# Approaches to Obtain a Large Number of Ranked Solutions to 3-Dimensional Assignment Problems

**LINGYI ZHANG**
**DAVID SIDOTI**
**SPANDANA VALLABHANENI**
**KRISHNA R. PATTIPATI**
**DAVID A. CASTAÑÓN**

**A generalized 3-dimensional assignment problem is a decision-making process that involves allocating limited resources to a set of tasks over time, where the objective is to optimize a cost function subject to a set of generalized assignment constraints. The 3-dimensional (3-D) assignment problems are known to be NP-hard. In this paper, we propose a novel approach to efficiently solve an $m$-best 3-D assignment problem with non-unity right-hand side constraints (also referred to simply as 3-D assignment problem), where $m$ may be large (as many as $10^4$ solutions), by decomposing it into two sequential phases. In phase I, we partition the original problem space into a series of subproblems via Murty's $m$-best search space decomposition procedure. Modifications previously proposed in the literature for the 2-dimensional (2-D) assignment problem are applied to optimize the search space decomposition for the 3-D assignment problem. In phase II, we solve each subproblem by using Lagrangian relaxation and solving the 3-D assignment problem as a combination of relaxed 2-D assignment problems and 2-D transportation problems. The 2-D assignment problem is solved by the JVC or auction algorithms, and the 2-D transportation problem is solved by the simplex-based transportation, Transauction or RELAX-IV algorithms. The sequence of relaxed 2-D problems are interchangeable, while adhering to the relaxed constraints. We validate and compare the performance and utility of the proposed algorithms and search space decomposition optimizations via extensive numerical experiments. Overall, the fully optimized algorithm took less than 50 seconds, on average, to obtain $10^4$ soutions for a tensor of dimension $30 \times 30 \times 8$.**

Manuscript received Sept. 9, 2016; revised February 7, 2017; released for publication June 26, 2017.

Refereeing of this contribution was handled by Stefano Coraluppi.

Authors' addresses: L. Zhang, D. Sidoti, S. Vallabhaneni, and K. R. Pattipati, Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT, 06269 USA (E-mail: lingyi.zhang@uconn.edu; krishna.pattipati@uconn.edu). D. A. Castañón, Boston University, Boston, MA, 02215 USA (E-mail: dac@bu.edu).

Research supported by the U.S. Office of Naval Research under contract #N00014-16-1-2036 and by the Department of Defense High Performance Computing Modernization Program under subproject contract #HPCM034125HQU.



## 1. INTRODUCTION

### 1.1. Motivation

Assignment problems are applicable to a diverse array of real world problems [1]–[3]. This set of problems takes the form of how best to assign a number of items or objects to some (possibly different) number of machines or people during different time periods. Assignment problems are of a combinatorial nature, each requiring some form of an objective function to indicate the value or utility of individual assignments. A sampling of how diverse and widely applicable such assignment problems are can be seen from the following: multi-target tracking, quadratic assignment problems, traveling salesman problems, or vehicle routing problems. Such problems also occur in academia or the military, where a set of military troops [1] or teachers [2] must be assigned to locations or classrooms that are temporally dependent in value or utility. Assignment problems have even been motivated from a telecommunications standpoint, where a set of satellites must be launched from a set of locations to maximize their coverage [1].

A 2-dimensional (2-D) assignment problem may be viewed as a weighted bipartite graph matching problem, where arcs must link two sets of nodes together such that an objective function is optimized, while satisfying a set of one-to-one constraints. The 3-dimensional (3-D) extension of this problem has been proven to be NP-hard [4]–[6]. In particular, one application that we focus on in this paper is a nuclear fuel assembly (FA) loading pattern optimization. The core of a nuclear reactor is formed by large sets of elongated, rectangular FAs arranged in a cylindrical fashion, as shown in Fig. 1.

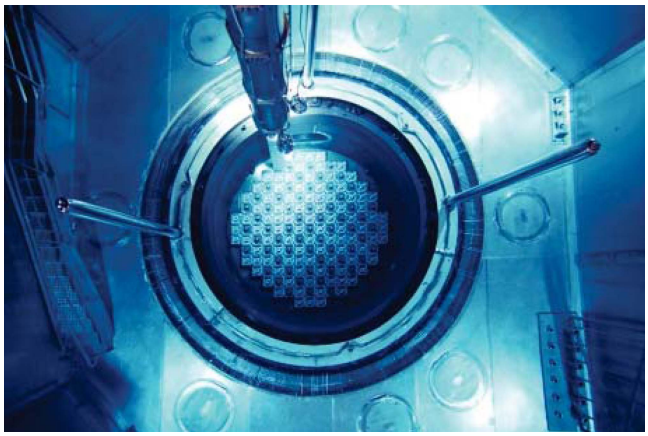


Fig. 1. The core of a nuclear reactor is formed by large sets of fuel assemblies where position, type, and rotation/orientation must be chosen for each one. Illustrated here is a nuclear fuel assembly loading operation at Fangchenggang nuclear power plant in China's Guangxi province [7].

The nuclear fuel assembly loading pattern optimization problem involves choosing: 1) the position of the FA in the nuclear reactor core, 2) the type of FA to put

in the chosen position, and 3) the rotation/orientation of the chosen FA type in the chosen position. Each dimension of the 3-D assignment corresponds to each of the decision variables above. In general, this problem is treated as a multiple objective combinatorial problem, but what separates it from the traditional 3-D assignment problems is the requirement for a dense set of new discrete loading patterns though a dynamically estimated probability distribution (represented by a reward tensor). This conversion to a 3-D assignment problem is a completely new approach for nuclear fuel loading pattern optimization. The reward tensor is dynamically updated based on the "best" solutions taken from the multi-objective Pareto front. "Best" in this case may not necessarily refer to the optimal, but one of a large number of solutions (assignments). By "large," we mean on the order of $10^4$ solutions. Evaluation of each loading pattern by reactor-physics-based external code may be very time consuming ($\approx 0.1$ to 10 minutes, depending on the required accuracy of loading pattern response evaluation), so there exists a need to evaluate only new (unique) loading patterns (assignments).

In such scenarios, an $m$-best 3-D assignment problem is needed, wherein a large set of solutions is generated in a reasonable amount of time ($< 10$ minutes for $10^4$ solutions), so that the set of assignments may be externally evaluated (each of which, in turn, may take 0.1 to 10 minutes). It may also be a viable approach to obtain a dense set of solutions that are near-optimal and satisfy the decision maker (such as in the case of resource allocation or military troop allocation problems) or customer preferences (as in [2], where they attempt to satisfy both student and tutor requirements or requests). Having a large set of solutions offers a range of options that may be of interest to a decision maker attempting to optimize with respect to multiple, possibly conflicting, objectives.

This paper offers an effective solution approach for finding a large number of $m$-best solutions to the 3-D assignment problems with non-unity right-hand side constraints with application to many real world challenges. The problem space may be decomposed into multiple partitions based on the optimal assignment, as detailed in [8]. Through a two-phase approach, we offer a method for rapidly generating large numbers of solutions to the 3-D assignment problems.

### 1.2. Related Research

There exist a number of well-known algorithms to obtain the optimal solution to a 2-D assignment problem, including the Hungarian algorithm [9], the Jonker-Volgenant-Castañón (JVC) algorithm [10], [11], the auction algorithm [12], and the signature method [13]. However, the assignment problem becomes NP-hard when a third dimension is added [4]–[6]. One of the first approaches for solving the 3-D assignment problem was developed by Pierskalla [1], where he proposed a tri-substitution algorithm based on the simplex method. Hansen [14] proposed a primal-dual implicit enumeration algorithm, while [15], [16] proposed branch-and-bound approaches to obtain the optimal solution to such 3-D assignment problems. However, branch-and-bound methods suffer from exponential computational complexity and are unsuitable for large-scale real-world applications where accurate bounds cannot be obtained.

In order to overcome the 3-D assignment problem's inherent computational intractability, a wide range of algorithms have been developed to obtain suboptimal solutions, including greedy heuristics, genetic algorithms, simulated annealing, tabu search, neural networks, and Lagrangian relaxation approaches [2], [17]–[20]. Mazzola [17] proposed a heuristic branch-and-bound method to reduce the computation time. In contrast, Frieze and Yadegar [2] applied Lagrangian relaxation theory to a more general 3-D assignment problem with application to teaching practice scheduling. The Lagrangian relaxation method of obtaining solutions to 3-D assignment problems has become extremely prevalent in data association applications due to the real time computation speed and solution quality [3], [18], [19]. Poore [20] combined these two approaches, proposing a hybrid branch-and-bound and Lagrangian relaxation algorithm to the 3-D assignment problem.

In this paper, we seek to solve the aforementioned 3-D assignment problem, but instead of finding a single solution, we aim to provide a large set of ranked solutions. The process of finding the first best, second best, third best, and so on, solution is known as the $m$-best optimization problem. The $m$-best optimization problem occurs in a variety of contexts, including the shortest path [21]–[23], spanning tree [24]–[26], traveling salesman [27], directed network [28], multi-target tracking [29]–[33] and many other problems. The general approach to the $m$-best optimization problem involves partitioning the solution space into smaller subspaces, which are subproblems of the original problem. Murty's search space decomposition [8] is the most common and widely used technique, where the best solution is found for each partitioned subproblem, given a modified solution subspace. Lawler [34] applied Murty's search space decomposition procedure within a more general framework for a discrete optimization problem. Pascoal [35] proposed a variant of Murty's search space decomposition to reduce the algorithm's complexity. This variant involved solving the partitioned subsets in reverse order. Miller et al. [36] proposed modifications to optimize Murty's search space decomposition procedure to the 2-D assignment problem via: 1) inherited dual variables and partial solutions from the initial subproblems; 2) sorting the subproblems based on lower bounds on the optimal reward before solving the assignment problem; and 3) partitioning in an order based on lower bounds on cumulative reward. These modifications substantially reduce the complexity of Murty's search space decomposition and are implemented in this paper.

Another alternative way to solve the $m$-best optimization problem is by Gabow's [24] binary heap partition method. Similarly, Hamacher [37] also proposed using a binary search tree procedure, while also combining an approach developed by Carraresi and Sodini [38] to rank the paths. Chegireddy and Hamacher [39] extended this work further and developed an $m$-best perfect matching algorithm based on the binary partition of the solution space to apply to a bipartite matching problem in $\mathcal{O}(kn^3)$ time. Recently, a modified version of the Chegireddy and Hammacher's algorithm was developed for large datasets [40]. We suggest comparison of our algorithm with those in [40] as future research.

### 1.3. Paper Organization

The primary focus of this paper is on combining a Lagrangian relaxation method and $m$-best optimization to obtain a very large number of ranked solutions. Motivated by an approach developed by Pattipati [18], we apply the Lagrangian relaxation approach that successively solves a series of 2-D problems, since a key advantage of using the Lagrangian relaxation method is that it prunes the solution space by computing the upper and lower bounds. The first 2-D problem is a bipartite graph matching problem (2-D assignment problem), which can be solved using either the auction algorithm or the JVC algorithm [10]; the latter is more efficient for dense problem spaces [11]. The feasible solution is obtained by solving a 2-D transportation problem (via a simplex algorithm or Transauction algorithm) reconstructed from the relaxed solution of the 2-D assignment problem. The second step corresponds to imposing the originally relaxed constraint on the first subproblem's solutions. As in [33], we generate $m$-best solutions by exploiting Murty's search space decomposition procedure; however, unlike the formulation in [33], the present 3-D formulation has general constraints that require a transportation problem to be solved. Moreover, we optimize Murty's search space decomposition via Miller's [36] proposed modifications, resulting in further speedup and improved computational performance. An alternate Lagrangian relaxation method involves first solving a 2-D transportation problem at each iteration of the 3-D assignment algorithm using either a simplex algorithm or the Transauction algorithm, and subsequently reconstructing the feasible solution via a 2-D assignment problem. We will show that the former Lagrangian relaxation method is two orders of magnitude faster than the latter.

This paper is organized as follows. We begin by introducing the problem formulation in Section 2. In Section 3, we solve the $m$-best 3-D assignment problem via Murty's search space decomposition and the Lagrangian relaxation method. In Section 4, we detail Miller et al.'s [36] search space optimizations and extend them to the 3-D assignment problem. We provide the pseudocode of the fully optimized $m$-best 3-D assignment solution algorithm in Section 5. In Section 6, we present the results of the $m$-best 3-D assignment algorithm and the performance of each different optimization technique. Finally, we provide concluding remarks in Section 7.

TABLE I
Summary of Notation

| | |
|---|---|
| $w_{ijk}$ | Reward of allocating resource $i$ to task $j$ at time $k$ |
| $x_{ijk}$ | Binary decision variable for the primal problem |
| $y_{ij}$ | Binary decision variable for the 2-D assignment problem |
| $z_{jk}$,$z_{ik}$ | Binary decision variables for the 2-D transportation problem |
| $i$ | Resource index |
| $j$ | Task index |
| $k$ | Time index |
| $m_k$ | Maximum number of assignment allowed for each $k$ |
| $W$ | Reward tensor |
| $N$ | Total number of tasks/resources |
| $R$ | Total number of time units |
| $\mu$ | Lagrange multiplier |
| $q$ | Upper bound found from the relaxed 2-D assignment problem (dual) |
| $f$ | Lower bound found via simplex-based transportation or Transauction problem (primal) |
| $\underline{g}$ | Gradient vector for the subgradient update |
| $\mathcal{P}_0$ | Original problem space |
| $A$ | Solution space |
| $S$ | Feasible assignment in solution space $A$ |
| $X$ | Solution tensor |
| $\Phi$ | Column for row solution |
| $\phi$ | Optimal reward from the 2-D assignment problem |
| $\Omega$ | Layer for row solution |
| $\omega$ | Optimal reward from the 2-D transportation problem |
| $B$ | Slack value for upper bound reward computation |
| $C$ | Relaxed 2-D reward matrix in the 2-D assignment problem |
| $T$ | 2-D reward matrix for the transportation problem |

## 2. PROBLEM FORMULATION

The notation used in the remainder of this paper is listed in Table I.

### 2.1. Problem Formulation

Given a 3-D reward tensor $W = [w_{ijk}]$ of dimension $N \times N \times R$, our problem is the following:

$$\max_{x_{ijk}\in\{0,1\}} \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{R} w_{ijk}x_{ijk} \tag{1}$$

$$s.t. \sum_{j=1}^{N}\sum_{k=1}^{R} x_{ijk} = 1,\ i = 1,\ldots,N \tag{2}$$

$$\sum_{i=1}^{N}\sum_{k=1}^{R} x_{ijk} = 1,\ j = 1,\ldots,N \tag{3}$$

$$\sum_{i=1}^{N}\sum_{j=1}^{N} x_{ijk} \le m_k,\ k = 1,\ldots,R \tag{4}$$

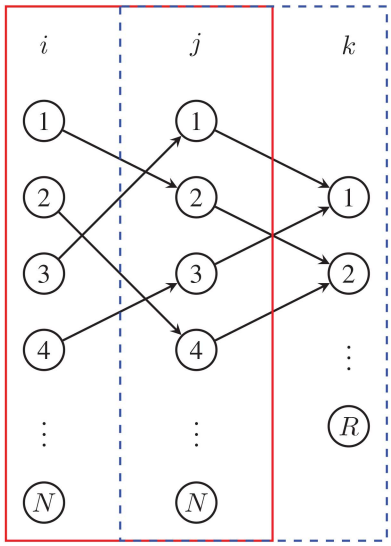



Fig. 2. Network flow view of the 3-D assignment problem, originally presented in [42].

where $x_{ijk}$ is a binary decision variable such that $x_{ijk} = 1$ if resource (row) $i$ is assigned to task (column) $j$ at time (layer) $k$, and 0 otherwise. Constraints (2) and (3) ensure that each resource $i$ is allocated to exactly one task $j$ and vice versa. Constraint (4) requires that there may be no more than $m_k$ assignments at each time $k$ and makes this assignment problem non-standard.

Figure 2 shows the 3-D assignment problem as a network flow problem. Consider the first set, indexed by $i$, and the second set, indexed by $j$, each consisting of $N$ nodes. Also, consider a third set, indexed by $k$, with a total of $R$ nodes. There are a total of $N$ assignments that may be made between sets $i$ and $j$ based on constraints (2) and (3). We view this as a 2-D assignment problem (indicated by the solid box in Fig. 2). Additionally, each node in set $j$ must be assigned to one of the nodes in set $k$ (indicated by the dashed (blue) box in Fig. 2). Due to constraint (4), for every $k$, there may be no more than $m_k$ assignment pairs of $(i,j)$ mapped to each layer. This may be viewed as an unbalanced transportation problem, where the nodes in set $(i,j)$ are the sources and the nodes in set $k$ are the sinks. Note that $\{m_k : k = 1,2,\ldots R\}$ should be such that $\sum_{k=1}^{R} m_k \geq N$ so that each $(i,j)$ can be assigned to a node $k$.

## 2.2. Special Cases

Note that our 3-D assignment problem formulation covers a wide range of problems.

2.2.1. Tri-index Assignment problem: The problem in (1)–(4) may be viewed as a traditional tri-index assignment problem by setting $m_k = 1$ and $R = N$ [1].

2.2.2. Scheduling problem: By setting $m_k = m$, the problem in (1)–(4) is related to some resource-constrained assignment scheduling problems [41].

2.2.3. Transportation problem: Note that our problem formulation is a special case of the transportation problem. The general transportation problem involves altering the unity constraint to some non-unity values.

2.2.4. Nuclear Fuel Loading Pattern Optimization: In some nuclear reactor fuel assembly loading pattern optimization problems, $m_k = N$ on the right hand side of the constraint (4). In this case, the problem can be reduced to the traditional 2-D assignment problem, since constraint (4) can be subsumed under constraints (2) and (3) and is, thus, unnecessary. The 3-D assignment problem posed in (1) then devolves to a 2-D assignment problem, detailed later in Section 3.1.4. An $m$-best 2-D assignment problem is adequate for this version of the problem.

# 3. SOLUTION APPROACH

In order to solve this NP-hard problem, we propose a two-phase solution approach. In phase I, we utilize Murty's search space decomposition to partition the original problem space into a series of subproblems. Each subproblem is then relaxed and solved by a 3-D assignment algorithm in phase II.

## 3.1. 3-D Assignment Relaxations

We adopt the solution approach of the 3-D assignment problem in [18] by relaxing one of the three constraints and solving the 3-D assignment problem as a series of 2-D subproblems. Since sets $i$ and $j$ have the unity constraint, a similar solution approach can be applied to the 3-D assignment problem here by relaxing either of the two sets of constraints. We then denote Relaxation Method I and Relaxation Method II as the solution approaches for the 3-D assignment problem when constraints (4) or (2)/(3) are relaxed, respectively.

3.1.1. Relaxation Method I: Relaxation Method I is developed by relaxing constraint (4) via a set of Lagrange multipliers $\{\mu_k : k = 1,2,\ldots,R\}$. The result is the Lagrangian function

$$L(x,\mu) = \max_{x_{ijk}\in\{0,1\}} \left( \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{R} (w_{ijk} - \mu_k) x_{ijk} \right) + m_k \sum_{k=1}^{R} \mu_k \tag{5}$$

Equation (5) is then a relaxed 2-D assignment problem of the form,

$$\max_{y_{ij}\in\{0,1\}} \sum_{i=1}^{N}\sum_{j=1}^{N} \max_{k} (w_{ijk} - \mu_k) y_{ij} \tag{6}$$

$$s.t. \sum_{i=1}^{N} y_{ij} = 1,\ j = 1,\ldots,N \tag{7}$$

$$\sum_{j=1}^{N} y_{ij} = 1,\ i = 1,\ldots,N \tag{8}$$

where,

$$y_{ij} = \sum_{k=1}^{R} x_{ijk};\ i,j = 1,\ldots,N. \tag{9}$$

The upper bound $q$ of the relaxed 2-D assignment problem is easily solvable via a 2-D assignment algorithm. To obtain a feasible solution, we reimpose constraint (4) by reconstructing the reward tensor and viewing the asymmetric bipartite graph as a transportation problem based on the solution of the relaxed 2-D assignment problem. For each $\langle i^*, j^* \rangle$ of the relaxed 2-D assignment problem at each iteration, the reward matrix is dynamically updated for each layer $k$. Given a new reward matrix $\tilde{w}_{\langle i,j \rangle k}$, the transportation variation of the problem is as follows.

$$\max_{z_{jk} \in \{0,1\}} \sum_{j=1}^{N} \sum_{k=1}^{R} \tilde{w}_{\langle i,j \rangle k} z_{jk} \tag{10}$$

$$s.t. \sum_{j=1}^{N} z_{jk} = 1,\ k = 1,\ldots,R \tag{11}$$

$$\sum_{k=1}^{R} z_{jk} \leq m_k,\ j = 1,\ldots,N \tag{12}$$

Through this sequence, we obtain a feasible solution and a lower bound $f$. The upper and lower bounds serve as measures of the solution quality. The distance between these bounds is referred to as the approximate duality gap (because it is overestimated by $(f - f^*)$, where $f^*$ is the optimal solution). For discrete 3-D assignment problems, the duality gap may be nonzero. The relative approximate duality gap is given by

$$gap = \frac{|q - f|}{f}, \tag{13}$$

where $q$ and $f$ are the upper and and lower bounds, respectively, obtained by solving the series of 2-D subproblems. The 3-D assignment algorithm terminates for a sufficiently small gap, which implies that a near-optimal solution has been obtained. In scenarios where the duality gap is large, the 3-D assignment algorithm updates its Lagrange multipliers via the method proposed in Pattipati [18]. Let us denote $\underline{g}$ as an $R$-dimensional subgradient vector with components given by

$$g_k = R - \sum_{i=1}^{N} \sum_{j=1}^{N} X_{ijk} \quad k = 1,\ldots,R, \tag{14}$$

where $X$ is the solution tensor related to the optimal value of the relaxed 2-D assignment variables $\{y_{ij}^*\}$ via

$$X_{ijk} = \begin{cases} y_{ij}^*, & \text{if } k = \arg\min_\alpha (w_{ij\alpha} - \mu_\alpha) \\ 0, & \text{otherwise} \end{cases}$$

We then update the Lagrange multipliers by

$$\mu_k = \max\left( \mu_k - \frac{(p - f)}{\|\underline{g}\|_2^2} g_k, 0 \right). \tag{15}$$

After updating the Lagrange multipliers, the algorithm iterates back to the relaxation step. The process continues until either the maximum number of iterations is reached or the duality gap is sufficiently small. The flow diagram of the 3-D assignment algorithm when the constraint in (4) is relaxed is shown in Fig. 3.

3.1.2. Relaxation Method II: Note that a relaxed problem is also obtainable by interchanging the sequence of 2-D subproblems. In other words, we may apply the Lagrangian relaxation on constraints (2) or (3). When constraint (3) is relaxed via Lagrange multipliers $\mu_j$, the Lagrangian function is:

$$L(x,\mu) = \max_{x_{ijk} \in \{0,1\}} \left( \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{R} (w_{ijk} - \mu_j) x_{ijk} \right) + \sum_{j=1}^{N} \mu_j \tag{16}$$

The 3-D assignment problem is then relaxed into a 2-D transportation problem of the form

$$\max_{z_{ik} \in \{0,1\}} \sum_{i=1}^{N} \sum_{k=1}^{R} \max_j (w_{ijk} - \mu_j) z_{ik} \tag{17}$$

$$s.t. \sum_{k=1}^{R} z_{ik} = 1,\ i = 1,\ldots,N \tag{18}$$

$$\sum_{i=1}^{N} z_{ik} \leq m_k,\ k = 1,\ldots,R, \tag{19}$$

where

$$z_{ik} = \sum_{j=1}^{N} x_{ijk};\ i = 1,\ldots,N;\ k = 1,\ldots,R \tag{20}$$

The upper bound $q$ can be obtained by solving the relaxed 2-D transportation problem. The 2-D assignment problem is obtained by reimposing constraint (3) and reconstructing the reward tensor based on the solution of the relaxed 2-D transportation problem. The assignment variation of the problem is as follows.

$$\max_{y_{ij} \in \{0,1\}} \sum_{i=1}^{N} \sum_{j=1}^{N} \tilde{w}_{\langle i,k \rangle j} y_{ij} \tag{21}$$

$$s.t. \sum_{i=1}^{N} y_{ij} = 1,\ j = 1,\ldots,N \tag{22}$$

$$\sum_{j=1}^{N} y_{ij} = 1,\ i = 1,\ldots,N \tag{23}$$

A feasible solution and a lower bound $f$ can be obtained through this sequence. The duality gap is then computed and compared for algorithm termination. The subgradient is updated in a similar fashion to the first relaxation

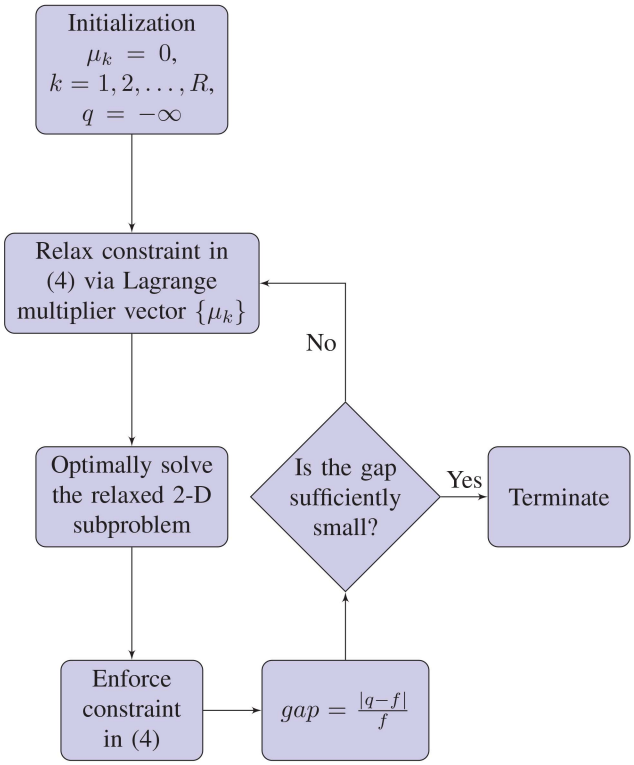



Fig. 3. Flow diagram of the 3-D assignment algorithm when relaxing constraint (4).

method, except that it is with respect to dimension $j$ and uses binary decision variables $\{z_{ik}\}$.

3.1.3. Algorithm selection for 2-D subproblems: To optimize the 3-D assignment algorithm, state-of-the-art 2-D assignment and transportation algorithms were selected for comparison purposes. The JVC and auction algorithms were selected for comparison when solving the 2-D assignment problem. We solve the 2-D transportation problem via three approaches. The first algorithm utilizes the Transauction algorithm developed by Bertsekas and Castañón [43], which solves the transportation problem by mapping it to an assignment problem and obtains a solution via a modified auction algorithm. In the second algorithm, we exploit the findings in [6], [44], [45], where the transportation problem was found to be equivalent to the minimum cost network flow problem, and solve the 2-D transportation problem via a (strongly polynomial) simplex-based method. We refer to this simply as simplex-based transportation. The third algorithm is the RELAX-IV algorithm developed by Bersekas and Tseng [46] and further detailed in [47]. It is one of the most efficient algorithms to solve problems of the network flow type.

3.1.4. Solution approach for a variant of the nuclear FA loading pattern optimization problems: for this problem, constraint (4) is such that $m_k = N$. In this case, the summations over sets $i$ and $j$ are always less than or equal to $N$ for each $k$, and, consequently, constraint (4) is always satisfied. This implies that the constraints in (4) are inactive and the Lagrange multipliers $\mu_k = 0$ for $k = 1, 2, \ldots, R$. Consequently, the 3-D assignment problem takes the form,

$$\max_{y_{ij} \in \{0,1\}} \sum_{i=1}^{N} \sum_{j=1}^{N} \max_{k}(w_{ijk}) y_{ij} \tag{24}$$

$$s.t. \sum_{i=1}^{N} y_{ij} = 1, \; j = 1, \ldots, N \tag{25}$$

$$\sum_{j=1}^{N} y_{ij} = 1, \; i = 1, \ldots, N \tag{26}$$

This problem can be easily solved by an $m$-best 2-D assignment algorithm. Furthermore, the approach is the same for the general case when $m_k \geq N$.

### 3.2. $m$-best 3-D Assignment

Let $\mathcal{P}_0$ be the original problem in equations (1)–(4) and let $A$ be the corresponding assignment solution space. Further, let $A_0^*$ be the best feasible assignment found by the 3-D assignment algorithm detailed in Section 3.1. In general, to find the $(n+1)$th best solution, we have to partition the $(n+1)$th problem space, $\mathcal{P}_n$, into $N$ subproblems, denoted by $\mathcal{P}_{nr}$, $1 \leq r \leq N$. Then, the complete solution space corresponding to problem space $\mathcal{P}_n$ is

$$A_n = \bigcup_{r=1}^{N} A_{nr} = A - \bigcup_{i=0}^{n-1} A_i^* \quad \text{for } n = 1, 2, \ldots, m \tag{27}$$

$$A_{nr} \cap A_{ns} = \emptyset \quad \text{for } r, s = 1, 2, \ldots, N \quad r \neq s, \tag{28}$$

where $A_{nr}$ denotes a set of tuples in which each $i$ and $j$ appear exactly once, but $k$ may be repeated. Equation (27) is a formalization of the constraint that the solution space $A_n$ for the $(n+1)$th best solution will not contain any of the best solutions obtained for the previous $n$ problems. Here, a complete feasible solution is assumed to be a set of tuples. Hence, some solutions may have a similarity, however, as seen in (28), the set of solution tuples as a whole are unique, differing by at least one element for each of the previous $n$ problems. Let an assignment $A_{nr}$ consist of multiple tuples (in this paper, triples), where we index the triples within by $t$. Let $\ell_{nr_t}$ be the individual reward of the $t$th triple, sub-indexed as $\langle i_{nr_t}, j_{nr_t}, k_{nr_t} \rangle$, in the solution space $A_{nr}$. We can then augment the triple into a 4-tuple and write a feasible assignment in $A_{nr}$ as

$$S_{nr} = \{\langle i_{nr_t}, j_{nr_t}, k_{nr_t}, \ell_{nr_t} \rangle\} \quad \text{for } t = 1, \ldots, N. \tag{29}$$

The primal value of the corresponding assignment $S_{nr}$ is denoted by $f_{nr}$, which can be obtained by summing $\ell_{nr_t}$ over $t = 1, 2, \ldots, N$.

$$f_{nr} = \sum_{t=1}^{N} \ell_{nr_t} \tag{30}$$

The best assignment $A^*_{nr}$ with the corresponding primal value $f^*_{nr}$ in the solution space $A_{nr}$ is found via the 3-D assignment algorithm described earlier and pertains specifically to partition $r$. The best assignment $A^*_n$ is found by iterating over all active partitions and finding the argument $r^*$ which has the maximum primal value.

$$A^*_n = A^*_{nr^*} \tag{31}$$

$$r^* = \arg\max_r f^*_{nr} \tag{32}$$

Given the original problem space and its optimal assignment, denoted by $\mathcal{P}_0$ and $A^*_0$, respectively, we partition $\mathcal{P}_0$ into $N$ problem subspaces $\mathcal{P}_{11}$ to $\mathcal{P}_{1N}$ in order to find the next best solution. To generate subproblem $\mathcal{P}_{11}$, we remove the first of $N$ tuples in the assignment $A^*_0$. We then use the 3-D assignment algorithm to obtain the best possible solution $A^*_{11}$ to problem $\mathcal{P}_{11}$. To partition the subspace $\mathcal{P}_{1s}$, $2 \leq s \leq N$, we remove the $s$th tuple in $A^*_0$ as a feasible assignment in $\mathcal{P}_{1s}$, while fixing the first $(s-1)$ triples to those in the original assignment $A^*_0$. Thus, as the solution and problem spaces are reduced at every search space decomposition, the complexity of the problem decreases substantially, since the first $(s-1)$ triples are reused from the previous assignments. We then only need to find assignments for the remaining $N-s$ assignments, such that the $s$th triple from the original assignment $A^*_0$ is not contained in the solution, while satisfying the constraints. The enforcement of tuples to be either in or be removed from the problem spaces $\mathcal{P}_{11}$ to $\mathcal{P}_{1N}$ during partitioning ensures the disjointness of the individual subproblems, as in equation (28).

Each of the best solutions $A^*_{11}$ to $A^*_{1N}$ is saved into a heap and accordingly sorted based on the respective primal values, $f^*_{11}$ to $f^*_{1N}$. The best solution within the heap is then removed and saved as the second best solution. The problem corresponding to the second best solution is then partitioned into the subproblems $\mathcal{P}_{21}$ to $\mathcal{P}_{2N}$. The best assignment from the top of the heap is then marked as the third best assignment with respect to the original problem $\mathcal{P}_0$. We continue to apply this process until the $m$th best solution is found or, alternatively, the heap becomes empty.

Murty's search space decomposition is an ingenious way of decomposing the search space, and has a number of applications in combinatorial optimization [34], [48]. Optimizations of the decomposition technique to improve the computational efficiency are discussed in Section 4.

REMARK For small size problems and large $m$, if we apply the Lagrangian relaxation on constraint (4), the transportation problem reconstructed from the best ($i$, $j$) pair of the 2-D assignment problem may contain too many removed arcs and, thus, no feasible solution may exist. In this case, by interchanging the sequence of relaxed problems solved (i.e., solve the 2-D transportation problem first, as opposed to the assignment problem (normally solved first)), we can obtain a feasible solution to the 3-D assignment problem. This situation arises in small size problems (e.g., of dimension $3 \times 3 \times 2$). However, since the tensor dimensions used in this paper are large, the solution space is vast and this anomaly did not arise.

## 4. OPTIMIZED IMPLEMENTATION OF MURTY'S SEARCH SPACE DECOMPOSITION

We extend the 2-D optimization modifications in [36] to the 3-D assignment problems. These include: 1) inheriting the dual variables and partial solutions from the subproblems being decomposed; 2) sorting the subproblems by an upper bound on reward before solving; and 3) partitioning the subproblems in an optimized order. All three modifications exploit the primal-dual aspects of the JVC algorithm. The following sections explain each modification in detail for the case when the constraints in (4) are relaxed in the $m$-best 3-D assignment algorithm. Similar optimization techniques can be applied for the case when the constraints in (3) are relaxed.

### 4.1. Inheriting dual variables and partial solutions during partitioning

Solving the 3-D assignment problem via the JVC algorithm provides dual variables $u$ and $v$, which can be inherited by the partitioned subproblem using Murty's search space decomposition. The solution tensor $X_n$, for the problem space $\mathcal{P}_n$ and the reward tensor $W$, contains $N$ solution triples $\langle i^*, j^*, k^* \rangle$. During each step of Murty's search space decomposition, a new subproblem $\mathcal{P}_{nr}$ is generated, associated with a new reward tensor $W'$. Removing the triple $\langle i^*, j^*, k^* \rangle$ from the subproblem space $\mathcal{P}_{nr}$ is equivalent to setting $w_{\langle i^*, j^*, k^* \rangle} = -\infty$. This implies we may skip the initialization step for the JVC algorithm and go directly to the augmentation step with only one arc left to assign in the 2-D assignment problem, following the procedure outlined in Algorithm 1. In this case, the initialization step is only required for the first feasible solution to the 3-D assignment problem.

ALGORITHM 1 *Upper bound reward calculation when inheriting dual variables*

1: **for each** $\langle i^*, j^*, k^* \rangle \in A$ **do**
2: $\quad w_{\langle i^*, j^*, k^* \rangle} = -\infty$
3: $\quad u' = u$, $v' = v$,
4: $\quad X' \leftarrow X - X_{\langle i^*, j^*, k^* \rangle}$
5: **end for**

Note that we can not inherit the Lagrange multipliers $\mu_k$ from the previous problem $\mathcal{P}_n$ in the process of partitioning the subproblems. The Lagrange multipliers from the previous problem $\mathcal{P}_n$ may be too large for the subproblems $\mathcal{P}_{nr}$, $r = 1, 2, \ldots, N$. This may cause the duality gap to remain above the threshold value required

to terminate. Thus, the algorithm will continue to run until the maximum iteration limit is reached.

### 4.2. Sorting subproblems via an upper bound

The upper bound reward of individual subproblems is easily obtainable and can be used to avoid solving subproblems that are unlikely to produce the next best solution. For an $m$-best assignment problem, the best solution from problem $\mathcal{P}_n$ is always better than the best solution obtained from the subproblems obtained by partitioning $\mathcal{P}_{nr}$, $r = 1,2,\ldots,N$. Therefore, for an $m$-best 2-D assignment problem, the objective function of the solution to $\mathcal{P}_n$ can be used as an initial upper bound on the objective function value of the best solution to its corresponding subproblems. Since 3-D assignment problems may have a nonzero duality gap, the computation of the upper bound can be determined using either the dual value (denoted by $\phi$) or the primal value (denoted by $\omega$) as initial upper bounds to the partitioned subproblems.

When a subproblem $\mathcal{P}_{nr}$ is created by removing a triple $\langle i^*, j^*, k^* \rangle$ from a copy of $\mathcal{P}$, we can compute the upper bound objective function value by finding the best slack (i.e., next possible best assignment) of all the alternative assignments for a row $i$. The upper bound objective function value will be the sum of the initial upper bound and the row slack, denoted by $B_r$. The calculation of the upper bound is shown in detail in Algorithm 2.

ALGORITHM 2 *Upper bound reward calculation when sorting subproblems*

1: **for each** row $i$ **do**
2: $\quad w_{\langle i^*, j^*, k^* \rangle} = -\infty$
3: $\quad B_r = \max_{j,k}\{w_{ijk} - u(i^*) - v(j) - \mu(k)\}$
4: $\quad f' = f + B_r$
5: **end for**

A similar procedure can be followed for column $j$ to find the column slack, $B_c$. By combining both the row and the column slack, a tighter upper bound can be obtained. The heap of subproblems can be modified to sort its elements (in descending order) based on each element's respective upper bound reward. This implies that the problems located at the top of the heap are most likely to have the best solutions.

In this optimization method, the initial problem is partitioned into a series of subproblems when it is solved by the 3-D assignment algorithm. Both the original problem and its corresponding subproblems are saved into a heap. During each iteration of Murty's search space decomposition, if the top problem $\mathcal{P}_n$ removed from the heap has a feasible solution, then the solution will be saved as the $m$th best assignment. If $\mathcal{P}_n$ has not yet been solved (i.e., it has a partial solution), then we find its best solution $A_n^*$ using the 3-D assignment algorithm and add it back into the heap. A new partitioning process is then invoked on $\mathcal{P}_n$ and its solution $A_n^*$. The process is repeated until the heap is empty or a total of $m$ solutions are obtained. This method allows us to eliminate subproblems by focusing on their corresponding upper bounds, thus reducing the number of problems needed to be solved by the 3-D assignment algorithm.

### 4.3. Partition in an optimized order

The third optimization method proposed here is to carefully select the *order* in which the partitioning is performed. This modification maximizes the probability that the subsequent smaller subproblems (with a greater number of fixed arcs) have better solutions. For problem $\mathcal{P}_n$ with solution $A_n^*$ that contains $N$ triples, we first compute each upper bound reward that would result from excluding each individual arc. These upper bounds are computed via the method explained in Section 4.2. We then select the triple that corresponds to the lowest upper bound reward computed and exclude it from the current subproblem, while fixing the corresponding arc in the next subproblem.

In this modification, the heuristic tends to ensure that the largest problem (maximum number of unassigned arcs) has the lowest upper bound. In other words, the largest problem has the highest probability of containing the worst solution and to be pushed to the bottom of the heap (and in turn, will most likely remain unsolved upon algorithm termination). The next worst problem will tend to be the second largest subproblem, and so on. By doing this, we increase the chance that the smallest problem (that which has the least amount of unassigned arcs) contains the best solution.

## 5. PSEUDOCODE

The following variants were used and/or combined for different optimization methods:

(A) Inheritance of the dual variables and partial solutions during partitioning
(B) Sorting subproblems by an upper bound reward before solving, where the upper bound is calculated via:
- i $\omega + B_r$
- ii $\omega + B_r + B_c$
- iii $\phi + B_r$
- iv $\phi + B_r + B_c$

(C) Partitioning the problem in an optimized order

These variants are denoted as listed for the remainder of the paper and may be combined, e.g., when combining variant A with variant B(ii) and variant C, the algorithm variant will be categorized as A+B(i)+C. The pseudocode for Murty's modified search space decomposition, optimized via variants A, B(ii), and C, is detailed in Algorithm 3. These variants assume JVC and Transauction to be applied in the $m$-best 3-D assignment algorithm.

ALGORITHM 3 *m-best 3D assignment algorithm*

$H \leftarrow \{\}$ Initialize binary heap
$U \leftarrow [\,]$ Initialize solution list
$\langle A_0^*, P_0, f_0^* \rangle$=3DASSIGN($w_{ijk}$)
PARTITION($H, \mathcal{P}_0, A_0^*$) Invoke PARTITION method
$H \leftarrow \langle A_0^*, \mathcal{P}_0, f_0^* \rangle$ Add to the heap
counter=0
**while** counter $\leq m-1$ and $H \neq \emptyset$ **do**
  $\langle A_n^*, \mathcal{P}_n, f_n^* \rangle = H.pop$
  **if** $A_n^*$ is feasible **then**
    counter=counter+1
    $U \leftarrow A_n^*, f_n^*$
  **else**
    $\langle A_n^*, \mathcal{P}_n, f_n^* \rangle$=3DASSIGN($w_{ijk}, \langle A_n^*, \mathcal{P}_n, f_n^* \rangle$)
    **if** $\exists$ solution **then**
      PARTITION($H, \langle A_n^*, \mathcal{P}_n, f_n^* \rangle$)
      $H \leftarrow \langle A_n^*, \mathcal{P}_n, f_n^* \rangle$
    **end if**
  **end if**
**end while**

**function** PARTITION($H, \langle A_n^*, \mathcal{P}_n, f_n^* \rangle, w_{ijk}$)
  **for each** $\langle i^*, j^*, k^* \rangle \in A_n^*$ **do**
    $w_{i^*,j^*,k^*} = -\infty$
  **end for**
  **for each** $\langle i^*, j^*, k^* \rangle \in A_n^*$ **do**
    **for each** row $i^* \in A^*$ **do**
      $B_r = \max_{j,k}\{w_{ijk} - u(i^*) - v(j) - \mu(k)\}$
      $B_c = \max_{i,k}\{w_{ijk} - u(i) - v(j^*) - \mu(k)\}$
      $B_{i^*} = B_r + B_c$
    **end for**
    $(B, i^*) = \min(B_{i^*} \neq -\infty)$
    $\langle i^*, j^*, k^* \rangle = A_n^*(i^*)$
    $f_{nr}^* = f_n^* + B$
    $A_{nr}^* \leftarrow A_n^* - \langle i^*, j^*, k^* \rangle$
    $\mathcal{P}_{nr} \leftarrow \mathcal{P}_n - \langle i^*, j^*, k^* \rangle$
    $H \leftarrow \langle A_{nr}^*, \mathcal{P}_{nr}, f_{nr}^* \rangle$
    $A_{n(r+1)}^*.FixList \leftarrow \langle i^*, j^*, k^* \rangle$
    **for each** $j,k$ **do**
      $w[i^*, j, k] = -\infty$
    **end for**
    **for each** row $\neq i^*, k$ **do**
      $w$[row, $j^*, k$] $= -\infty$
    **end for**
  **end for**
**end function**

**function** 3DASSIGN($w_{ijk}, \langle A_n^*, \mathcal{P}_n, f_n^* \rangle$)
  $f^* = -\infty$; lb= $-\infty$; $q^* = \infty$; maxIter= 20
  MAX= $true, n_3 = R$
  $FixList, v, \Phi \leftarrow A_n^*$
  **for** curIter= 1 to maxIter **do**
    $C = \max_k(w_{ijk} - \mu_k)$
    **for** $\langle i^*, j^*, k^* \rangle \in A_n^*.FixList$ **do**
      $C[i^*, j^*] = w[i^*, j^*, k^*]$
    **end for**
    **if** $\Phi == \emptyset$ **then**
      $(\Phi, u, v, \phi)$ =JVC($C$, MAX)
    **else**
      $(\Phi, u, v, \phi)$ =Augment($C, \Phi, v$, MAX)
    **end if**
    $q^* = \min(q, \phi + n_3 * \sum_k(\mu_k))$
    **for each** row **do**
      $T$[row] = $w$[row, $\Phi$[row], $k$] $\forall k$
    **end for**
    $(\Omega, \omega)$ =Transportation($T$, MAX)
    **if** $\omega \geq$ lb **then**
      lb= $\omega$
      $f^*$ =lb
    **end if**
    gap= $\dfrac{|q^* - f^*|}{|f^*|}$
    **if** gap $\leq 0.05$ **then**
      **return**
    **end if**
    Update Lagrangian multiplier
  **end for**
**end function**

## 6. RESULTS

The proposed $m$-best 3-D assignment algorithm was implemented in the MATLAB 2016b and runs on an Intel Core i7-4712HQ CPU processor @2.30 GHz with 16 GB RAM. In all experiments, the top $10^4$ ranked solutions were computed.

### 6.1. Relaxation Method I vs. Relaxation Method II

We first performed 10 Monte Carlo runs to compare the simulation runtimes of the 3-D assignment algorithm when relaxing either constraint (3) or constraint (4). The reward tensor elements were uniformly distributed in the interval $[0,1]$ and of dimension $60 \times 60 \times 8$. The JVC and Transauction algorithms were implemented to solve the 2-D assignment and the transportation problems, respectively. As shown in Table II, speedup of as much as 2.28 and an average speedup of 1.63 were observed when comparing the two relaxation methods. In general, solving a 2-D assignment problem is significantly faster than solving a transportation problem. The transportation problem obtained from relaxing constraint (3) is complex, and thus takes a longer time to solve compared to the transportation problem reconstructed from the best 2-D assignment solution when constraint (4) is relaxed. Relaxing constraint (4) also consistently resulted in a smaller duality gap compared to when constraint (3) was relaxed due to the fact that $|k| = R < N = |j|$. This implies that when constraint (4) is relaxed, a smaller number of elements in the 3-D reward tensor are removed when constructing the 2-D subproblem, i.e., since a smaller number of elements are removed, there is a higher likelihood that a better

TABLE II
10 Monte Carlo Runs for different Lagrangian Relaxation methods

| MC | Relaxation on constraint (3) | | | Relaxation on constraint (4) | | | Speedup |
|---|---|---|---|---|---|---|---|
| | Gap | Objective Function Values | Runtime (CPU s) | Gap | Objective Function Values | Runtime (CPU s) | |
| 1 | 0.015 | 59.021 | 0.065 | 0.003 | 59.627 | 0.040 | 1.628 |
| 2 | 0.010 | 59.281 | 0.064 | 0.004 | 59.599 | 0.045 | 1.422 |
| 3 | 0.013 | 59.100 | 0.054 | 0.003 | 59.618 | 0.038 | 1.413 |
| 4 | 0.011 | 59.219 | 0.053 | 0.002 | 59.664 | 0.033 | 1.613 |
| 5 | 0.011 | 59.209 | 0.053 | 0.001 | 59.764 | 0.031 | 1.692 |
| 6 | 0.010 | 59.249 | 0.053 | 0.002 | 59.699 | 0.032 | 1.664 |
| 7 | 0.009 | 59.335 | 0.070 | 0.002 | 59.719 | 0.031 | 2.279 |
| 8 | 0.013 | 59.124 | 0.057 | 0.005 | 59.513 | 0.034 | 1.700 |
| 9 | 0.012 | 59.184 | 0.053 | 0.008 | 59.347 | 0.034 | 1.563 |
| 10 | 0.010 | 59.290 | 0.049 | 0.003 | 59.632 | 0.036 | 1.344 |

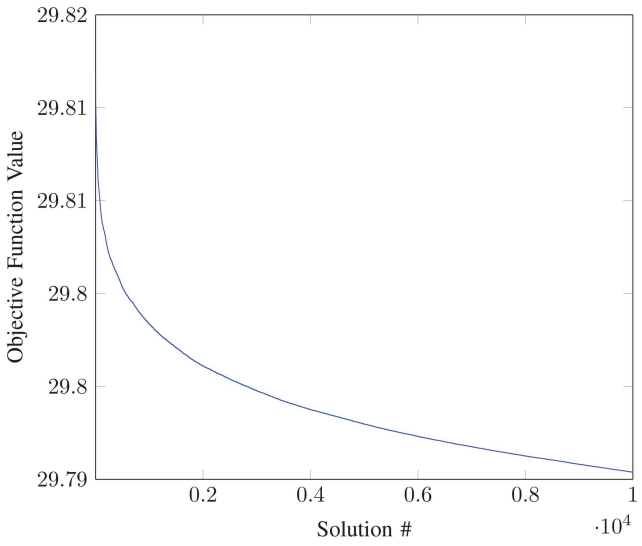



Fig. 4. Example objective function values for a tensor of dimension $30 \times 30 \times 8$ with values uniformly distributed on the interval $[0,1]$.

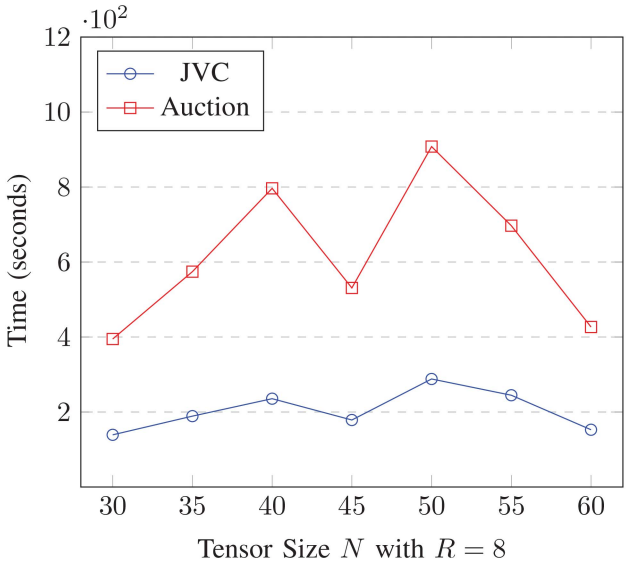



Fig. 5. The CPU runtime for the JVC and auction algorithms were compared as a function of varying tensor dimensions. The JVC algorithm consistently outperformed the auction algorithm.

solution remains. For these reasons, the remaining experiments used the $m$-best 3-D assignment algorithm with the relaxation of constraint (4) only.

### 6.2. JVC vs. Auction Algorithm

To measure and quantify which algorithms best solve the 2-D assignment and transportation problems within the 3-D assignment problem, we compared the runtimes of the 3-D assignment algorithm when using the JVC or the auction algorithms for the 2-D assignment problem, and Transauction or simplex-based transportation algorithms for the transportation problem. A tensor was generated with elements sampled from a uniform distribution in the interval $[0,1]$ for tensor sizes ranging from $30 \times 30 \times 8$ to $60 \times 60 \times 8$ with increments of $N = 5$. Any combination of the 2-D assignment algorithms with the transportation algorithms resulted in the same assignments and objective function values. An example of the objective function values of a sample tensor of dimension $30 \times 30 \times 8$ is shown in Fig. 4 when the algorithm was run to obtain the top $10^4$ solutions. As shown in Fig. 4, even when $10^4$ assignments were obtained, the maximum and minimum objective function values obtained from the assignment solutions had minimal variation and the difference was relatively small for all tensor dimensions tested; however, as shown in Fig. 5, the $m$-best 3-D assignment algorithm, which invoked the JVC algorithm was, on average, 3 times faster when compared to the case when the auction algorithm was used. The RELAX-IV algorithm was used to solve the transportation problem in this experiment.

### 6.3. Transportation vs. Transauction vs. RELAX-IV Algorithm

Similar tests were performed to evaluate the best algorithm to solve the transportation problem. Assuming that the JVC algorithm would be invoked to solve the 2-D assignment portion of the problem, Fig. 6

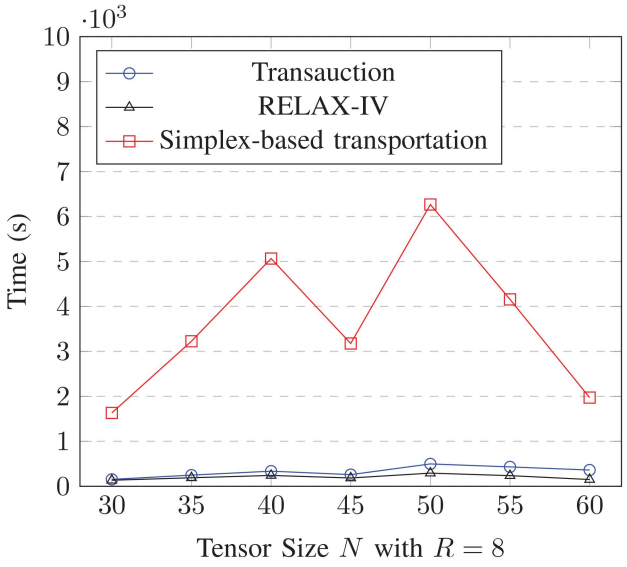



Fig. 6. The CPU runtime for the Transauction and simplex-based transportation algorithms were compared as a function of differing tensor dimensions. The Transauction algorithm remained relatively unaffected by the increase in the reward tensor size, while the transportation algorithm took orders of magnitude more time to find the same assignments.

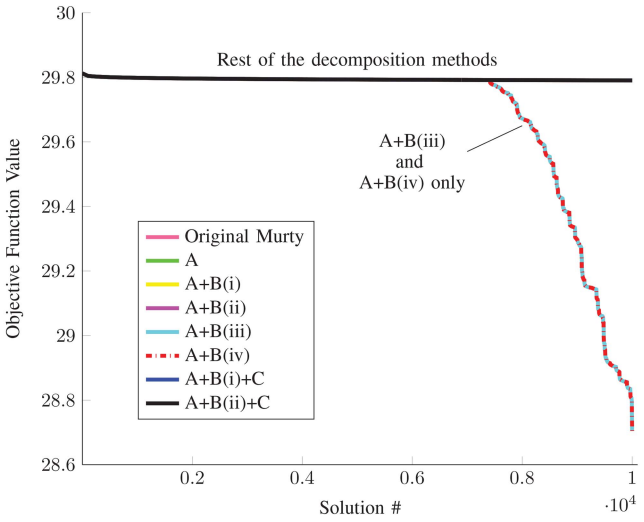



Fig. 7. Objective function values for tensor size $30 \times 30 \times 8$ with various optimization combinations.

demonstrates that the simplex-based transportation algorithm was significantly slower compared to both the Transauction and RELAX-IV algorithms. In general, the RELAX-IV algorithm had the fastest runtime speed. The maximum observed speedup of RELAX-IV in comparison to the simplex-based transportation and the Transauction algorithms was 21.4 and 2.4, respectively. Overall, the RELAX-IV algorithm dominated both the simplex-based transportation and the Transauction approaches to the transportation problem, on average solving it nearly 17 and 1.6 times faster, respectively. Based on these findings, the JVC and RELAX-IV algorithms were selected to solve the 2-D assignment and transportation problems within the $m$-best 3-D assignment problem, respectively, for the remaining computational experiments. We optimized the $m$-best 3-D assignment algorithm via the methods detailed in Section 4.

### 6.4. Solution quality evaluation for decomposition methods

A sample test tensor of dimension $30 \times 30 \times 8$ with elements uniformly distributed in the interval $[0, 1]$ was used to measure the solution quality and to compare the simulation runtimes of the $m$-best 3-D assignment algorithm when exploiting different combinations of the search space decomposition optimization methods. As shown in Fig. 7, the optimization combinations of A+B(iii) and A+B(iv) resulted in approximately a 10% reduction in the solution quality compared to the original Murty's proposed search space decomposition method. This is because the dual value in our problem setup did not serve as an accurate estimate of the initial upper bound. All other combinations of optimization methods were comparable to the Murty's search space decomposition. Therefore, optimization method combinations A+B(iii) and A+B(iv) were removed from the remaining tests.

### 6.5. Runtime comparison for decomposition methods

Similar tests were performed on all the remaining combinations of optimization methods for tensor sizes varying from $30 \times 30 \times 8$ to $60 \times 60 \times 8$ with an increment of $N = 5$. Table III shows the simulation runtime in CPU seconds for $10^4$ solutions and for the various search space optimization combinations, given a sample test tensor for each incremented dimension. Methods A, A+B(i) and A+B(ii) on average ran 20%, 52% and 32% slower, respectively, within the $m$-best 3-D assignment algorithm, as compared to the original Murty's search space decomposition. As shown in Fig. 8, methods A+B(i)+C and A+B(ii)+C were able to obtain speedups with very minimum variation in the objective function values originally found by Murty's search space decomposition for all tensor sizes except that of dimension $60 \times 60 \times 8$. The reason for such a slow down is explained later in Section 6.6. Furthermore, combinations A+B(i)+C and A+B(ii)+C were able to obtain objective function values slightly better (higher) than the proposed method by Murty (on the order of $10^{-6}$). This phenomenon is due to the Lagrangian relaxation algorithm's approximation of the 3-D assignment problem. The search space decomposition method is suboptimal when applied to the 3-D assignment problem (due to the suboptimal nature of the Lagrangian relaxation algorithm), and so from our analysis we observed that, through the particular optimization method combinations of A+B(i)+C and A+B(ii)+C, better feasible solutions were found. These methods were also significantly faster, offering an average of 2.1 and 2.4

TABLE III
Simulation runtime in CPU seconds for various combinations of decomposition methods

| | | Decomposition Methods | | | | |
|---|---|---|---|---|---|---|
| Tensor Size | Original Murty | A | A+B(i) | A+B(ii) | A+B(i)+C | A+B(ii)+C |
| $30 \times 30 \times 8$ | 139.17 | 157.32 | 193.45 | 124.30 | 51.69 | 44.96 |
| $35 \times 35 \times 8$ | 189.03 | 230.78 | 400.78 | 284.66 | 74.95 | 62.67 |
| $40 \times 40 \times 8$ | 235.56 | 293.40 | 429.86 | 391.12 | 76.85 | 73.50 |
| $45 \times 45 \times 8$ | 178.70 | 226.16 | 398.22 | 337.41 | 102.21 | 87.33 |
| $50 \times 50 \times 8$ | 288.11 | 370.07 | 803.69 | 491.81 | 99.15 | 79.73 |
| $55 \times 55 \times 8$ | 244.75 | 303.14 | 489.72 | 477.44 | 207.71 | 204.60 |
| $60 \times 60 \times 8$ | 152.72 | 209.78 | 463.33 | 427.37 | 309.65 | 232.06 |

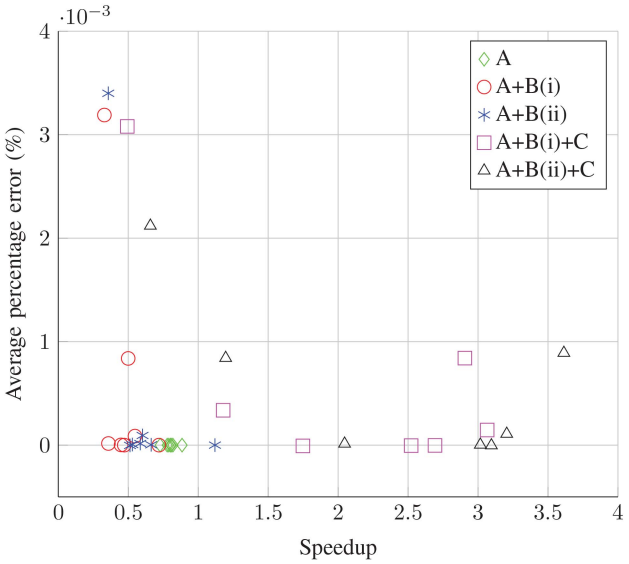



Fig. 8. Percentage error compared against the speedup for the combinations of optimization methods tested for all tensor sizes, varied from $30 \times 30 \times 8$ to $60 \times 60 \times 8$ with an increment of $N = 5$.

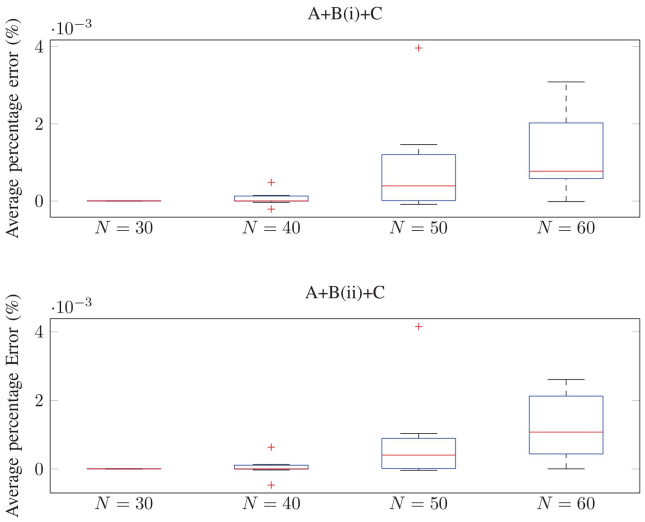



Fig. 9. Box plot for the average percentage error (as compared to the original Murty search space decomposition method) for the optimization method combinations A+B(i)+C and A+B(ii)+C.

speedup, respectively, as illustrated in Fig. 8. To investigate these combinations more thoroughly, Monte Carlo runs were performed on these two combinations only.

### 6.6. Scalability with N

To measure both the overall scalability and consistency, 10 Monte Carlo runs were performed for each tensor size varying from $30 \times 30 \times 8$ to $60 \times 60 \times 8$ in increments of $N = 10$ and using the two specific optimization method combinations of A+B(i)+C and A+B(ii)+C. Each test tensor was generated with elements uniformly distributed in the interval $[0, 1]$ and $10^4$ solutions were obtained for each tensor. In each run, the objective function values and the simulation runtime, were monitored and compared against both combinations, as well as with respect to Murty's decomposition method. Fig. 9 shows the percentage error of the two optimization methods as compared to the original Murty's search space decomposition. The average percentage error increased with each increment in the tensor dimensions. Overall, the optimization method combination of A+B(ii)+C had a lower median compared to the combination of A+B(i)+C; however, the combination of variants A+B(i)+C had less variation with respect to the average percentage error. Table IV details the minimum, maximum, and average runtimes observed in CPU seconds for the Monte Carlo runs. The method A+B(ii)+C had the fastest runtime, as shown in Fig. 10, with an observed maximum average of 3.1 speedup over Murty's search space decomposition method, while having 2.14 speedup on average, when averaged over *all* Monte Carlo runs. The tensor of dimension $60 \times 60 \times 8$ resulted in a slow down of 32% and 25%, respectively, with optimizations A+B(i)+C and A+B(ii)+C.

In the fully optimized $m$-best 3-D assignment algorithm, there exists a tradeoff (when $N \approx 55$) in computation time between obtaining a feasible solution and the $m$-best optimization methods (e.g., partitioning or sorting), as seen in Fig. 10. The increase in dimension $N$ does not necessarily mean an increase in the computation time of the 3-D assignment algorithm, since both optimization methods A+B(i)+C and A+B(ii)+C reduce the frequency of calling the 3-D assignment algorithm; however, partitioning and/or sorting the larger subproblems may become more difficult. A more favorable speedup may be observed if the algorithms were to be implemented in a fast object oriented-programming

TABLE IV
Minimum, maximum, and average runtimes in CPU seconds to obtain $10^4$ solutions

| $30 \times 30 \times 8$ | Original Murty | A+B(i)+C | A+B(ii)+C |
|---|---|---|---|
| Min | 111.84 | 53.72 | 45.01 |
| Mean | 146.03 | 63.50 | 49.64 |
| Max | 195.41 | 76.68 | 53.55 |
| $40 \times 40 \times 8$ | Original Murty | A+B(i)+C | A+B(ii)+C |
| Min | 147.44 | 78.07 | 69.83 |
| Mean | 245.27 | 103.74 | 79.19 |
| Max | 284.48 | 128.17 | 92.61 |
| $50 \times 50 \times 8$ | Original Murty | A+B(i)+C | A+B(ii)+C |
| Min | 151.15 | 98.25 | 80.24 |
| Mean | 247.28 | 155.13 | 139.05 |
| Max | 320.79 | 271.20 | 243.48 |
| $60 \times 60 \times 8$ | Original Murty | A+B(i)+C | A+B(ii)+C |
| Min | 142.19 | 226.88 | 203.46 |
| Mean | 214.01 | 314.57 | 284.93 |
| Max | 294.43 | 586.10 | 542.90 |

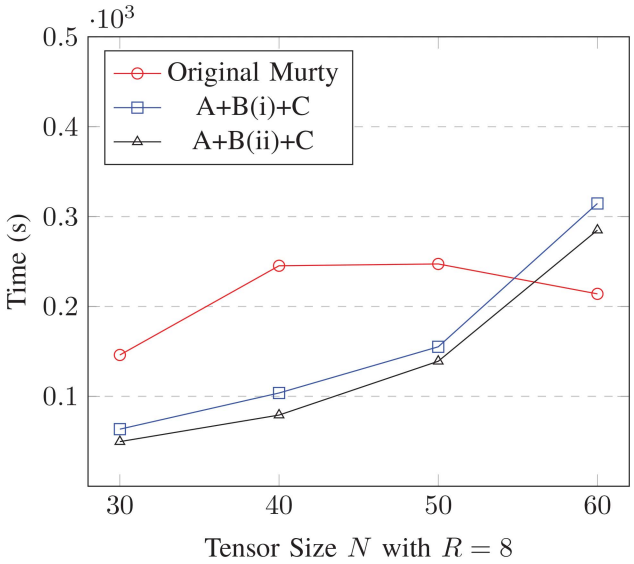



Fig. 10. The average CPU runtimes for 10 Monte Carlo runs for the two optimization method combinations tested.

language. Overall, for a $30 \times 30 \times 8$ tensor, the $m$-best 3-D assignment algorithm utilizing optimization method A+B(ii)+C took an average of 4.9 milliseonds to obtain a single solution to the 3-D assignment problem.

### 6.7. Scalability with R

As mentioned in Sections 2.1 and 3.1.4, the value of $m_k$ should be such that $\sum_{k=1}^{R} m_k \geq N$. For problems where $m_k = R \geq N$, the 3-D assignment problem reduces to a 2-D assignment problem. We analyze the scalability of the algorithms with respect to incrementing $R$ by performing 10 Monte Carlo runs for each increment and requesting $10^4$ solutions with tensor size $N = 30$ and $R = 6, 10, 15, 20, 25, 29$ ($R = 30$ is omitted because, as mentioned earlier, this devolves into a 2-D assignment problem). Fig. 11 shows the relative percentage error of the two optimization methods, as compared to the original Murty's search space decomposition. The average percentage error is zero for $R \geq 10$, since the problem constraint (4) is less likely to be violated, and therefore, the duality gap is zero. The minimum, average and maximum runtimes are listed in Table V. As shown in Fig. 12, the speed of the original $m$-best 3-D assignment algorithm increases significantly with $R$. A maximum speedup of 10.8 and an average speedup of 7.5 were observed when comparing the optimization method A+B(ii)+C with the original $m$-best 3-D assignment algorithm. The total number of arcs input to the RELAX-IV algorithm is bounded above by $R^3$; hence increasing $R$ has an exponential impact on the complexity of the problem solved by the algorithm and, in turn, the CPU runtime of the original $m$-best 3-D assignment. On the other hand, the optimized $m$-best 3-D assignment is able to reduce the need for the 3-D assignment routine invocation and, therefore, is able to obtain $10^4$ solutions in a relatively short amount of time ($< 5$ minutes). The tensor of dimension $30 \times 30 \times 6$ had an increase in average CPU runtime compared to a tensor of dimension $30 \times 30 \times 10$ when considering the optimized methods A+B(i)+C and A+B(ii)+C. This is due to nonzero duality gap which impacts the partitioning procedure and subsequently requires more subprob-

TABLE V
Minimum, maximum, and average runtimes in CPU seconds to obtain $10^4$ solutions

| $30 \times 30 \times 6$ | Original Murty | A+B(i)+C | A+B(ii)+C |
|---|---|---|---|
| Min | 83.33 | 78.24 | 62.51 |
| Mean | 103.77 | 93.48 | 79.00 |
| Max | 119.63 | 108.27 | 93.91 |
| $30 \times 30 \times 10$ | Original Murty | A+B(i)+C | A+B(ii)+C |
| Min | 161.03 | 56.47 | 42.03 |
| Mean | 232.45 | 65.78 | 51.33 |
| Max | 276.15 | 73.7 | 56.32 |
| $30 \times 30 \times 15$ | Original Murty | A+B(i)+C | A+B(ii)+C |
| Min | 477.91 | 87.00 | 68.20 |
| Mean | 640.89 | 102.21 | 75.71 |
| Max | 886.40 | 113.97 | 84.40 |
| $30 \times 30 \times 20$ | Original Murty | A+B(i)+C | A+B(ii)+C |
| Min | 747.44 | 133.35 | 108.31 |
| Mean | 1077.13 | 150.95 | 115.24 |
| Max | 1587.85 | 172.55 | 125.67 |
| $30 \times 30 \times 25$ | Original Murty | A+B(i)+C | A+B(ii)+C |
| Min | 1519.77 | 186.74 | 162.13 |
| Mean | 1902.02 | 218.85 | 176.63 |
| Max | 2507.46 | 230.53 | 187.83 |
| $30 \times 30 \times 29$ | Original Murty | A+B(i)+C | A+B(ii)+C |
| Min | 1704.44 | 265.57 | 235.14 |
| Mean | 2572.39 | 298.28 | 249.87 |
| Max | 3343.91 | 330.11 | 268.99 |

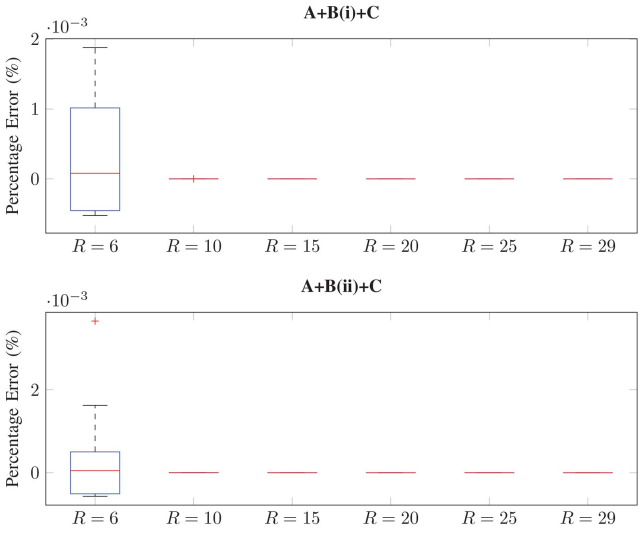



Fig. 11. Box plot for the average percentage error (as compared to the original Murty search space decomposition method) for the optimization method combinations A+B(i)+C and A+B(ii)+C, where $N = 30$.

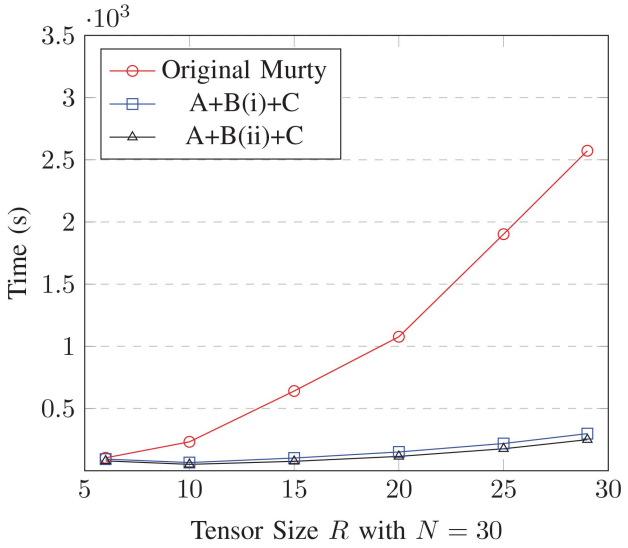



Fig. 12. The average CPU runtimes for 10 Monte Carlo runs with each increment of $R$ for the two optimization method combinations tested.

lems to be solved before obtaining all $m$-best solutions. Intuitively, due to the nature of the problem, a tensor of dimension $30 \times 30 \times 6$ is more likely to violate constraint (4).

## 7. CONCLUSION

In this paper, we formulated a 3-D assignment problem and developed an efficient method to solve the problem, including 1) a rigorous mathematical formulation that is applicable to multiple domains; 2) a novel two-phase solution approach to obtain a large number of ranked solutions. The first phase of our solution approach involves partitioning the original problem space into a series of subproblems via Murty's $m$-best decomposition procedure, while in the second phase, we solve each of these subproblems using a combination of relaxed 2-D assignments through reformulation into either a transportation problem. The solution converges with a sufficiently small duality gap.

We compared the simulation runtime of the $m$-best 3-D assignment algorithm when relaxing either the assignment constraints or the transportation constraints. We also compared the performance of different combinations of 2-D assignment algorithms with a given transportation algorithm and found the combination of JVC and RELAX-IV algorithms, while relaxing the transportation constraints, to be the best performing combination when one is interested in solving the $m$-best 3-D assignment problem for a large number of solutions (in this paper, we were interested in obtaining $10^4$ ranked solutions).

We also evaluated different decomposition methods and compared their scalability and consistency with Murty's search space decomposition. From our analysis, it can be seen that, when solving for a large number of solutions within a 3-D assignment problem, utilizing dual variable inheritance, tight upper bounds on the feasible reward, and partitioning in an optimized order offer the best performance, solving for all $m$-best solutions in a fraction of the time of the original Murty's decomposition method, with little to no sacrifice in solution quality. These optimizations offered a maximum speedup of 10.8 over Murty's search space decomposition. On average, it took 49.64 s to obtain $10^4$ solutions for a tensor of dimension $30 \times 30 \times 8$ required for the nuclear fuel loading problem, which was well within the 10 minute time limit placed on the algorithm.

## ACKNOWLEDGMENT

We would like to thank Dr. Michal Kvasnicka for bringing this unique variation of the $m$-best 3-D assignment problem to our attention and Dr. David Crouse for initial discussions. We thank them for their valuable comments and suggestions during the planning and development of this research work. We also would like to thank Antonio Frangioni for providing the C++ version of the RELAX-IV algorithm. This research was supported by the U.S. Office of Naval Research under contract #N00014-16-1-2036 and by the Department of Defense High Performance Computing Modernization Program under subproject contract #HPCM034125HQU.

## REFERENCES

[1] W. P. Pierskalla
"The tri-substitution method for the three-dimensional assignment problem,"
*CORS Journal*, vol. 5, pp. 71–81, 1967.

[2] A. M. Frieze and J. Yadegar
"An Algorithm for Solving 3-Dimensional Assignment Problems with Application to Scheduling a Teaching Practice,"
*Palgrave Macmillan Journal*, vol. 32, no. 11, pp. 989–995, 1981.

[3] S. Deb, M. Yeddanapudi, K. Pattipaii, and Y. Bar-Shalom
"A generalized s-d assignment algorithm for multisensor-multitarget state estimation,"
*IEEE Transactions on Aerospace and Electronic Systems*, vol. 33, no. 2 PART 1, pp. 523–538, 1997.

[4] R. M. Karp
"Reducibility among combinatorial problems,"
in *Complexity of computer computations*. Springer, 1972, pp. 85–103.

[5] A. Frieze
"Complexity of a 3-dimensional assignment problem,"
*European Journal of Operational Research*, vol. 13, no. 2, pp. 161–164, 1983.

[6] C. H. Papadimitriou and K. Steiglitz
*Combinatorial optimization: algorithms and complexity*.
Courier Corporation, 1982.

[7] "Fuel loading completed at fangchenggang 2,"
http://www.world-nuclear-news.org/NN-Fuel-loading-completed-at-Fangchenggang-2-2505164.html, accessed: 2016-09-22.

[8] K. G. Murty
"An Algorithm for Ranking all the Assignments in Order of Increasing Cost,"
*Operations Research*, vol. 16, no. 3, pp. 682–687, 1968.

[9] H. W. Kuhn
"The hungarian method for the assignment problem,"
*Naval research logistics quarterly*, vol. 2, no. 1–2, pp. 83–97, 1955.

[10] R. Jonker and A. Volgenant
"A shortest augmenting path algorithm for dense and sparse linear assignment problems,"
*Computing*, vol. 38, no. 4, pp. 325–340, 1987.

[11] O. Drummond, D. A. Castanón, and M. Bellovin
"Comparison of 2-d assignment algorithms for sparse, rectangular, floating point, cost matrices,"
in *Proceedings of the SDI Panels on Tracking*, vol. 4, 1990, pp. 4–81.

[12] D. P. Bertsekas
"The auction algorithm: A distributed relaxation method for the assignment problem,"
*Annals of operations research*, vol. 14, no. 1, pp. 105–123, 1988.

[13] M. Balinski
"Signature methods for the assignment problem,"
*Operations research*, vol. 33, no. 3, pp. 527–536, 1985.

[14] P. Hansen and L. Kaufman
"A primal-dual algorithm for the three-dimensional assignment problem,"
*Cahiers du CERO*, vol. 15, pp. 327–336, 1973.

[15] E. Balas and M. J. Saltzman
"An algorithm for the three-index assignment problem,"
*Operations Research*, vol. 39, no. 1, pp. 150–161, 1991.

[16] R. Burkard and R. Rudolf
"Computational investigations on 3-dimensional axial assignment problems,"
*Belgian Journal of Operations Research, Statistics and Computer Science*, vol. 32, pp. 85–98, 1992.

[17] J. B. Mazzola and A. W. Neebe
"Resource-constrained assignment scheduling,"
*Operations Research*, vol. 34, no. 4, pp. 560–572, 1986.

[18] K. R. Pattipati, S. Deb, Y. Bar-Shalom, and R. B. Washburn
"A new relaxation algorithm and passive sensor data association,"
*IEEE Transactions on Automatic Control*, vol. 37, no. 2, pp. 198–213, 1992.

[19] A. P. Poore and N. Rijavec
"A lagrangian relaxation algorithm for multidimensional assignment problems arising from multitarget tracking,"
*SIAM Journal on Optimization*, vol. 3, no. 3, pp. 544–563, 1993.

[20] A. B. Poore and X. Yan
"K-near optimal solutions to improve data association in multiframe processing,"
in *SPIE's International Symposium on Optical Science, Engineering, and Instrumentation*. International Society for Optics and Photonics, 1999, pp. 435–443.

[21] W. Hoffman and R. Pavley
"A method for the solution of the n th best path problem,"
*Journal of the ACM (JACM)*, vol. 6, no. 4, pp. 506–514, 1959.

[22] E. de Queirós Vieira Martins, M. M. B. Pascoal and J. L. E. D. Santos
"Deviation algorithms for ranking shortest paths,"
*International Journal of Foundations of Computer Science*, vol. 10, no. 03, pp. 247–261, 1999.

[23] K. N. Androutsopoulos and K. G. Zografos
"Solving the k-shortest path problem with time windows in a time varying network,"
*Operations Research Letters*, vol. 36, no. 6, pp. 692–695, 2008.

[24] H. N. Gabow
"Two algorithms for generating weighted spanning trees in order,"
*SIAM Journal on Computing*, vol. 6, no. 1, pp. 139–150, 1977.

[25] H. W. Hamacher and G. Ruhe
"On spanning tree problems with multiple objectives,"
*Annals of Operations Research*, vol. 52, no. 4, pp. 209–230, 1994.

[26] Ž. Agić
"K-best spanning tree dependency parsing with verb valency lexicon reranking,"
in *24th International Conference on Computational Linguistics (COLING 2012)*, 2012.

[27] E. S. Van der Poort, M. Libura, G. Sierksma, and J. A. van der Veen
"Solving the k-best traveling salesman problem,"
*Computers & operations research*, vol. 26, no. 4, pp. 409–425, 1999.

[28] P. M. Camerini, L. Fratta, and F. Maffioli
"The k best spanning arborescences of a network,"
*Networks*, vol. 10, no. 2, pp. 91–109, 1980.

[29] I. J. Cox and M. L. Miller
"On Finding Ranked Assignments with Application to Multitarget Tracking and Motion Correspondence,"
*IEEE Transactions on Aerospace and Electronic Systems*, vol. 31, no. 1, pp. 486–489, 1995.

[30] I. J. Cox, M. L. Miller, R. Danchick, and G. E. Newnam
"A comparison of two algorithms for determining ranked assignments with application to multitarget tracking and motion correspondence,"
*IEEE Transactions on Aerospace and Electronic Systems*, vol. 33, no. 1, pp. 295–301, 1997.

[31] R. L. Popp
"Dynamically adaptable m-best 2-d assignment algorithm and multilevel parallelization,"
*IEEE Transactions on Aerospace and Electronic Systems*, vol. 35, no. 4, pp. 1145–1160, 1999.

[32] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua
"Multiple object tracking using k-shortest paths optimization,"
*IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 9, pp. 1806–1819, 2011.

[33] R. L. Popp, K. R. Pattipati, and Y. Bar-Shalom
"m-best S-D assignment algorithm with application to multitarget tracking,"
*IEEE Transactions on Aerospace and Electronic Systems*, vol. 37, no. 1, pp. 22–39, 2001.

[34] E. L. Lawler
"A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem,"
*Management science*, vol. 18, no. 7, pp. 401–405, 1972.

[35] M. Pascoal, M. E. Captivo, and J. Clímaco
"A note on a new variant of murty's ranking assignments algorithm,"
*Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, vol. 1, no. 3, pp. 243–255, 2003.

[36] M. L. Miller, H. S. Stone, and I. J. Cox
"Optimizing Murty's ranked assignment method,"
*IEEE Transactions on Aerospace and Electronic Systems*, vol. 33, no. 3, pp. 851–862, 1997.

[37] H. W. Hamacher and M. Queyranne
"K best solutions to combinatorial optimization problems,"
*Annals of Operations Research*, vol. 4, no. 1, pp. 123–143, 1985.

[38] P. Carraresi and C. Sodini
"A binary enumeration tree to find k shortest paths,"
in *Proc. 7th Symp. operations research*, 1983, pp. 177–188.

[39] C. R. Chegireddy and H. W. Hamacher
"Algorithms for finding k-best perfect matchings,"
*Discrete applied mathematics*, vol. 18, no. 2, pp. 155–165, 1987.

[40] Y. Lin and K. Mouratidis
"Shortlisting top-k assignments,"
in *Proceedings of the 25th International Conference on Scientific and Statistical Database Management*. ACM, 2013, p. 21.

[41] J. B. Mazzola and A. W. Neebe
"Resource-constrained assignment scheduling,"
*Operations Research*, vol. 34, no. 4, pp. 560–572, 1986.

[42] L. Zhang, D. Sidoti, K. R. Pattipati, and D. Castaỳỳn
"Approaches for solving m-best 3-dimensional dynamic scheduling problems for large m,"
in *Information Fusion (FUSION), 2016 19th International Conference on*. ISIF, 2016, pp. 53–58.

[43] D. P. Bertsekas and D. A. Castanon
"the Auction Algorithm for the Transportation Problem,"
*Annals of Operations Research*, vol. 20, pp. 67–96, 1989.

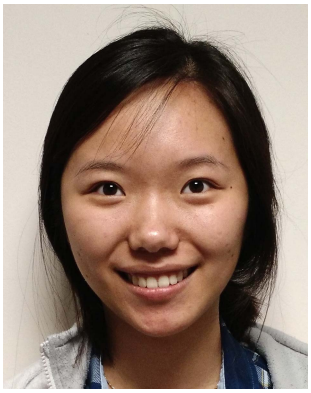
[44] J. B. Orlin, S. A. Plotkin, and É. Tardos
"Polynomial dual network simplex algorithms,"
*Mathematical programming*, vol. 60, no. 1, pp. 255–276, 1993.

[45] H. A. Taha
*Operations research: an introduction*.
Macmillan,, 1992.

[46] D. P. Bertsekas, P. Tseng et al.
*RELAX-IV: A faster version of the RELAX code for solving minimum cost flow problems*.
Massachusetts Institute of Technology, Laboratory for Information and Decision Systems Cambridge, MA, 1994.

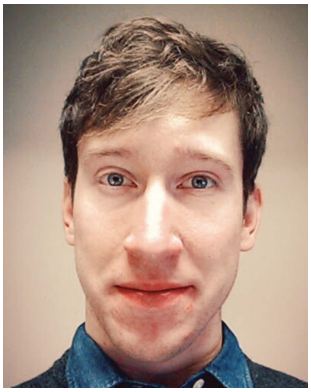
[47] A. Frangioni and A. Manca
"A computational study of cost reoptimization for min-cost flow problems,"
*INFORMS Journal on Computing*, vol. 18, no. 1, pp. 61–70, 2006.

[48] X. Han, H. Bui, S. Mandal, K. R. Pattipati, and D. L. Kleinman
"Optimization-based decision support software for a team-in-the-loop experiment: Asset package selection and planning,"
*IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 2, pp. 237–251, 2013.



**Lingyi Zhang** received the B.S. degree in Electrical and Computer Engineering from the University of Connecticut, Storrs in 2014. She is currently pursuing the Ph.D. degree from the Department of Electrical and Computer Engineering at the same university.

Her current research interests include modeling dynamic and uncertain environments for asset allocation and path planning, context-aware decision support systems, and optimization-based techniques for mission planning and coordination.



**David Sidoti** received the B.S. and M.S. degrees in Electrical and Computer Engineering from the University of Connecticut, Storrs in 2011 and 2016. He is currently a Ph.D. candidate in the Electrical and Computer Engineering Department working under the advisement of Dr. Krishna R. Pattipati at the same university.

His work in dynamic resource management has resulted in numerous transitions to the real world (Naval Research Laboratory–Monterey, Joint Interagency Task Force–South, etc.). He has more than 30 research articles in peer-reviewed journals and international conference proceedings. He was an invited speaker at NATO's *Decision Support and Risk Assessment for Asset Planning 2015 Workshop*. His current interests include multi-objective algorithms for dynamic scheduling and resource management in weather-impacted environments.

**Spandana Vallabhaneni** received her B.S. degree in Computer Science and Engineering from GITAM University, Visakhapatnam, Andhra Pradesh, India in 2015, and the M.S. degree in Computer Science and Engineering from the University of Connecticut, Storrs, CT, USA, in 2017. She is currently a Software Developer with Cigna, Windsor, CT, USA.

Her current research interests include multi-objective path planning, optimization-based techniques, multiprocessing, and efficient algorithm development and implementation of dynamic resource management applications.

**Krishna R. Pattipati** (S'77–M'80–SM'91–F'95) received the B.Tech. degree in Electrical Engineering (with highest honors) from the Indian Institute of Technology, Kharagpur, India, in 1975 and the M.S. and Ph.D. degrees in Systems Engineering from the University of Connecticut, Storrs, in 1977 and 1980, respectively.

From 1980 to 1986, he was with ALPHATECH, Inc., Burlington, MA. Since 1986, he has been with the University of Connecticut, where he is currently the Board of Trustees Distinguished Professor and the UTC Professor in Systems Engineering in the department of Electrical and Computer Engineering, and was the founding director of the UTC Institute for Advanced Systems Engineering there during 2013–2015. His research interests are in the application of systems theory and optimization techniques to large-scale systems.

Dr. Pattipati was selected by the IEEE Systems, Man, and Cybernetics Society as the Outstanding Young Engineer of 1984. He was also a recipient of the Centennial Key to the Future award. He has served as the Editor-in-Chief of the IEEE Transactions on Systems, Man, and Cybernetics: Part B–Cybernetics in 1998–2001, Vice-President for Technical Activities of the IEEE SMC Society in 1998–1999, and Vice-President for Conferences and Meetings of the IEEE SMC Society in 2000–2001. He was the co-recipient of the Andrew P. Sage Award for the Best SMC Transactions Paper for 1999, the Barry Carlton award for the Best AES Transactions Paper for 2000, the 2002 and 2008 NASA Space Act Awards for "A Comprehensive Toolset for Model-based Health Monitoring and Diagnosis," the 2003 AAUP Research Excellence Award, and the 2005 School of Engineering Teaching Excellence Award at the University of Connecticut. He was also the recipient of the Best Technical Paper Awards at the 1985, 1990, 1994, 2002, 2004, 2005, and 2011 IEEE AUTOTEST Conferences, and at the 1997 and 2004 Command and Control Conferences. He is recognized for his contributions to discrete-optimization algorithms for large-scale systems and team decision making.

**David A. Castañón** received his B.S. degree in Electrical Engineering from Tulane University in 1971, and his Ph.D. degree in Applied Mathematics from the Massachusetts Institute of Technology in 1976. He was chief scientist at ALPHATECH, Inc. in Burlington, MA until 1990, when he joined Boston University's Department of Electrical and Computer Engineering. He is a member of the IEEE Control Systems Society and the IEEE Signal Processing Society. He has served as general chair of the IEEE Conference on Decision and Control, and in numerous positions on the IEEE Control Systems Society, including Vicepresident for Finance, and President. Within IEEE, he served as Chair of the Society Review Committee, and the Conference Publications Committee. He also served on the Air Force Scientific Advisory Board, and received the Society's Distinguished Member Award. At Boston University, he served twice as interim Department Chair, and subsequently as Department Chair of the Department of Electrical and Computer Engineering. He also served as co-director of the Center for Information and Systems Engineering. He served Deputy Director of the NSF Engineering Research Center for Subsurface Sensing and Imaging, and is currently Associate Director of the Department of Homeland Securty's ALERT Center of Excellence on Explosives Detection and Mitigation. His research interests include stochastic control, estimation, game theory, optimization, and distributed computing, with applications to inverse problems, multitarget tracking, object recognition, sensor management, and security.