52nd IEEE Conference on Decision and Control
December 10-13, 2013. Florence, Italy

Forward and Reverse Auction Algorithms for Nonlinear Resource Allocation

Ajay Kumar Bangla and David A. Castañón

***Abstract*— We study the problem of optimally assigning N divisible resources to M competing tasks, where the performance cost of each task is a convex function of the resources allocated. This is called the Nonlinear Resource Allocation Problem (RAP). This class of problems arises in diverse fields such as search theory, statistics, finance, economics, logistics, sensor and wireless networks. In our recent work, we proposed a class of algorithms, RAP Auction, which were based on extensions of Auction algorithms for linear assignment problems. RAP Auction was shown to find a near optimal solution in finite time and converge under asynchronous computation. However, major limitations of RAP auction were the lack of stronger complexity results and mediocre empirical performance compared to alternative algorithms. In this paper, we develop a new class of algorithms for the solution RAP problems, based on the use of forward and reverse auction principles, along with scaling techniques. The new algorithms can be shown to have pseudo-polynomial complexity and are significantly faster in standard benchmarks than competing special purpose and general purpose state of the art methods.**

I. INTRODUCTION

Nonlinear Resource Allocation Problems (RAP) are a class of optimization problems where heterogeneous resources have to be allocated to a diverse set of tasks. The underlying performance of executing a task is a convex function of the bundle of resources assigned to it. Interest in RAP is motivated by diverse applications such as in search theory [1]–[3], weapon target assignment [4], [5], sensor management, market equilibria [6], production planning [7], [8], scheduling of mass screening tests [9] and allocation of software-testing resources [10]. The linear cost generalized assignment [11] and transportation [12] problems can be seen as special cases of these problems.

In [13], [14], we developed a new class of auction algorithms called RAP Auction for solving such problems in centralized as well as distributed fashion. They were inspired by success of the auction algorithm for linear assignment and transportation problems [15], [16] and illustrated how ideas from linear problems with bipartite graphs can be extended to nonlinear problems. In RAP Auction, sources with surplus, bid for sinks offering them the highest value. Unlike most existing algorithms for this class of problems, our algorithms work for all convex monotonic utilities including non-differentiable and non-strictly convex functions.

Despite their novelty, centralized implementations of RAP Auction did not compare well both in terms of computational complexity and empirical performance with state-of-art techniques such as the algorithms in [17], [18], [3]. RAPs are convex optimization problems on generalized network (network with gains). Problems in generalized networks are harder than their ordinary network counterparts because cycles in generalized networks can generate or absorb net flow. It is the presence of such flow generating cycles that prevents RAP Auction from having stronger complexity results than finite termination as illustrated in [19]. The results in [19] claimed a bound in the number of iterations required to resolve a given cycle, which was later found to be in error. Furthermore, the empirical implementations of RAP Auction depend on the accuracy parameter for the solution, ϵ, which required many iterations to achieve high accuracy.

In this paper, we propose a new variation of auction algorithms for RAP, which use combine principles from forward and reverse auction algorithms plus ϵ-scaling to achieve worst-case complexity bounds equivalent or better than those obtained for ϵ-relaxation algorithms for generalized nonlinear networks [17]. The forward algorithm derives from RAP Auction which appropriate modifications which enable us to avoid infinite circulations in cycles, a limitation of our previous algorithms. This forward auction can be seen as a hybrid of auction and ϵ-relaxation: auction ideas accelerate empirical performances while ϵ-relaxation ideas bound the worst case performance. A similar hybrid algorithm was used for improving the performance of solving convex network and generalized network problems in [20], [21]. In the forward auction, sources with surplus allocate resources to sinks. Reverse auction has the opposite interpretation, where sources with negative surplus withdraw resources from sinks. Reverse auction is essential for implementing ϵ-scaling, which leads to pseudo polynomial complexity as well as accelerating the empirical performance. We describe the algorithms, establish the theoretical bounds, we demonstrate the superior empirical performance of our algorithm on a wide range of randomly generated search problems when compared with alternative state of the art algorithms.

The remainder of this paper is organized as follows. In section II, we formulate the RAP and briefly discuss duality for RAP. Forward and reverse auction algorithms are presented and analyzed in sections III and IV, respectively. In section V, we propose ϵ-Scaling. Some numerical experience is reported in section VI. Section VII summarizes our results.

II. PROBLEM FORMULATION

Consider a bipartite graph $\mathcal{G} = (\mathcal{W}, \mathcal{T}, \mathcal{E})$, consisting of a set $\mathcal{W}$ of N source nodes , a set $\mathcal{T}$ of M sink nodes and a set $\mathcal{E}$ of A arcs from $\mathcal{W}$ to $\mathcal{T}$. For each source $i \in \mathcal{W}$, let s_i denote the (*supply* of i). For each arc $(i, j) \in \mathcal{E}$, c_{ij}

This work was supported by AFOSR grants FA9550-07-1-0361 and by ODDR&E MURI Grant FA9550-06-1-0324

The authors are with Google, and Boston University, respectively, ajay.bangla@gmail.com, dac@bu.edu

is the (*gain* of (i,j)). For each sink $j \in \mathcal{T}$, we have a closed, convex *non-increasing cost function* $f_j : \Re^+ \mapsto \Re$. The nonlinear Resource Allocation Problem (RAP) is

$$\min_{\mathbf{x},\mathbf{z}} \quad f(\mathbf{z}) := \sum_{j \in T} f_j(z_j) \tag{1a}$$

$$\text{subject to} \quad \sum_{j \in \mathcal{T}_i} x_{ij} = s_i \qquad \forall\, i \in \mathcal{W} \tag{1b}$$

$$\sum_{i \in W_j} c_{ij} x_{ij} = z_j \qquad \forall\, j \in \mathcal{T} \tag{1c}$$

$$\mathbf{x} \geq \mathbf{0},\ \mathbf{z} \geq \mathbf{0} \tag{1d}$$

where $\mathcal{T}_i = \{j : (i,j) \in E\}$ are the sinks connected to the i^{th} source, $\mathcal{W}_j = \{i : (i,j) \in \mathcal{E}\}$ are the sources connected to j^{th} sink, $\mathbf{x} \triangleq \{x_{ij} | (i,j) \in \mathcal{E}\}$ is the flow vector, and $\mathbf{z} \triangleq \{z_j | j \in \mathcal{T}\}$ is the *demand* vector.

Introducing multipliers μ_i and p_j (also called *prices*) for the flow conservation constraints at the sources $\mathcal{W}$ and sinks $\mathcal{T}$, as illustrated in Fig. 1, we get the dual of RAP as $\max_{\boldsymbol{\mu},\mathbf{p}} q(\boldsymbol{\mu},\mathbf{p})$, where the dual function q is given by $q(\boldsymbol{\mu},\mathbf{p}) = \sum_{j \in T} q_j(\boldsymbol{\mu}_{W_j}, p_j) - \boldsymbol{\mu}'\mathbf{s}$ and q_j is defined as

$$q_j(\boldsymbol{\mu}_{W_j}, p_j) = \inf_{z_j \geq 0} \{f_j(z_j) + p_j z_j\} + \sum_{i \in W_j} \inf_{x_{ij} \geq 0} \{(\mu_i - c_{ij} p_j) x_{ij}\}.$$

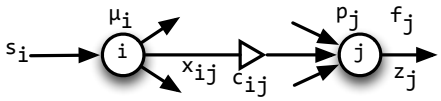


Fig. 1: Visualization of primal variables (flow x_{ij}, demand z_j) and dual variables (source price μ_i, sink price p_j).

We say that a flow-demand vector pair $(\mathbf{x},\mathbf{z})$ and a price vector pair $(\boldsymbol{\mu},\mathbf{p})$ satisfy *Complementary Slackness* (CS) if $\mathbf{x} \geq 0$, $\mathbf{z} \geq 0$, $\boldsymbol{\mu} \geq 0$, and

$$\mu_i \geq c_{ij} p_j \qquad \forall\, \{i,j\} \in \mathcal{E}, \tag{2a}$$

$$\mu_i = c_{ij} p_j \qquad \forall\, \{i,j\} \in \mathcal{E}\ \&\ x_{ij} > 0, \tag{2b}$$

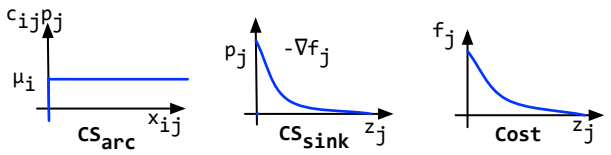
$$p_j \in -[f_j^-(z_j), f_j^+(z_j)] \quad \forall\, j \in \mathcal{T} \tag{2c}$$

where $f_j^-(z_j)$ and $f_j^+(z_j)$ are the *left derivative* and *right derivative* of f_j at z_j, respectively. This can be broken down in two conditions one related to the arcs and one to the sinks:

- The flow-price pair $(\mathbf{x},\mathbf{p})$ satisfies *CS$_{arc}$* if $\mathbf{x} \geq 0$ and (2a) and (2b) hold. This is illustrated in Fig. 2a.
- The demand-price pair $(\mathbf{z},\mathbf{p})$ satisfies *CS$_{sink}$* if $\mathbf{z} \geq 0$ and (2c) holds. This is shown in Fig. 2b.

Strong duality, existence of both primal and dual optimal solutions and existence of multipliers which satisfy *Complementary Slackness* (CS) for any primal feasible solutions were established in [13]. A *feasible* flow-demand pair $(\mathbf{x}^*,\mathbf{z}^*)$ is optimal if there exists a price-vector pair $(\boldsymbol{\mu}^*,\mathbf{p}^*)$ which satisfies the complementary slackness conditions.

The sink price p_j using (2c) can be interpreted as the marginal rate of return with respect to demand z_j, referred



(a) CS$_{arc}$ (b) CS$_{sink}$ (c) Cost Function

Fig. 2: Illustration of CS. All pairs of arc flows x_{ij} and $c_{ij}p_j$ should lie on the blue line, (b) All pairs of demand z_j and price p_j should lie on the blue line and (c) Sample cost function for which (b) is derived.

henceforth as *value*. From monotonicity and convexity of the cost function, this value follows the law of diminishing returns and is always non negative. So as the demand at a sink increases, its value falls. The value offered to the source i by sink j gets scaled by the arc gain c_{ij}. Each source with finite resource must choose between which sinks to allocate to each offering different values. (2a) and (2b) require that sources allocate only to sinks with best value.

ϵ-CS [13] relaxes CS by allowing sources to allocate to sinks with "approximately" best value. We say for any positive scalar ϵ, a flow-price quartet $(\mathbf{x},\mathbf{z},\boldsymbol{\mu},\mathbf{p})$ satisfies ϵ-CS if $(\mathbf{z},\mathbf{p})$ satisfies (2c), $\mathbf{x} \geq 0$, $\mathbf{z} \geq 0$, and flow-price triple $(\mathbf{x},\boldsymbol{\mu},\mathbf{p})$ satisfies ϵ-CS$_{arc}$ below:

$$\mu_i \geq c_{ij} p_j \qquad \forall\, \{i,j\} \in \mathcal{E} \tag{3a}$$

$$c_{ij} p_j \geq \mu_i - \epsilon \quad \forall\, \{i,j\} \in \mathcal{E} \text{ with } x_{ij} > 0 \tag{3b}$$

Note that this definition of ϵ-CS is not the same as originally proposed in [22]; it is a combination of standard CS at the sources and sinks (2a),(2c) with $\epsilon-$CS on the arcs (3b).

The intuition behind the ϵ-CS conditions is that a feasible flow-price pair is "approximately" primal and dual optimal if the ϵ-CS conditions are satisfied as shown in this proposition which was proved in [13], [21].

Proposition 1: Let $(\mathbf{x}^*,\mathbf{z}^*,\boldsymbol{\mu}^*,\mathbf{p}^*)$ be a flow-demand-price quartet satisfying ϵ-CS such that $(\mathbf{x}^*,\mathbf{z}^*)$ is primal feasible (satisfy (1b), (1c) and (1d)), then

$$0 \leq f(\mathbf{z}^*) - q(\boldsymbol{\mu}^*,\mathbf{p}^*) \leq \epsilon\ \mathbf{s}'\mathbf{1}. \tag{4}$$

In this paper, we are interested in primal-dual methods which maintain a flow-price quartet $(\mathbf{x},\mathbf{z},\boldsymbol{\mu},\mathbf{p})$ that satisfies ϵ-CS but is not primal feasible, and then iterate over prices and flows with ϵ-CS as an invariant while approaching primal feasibility. These methods terminate when all the resources have been fully allocated and satisfy primal feasibility. In a typical iteration, prices have to be computed for given demands, and demands for given prices which are CS$_{sink}$ consistent. This requires maps Φ_j and its inverse Θ_j:

$$\Phi_j : Z_j \mapsto P_j \qquad \Theta_j : P_j \mapsto Z_j \tag{5}$$

where $Z_j = \Re^+$ is the demand space and $P_j = \Phi_j(Z_j)$ is the price space. From convexity and monotonicity of cost functions, we have $P_j \in \Re^+$. For continuously differentiable and strictly convex cost functions, $\Phi_j(z_j) = -\nabla f_j(z_j)$ (illustrated in Fig. 2b) and $\Theta_j(p_j) = \nabla {f_j}^{-1}(-p_j)$. In [13], [21], we define these mappings without these assumptions.

III. Forward Auction

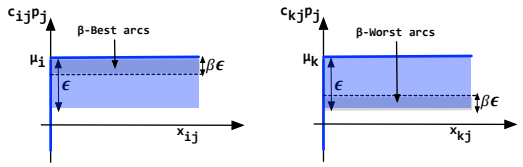
Though the forward auction derives from the previous RAP Auction, it modifies this in a careful way so as not to inherit RAP Auction's limitations concerning cycles. Ginve a current set of flows $\mathbf{x}$, define the surplus of source i as

$$g_i = s_i - \sum_{j \in \mathcal{T}_i} x_{ij}$$

In RAP Auction, a source i with surplus bids for sink j offering it the best value i.e., $j = \arg\max_{j \in T_i} c_{ij} p_j$ and sinks accept flow by increasing demand, dropping their price. We modify this by allowing bids on "nearly" the best arc: Given $\epsilon > 0$, $\beta \in (0, 0.5)$, and flow-price quartet $(\mathbf{x}, \mathbf{z}, \boldsymbol{\mu}, \mathbf{p})$ satisfying ϵ-CS, an arc $(i, j) \in \mathcal{E}$ is *β-best* if

$$c_{ij} p_j > \mu_i - \beta\epsilon. \tag{6}$$

As shown in Fig. 3, β-best arcs offer "almost" the best value ($c_{ij} p_j$) to the source i and the maximum possible drop in sink price before violating ϵ-CS, and thus are near-optimal arcs to allocate new flow on while preserving ϵ-CS. For each source



(a) β-best (b) β-worst

Fig. 3: Visualization of ϵ-CS$_{\text{arc}}$, β-best and β-worst arcs. Blue region are flow-price pairs that satisfy ϵ-CS$_{\text{arc}}$. Darker strips are flow-price pairs for β-best and β-worst arcs.

k its *bid list*, denoted as $\mathcal{L}^+(k)$, is defined as the set of all β-best outgoing arcs.

As a sink j gets new flow allocated to it, its price drops according to the mapping (5). If this sink has received flow from any other source, say k, then (2b) its the possible price drop before ϵ-CS$_{\text{arc}}$ is violated. Such flows have to be reversed before p_j falls below $(\mu_k - \epsilon)/c_{kj}$. An arc (k, j) satisfies this relation if it lies on the lower boundary of the blue region in Figure 3b with $x_{kj} > 0$. Hence this arc offers the worst value to the source k among all the arcs which have received flow from it. However rather than checking this condition exactly, we allow sinks to reverse flow on "almost" the worst arcs which are defined as: for any flow-price quartet $(\mathbf{x}, \mathbf{z}, \boldsymbol{\mu}, \mathbf{p})$ satisfying ϵ-CS, we say that an arc $(k, j) \in \mathcal{E}$ is *β-worst* if $x_{kj} > 0$ and

$$c_{kj} p_j < \mu_k - (1 - \beta)\epsilon. \tag{7}$$

β-worst arcs offer "almost" the worst value to source k among arcs with flow, and prevent the sink j from dropping its price significantly without violating e-CS. Thus they are near-optimal arcs to reverse flow on.

Since all flow changes are restricted to be on the β-best and worst arcs, the *admissible graph* is defined as

$$\mathcal{G} = \{(i,j) | (i,j) \in \mathcal{E} \text{ is } \beta\text{-best}\} \cup \{(j,i) | (i,j) \in \mathcal{E} \text{ is } \beta\text{-worst}\} \tag{8}$$

For each β-best arc (i, j) we introduce a forward arc (i, j) in $\mathcal{G}$, and for each β-worst arc (i, j) we introduce a reverse arc (j, i) in $\mathcal{G}$. In what follows, by a *path* $\mathcal{P}$ in $\mathcal{G}$, we mean a sequence of nodes $(i_1, i_2 ..., i_m)$ in $\mathcal{W} \cup \mathcal{T}$ and an associated sequence of $(m-1)$ arcs in $\mathcal{E}$ such that, for each $k = 1, ..., m-1$, (i_k, i_{k+1}) or (i_{k+1}, i_k) is an arc of the sequence. The set of forward arcs of $\mathcal{P}$ (those of the form (i_k, i_{k+1}) is denoted by $\mathcal{P}^+$ and the set of reverse arcs of $\mathcal{P}$ (those of the form (i_{k+1}, i_k)) is denoted by $\mathcal{P}^-$. We define the *gain of the path* $\mathcal{P}$ by

$$\gamma_{\mathcal{P}} = \prod_{(i,j) \in \mathcal{P}^+} c_{ij} / \prod_{(i,j) \in \mathcal{P}^-} c_{ij}, \tag{9}$$

with $\gamma_{\mathcal{P}} = 1$ if $\mathcal{P}$ comprises of a single node. A *cycle* is a path whose starting node equals the ending node. Note that admissible graph may contain cycles. Let $\mathcal{C}$ denote one such cycle. If $\gamma_{\mathcal{C}} > 1$, the cycle is said to be *flow generating* else it is *flow absorbing*. We now show that cycles in $\mathcal{G}$, if any, are flow generating.

Lemma 1: Cycles in admissible graph are flow generating.

Proof: Let $\mathcal{C}$ be one such cycle in $\mathcal{G}$ where $\mathcal{C} = (i_1, j_1, i_2, \ldots, j_m, i_{m+1} = i_1)$ and $\forall\ k = 1, \ldots, m$ and $(i_k, j_k) \in \mathcal{C}^+$ are the β-best arcs while $(i_{k+1}, j_k) \in \mathcal{C}^-$ are the β-worst arcs. From (6) and (7), we have

$$\frac{c_{i_1 j_1}}{c_{i_1 j_m}} > \frac{p_{j_m}}{p_{j_1}} \ \& \ \frac{c_{i_k j_k}}{c_{i_k j_{k-1}}} > \frac{p_{j_{k-1}}}{p_{j_k}} \quad \forall\ k = 2, \ldots, m$$

So $\gamma_{\mathcal{C}} \triangleq \frac{c_{i_1 j_1}}{c_{i_1 j_m}} \prod_{k=2}^{m} \frac{c_{i_k j_k}}{c_{i_k j_{k-1}}} > \frac{p_{j_m}}{p_{j_1}} \prod_{k=2}^{m} \frac{p_{j_{k-1}}}{p_{j_k}} = 1.$ ■

Forward auction has two phases: Bidding and Allocation. During the bidding phase a source with surplus bids on its β-best arcs until all the surplus is exhausted. However if a cycle is detected during this phase, then to avoid large number of circulations, we introduce a special step. Lemma 1 proved that all such cycles are flow generating. So it is possible to saturate cycles by pushing large enough flow into it. A *δ-flow push* along a cycle $\mathcal{C}$ increases (respectively, decreases) flow x_{kl} by a positive amount $\gamma_{\mathcal{C}_k} \delta$ for all forward arcs $(k, l) \in \mathcal{C}^+$ (respectively, for all reverse arcs $(k, l) \in \mathcal{C}^-$). A cycle is said to be *saturated* during such a flow push when one of its reverse arcs is saturated. A reverse arc is said to be *saturated* when its flow becomes zero. The minimum flow push to saturate a cycle $\mathcal{C}$ can be computed as $\delta = \min_{(k,l) \in \mathcal{C}^-} x_{kl} / \gamma_{\mathcal{C}_k}$ where $\mathcal{C}_k$ denotes the portion of $\mathcal{C}$ from i to k.

Bidding Phase:

Select a source $i \in \mathcal{W}$ with $g_i > 0$; if no such source can be found, the bidding phase terminates.

Step 1: (Choose a β-best arc) If the bid list $\mathcal{L}^+(i)$ is empty, go to Step 2. Else, choose any β-best arc (i, j). Check if this arc is still β-best, else remove it from the bid list and restart Step 1. If this arc belongs to some cycle $\mathcal{C}$ of the admissible graph $\mathcal{G}$, go to Step 3b. Otherwise go to Step 3a.

Step 2: (Update price of source i) Drop μ_i to the level

$$\bar{\mu}_i = \max_{j \in \mathcal{T}_i} c_{ij} p_j \tag{10}$$

Compute the bid list of i. If $g_i > 0$ go to Step 1 else exit.

Step 3a: (Bid for sink j) Increase x_{ij} by g_i, set $g_i = 0$ and set the bid price of this flow to

$$b_i = \begin{cases} (\mu_i - \epsilon)/c_{ij} & \text{if } |\mathcal{L}^+(i)| > 1 \\ (v_{sec} - \epsilon)/c_{ij} & \text{else} \end{cases} \tag{11}$$

where

$$v_{sec} = \begin{cases} \max_{k \in \mathcal{T}_i \backslash j} c_{ik} p_k & \text{if } |\mathcal{T}_i \backslash j| > 0 \\ 0 & \text{else} \end{cases}.$$

Run allocation phase at sink j. Check if (i, j) is still β-best, else remove it from $\mathcal{L}^+(i)$. Now if $\mathcal{L}^+(i) = \emptyset$ go to Step 2. If $g_i > 0$ go to Step 1 else exit.

Step 3b: (Push δ-flow along cycle $\mathcal{C}$), where

$$\delta = \min_{(k,l) \in \mathcal{C}^-} x_{kl}/\gamma_{\mathcal{C}_k}.$$

Update $g_i = g_i + (\gamma_C - 1)\delta$ and go to Step 1.

A sink which receives a bid, now has a surplus $g_j = c_{ij} g_i'$ where g_i' denotes the surplus at source i before executing Step 3a. This surplus is accepted during the allocation phase.

Allocation Phase:

Step 1: (Choose a β-worst arc) If the reject list of sink j, $\mathcal{L}^+(j)$, is empty, go to Step 2. Otherwise, choose any β-worst arc, say (k, j). Check if it's still β-worst, else remove this arc from the reject list and restart Step 1. Go to Step 3.

Step 2: (Update price of sink j) Drop p_j to the level

$$\bar{p}_j = \max\{\max_{k \in \mathcal{W}_j \backslash i \ \&\ x_{kj} > 0} (\mu_k - \epsilon)/c_{kj}, b_i, \Phi_j(z_j + g_j)\} \tag{12}$$

while absorbing flow $\Delta z = \min\{g_j, \Theta_j(\bar{p}_j) - z_j\}$. So the demand increases to $z_j = z_j + \Delta z$ and surplus reduces to $g_j = g_j - \Delta z$. If the cumulative price drop since the last iteration when its reject list was computed is substantive i.e., $\hat{p}_j - p_j \geq \beta \epsilon_{\min}$ where $\hat{p}_j$ denotes the price at which current reject list was computed, then rebuild the reject list. If $p_j < b_i + \beta\epsilon/c_{ij}$, then add (i, j) to the reject list. If $g_j = 0$ stop; else go to Step 1.

Step 3: (Reverse flow along arc (k, j)) Decrease x_{kj} by $\delta = \min\{g_j/c_{ij}, x_{kj}\}$, decrease $g_j = g_j - c_{ij}\delta$, and increase $g_k = g_k + \delta$. If now $g_j = 0$ stop; else go to Step 1.

At the sink there are two notions of price drops: one is the difference between previous price and current price while cumulative price drop is the difference between the sink price at which the current reject list was computed and the current sink price. A price drop at a source (sink, respectively) is *substantive* if it is at least $\beta\epsilon$ ($\beta\epsilon_{\min}$, respectively).

Note that during the allocation phase, the β-best arc (i, j) on which the bid was received, may end up becoming a β-worst arc due to price drop at sink j. In such a case, we say that the *forward arc is saturated*. One of the scenarios when a forward arc (i, j) is saturated is when sink j is unable to accept all the surplus being bid to it by source i as at the end of such an allocation phase we have $p_j = b_i$. We refer to this particular scenario as *saturated bid* from source i to j. A saturated bid results in a substantive price drop at sink j. Also non substantive bids are exhausting (surplus of the bidding source becomes zero).

The way the algorithm has been defined, each bidding phase may result in multitude of bids to different sinks until the source is exhausted. This constitutes one iteration of the forward auction. Each iteration consists of at least one bid and at least one allocation phase. We make the following observations about these iterations which are proved in [21].

1) The iterations preserve ϵ-CS and the prices are monotonically non-increasing.
2) Every source price drop is substantive. However, this may not be true for sinks but reject lists are only computed after substantive cumulative sink price drops.
3) Every iteration preserves flow conservation at sinks (1c) if the initial solution satisfies this. At the end of allocation phase, sink surplus is zero.
4) If any source or sink in a given iteration performs flow pushes on multiple arcs, then all but the last are saturating. Flow pushes collectively refers to all possible arc flow changes in the algorithm such as sources biding to sinks on β-best arcs, sources pushing flow into cycles to saturate them, and sinks reversing flows on β-worst arcs. This is simply because non saturating flow pushes (both non saturating bids and reversals) are exhausting. Also a flow push in a cycle, results in saturation of at least one β-worst arc.

In general, finding the cycle $\mathcal{C}$ in Step 1 of bidding is expensive. Due to observation 4, such a cycle can be found without excessive overhead by growing a path as opposed to growing a tree. The following method of choosing sources during the bidding phase facilitates cycle detection.

(a) Select any source i_0 with positive surplus. If no such node exists, terminate the method. Else let $k = 0$.
(b) Perform a bidding iteration at i_k. If there are any changes in the admissible graph (happens if there are any saturated flow pushes or source price drops or substantive cumulative sink price drops) then go to (a) else let j denote the unique sink being bid for and i denote the source to which flow has been reversed during the allocation iteration at j. If there is no flow reversal, then go to (a). Set $j_k = j$. If $i = i_l$ for some $l < k$, go to (c). Else let $i_{k+1} = i$, increment k by 1 and go to (b).
(c) A cycle whose forward arcs are all β-best and whose reverse arcs are all β-worst is $\mathcal{C} : i_l, j_l, i_{l+1}, j_{l+1}, \ldots, i_k, j_k, i_l$.

A. Termination of forward auction

Since each source price drop is substantive and prices are constrained to be non negative, there can only be finitely many price drops. By analyzing changes in the admissible graph, we have the following proposition which bounds the number of operations between successive source price drops.

Proposition 2: The number of flow pushes along arcs (respectively, cycles) between two successive source price drops (not necessarily at the same source) is at most $N^2(2A + MC/\beta)$ (respectively, $2A$) while the number of substantive and non substantive cumulative sink price drops are at most MC/β and NA, respectively where

$$C = \max_{(i,j)\in\mathcal{E}} c_{ij} / \min_{(i,j)\in\mathcal{E}} c_{ij}$$

.

Proof: For lack of space, we only outline the proof here. The detailed proof can be found in [21]. Between two successive source price changes, the admissible graph $\mathcal{G}$ changes only when there are saturated flow pushes or substantive cumulative sink price drops. We bound the number of such operations by $2A+MC/\beta$ and the number of flow pushes in a fixed $\mathcal{G}$ to at most $2(N-1)N$. Thus, between successive source price drops, the admissible graph can change at most $2A+MC/\beta$ times; between successive changes in the admissible graph, there are at most N^2 flow pushes.

With this result, we have finite termination of forward auction in a near optimal feasible solution provided the initial solution $(\mathbf{x}^0, \mathbf{z}^0, \boldsymbol{\mu}^0, \mathbf{p}^0)$ satisfies ϵ-CS, sink flow conservation (1c) and all the source surpluses are non-negative. One possible initial solution can be computed as follows:

$$\begin{aligned} &\mathbf{z}^0 = \mathbf{0},\ \mathbf{x}^0 = \mathbf{0}, \\ &\quad p_j^0 = \Phi_j(0) \qquad \forall\ j \in \mathcal{T},\ \text{and} \\ &\mu_i^0 = \max_{j\in\mathcal{T}_i} c_{ij} p_j^0 \quad \forall\ i \in \mathcal{W}. \end{aligned}$$

IV. Reverse Auction

In forward auction, iterations are performed only on sources with surplus. If any of the sources have deficits (negative surplus), then forward auction will ignore them, and will terminate with some sources with deficits. Assume some sources have deficits. To handle this case, we need a reverse algorithm where sources with deficits withdraw flow from sinks. This can be also seen as allocation of deficits to sinks as opposed to surpluses during forward auction. A sink receiving a deficit can try to absorb it by reducing its demand and hence raising its price. In this section, we propose such an algorithm called reverse auction.

During the bidding phase, a source i with deficit picks arcs on which to bid this deficit. Among arcs on which it has allocated flow, β-worst arcs are among those offering it the lowest value and possibility of largest ϵ-CS preserving sink price rise as shown in Figure 3b. Hence they are near-optimal candidates for *deficit push* (negative flow push). Now the set of β-worst arcs is the bid list $\mathcal{L}^-(i)$ for source i.

During the allocation phase, the presence of β-best arcs prevents the sink j from raising its prices significantly to absorb the deficit allocated to it as shown in Figure 3. Hence, the deficit is reversed (flow is pulled) on these arcs. The set of all such β-best arcs is the reject list $\mathcal{L}^-(j)$ of the sink j.

Since all flow changes due to deficit pushes are restricted to be on the β-best and worst arcs, the admissible graph is now defined as

$$\mathcal{G} = \{(i,j)|(j,i) \in \mathcal{E} \text{ is } \beta\text{-best}\} \cup \{(i,j)|(i,j) \in \mathcal{E} \text{ is } \beta\text{-worst}\}. \tag{13}$$

Unlike the admissible graph of forward auction, this graph captures the possible deficit pushes as opposed to flow pushes. The role of β-best and worst arcs has been swapped as compared to that in forward auction. This swapping has the following impact of the cycles in the admissible graph which follows from application of (6) and (7),

Lemma 2: Cycles in admissible graph are flow absorbing.

Reverse auction also has two phases: Bidding and Allocation. During the bidding phase a source with deficits bids on its β-worst arcs until all the deficit is exhausted. However if a cycle is detected, deficit is pushed till either the cycle saturates or the source is exhausted whichever happens earlier. A δ-*deficit push* along a cycle $\mathcal{C}$ decreases (respectively, increases) flow x_{kl} by a positive amount $\gamma_{\mathcal{C}_k}\delta$ for all forward arcs $(k,l) \in \mathcal{C}^+$ (respectively, for all reverse arcs $(k,l) \in \mathcal{C}^-$). A *cycle is said to be saturated* during such a deficit push when flow in one of its forward arcs becomes zero. The minimum deficit push to saturate a cycle $\mathcal{C}$ can be computed as $\delta = \min_{(k,l)\in\mathcal{C}^+} x_{kl}/\gamma_{\mathcal{C}_k}$.

Bidding Phase:

Select a source $i \in \mathcal{W}$ with $g_i < 0$; if no such source can be found, the bidding phase terminates.

Step 1: (Choose a β-worst arc) If the bid is list $\mathcal{L}^-(i)$ is empty, go to Step 2. Otherwise, choose any β-worst arc, say (i,j). Check if this arc is still β-worst, else remove it from the bid list and restart Step 1. If this arc belongs to some cycle $\mathcal{C}$ of the admissible graph $\mathcal{G}$, go to Step 3b. Otherwise go to Step 3a.

Step 2: (Update price of source i) Raise μ_i to the level

$$\bar{\mu}_i = \min_{j\in\mathcal{T}_i|x_{ij}>0} c_{ij}p_j + \epsilon \tag{14}$$

and populate the bid list of i. If $g_i < 0$ go to Step 1 else exit.

Step 3a: (Bid for sink j) Decrease x_{ij} by δ where $\delta = \min\{-g_i, x_{ij}\}$, increase g_i by δ, and set the bid price of this flow to

$$b_i = \begin{cases} \mu_i/c_{ij} & |\mathcal{L}^-(i)| > 1 \\ v_{sec}/c_{ij} & \text{else} \end{cases}. \tag{15}$$

where

$$v_{sec} = \begin{cases} \min_{k\in\{\mathcal{T}_i\setminus j|x_{ik}>0\}} c_{ik}p_k + \epsilon & \text{if } |\{\mathcal{T}_i\setminus j|x_{ik}>0\}| > 0 \\ c_{ij}p_{\max} & \text{else} \end{cases}.$$

Here $p_{\max} \triangleq \max_{j\in\mathcal{T}} \inf P_j$. Run allocation phase at sink j which now has deficit $g_j = -c_{ij}\delta$. Check if (i,j) is still β-worst, else remove it from $\mathcal{L}^-(i)$. Now if $\mathcal{L}^-(i) = \emptyset$ go to Step 2. If $g_i < 0$ go to Step 1 else exit.

Step 3b: (Push δ-deficit along cycle $\mathcal{C}$), where

$$\delta = \min\{-g_i/(1-\gamma_{\mathcal{C}}) \min_{(k,l)\in\mathcal{C}^+} x_{kl}/\gamma_{\mathcal{C}_k}\}.$$

Update $g_i = g_i + (1-\gamma_C)\delta$. If $g_j < 0$, go to Step 1.

A sink from which receives a bid has a deficit. It can accept this deficit by absorption or reversal during allocation.

Allocation Phase:

Step 1: (Choose a β-best arc) If the reject list of sink j, $\mathcal{L}^-(j)$, is empty, go to Step 2. Otherwise, choose from the reject list any β-best arc, say (k, j). Check if it's still β-best, else remove this arc from the reject list and restart Step 1. Go to Step 3.

Step 2: (Increase price of sink j) Increase p_j to the level

$$\bar{p}_j = \min\{\min_{k \in \mathcal{W}_j \setminus i} \mu_k / c_{kj}, b_i, \Theta_j(z_j + g_j)\} \qquad (16)$$

while withdrawing flow $\Delta z = \min\{-g_j, z_j - \Phi_j(\bar{p}_j)\}$. So the demands reduces to $z_j = z_j - \Delta z$ and $g_j = g_j + \Delta z$. If the cumulative price rise since the last iteration when its reject list was computed is substantive, then rebuild the reject list. If $p_j > b_i - \beta\epsilon/c_{ij}$, then add (i, j) to the reject list. If now $g_j = 0$ stop; else go to Step 1.

Step 3: (Reverse deficit along arc (k, j)) Increase x_{kj} by $-g_j/c_{kj}$, set $g_k = g_k + g_j/c_{kj}$ and set $g_j = 0$.

By analyzing changes in the admissible graph, we have the following proposition which bounds the number of operations between successive source price rises derived along the lines of proposition 2. In this proposition, flow pulls collectively refers to all flow changes in the algorithm such as sources bidding to sinks on β-worst arcs, sources pushing deficit into cycles, and sinks reversing deficits on β-best arcs.

Proposition 3: The number of flow pulls along arcs (respectively,cycles) between two successive source price rises (not necessarily at the same source) is at most $N^2(A + MC/\beta)$ (respectively, $N(A+MC/\beta)$) while the number of substantive and non substantive cumulative sink price rises are at most MC/β and NA.

With this result, we have finite termination of reverse auction in a near optimal feasible solution provided the initial solution $(\mathbf{x}^0, \mathbf{z}^0, \boldsymbol{\mu}^0, \mathbf{p}^0)$ satisfies ϵ-CS, sink flow conservation (1c) and all the source surpluses are non-positive.

V. ϵ-Scaling

In this section, we will illustrate how the performance of our algorithms can be improved by ϵ-scaling, applying the above algorithms starting with a large value of ϵ and successively reducing ϵ up to some final value $\bar{\epsilon}$ deemed sufficiently small. Let ϵ^k denote the value used for the $(k+1)$st scaling phase. We define

$$\epsilon^{k+1} = \epsilon^k/\theta, \quad k = 0, 1, \ldots$$

where ϵ^0 is a suitably chosen starting value of ϵ, and $\theta > 1$. Then the total number of such scaling phases is $O(\ln(\epsilon^0/\bar{\epsilon}))$ where final value $\bar{\epsilon}$ deemed sufficiently small.

A. Generating the starting solution

The starting solution for a given scaling phase is generated from the solution of the previous scaling phase by recalling flows on arcs violating ϵ-CS without changing the prices. Recalling flows creates *deficits* (negative surpluses) at sinks and both forward and reverse auction need the starting solutions to have no deficits at sinks. There are different ways to handle sink deficits as we saw in the allocation phase of reverse auction: either the sinks can absorb them by raising prices or transfer them to sources. Deficits at sources can be exhausted by applying the reverse auction.

Let $(\mathbf{x}^k, \mathbf{z}^k, \boldsymbol{\mu}^k, \mathbf{p}^k)$ be the flow-price quartet obtained at the k^{th} scaling phase. Then $\{\mathbf{x}^k, \mathbf{z}^k\}$ is feasible and satisfies ϵ^k-CS with $(\boldsymbol{\mu}^k, \mathbf{p}^k)$. To start the $(k+1)$th scaling phase, we need a starting solution which satisfies ϵ^{k+1}-CS. We propose the following procedure to generate such a starting point.

Generate starting solution for $k+1$ stage from $(\mathbf{x}^k, \mathbf{z}^k, \boldsymbol{\mu}^k, \mathbf{p}^k)$:

Step 1: Recall flow on all the arcs which violate ϵ^{k+1}-CS. This introduces deficits at a set of sinks, say, $\hat{\mathcal{T}}$. For each sink $j \in \hat{\mathcal{T}}$, let $\hat{\mathcal{W}}_j$ denote the set of sources which have recalled flow from it. Assume that sources in $\hat{\mathcal{W}}_j$ are sorted in descending order of $(\mu_i - \epsilon^{k+1})/c_{ij}$.

Step 2: For each sink $j \in \hat{\mathcal{T}}$,

- Compute $\hat{p}_j = \min_{i \in \mathcal{W}_j} \mu_i/c_{ij}$ and let $\hat{i}$ denote any one of the arguments this minimization. This is an ϵ-CS satisfying upper bound on p_j.
- For each source $i \in \hat{\mathcal{W}}_j$ such that $(\mu_i - \epsilon^{k+1})/c_{ij} \leq \hat{p}$:
  - Raise price: $p_j = \min\{(\mu_i - \epsilon^{k+1})/c_{ij}, \Theta_j(z_j + g_j)\}$ while withdrawing flow $\Delta z = \min\{-g_j, z_j - \Phi_j(p_j)\}$. So the demands reduces to $z_j = z_j - \Delta z$ and $g_j = g_j + \Delta z$. If g_j is zero, exit.
  - Reclaim recalled flow: Since $p_j = (\mu_i - \epsilon^{k+1})/c_{ij}$, ϵ^{k+1}-CS is satisfied on (i, j). Increase x_{ij} by $\delta = \min\{g_i, -g_j/c_{ij}\}$ reducing $g_j = g_j + c_{ij}\delta$ and $g_i = g_i - \delta$. If g_j is zero, exit.
- If still $g_j < 0$,
  - Raise price: $p_j = \min\{\hat{p}_j, \Theta_j(z_j + g_j)\}$ while withdrawing flow $\Delta z = \min\{-g_j, z_j - \Phi_j(p_j)\}$. So the demands reduces to $z_j = z_j - \Delta z$ and $g_j = g_j + \Delta z$. If g_j is zero, exit.
  - Transfer deficit to source $\hat{i}$: Increase $x_{\hat{i}j}$ by $-g_j/c_{\hat{i}j}$, set $g_{\hat{i}} = g_{\hat{i}} - g_j/c_{\hat{i}j}$ and $g_j = 0$. Exit.

We make the following observations about this procedure.

1) The output of this procedure satisfies ϵ^{k+1}-CS. Step 1 ensures this by recalling all flow on arcs which violate it and Step 2 preserves this property.
2) All the sinks have zero deficits while the sources may have surpluses or deficits. Step 2 may transfer the deficits from the sinks to the sources.
3) Source prices are unchanged.
4) The computation complexity of this procedure is $O(A)$.

The procedure to generate starting solution takes the feasible ϵ^k-CS solution to generate a starting solution for next scaling phase such that there is flow conservation at the sinks but the sources may have surpluses or deficits. Now we can use the forward and reverse auction algorithms to exhaust the surpluses and deficits.

Polynomial RAP Auction

Initialization: Choose $\epsilon > 0$ and any ϵ-CS satisfying flow-price quartet $(\mathbf{x}, \mathbf{z}, \boldsymbol{\mu}, \mathbf{p})$. Fix any scalar $\theta \in (0, 1)$.
First phase: Repeatedly choose a source i with positive surplus and run the forward auction algorithm.
Second phase: Repeatedly choose a node i with negative surplus g_i and run the reverse auction algorithm.
ϵ-Scaling. If ϵ is below a desired tolerance, terminate. Otherwise, scale ϵ by θ. Run the procedure to generate starting solution for next scaling phase. Return to First phase.

To show finite termination of a given scaling phase in a feasible solution, we make additional observations about forward auction iterations:

- Once the surplus of a source becomes non negative, it remains non negative for all subsequent forward iterations. The reason is that a flow push from a source cannot make its surplus negative and cannot decrease the surplus of any other source. So no new deficit can be introduced by forward iterations.
- If at some iteration a source has negative surplus, then its price must be equal to its initial price. This is a consequence of above observation and the fact that price rises occur only on sources with positive surplus.

Due to these properties, the first phase terminates with a ϵ-CS satisfying solution where none of the sources have positive surplus. We have analogous properties for the reverse auction iterations which enable us to conclude that the second phase terminates with a feasible ϵ-CS satisfying solution as no new surplus can be introduced during reverse iterations.

For a path $\mathcal{P}$ in $(\mathcal{W}, \mathcal{T}, \mathcal{E})$ with a set of sources $\mathcal{W}_\mathcal{P}$ and starting node s, denote

$$\Gamma_\mathcal{P} = \sum_{i \in \mathcal{W}_\mathcal{P}} \gamma_{\mathcal{P}_i} \tag{17}$$

where $\mathcal{P}_i$ denotes the portion of path $\mathcal{P}$ from s to source i. This is the sum of the cumulative path gains from s to each of the sources in $\mathcal{P}$. The following proposition bounds the total number of price drops and rises for forward and reverse auction, respectively, in terms of $\Gamma_\mathcal{P}$ and other network parameters during a scaling phase starting for the source price vector $\boldsymbol{\mu}^0 = \boldsymbol{\mu}^k$.

Proposition 4: Let θ be any non negative scalar such that the initial source price vector $\boldsymbol{\mu}^0$ for RAP Auction satisfies $\theta\epsilon$-CS together with some feasible flow vector $\mathbf{x}^0$ and price vector $\mathbf{p}^0$. Then, forward and reverse RAP Auction perform at most $((\theta+1)\Gamma-1)/\beta$ price drops and $((\theta+1)(2\Gamma-1))/\beta$ price rises at each source, respectively, where

$$\Gamma := \max_{\mathcal{H}:\text{simple path}} \left\{ \Gamma_\mathcal{H} + \gamma_\mathcal{H} \left(\max_{\mathcal{C}:\text{simple cycle with } \gamma_\mathcal{C} < 1} \frac{\Gamma_\mathcal{C}}{1-\gamma_\mathcal{C}} \right) \right\}$$

where $\gamma_\mathcal{H}$, $\gamma_\mathcal{C}$, $\Gamma_\mathcal{H}$, $\Gamma_\mathcal{C}$ are given by (9) and (17).

Proof: The proof is in [21], and we provide a brief outline here. Consider the flow-price triple $(\mathbf{x}, \boldsymbol{\mu}, \mathbf{p})$ at the beginning of a forward iteration. Consider the flow $\mathbf{x}^0 - \mathbf{x}$. Since $\mathbf{x}^0$ is feasible, the divergence of a source node is $-g_i$ while that of all the sinks is zero. From the conformal realization theorem citeTseng00 to $\mathbf{x}^0 - \mathbf{x}$, for each source s with $g_s > 0$, there a node t and a simple path $\mathcal{H}$ from s to t that conforms to $\mathbf{x}^0 - \mathbf{x}$. Such paths are either simple flow-absorbing cycles or paths that end in nodes with negative surplus. This allows us to bound the number of price drops per source in the forward auction part. Similarly, we can bound the number of price rises per source in the reverse auction part.

It follows from Propositions 2 and 4 that the forward auction in each scaling phase terminates after at most $O(N\Gamma)$ source price drops, $O(A\Gamma C)$ substantive sink price drops, $O(N^3A\Gamma + N^2AC)$ flow pushes along arcs, and $O(N^2A\Gamma)$ flow pushes along cycles. Similarly the reverse phase terminates after at most $O(N\Gamma)$ source price rises, $O(A\Gamma C)$ substantive sink price rise, $O(N^3A\Gamma + N^2AC)$ flow pulls along arcs, and $O(N^2A\Gamma + NA\Gamma C)$ flow pulls along cycles.

The best known network flow based bounds for RAP are obtained by transforming RAP into a min-cost flow problem by introducing flow absorbing cycles at sinks [21]. This can be solved by ϵ-relaxation, one of the fastest algorithms for generalized min-cost flow [23]. The bounds per scaling phase on the number of prices rises (sources and sinks) is $O((N+M)\hat{\Gamma})$, flow pushes on arcs is $O((N+M)^3A\hat{\Gamma})$, and flow pushes in cycles is $O((N+M)^2A\hat{\Gamma})$ while the bounds on the number of prices drops is $O((N+M)\hat{\Gamma}^2)$, flow pulls on arcs is $O((N+M)^3A\hat{\Gamma}^2)$, and flow pulls in cycles is $O((N+M)^2A\hat{\Gamma}^2)$ where

$$\hat{\Gamma} := C. \max_{\mathcal{H}:\text{simple path}} \left\{ \Gamma_\mathcal{H} + \gamma_\mathcal{H} \max_{\mathcal{C}:\text{simple cycle with } \gamma_\mathcal{C} < 1} \frac{\Gamma_\mathcal{C}}{1-\gamma_\mathcal{C}} \right\}$$

Our complexity results are comparable, with a smaller constant $\Gamma < \hat{\Gamma}$, and a tighter bound on accuracy.

VI. Simulations

We implemented the following using Xcode 4.5.2 for benchmarking the performance of our algorithms. It includes

- RAPGen: A generator for generating random instances of large RAPs. The logic for generating the underlying bipartite graph is identical to that used by NETGENG [24] for generating transportation problems.
- RAPA: Our ϵ-scaling auction in this paper.
- RWOA: Our implementation of Resource-Wise Optimization Algorithm [25].
- SPLX: Our implementation of a simplex type algorithm for RAP proposed in [3].
- CVX: A Matlab based convex modeling framework configured to use the convex solver from Mosek, astate-of-art interior point based solver [18].
- ERXG: Implementation of $e-relaxation$ for min-cost flow generalized networks used in [17], [23]

Table I lists the solutions times for solving randomly generated symmetric i.e., $M = N$, search problems [3], on

MacBook Air 5.2 running OS X 10.8.2 operating system. In these problems the cost functions are exponential

$$f_j(z_j) = v_j \exp(-z_j)$$

where v_j are uniformly distributed on $(0, 1)$. For ERXG and RAPA, the final ϵ was chosen for roughly 4 digits of accuracy. ERXG was unable to achieve higher accuracy due to machine precision. RWOA was terminated to roughly achieve the same precision. Whenever possible, the different algorithms were implemented using similar data structures for network access and logic control.

RAPA outperforms every algorithm in all cases and on average is 21, 26, 11, and 286 times faster than ERXG, RWOA, SPLX, and CVX respectively. Both ERXG and RAPA use ϵ-scaling but ERXG needs more scaling phases to achieve the same accuracy. The behavior of RWOA is typical of coordinate descent algorithms. It makes rapid progress in the beginning but suffers a slow convergence.

TABLE I: Solution times (in seconds) for randomly generated symmetric ($N = M$) search problems where N, M and A are the number of sources, sinks and arcs, respectively.

N 100s	**A** 100s	**Gain**	**RAPA** sec.	**RWOA** sec.	**ERXG** sec.	**SPLX** sec.	**CVX** sec.
16	160	$.9-1.1$	0.072	1.925	1.167	0.570	47.53
		$.9-1.2$	0.074	2.321	1.022	0.535	48.44
		$.9-1.5$	0.074	1.478	1.101	0.477	46.48
		$.5-1.5$	0.059	1.258	0.84	0.453	46.88
16	320	$.9-1.1$	0.091	1.015	1.777	0.870	59.84
		$.9-1.2$	0.092	2.128	2.877	0.851	58.04
		$.9-1.5$	0.082	2.221	2.846	0.766	56.41
		$.5-1.5$	0.103	2.272	2.373	0.727	56.43
16	640	$.9-1.1$	0.155	1.620	3.441	1.416	56.43
		$.9-1.2$	0.174	1.458	6.002	1.587	56.43
		$.9-1.5$	0.137	2.282	7.840	1.431	56.43
		$.5-1.5$	0.144	2.462	5.414	1.296	56.43
20	700	$.9-1.1$	0.228	1.876	5.135	2.350	139.41
		$.9-1.2$	0.185	2.633	6.221	3.698	129.07
		$.9-1.5$	0.191	2.213	8.435	3.426	127.95
		$.5-1.5$	0.182	3.306	6.210		124.66
30	700	$.9-1.1$	0.247	5.618	5.185	3.985	415.22
		$.9-1.2$	0.252	3.901	6.656	2.255	407.67
		$.9-1.5$	0.235	6.728	5.848	2.049	373.08
		$.5-1.5$	0.231	6.754	5.267	3.128	375.12
40	700	$.9-1.1$	0.315	8.437	4.687	5.752	938.64
		$.9-1.2$	0.288	8.199	6.184	5.426	941.41
		$.9-1.5$	0.293	10.80	6.616	4.919	916.77
		$.5-1.5$	0.263	9.897	4.916	4.458	952.83

VII. CONCLUSIONS

We have developed a new class of algorithms which can solve RAPs with pseudo-polynomial complexity. The algorithm achieve the best know bounds available for solving RAPs using network flow approaches. In addition, our method outperforms all the other algorithms in standard benchmark problems for generalized networks.

There are several directions in which the work in this paper can be extended. First, our current model does not consider arc costs, but only uses costs associated with the sinks in the bipartite network. Extensions to handle this case are discussed in [21]. One can also extend the techniques discussed here for distributed implementation. Finally, one can explore applications of these techniques to large scale resource allocation problems such as sensor management and investment allocation.

REFERENCES

[1] B. O. Koopman, "The theory of search. III. The optimum distribution of searching effort," *Operations Research*, vol. 5, no. 5, pp. 613–626, Oct 1957.

[2] A. Charnes and W. W. Cooper, "The theory of search: Optimal distribution of search effort," *Management Science*, vol. 5, no. 1, pp. 44–50, 1958.

[3] A. R. Washburn, "Finite method for a nonlinear allocation problem," *Journal of Optimization Theory and Applications*, vol. 85, no. 3, pp. 705–726, June 1995.

[4] ——, "Sortie optimization and munitions planning," *Military Operations Research*, vol. 1, no. 1, pp. 13–18, 1994.

[5] R. K. Ahuja, A. Kumar, K. C. Jha, and J. B. Orlin, "Exact and heuristic algorithms for the weapon-target assignment problem," *Operations Research*, vol. 55, no. 6, pp. 1136–1146, Nov 2007.

[6] N. R. Devanur, C. H. Papadimitriou, A. Saberi, and V. V. Vazirani, "Market equilibrium via a primal–dual algorithm for a convex program," *Journal of ACM*, vol. 55, no. 5, pp. 22:1–22:18, Nov. 2008. [Online]. Available: http://doi.acm.org/10.1145/1411509.1411512

[7] T. Ibaraki and N. Katoh, *Resource Allocation Problems: Algorithmic approaches*. The MIT Press, 1988.

[8] M. Patriksson, "A survey on the continuous nonlinear resource allocation problem," *European Journal of Operational Research*, vol. 185, no. 1, pp. 1–46, Feb. 2008.

[9] H. L. Lee and W. P. Pierskalla, "Mass screening models for contagious diseases with no latent period," *Operations Research*, vol. 36, no. 6, pp. 917–928, 1988.

[10] H. Ohtera and S. Yamada, "Optimal allocation and control problems for software-testing resources," *IEEE Transactions on Reliability*, vol. 39, no. 2, pp. 171–176, Jun 1990.

[11] D. P. Bertsekas, *Network Optimization: Continuous and Discrete Models*. Athena Scientific, 1998.

[12] Y. Censor and S. A. Zenios, *Parallel Optimization: Theory, Algorithms and Applications*. Oxford University Press, 1997.

[13] A. K. Bangla and D. A. Castañón, "Auction algorithm for nonlinear resource allocation problems," in *Proc. of 49th IEEE Conference on Decision and Control*, Atlanta, Georgia, Dec. 2010.

[14] ——, "Asynchronous auction for distributed nonlinear resource allocation," in *Proc. of 50th IEEE Conference on Decision and Control*, Orlando, Florida, Dec. 2011.

[15] D. P. Bertsekas, "Auction algorithms for network flow problems: A tutorial introduction," *Computational Optimization and Applications*, vol. 1, pp. 7–66, 1992.

[16] D. Bertsekas and D. Castañón, "The auction algorithm for the transportation problem," *Annals of Operations Research*, vol. 20, no. 1, pp. 67–96, December 1989.

[17] P. Tseng and D. P. Bertsekas, "An ϵ-relaxation method for separable convex cost generalized network flow problems," *Mathematical Programming*, vol. 88, no. 1, pp. 85–104, June 2000.

[18] E. D. Andersen, "The MOSEK optimization tools manual, version 6.0," http://www.mosek.com.

[19] A. K. Bangla and D. A. Castañón, "Pseudo-polynomial auction algorithm for nonlinear resource allocation problems," in *American Controls Conference*, San Francisco, CA, June. 2011.

[20] ——, "Accelerated dual coordinate algorithms for separable convex cost network flow problems," in *Proc. of 51th IEEE Conference on Decision and Control*, Maui, Hawaii, Dec. 2012.

[21] A. K. Bangla, "Auction algorithms for generalized network flow problems." Ph.D. dissertation, Boston University, 2013.

[22] D. P. Bertsekas, P. A. Hossein, and P. Tseng, "Relaxation methods for network flow problems with convex arc costs," *SIAM Journal of Control and Optimization*, vol. 25, no. 5, pp. 1219–1243, 1987.

[23] F. Guerriero and P. Tseng, "Implementation and test of auction methods for solving generalized network flow problems with separable convex cost," *Journal of Optimization Theory and Applications*, vol. 115, no. 1, pp. 113–144, Oct 2002.

[24] J. W. Hultz, "Algorithms and applications for generalized networks." Ph.D. dissertation, The University of Texas at Austin, 1976.

[25] H. Luss and S. K. Gupta, "Allocation of effort resources among competing activities," *Opx. Res.*, vol. 23, no. 2, pp. 360–366, 1975.