30)$$

To test the effectiveness of the proposed algorithms, we have performed two sets of experiments. The first set of experiments is conducted with small size asset-task problem sets where the optimal solution can be obtained, while the second set is conducted with large size problems and the results are compared against the lower bounding algorithm (LBA).

For the small size problems, we have generated problem instances for constant number of assets, $|A| = 6$, and varying number of tasks $|T| = 3, 6, 9, 12, 15$ as well as for constant number of tasks, $|T| = 6$, and varying number of assets $|A| = 3, 9, 12, 15$, for both data families described above. To compare the performance of the various algorithms we have used as
while for the other cases the best performing algorithm is better only for the cases that worse than those of data family 1. The MCFmin algorithm and in all cases other cases. The RNN approach clearly outperforms the number of tasks. MCFmax performs well only when cases that the number of assets is smaller than the MCFmin and MCFrnn obtain optimal results in all which achieves slightly worse results. Remarkably, both exceed 1% in all cases, as is the case for the MCFrnn relative percentage deviation from the optimal not best performing method is the MCRmin with average each (denote the best performing approach or approaches for data family 1 and 2 respectively. The bold letters i
\[ C \]
\[ C \] where \( C \) is the cost of the lower bound obtained from the heuristic algorithm used, and \( N_{PI} \) is the total number of problem instances considered in each case. The number of problem instances used was \( N_{PI} = 300 \) in all cases.

The results are summarised in Tables 1 and 2 for data family 1 and 2 respectively. The bold letters denote the best performing approach or approaches for each (\(|A|, |T|\)) pair. For data family 1, clearly the best performing method is the MCRmin with average relative percentage deviation from the optimal not exceeding 1% in all cases, as is the case for the MCFmin which achieves slightly worse results. Remarkably, both MCFmin and MCFrnn obtain optimal results in all cases that the number of assets is smaller than the number of tasks. MCFmax performs well only when |A| > |T|, while its performance is the worst for the other cases. The RNN approach clearly outperforms the maximum marginal return algorithm and in all cases \( \sigma_{opt} < 5\% \).

The results obtained for data family 2 are overall worse than those of data family 1. The MCFmin algorithm is better only for the cases that \(|A| \geq 2|T|\), while for the other cases the best performing algorithm is MCFrnn. The RNN approach again outperforms the MMR algorithm, and its performance remains within 5% of optimality. Moreover, RNN’s effectiveness is comparable to the best performing methods MCFmin and MCFrnn. From this first set of experiments we can conclude that the MCFmin and MCFrnn approaches are the best performing algorithms, whilst the performance of the algorithms significantly depends on the ratio \(|A|/|T|\).

We have performed a second set of experiments with large problem instances, for several (|A|, |T|) pairs with up to 200 assets and 200 tasks. Due to the large size of the problems, the optimal solution is difficult to be derived, so the performance of the algorithms is compared against tight lower bounds obtained from the solution of (19). To obtain tight lower bounds in relatively short execution period, the parameter \( \epsilon_{max} \) has been carefully selected for each (|A|, |T|) pair to accomplish a good trade-off between execution time and optimality. The performance measure that we use is the average relative percentage deviation from the lower bound, \( \sigma_{LB} \), defined as:

\[
\sigma_{LB} = \frac{1}{N_{PI}} \sum_{i=1}^{N_{PI}} \frac{C_{alg,i} - C_{LB,i}}{C_{LB,i}} \times 100
\]  

where \( C_{LB,i} \) is the cost of the lower bound obtained from the solution of (19).

We report \( \sigma_{LB} \) for the various algorithms in Tables 3 and 4. Column LBA corresponds to the cost of the original problem (2), computed using the solution obtained from the lower bounding algorithm. LBA

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Average relative percentage deviation from the optimal solutions of data family 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td>A</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>15</td>
<td>6</td>
</tr>
<tr>
<td>Overall Perf.</td>
<td>1.674</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Average relative percentage deviation from the optimal solutions of data family 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td>A</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>15</td>
<td>6</td>
</tr>
<tr>
<td>Overall Perf.</td>
<td>1.698</td>
</tr>
</tbody>
</table>
is only considered to demonstrate the tightness of the lower bounds and is not compared with the other methods, as it is not of polynomial complexity. Moreover, the MMR is a greedy heuristic, where in each iteration we select the assignment corresponding to the maximum marginal return, represented by term $\max \{0, b(a, t)\}$.

For data family 1 the most effective algorithms are the network flow approaches MCFmin and MCFrnn for all $(|\mathcal{A}|, |\mathcal{T}|)$ pairs, which have almost the same efficiency and achieve $\sigma_{LB} < 1.3\%$ in all cases. In addition, these algorithms even outperform the LBA approach for the cases that $|\mathcal{A}|/|\mathcal{T}| \leq 1$. Additionally, MCFmax only performs well when $|\mathcal{A}|/|\mathcal{T}| \leq 1$. The RNN approach is worse than the network flow heuristics, but performs by approximately 4% better than the MMR algorithm.

Interestingly, for data family 2 the best overall performing algorithm is the RNN. RNN performs better than the other approaches for large problem instances when $|\mathcal{A}| = |\mathcal{T}|$ and $2|\mathcal{A}| = |\mathcal{T}|$, while for the other problems its performance is highly competitive.

| $|\mathcal{A}|$ | $|\mathcal{T}|$ | $|\mathcal{A}|/|\mathcal{T}|$ | RNN | MMR | MCFmax | MCFmin | MCFrnn | LBA |
|-----|-----|-----|-----|-----|------|------|------|------|
| 20  | 80  | 0.25 | 0.273 | 0.279 | 0.273 | 0.273 | 0.273 | 0.286 |
| 40  | 120 | 0.33 | 0.421 | 0.434 | 0.423 | 0.423 | 0.423 | 0.447 |
| 10  | 20  | 0.5  | 0.489 | 0.534 | 0.482 | 0.482 | 0.482 | 0.514 |
| 20  | 40  | 0.5  | 0.486 | 0.540 | 0.486 | 0.486 | 0.486 | 0.527 |
| 40  | 80  | 0.5  | 0.505 | 0.561 | 0.509 | 0.508 | 0.508 | 0.553 |
| 80  | 160 | 0.5  | 0.684 | 0.740 | 0.683 | 0.683 | 0.683 | 0.751 |
| 100 | 200 | 0.5  | 0.717 | 0.773 | 0.644 | 0.716 | 0.722 | 0.801 |
| 10  | 10  | 1.0  | 0.833 | 2.554 | 0.638 | 1.336 | 0.467 | 0.067 |
| 20  | 20  | 1.0  | 0.708 | 3.031 | 0.765 | 1.577 | 0.589 | 0.094 |
| 40  | 40  | 1.0  | 0.673 | 3.384 | 0.822 | 1.710 | 0.652 | 0.140 |
| 80  | 80  | 1.0  | 0.777 | 3.719 | 1.007 | 1.826 | 0.806 | 0.518 |
| 100 | 100 | 1.0  | 0.932 | 3.927 | 1.192 | 2.018 | 1.640 | 0.789 |
| 10  | 5   | 2.0  | 3.301 | 3.610 | 4.896 | 2.644 | 2.540 | 0.040 |
| 20  | 10  | 2.0  | 3.024 | 3.376 | 5.581 | 3.040 | 2.894 | 0.053 |
| 40  | 20  | 2.0  | 2.619 | 3.428 | 5.980 | 3.149 | 3.039 | 0.077 |
| 80  | 40  | 2.0  | 2.322 | 3.324 | 6.266 | 3.258 | 3.086 | 0.784 |
| 100 | 50  | 2.0  | 2.382 | 3.393 | 6.441 | 3.429 | 3.307 | 0.580 |
| 200 | 100 | 2.0  | 2.115 | 3.298 | 6.333 | 3.533 | 3.320 | 1.213 |
| 40  | 10  | 4.0  | 6.840 | 6.764 | 9.930 | 7.473 | 7.675 | 0.064 |
| 80  | 20  | 4.0  | 6.279 | 6.078 | 9.825 | 7.132 | 7.445 | 0.093 |

TABLE 3. Average relative percentage deviation from the lower bound of data family 1

TABLE 4. Average relative percentage deviation from the lower bound of data family 2

| $|\mathcal{A}|$ | $|\mathcal{T}|$ | $|\mathcal{A}|/|\mathcal{T}|$ | Overall Perf. | Overall Perf. |
|-----|-----|-----|-----------------|-----------------|
| 20  | 80  | 0.25 | 2.417 | 5.500 | 6.499 | 0.449 | 0.468 | 0.336 |
MCFrnn achieves better results than the MCFmin approach, especially for the problem sets with equal number of assets and tasks. The performance of the MMR approach is well improved compared to data family 1; in fact the MMR obtains the best results for problem instances with $|\mathcal{A}|/|\mathcal{T}| = 4$. Finally, the MCFmax is again the least effective approach.

Summarising the results, we could argue that the best performing method is the MCFrnn approach since it performs similar to the MCFmin for data family 1, whilst it is more effective on data family 2. At this point, it is important to highlight that the RNN solution attained in MCFrnn is not considered to improve the performance of the MCFrnn. If this was taken into account, then the outcome from the MCFrnn algorithm would always be at least equal to the outcome of RNN, as we could choose the best solution amongst the two approaches; in that case, the MCFrnn would be the best performing method for data family 2. Finally, the performance of the RNN approach should also be emphasised as it is the most efficient method for data family 2, while it is also of low polynomial complexity.

8. CONCLUSIONS

In this paper we have studied an asset-task allocation problem when an asset may fail to execute an assigned task. For its solution, we have proposed algorithms three different network flow approaches. We have also proposed an algorithm for obtaining tight lower bounds to the optimal solution of the examined problem.

We have tested the efficiency of the various algorithms with respect to two different data families, the one with all parameters being independent and the other with the assignment costs having a positive correlation to the execution success probabilities. Summarising the results, we could argue that the proposed network flow algorithms MCFmin and MCFrnn have better overall performance than the MMR and RNN heuristics, with the MCFrnn being the most successful; it performs equally well with the MCFmin for the independent data family and it is more effective on the dependent data. This indicates that the incorporation of RNN into the MCF approach has a positive effect. The performance of the RNN algorithm should also be highlighted, as it is the most efficient method for the dependent data family.

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


