Pseudo-Polynomial Auction Algorithm 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. This is called the Nonlinear Resource Allocation Problem (RAP). Recently, we proposed a new algorithm, RAP Auction [1], which finds a near optimal solution in finite time. It works for monotonic convex cost functions and has a simple computation structure. In this paper, we propose an elegant extension which enables RAP Auction to have pseudo-polynomial complexity.

I. INTRODUCTION

Let us consider the problems where heterogeneous resources have to be allocated in continuous amounts to a diverse set of tasks. The underlying performance of executing a task is a nonlinear function of the bundle of resources assigned to it. Problems of this type arise in diverse applications such as search theory [2]–[4], weapon target assignment [5], [6], sensor management, and market equilibria [7].

In [1], [8], we developed a new class of finite time algorithms called RAP Auction for solving such problems. This algorithm was inspired by success of the auction algorithm for linear assignment problems. In essence, there is a price for each task node and in each iteration, source nodes with surpluses bid for their best tasks. The task node being bid for, decides on how much resource to accept from the bidding source node. This simple compute structure inherently makes this algorithm suitable for distributed implementation. It works for generalized monotonic convex functions including non-differential or/and non-strictly convex functions.

RAPs are convex optimization problems on generalized network (network with gains). Such network are usually harder than their ordinary network counterparts because cycles in generalized networks can generate or absorb flow. It is the presence of such cycles that prevents RAP Auction from having stronger complexity results than finite termination. One resolution to this dilemma is to detect and resolve cycles as proposed in [9]. However, this requires additional bookkeeping and imposes a certain sequential bidding order.

In this paper, we propose a simple yet potent extension to RAP Auction which enables it to have pseudo polynomial complexity. This is achieved by addition of a single new step without any need for cycle detection or additional bookkeeping or imposing a bidding order. Thus it preserves all the benefits of the existing computation structure. The convergence proof for this simple extension is novel and non-trivial.

The remainder of this paper is organized as follows. In section II, we formulate the RAP and briefly discuss duality for RAP. Section III reviews RAP Auction algorithm. In section IV, we propose our extensions to RAP Auction and prove its convergence. Some numerical experience is reported in section V. Section VI summarizes our results.

II. PROBLEM FORMULATION

Consider a bipartite graph $G = (W,T,E)$, a triple, consisting of a set of $N$ source nodes, a set of $M$ sink nodes and a set of arcs, respectively. We are given, for each source $i \in W$, a scalar $s_i$ (supply of $i$), for each arc $(i,j) \in E$, a positive scalar $c_{ij}$ (gain of $(i,j)$) and at each sink $j \in T$ a non-increasing, closed, convex cost function $f_j : \mathbb{R}^+ \mapsto \mathbb{R}$. We now define the nonlinear Resource Allocation Problem (RAP) as

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

subject to

$$\sum_{i \in T_i} x_{ij} = s_i \quad \forall \ i \in W$$

and

$$\sum_{i \in W_j} c_{ij}x_{ij} = z_j \quad \forall \ j \in T$$

where $T_i = \{j : (i,j) \in E\}$ is the set of sinks connected to the $i$th source, $W_j = \{i : (i,j) \in E\}$ is the set of sources connected to the $j$th sink, $x \triangleq \{x_{ij} : (i,j) \in E\}$ is the flow vector, and $z \triangleq \{z_j : j \in T\}$ is the demand vector.

Introducing multipliers $\mu_i$ and $p_j$ (also called sink prices) for the flow conservation constraints at the source $W$ and sink $T$ nodes, respectively, we get the dual of RAP as

$$\max_{\mu,p} q(\mu, p)$$

where the dual function $q$ is given by

$$q(\mu, p) = \sum_{j \in T} q_j(\mu_{W_j}, p_j) - \mu's$$

and $q_j$ is defined as

$$q_j(\mu_{W_j}, p_j) = \inf_{z_j \geq 0} \{f_j(z_j) + p_jz_j\}$$

$$+ \sum_{i \in W_j} \inf_{x_{ij} \geq 0} \{(\mu_i - c_{ij}p_j)x_{ij}\}.$$
Following [1], we say for any positive scalar $\epsilon$, a triple $(x, z, p)$ is $\epsilon$-CS iff $(x, p)$ and $(z, p)$ are $\epsilon$-CSarc and $\epsilon$-CSsink, respectively, where

- flow-price pair $(x, p)$ is $\epsilon$-CSarc if $x \geq 0$, $p \geq 0$, & $c_{ij}p_j \geq \max_{k \in T_i} c_{ik}p_k - \epsilon \forall \{i, j\} \in E$ with $x_{ij} > 0$,
- demand-price pair $(z, p)$ is $\epsilon$-CSsink if $z \geq 0$, $p \geq 0$, & $-f_j^+(z_j) \leq p_j \leq -f_j^-(z_j) \forall j \in T$

where $f_j^+(z_j)$ and $f_j^-(z_j)$ are the left and right derivatives of $f_j$, respectively.

The intuition behind the $\epsilon$-CS conditions is that a feasible flow-price pair is "approximately" primal and dual optimal if the $\epsilon$-CS conditions are satisfied as shown in this proposition which was proved in [8].

**Proposition I:** Let $(x^*, z^*, p^*)$ be a flow-demand-price triple satisfying $\epsilon$-CS such that $(x^*, z^*)$ is primal feasible, then

$$0 \leq f(x^*) - g(\mu^*, p^*) \leq \epsilon \sec' 1$$

where $\mu^*$ is defined as $\mu_i \triangleq \max_{j \in T_i} c_{ij}p_j \forall i \in W$.

### III. RAP Auction Algorithm

RAP Auction is a primal-dual method. In a typical iteration, prices have to be computed for given demands, and demands for given prices which are $\epsilon$-CSsink consistent. So we need a map $\Phi_j : Z_j \rightarrow P_j$ where $Z_j = \mathbb{R}^+$ is the demand space and $P_j = \Phi_j(Z_j)$ is the price space and its inverse $\Theta_j : P_j \rightarrow Z_j$. We use the following notation to describe the RAP Auction Algorithm:

- $p_{\min} \triangleq \min_{j \in T} \inf P_j \geq 0$,
- $\epsilon_{\min} \triangleq \epsilon / \max_{i,j \in E} c_{ij}$,
- $p(t), x(t), z(t)$ denote the price, flow, and demand vectors at the beginning of the $t$th iteration, respectively,
- $g_i$ denotes the surplus of source $i \in W$ : $g_i(t) \triangleq s_i - \sum_{j \in T_i} x_{ij}(t) \forall i \in W$.

At the start of the generic $t$th iteration we have flow-demand-price vector $(x(t), z(t), p(t))$ that satisfies $\epsilon$-CS, the flow-demand pair $(x(t), z(t))$ satisfies (1c) and $g(t) \geq 0$. A generic iteration consists of two phases: the bidding phase and the allocation phase, which we now describe.

#### Bidding Phase:

Select a source $i \in W$ with $g_i(t) > 0$; if no such source can be found, the algorithm terminates.

1) Compute the current value $v_{ij} = c_{ij}p_j(t)$ of each sink $j \in T_i$. Find a sink $j_1$ offering the best value

$$v_{\text{best}} = \max_{j \in T_i} v_{ij},$$

and a sink $j_2$ offering second best value

$$v_{\text{sec}} = \max_{j \in T_i \setminus j_1} v_{ij}.$$

2) Compute $i$’s bid price for $j_1$ as

$$b = \begin{cases} v_{\text{sec}} - \epsilon / c_{ijj}, & \#T_i \geq 2 \\ p_{\min} & \text{else} \end{cases}. \tag{2}$$

3) Compute $i$’s bid surplus $y = c_{ijj}g_i$.

4) Submit the bid $\{y, b\}$ to sink $j_1$.

When a sink, say $j$, receives a bid, it runs the allocation phase. For ease of exposition, we use the following representation for previously accepted and still valid bids at a given sink $j$:

- $B_j = \{b_1, \ldots, b_{\numb} \}$: List of accepted bid prices in decreasing order,
- $Y_j = \{y_1, \ldots, y_{\numb} \}$: List of flows received where $y_{i_k}$ is an alias for $c_{i_kj}x_{i_kj}(t)$ with $b_{i_k}$ as the corresponding bid price,
- We use $b_{i_0}$ as an alias for $p_j(t)$.

A source $i_k$’s bid $\{y_{i_k}, b_{i_k}\}$ is valid if $p_j(t) \geq b_{i_k}$ and $y_{i_k} \geq 0$. The state at sink $j$ at the beginning of the allocation phase is illustrated in Fig. 1. As flow starts getting accepted from $i$ during the current iteration, the demand $z_j$ increases and the demand-price pair slides to the right of ‘A’ along the blue curve. We call this mode of acceptance as absorption. Up to $\delta_0$ of flow can be absorbed before reaching ‘B’. This is called the demand margin between successive prices $p_j(t)$ and $b_{i_1}$. In general, we define demand margin, $\delta_k$, as

$$\delta_k := \Theta_j(b_{i_k+1}) - \Theta_j(b_{i_k}).$$

If $y > \delta_0$, then the demand-price pair will reach ‘B’. At ‘B’, flow from $i$ is accepted by reversing flow to $i_1$. We call this mode of flow acceptance as reversal. The price and demand don’t change during this mode. If $y \geq \delta_0 + y_{i_1}$, then there is completely reversible to $i_1$. In this case, we say reverse arc $(j, i_1)$ has saturated. Now between $b_{i_j}$ and $b_{i_2}$ up to $\delta_1$ can be absorbed before reversing $i_2$’s bid and so forth till either
Allocation Phase:

Select a sink \( j \in T \) which received a new bid \( \{ y, b \} \).
1) If sink \( j \) has previously accepted bid from \( i \)
   a) Update \( b_i = b \) and re-sort \( B_j \) and \( Y_j \), accordingly.
2) Else insert \( b_i = 0 \) in \( B_j \) and \( Y_j \), respectively, in sorted order and let \( n' \) be the index such that \( i_{n'} = i \).
3) If \( y \geq y_{\max} \) where \( y_{\max} = \sum_{k=1}^{n'} y_{ik} + \Phi_j(b) - z_j(t) \)
   Non-exhausting push:
   a) Reverse bids from \( \{ i_1, \dotsc, i_{n'-1} \} \) while accepting \( y_{\max} \) from \( i \),
   b) Set \( p_j(t+1) = b \) and \( z_j(t+1) = \sum_{k=n'}^{n} y_{ik} \).
4) Else
   Exhausting push: Determine sources \( \{ i_k : 1 \leq k \leq n'' \} \) whose bids will be completely reversed where \( n'' < n' \).
i.e., if \( k \) is reversed if \( k < n'' \) and \( \sum_{i=1}^{n} (y_{ii} + \delta_{i-1}) \leq y \).
   a) If \( y \leq \sum_{k=1}^{n''} y_{ik} + \delta_{i-1} + \delta_{n''} \)
      Allocation with complete reversals and absorptions:
      i) Reverse bids from \( \{ i_1, \dotsc, i_{n''} \} \) completely while accepting \( y \) from \( i \),
      ii) Set \( z_j(t+1) = \sum_{k=n''+1}^{n} y_{ik} \) and \( p_j(t+1) = \Phi_j(z_j(t+1)) \).
   b) else
      Allocation with at least one partial reversal:
      i) Reverse \( \{ i_1, \dotsc, i_{n''} \} \) bids completely and \( b_{i_{n''+1}} \) partially while accepting \( y \) from \( i \),
      ii) Set \( z_j(t+1) = \sum_{k=n''+1}^{n} y_{ik} \) and \( p_j(t+1) = \Phi_j(z_j(t+1)) \).

We showed that RAP terminates in a finite number of iteration with a near optimal solution.

Proposition 2: The RAP auction terminates after finite iterations with an \( \epsilon \)-CS satisfying flow-demand-price triple \((x^*, z^*, p^*)\) such that \((x^*, z^*)\) is primal feasible.

This was derived by first establishing the following lemmas:

Lemma 1: Every iteration preserves \( \epsilon \)-CS, (1c), & \( g \geq 0 \).

Lemma 2: After finite number of iteration, price drops by at least \( \epsilon_{\min} \).

IV. REVISITED RAP AUCTION

In this section, we illustrate why RAP Auction can’t have stronger complexity bounds and then propose our extension.

First we define necessary terminology. At any iteration, we can define the set \( A \) that contains arcs oriented in the direction of flow change. In particular, for each source \( i \in W \), \( A \) contains one forward arc \( (i, j) \in E \) such that if \( i \) were to bid, it would bid for \( j \), i.e.,
\[
j = \arg \max_{k \in T_i} c_{ik} b_k.
\]
and for each sink \( j \in T \), \( A \) contains a backward arc for each valid bid leveled with \( p_j \), i.e., if \( p_j = b_{ij} \), then \((j, i) \in A \).

Now \((W, T, A)\) defines the admisible graph \( G \). This \( G \) can be cyclic. Due to \( \epsilon \)-CS, all such cycles are flow generating, i.e., cycle gain \( \geq 1 \). As illustrated in Fig. 2, the complexity of resolving such cycles though finite can be arbitrarily large. This prevents RAP Auction from having stronger than finite termination.

We propose appending this step to the bidding phase which resolves such cycles.

Bidding Phase (contd.):

5) Update all the valid bids of \( i \)
\[
b_i = (v_{\text{best}} - \epsilon)/c_{ij} \quad \forall \ j \in T_i \setminus j_1 : b_i \in B_j.
\]

The bid prices determine the sink prices at which flows have to be reversed as so not to violate \( CS_{\text{sink}} \). In this new step, the bidding source revises the bid prices for all its valid bids, excluding the current bid as shown in Fig. 3. If its current best value, \( v_{\text{best}} \), has strictly reduced, then all its valid bid prices are strictly lowered.

![Fig. 2: Impact of cyclic admissible graph. Assume at \( t \) we have this graph. In this iteration source 1 bids for sink 1 increasing \( x_{11} \) by \( g_1 \) while reducing \( x_{21} \) by \( g_1 c_{11}/c_{21} \). Next 2 bids for 2 increasing \( x_{22} \) by \( g_1 c_{11}/c_{21} \) and reducing \( x_{12} \) by \( g_1 \gamma \) where \( \gamma = c_{22} c_{11}/c_{21} c_{12} \), the cycle gain. After \( k \) such circulations \( x_{21} \) and \( x_{12} \) are reduced by \( g_1 \gamma^{k-1} c_{11}/c_{21} \) and \( g_1 \gamma^k \), respectively. This sequence continues till one of the reverse arcs saturates making the graph acyclic. Since \( \gamma \geq 1 \), this happens in order \( O(1/g_1) \) iterations.](image)

![Fig. 3: Illustration of bid update step. Let \( t_1 < t_2 < t_3 \) be iterations in which \( i \) bids subsequently. In \( t_1 \), \( i \) bids for \( j_1 \) setting \( b_{ij_1} \) using (2). At \( t_2 \), if \( i \) bids for some sink other than \( j_1 \) and \( v_{sec}(t_1) > v_{\text{best}}(t_2) \), then \( b_{ij_1} \) is strictly lowered in the bid update step. This step again revises \( b_{ij_1} \) at \( t_4 \) if \( v_{\text{best}}(t_4) < v_{\text{best}}(t_2) \).](image)
Lemma 3: The number iterations between two successive price drops is bounded by $O(N^2)$.

Proof: An iteration in which the prices don’t change is called Non Price Reducing (NPR). Every NPR iteration is exhausting and flow is completely accepted by reversal. Let $\{1, 2, \ldots, \chi\}$ denote a sequence of successive iterations between two non zero price drops. In this sequence, let $W = \{i_1, \ldots, i_\eta\}$ and $\hat{T} = \{j_1, \ldots, j_\eta\}$ denote the sources and sinks making and receiving bids, respectively such that $i_1$ bids for $j_1, i_2$ bids for $j_2$. Since more than one sources can bid for the same sink, $\hat{T}$ may not contain all unique sinks. Let $\hat{G} = (W, \hat{T}, A)$ denote the corresponding admissible graph. If this graph is acyclic, then from Lemma 4, the number of NPR iterations is bounded by

$$\chi_{\text{acyclic}}(\eta) \leq \eta(\eta+1)/2.$$ 

However if there are cycles, for RAP Auction we only have $\chi < \infty$. With the new proposed bid update step, we obtain a polynomial bound as we now show.

For each sink $j \in \hat{T}$, let $t_j < 1$ denote the latest price reducing iteration, i.e., $p_j(t_j) - p_j(t_j+1) > 0$. Assume that set of sinks $\hat{T}$ is labeled such that $t_{j_1} \leq t_{j_2} \leq \cdots \leq t_{j_\eta}$. For each sink $j$, we define its push list as set of following sources whose flows can be reversed

$$L_j = \{i \in W | p_j = b_{ij}\}.$$ 

Now consider one such bid $b_{ij}$ where $i \in L_j$. Then this bid price hasn’t changed since $t_j + 1$. Due to the new bid update step, this is either because the source $i$ hasn’t bid again or its best value hasn’t changed since $t_j + 1$. So for each source $i \in W$, we can define

$$t_i' = \begin{cases} 0 & i \notin L_j \forall j \in \hat{T} \\ \min_{j \in \hat{T} | i \in L_j} t_j & \text{else} \end{cases}.$$

Then all the bid prices of $i$ haven’t changed since $t_i' + 1$. If $i$ were new to bid for a sink $j$ whose price has changed after $t_i' + 1$ ($t_i' \leq t_j$), it means that the best value for $i$ has strictly reduced. Accordingly, in the new step 5, the bid prices are strictly lowered and the corresponding reverse arcs are removed from $\hat{G}$. For example in Fig. 4, $i_2$’s bid prices for its allocations to $j_1$ and $j_\eta$ hasn’t changed since $t_1 + 1$ and when it bids again, these bid prices are strictly lowered removing $(j_1, i_2)$ and $(j_\eta, i_2)$ from $\hat{G}$. Such a source can’t bid again in this sequence as it is exhausted in the current iteration and its flow can’t be reversed without a drop in prices. We call such a source as Single Push Source (SPS).

Fig. 5 is the biadjacency matrix of Fig. 4. Because of the ordering of the sink nodes, all the SPSs correspond to rows with negative lower diagonal elements. From this, it is easy to make the following observations:

1) $\hat{G}$ is acyclic if it has no SPSs.
2) Every cycle in $\hat{G}$ has at least one SPS.

When an SPS bids, then all the non-diagonal entries in its row are set to zero. So there can be at most one circulation in any cycle. Without the SPSs participating, the admission graph is essentially acyclic and after all the SPSs have bid once, $\hat{G}$ becomes acyclic, i.e.

$$\chi \leq \chi_{\text{acyclic}}(\eta - \eta_{\text{SPS}}) + \eta_{\text{SPS}} + \chi_{\text{acyclic}}(\eta - \eta_{\text{SPS}}) \leq (\eta + 1)^2$$

where $\eta_{\text{SPS}}$ is the number of SPSs. So finally we have $\chi$ is $O(N^2)$.

Lemma 4: Number of iterations in an acyclic admissible graph with $\eta$ sources is bounded by $\eta(\eta+1)/2$.

Proof: Let $\{1, 2, \ldots, \chi_{\text{acyclic}}\}$ denote a sequence of successive NPR iterations and $\hat{G} = (W, \hat{T}, A)$ the corresponding admissible graph. If this admissible graph consists of a single path $(i_1, j_1, i_2, j_2, \ldots, i_\eta, j_\eta)$, then the bound on iterations can be calculated using a directed tree as shown in Fig 6. If the admissible graph has multiple paths, then we have to form directed trees for each path and merge these trees to form a polytree as illustrated in Fig. 7. Let $\alpha(i)$ denote the number of ancestors of node $i$ and $\alpha = \{\alpha(i) | i = 1, 2, \ldots, \eta\}$. Relabel the nodes such that $\alpha$ is in descending order. We have this inequality

$$\alpha(i) \leq \eta - i$$

from the fact that in a directed acyclic graph if $i$ is the ancestor of $j$, then $j$ can’t be the ancestor of $i$. $\chi_{\text{acyclic}}$ is bounded by

$$\sum_{i=1}^{\eta} \{\alpha(i) + 1\} \leq \eta + \sum_{i=1}^{\eta} (\eta - i) = \eta(\eta + 1)/2.$$
This happens if at least two flow reversals (one complete and one at least partial) take place during a given iteration. Since there are only \( m_j \) bids which can be reversed, we have

\[
\#\Delta_j^i \leq \max\{m_j - 1, 0\}
\]

and the positive variation is also bounded as an arc once saturated remains saturated.

\[
\frac{1}{\epsilon} \leq \min_{\Delta} \sum_{t \in \Delta} \nabla n(t) \leq m_j - 1.
\]

Based on the what happens during the allocation phase at a sink node, an iteration can be classified as:

1) NPR: Flow is entirely accepted by reversal without any price drop. So

\[
0 \leq \nabla n(t) \quad \forall t \in \Delta_{NPR}
\]

where \( \Delta_{NPR} \subset \Delta \) is the subsequence of NPR iterations.

2) Price Reducing (PR): These are of two types:

a) Price Reducing with bid Reversal (PRR): These are strictly price reducing with some flow reversal.

\[
0 \leq \nabla n(t) \quad \forall n \in \Delta_{PRR}.
\]

where \( \Delta_{PRR} \subset \Delta \) is the subsequence of PRR iterations. This can only happen in two ways, a price drop followed by reversal (a bid starts getting reversed for the first time) or vice-versa (a bid is completely reversed). So each reversible bid can result at most 2 PRR iterations and

\[
\#\Delta_{jPRR}^i \leq 2m_j.
\]

b) Price Reducing Without bid Reversal (PRWR): In such iterations, there is no reversal. So

\[
\nabla n(t) = -1 \quad \forall t \in \Delta_{PRWR}
\]

where \( \Delta_{PRWR} \subset \Delta \) is the subsequence of PRWR iterations.

From (5), (7), (8), and (10),

\[
\Delta_{jNPR}^i \cup \Delta_{jPRR}^i = \Delta_{jPRWR}^i \cup \{t \in \Delta^j : \nabla n(t) = 0\}.
\]

So using (6)

\[
\frac{1}{\epsilon} \leq \min_{\Delta} \sum_{t \in \Delta_{jNPR}^i \cup \Delta_{jPRR}^i} \nabla n(t) \leq m_j - 1.
\]

Now we derive a bound for PRWRs using (4) and the fact that \( n(\chi) > 0 \),

\[
-n(1) \leq n(\chi) - n(1) = \sum_{t \in \Delta} \nabla n(t) = \sum_{t \in \Delta_{PRR} \cup \Delta_{PRWR}} \nabla n(t) \leq \sum_{j \in T} (m_j - 1) - \#\Delta_{PRWR}^i.
\]
TABLE I: Solution times (in seconds) for RAP Auction, Revised RAP Auction, CVX, and RWOA for solving randomly generated search problems. $P_{edge}$ is the probability of an edge between a source and sink.

<table>
<thead>
<tr>
<th>N</th>
<th>M</th>
<th>$P_{edge}$</th>
<th>Supply range</th>
<th>Gain range</th>
<th>$\epsilon$</th>
<th>RAP</th>
<th>Revised RAP</th>
<th>CVX</th>
<th>RWOA</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>0.4</td>
<td>1−7</td>
<td>0−2.3026</td>
<td>0.01</td>
<td>0.00171</td>
<td>0.00219</td>
<td>0.15647</td>
<td>0.00130</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0.4</td>
<td>1−7</td>
<td>0−2.3026</td>
<td>0.001</td>
<td>0.000602</td>
<td>0.00834</td>
<td>0.16417</td>
<td>0.00190</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0.4</td>
<td>1−7</td>
<td>0−2.3026</td>
<td>0.0001</td>
<td>0.04508</td>
<td>0.06979</td>
<td>0.17767</td>
<td>0.00208</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>0.4</td>
<td>1−7</td>
<td>0−2.3026</td>
<td>0.001</td>
<td>0.04650</td>
<td>0.07731</td>
<td>0.98179</td>
<td>0.00372</td>
</tr>
<tr>
<td>20</td>
<td>5</td>
<td>0.4</td>
<td>1−7</td>
<td>0−2.3026</td>
<td>0.001</td>
<td>0.04004</td>
<td>0.06240</td>
<td>0.15337</td>
<td>0.00619</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>0.25</td>
<td>1−10</td>
<td>0−2.3026</td>
<td>0.001</td>
<td>0.01360</td>
<td>0.01949</td>
<td>0.17066</td>
<td>0.00480</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>0.5</td>
<td>1−10</td>
<td>0−2.3026</td>
<td>0.001</td>
<td>0.01387</td>
<td>0.02111</td>
<td>0.17749</td>
<td>0.00497</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>0.75</td>
<td>1−10</td>
<td>0−2.3026</td>
<td>0.001</td>
<td>0.02174</td>
<td>0.02859</td>
<td>0.18594</td>
<td>0.00370</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>1</td>
<td>1−10</td>
<td>0−2.3026</td>
<td>0.001</td>
<td>0.02960</td>
<td>0.03642</td>
<td>0.16627</td>
<td>0.00364</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>0.4</td>
<td>1−10</td>
<td>0−2.3026</td>
<td>0.0001</td>
<td>0.05600</td>
<td>0.07847</td>
<td>0.25108</td>
<td>0.01069</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>0.4</td>
<td>1−10</td>
<td>1</td>
<td>0.0001</td>
<td>1.26861</td>
<td>1.41890</td>
<td>-</td>
<td>0.00202</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>0.4</td>
<td>1−10</td>
<td>0−2.3026</td>
<td>0.001</td>
<td>0.01272</td>
<td>0.01752</td>
<td>0.34665</td>
<td>0.00870</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>0.4</td>
<td>1−10</td>
<td>0−2.3026</td>
<td>0.001</td>
<td>0.01345</td>
<td>0.01715</td>
<td>0.23284</td>
<td>0.00178</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>0.4</td>
<td>1−100</td>
<td>0−2.3026</td>
<td>0.001</td>
<td>0.00825</td>
<td>0.01030</td>
<td>0.23256</td>
<td>0.00510</td>
</tr>
</tbody>
</table>

Last inequality follows from (10) and (11). So

$$\#\Delta_{PRWR} \leq \sum_{j \in T} (m_j - 1) + n(1). \quad (12)$$

From (9) and (12) we have for all the PR iterations,

$$\#\Delta_{PR} \leq \#\left(\Delta_{PRR} \bigcup \Delta_{PRWR}\right) \leq \sum_{j \in T} (3m_j - 1) + n(1) \leq 3MN + N - 4M.$$

From lemma 3, we have a bound on the number of successive NPRs iterations. So

$$\chi \leq \#\Delta_{PR} \times \text{max number of successive NPRs} \leq (3MN + N - 4M)O(N^2) = O(MN^3).$$

**Proposition 3:** The revised RAP auction algorithm terminates in $O(N^3M^2p_{max}c_{max}/\epsilon)$ where $p_{max} \triangleq \max_{j \in T} P_j(1)$ and $c_{max} \triangleq \max_{i,j \in E} c_{ij}$.

**Proof:** The sequence $P_j(t)$ is a non increasing sequence and lower and upper bounded by $p_{min} \geq 0$ and $p_{max}$ respectively. Every $O(MN^3)$ iterations the price of at least one sink node drops by $\epsilon_{min}$. So maximum iterations is $O(N^3M^2p_{max}/\epsilon_{min})$.

The proofs for lemmas 3, 4 and 5 assume strict convexity. However this can be relaxed similar to how we generalized lemma 2 in [8].

V. EXPERIMENTS

In [1], we benchmarked the original RAP Auction for randomly generated instances of search theory problems [4] where the cost functions are exponentials. These tests were designed to study its performance relative to CVX [10], a MATLAB based generic convex solver, and Resource-Wise Optimization Algorithm (RWOA) [11] and the dependence of this performance on network topology, arc gains, supply and $\epsilon$. In general, CVX was an order of magnitude slower than RAP Auction which was slower than RWOA. RWOA benefits from the fact that it has been customized for exponential cost functions.

Continuing along this line, we now report the performance for Revised RAP Auction. We have also improved the performance of original RAP Auction by efficient sorting. Table I lists the solutions times on MacBook Pro 4.1 running OS X 10.6.4 operating system. The performance of revised RAP Auction is only slightly more than the original RAP Auction.

VI. CONCLUSIONS

We have successfully proposed an extension to RAP Auction which enables it to have provably pseudo-polynomial complexity as opposed to finite termination. This extension takes the form of a single additional step to the bidding phase with negligible computation.

REFERENCES