Optimal Relay Assignment for Cooperative Communications

Yi Shi† Sushant Sharma† Y. Thomas Hou†
†Department of Electrical and Computer Engineering
†Department of Computer Science
Virginia Tech, Blacksburg, VA, USA
{yshi,sushant,thou}@vt.edu

Sastry Kompella
Information Technology Division
Naval Research Laboratory
Washington DC, USA
kompella@itd.nrl.navy.mil

ABSTRACT
Recently, cooperative communications, in the form of keeping each node with a single antenna and having a node exploit a relay node’s antenna, is shown to be a promising approach to achieve spatial diversity. Under this communication paradigm, the choice of relay node plays a significant role in the overall system performance. In this paper, we study the relay node assignment problem in a network environment, where multiple source-destination pairs compete for the same pool of relay nodes in the network. The main contribution of this paper is the development of a polynomial time algorithm to solve this problem. A key idea in this algorithm is a “linear marking” mechanism, which is able to offer a linear complexity for each iteration. We give a formal proof of optimality for this algorithm. We also show several attractive properties associated with this algorithm.

Categories and Subject Descriptors
C.2.1 [Network Architecture and Design]: Wireless communications

General Terms
Algorithm

Keywords
Cooperative Communications, Wireless Networks, Network Capacity

1. INTRODUCTION
Spatial diversity, in the form of employing multiple transceiver antennas, is shown to be very effective in coping fading in wireless channel. However, equipping a wireless node with multiple antennas may not be practical, as the footprint of multiple antennas may not fit on a wireless node (particularly handheld wireless device). To achieve spatial diversity without requiring multiple transceiver antennas on the same node, the so-called cooperative communications has been introduced [6, 8]. Under cooperative communications, each node is equipped with only a single transceiver and spatial diversity is achieved by exploiting the antenna on another (cooperative) node in the network.

There are two categories of cooperative communications, namely, amplify-and-forward (AF) and decode-and-forward (DF) [6]. Under AF, the cooperative relay node performs a linear operation on the signal received from the information source before forwarding it to the destination node. Under DF, the cooperative relay node decodes the received signal, and re-encodes it before forwarding it to the destination node. Regardless of AF or DF, the choice of a relay node plays a significant role in the final performance of cooperative communications [2, 3, 10]. As we shall see in Section 2, an improperly chosen relay node may offer a smaller capacity for a source-destination pair than that under direct transmissions.

In this paper, we study relay node assignment problem in a network environment. Specifically, we consider an ad hoc network environment where there are multiple active source-destination pairs and the remaining nodes can be exploited as relay nodes. We want to find out how to optimally assign relay nodes to the source and destination pairs so as to maximize the minimum capacity among all pairs. Although solution to this problem can be found via exhaustive search (among all possible relay node assignments), the complexity of this approach is exponential. Our goal in this paper is to find a polynomial-time complexity algorithm to this problem.

1.1 Related Work
The cooperative communication paradigm can trace back to the pioneering works done by van der Meulen [9] and Cover and El Gamal [4]. The readers are also referred to [1, 5] for some recent work on this subject. In this section, we focus our attention on related work for the relay node assignment problem.

Although it is possible for a source-destination pair to employ multiple relay nodes for cooperative communications, the benefit of such approach appears limited, as shown in a recent work by Zhao et al. [10]. In this work, Zhao et al. showed that for a source-destination pair, in the presence of multiple relay nodes, it is sufficient to choose the “best” relay to achieve full diversity order than to have multiple relay nodes participate. This result is interesting, as it paves the way for research on assigning no more than one relay node to a source-destination pair, which is the setting that we will adopt in this paper.

There has been much effort on selecting an optimal relay node (among a set of relay nodes) for a single source-destination pair (see, e.g., [2]). However, these schemes are limited to a single source-destination pair and cannot be easily extended to a network environment where there are multiple source-destination pairs competing for the same pool of relay nodes, which is the focus of this paper.

In [7], Ng et al. studied a utility maximization problem for the joint optimization of relay node selection, cooperative communica-
tions, and resource allocation in a cellular network. A key assumption in the solution to the optimization problem is infinite number of channels in the network (so that the duality gap of the optimization problem is zero). But this assumption may not hold in practice. Also, the complexity of the proposed solution is not polynomial.

In [3], Cai et al. studied relay node selection and power allocation for AF based wireless relay network. A simple network consisting of single source-destination pair was first studied. Then, the authors considered multiple source-destination pairs and proposed a semi-distributed algorithm on relay node selection. This algorithm is heuristic in nature and there is no performance guarantee (in terms of optimality).

1.2 Main Contribution of This Paper

In this paper, we study the optimal relay node assignment problem in a network setting. Specifically, we consider how to assign a set of relay nodes to a set of source-destination pairs so as to maximize the minimum capacity among the pairs. The main contributions of this paper are the following.

- We develop an algorithm, called Optimal Relay Assignment (ORA) algorithm, to solve the relay node assignment problem. A key idea in ORA is a “linear marking” mechanism, which is able to offer a linear complexity at each iteration. Due to this mechanism, ORA is able to achieve polynomial time complexity.

- We offer a formal proof of optimality for the ORA algorithm. The proof is based on contradiction and hinges on a clever recursive trace-back of source nodes and relay nodes in the solution by ORA and another hypothesized better solution.

- We show a number of nice properties associated with ORA. These include: (i) the algorithm works regardless whether the number of relay nodes in the network is more than or less than the number of source-destination pairs; (ii) the final capacity for each source-destination pair is guaranteed to be no less than that under direct transmissions; (iii) the algorithm is able to find the optimal objective regardless of initial relay node assignment.

1.3 Paper Organization

In Section 2, we give a brief overview of cooperative communications, which includes capacity calculation for both AF and DF. In Section 3, we present mathematical model for the relay node assignment problem in a network environment. Section 4 presents our Optimal Relay Assignment (ORA) algorithm. In Section 5, we give a proof of ORA’s optimality. Section 6 presents some numerical results to demonstrate the capabilities of the ORA algorithm. Section 7 concludes this paper.

2. COOPERATIVE COMMUNICATIONS: A PRIMER

The essence of cooperative communications is best explained by a three-node example in Fig. 1. In this figure, node s is the source node, node d is the destination node, and node r is a relay node. Transmission from s to d is done on a frame-by-frame basis. Within a frame, there are two time slots. In the first time slot, source node s makes a transmission to destination node d. This transmission is also overheard by relay node r, due to the broadcast nature of wireless communications. In the second time slot, node r forwards the data received in the first time slot to d. Note that such a two-slot structure is necessary for cooperative communications due to the half-duplex nature of most wireless transceivers.

Figure 1: A three-node schematic for cooperative communications.

In this section, we give expressions for capacity under cooperative communications and direct transmissions (i.e., no cooperation). For cooperative communications, we consider both the so-called amplify-and-forward (AF) and decode-and-forward (DF) modes [6].

Amplify-and-Forward (AF) Under this mode, let \( h_{sd}, h_{sr}, h_{rd} \) capture the effect of path-loss, shadowing, and fading between nodes s and d, s and r, and r and d, respectively. Also denote \( z_d \) and \( z_r \) the zero-mean background noise at nodes d and r, with variance \( \sigma_d^2 \) and \( \sigma_r^2 \), respectively.

Denote \( x_s \) the signal transmitted by source node s in the first time slot. Then the received signal at destination node d, \( y_{sd} \), can be expressed as

\[ y_{sd} = h_{sd}x_s + z_d, \]

and the received signal at the relay node r, \( y_{sr} \), is

\[ y_{sr} = h_{sr}x_s + z_r. \]

In the second time slot, relay node r transmits to destination node d. The received signal at d, \( y_{rd} \), can be expressed as

\[ y_{rd} = h_{rd}x_r + y_{sr} + z_d, \]

where \( \alpha_r \) is the amplifying factor at relay node r and \( y_{sr} \) is given in (2). Thus, we have

\[ y_{rd} = h_{rd}\alpha_r \cdot (h_{sr}x_s + z_r) + z_d. \]

The amplifying factor \( \alpha_r \) at relay node r should satisfy power constraint \( \alpha_r^2|y_{sr}|^2P_r + \sigma_r^2 = P_r \), where \( P_s \) and \( P_r \) are the transmission powers at nodes s and r, respectively. So, \( \alpha_r \) is given by

\[ \alpha_r^2 = \frac{P_r}{|y_{sr}|^2P_s + \sigma_r^2}. \]

We can re-write (1), (2) and (3) into the following compact matrix form

\[ Y = Hx_s + BZ, \]

where

\[ Y = \begin{bmatrix} y_{sd} \\ y_{rd} \end{bmatrix}, \quad H = \begin{bmatrix} h_{sd} & h_{rd} \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 1 & 0 \\ \alpha_r h_{rd} & 0 & 1 \end{bmatrix}, \quad Z = \begin{bmatrix} z_r \\ z_d \end{bmatrix}. \]

\[ \text{(4)} \]

It has been shown in [6] that, the above channel, which combines both the direct path (s to d) and relay path (s to r to d), can be modeled as a one-input, two-output complex Gaussian noise channel. The capacity \( C_{AF}(s, r, d) \) from s to d can be given by

\[ C_{AF}(s, r, d) = \frac{W}{2} \log_2|\det(I + (P_r HH^H)(B E[ZZ^H B^H])^{-1})|, \]

\[ \text{(5)} \]

where \( W \) is the bandwidth, \( \det(\cdot) \) is the determinant function, I is the identity matrix, the superscript “*” represents the complex conjugate transposition, and \( E[\cdot] \) is the expectation function.
After putting (4) into (5) and performing algebraic manipulations, we have
\[ C_{\text{AF}}(s, r, d) = \frac{W}{2} \log_2 \left( 1 + \frac{P_s|h_{sd}|^2}{\sigma^2} \right) - \frac{P_r|h_{sr}|^2}{P_s|h_{sd}|^2} - \frac{P_r|h_{sr}|^2}{P_s|h_{sd}|^2} \].
Denote SNR_{sd} = \frac{P_s|h_{sd}|^2}{\sigma^2},
SNR_{sr} = \frac{P_r|h_{sr}|^2}{\sigma^2},
SNR_{rd} = \frac{P_r|h_{rd}|^2}{\sigma^2}. We have
\[ C_{\text{AF}}(s, r, d) = W \cdot I_{\text{AF}}(\text{SNR}_{sd}, \text{SNR}_{sr}, \text{SNR}_{rd}) \]
where \( I_{\text{AF}}(\text{SNR}_{sd}, \text{SNR}_{sr}, \text{SNR}_{rd}) = \frac{1}{2} \min \{ \log_2 (1 + \text{SNR}_{sr}), \log_2 (1 + \text{SNR}_{sd} + \text{SNR}_{rd}) \} \).

**Decode-and-Forward (DF)** Under this mode, relay node \( r \) decodes and estimates the received signal from source node \( s \) in the first time slot, then transmits the estimated data to destination node \( d \) in the second time slot. The capacity for DF under the two time-slot structure is given by [6]
\[ C_{\text{DF}}(s, r, d) = W \cdot I_{\text{DF}}(\text{SNR}_{sd}, \text{SNR}_{sr}, \text{SNR}_{rd}) \]
where \( I_{\text{DF}}(\text{SNR}_{sd}, \text{SNR}_{sr}, \text{SNR}_{rd}) = \frac{1}{2} \min \{ \log_2 (1 + \text{SNR}_{sr}), \log_2 (1 + \text{SNR}_{sd} + \text{SNR}_{rd}) \} \).

Note that \( I_{\text{AF}}(\cdot) \) and \( I_{\text{DF}}(\cdot) \) are increasing functions of \( P_s \) and \( P_r \), respectively. This suggests that, in order to achieve the maximum capacity under either mode, both source node and relay node should transmit at the maximum power. In this paper, we let \( P_s = P_r = P \).

**Direct Transmissions** When cooperative communications (i.e., relay node) are not used, source node \( s \) transmits to destination node \( d \) in both time slots. The capacity from source node \( s \) to destination node \( d \) is
\[ C_D(s, d) = W \log_2 (1 + \text{SNR}_{sd}) \].

Based on the above results, we have two observations. First, comparing \( C_{\text{AF}} \) (or \( C_{\text{DF}} \)) to \( C_D \), it is hard to say cooperative communications is always better than direct transmissions. In fact, a poor choice of relay node could make the capacity under cooperative communications be smaller than direct transmissions. This fact underlines the significance of relay node selection in cooperative communications. Second, although AF and DF are different mechanisms, the capacities for both of them have the same form, i.e., a function of SNR_{sd}, SNR_{sr}, and SNR_{rd}. Therefore, a relay node assignment algorithm designed for AF can be easily extended for DF. In this paper, we develop a relay node assignment algorithm for AF, which can also be used should DF be employed.

### 3. THE RELAY NODE ASSIGNMENT PROBLEM

Based on the background in the last section, we consider relay node assignment problem in a network setting. There are \( N \) nodes in an ad hoc network, with each node being either a source node, a destination node, or a potential relay node (see Fig. 2). In order to avoid interference, we assume that orthogonal channels are available in the network (e.g., using OFDMA), which is used for cooperative communications [6]. The path loss between nodes \( u \) and \( v \) is captured in \( h_{uv} \) and is given a priori. The discussion of channel measurement techniques is beyond the scope of this paper.

Denote \( \mathcal{N}_s = \{ s_1, s_2, \ldots, s_N \} \) the set of source nodes, \( \mathcal{N}_d = \{ d_1, d_2, \ldots, d_N \} \) the set of destination nodes, and \( \mathcal{N}_r = \{ r_1, r_2, \ldots, r_N \} \) the set of relays (see Fig. 2). We consider unicast where every source node \( s_i \) is paired with a destination node \( d_i \), i.e., \( N_d = N_s \). Each node is equipped with a single transceiver and can transmit/receive within one channel at a time. Further, we assume that each node can only serve a unique role of source, destination, or relay. That is, \( N_r + 2N_s = N \).

Note that a source node may not always get a relay node. This is because there may not be sufficient number of relay nodes in the network (e.g., \( N_r < N_s \)). Even if there are enough relay nodes, a sender may still not use a relay node if it leads to a smaller capacity than direct transmissions (see discussion at the end of Section 2).

We now discuss the objective function of our problem. Although different objectives can be used, a widely-used objective for cooperative communications is capacity. For the multi-pair network environment considered in this paper (see Fig. 2), each source-destination pair will have a different capacity after we apply a relay node assignment algorithm. So a plausible objective is to maximize the minimum capacity among all the source-destination pairs.

More formally, denote \( \mathcal{R}(s_i) \) the relay node assigned to \( s_i \). For both AF and DF, its capacity can be written as (see Section 2)
\[ W R_{\text{AF}}(\text{SNR}_{s_i,d_i}, \text{SNR}_{s_i,r_i}, \mathcal{R}(s_i), \text{SNR}_{\mathcal{R}(s_i),d_i}) \]
with \( R_{\text{DF}}(\cdot) = I_{\text{DF}}(\cdot) \) for AF and \( R_{\text{AF}}(\cdot) = I_{\text{AF}}(\cdot) \) for DF. In the case that \( s_i \) does not use a relay, we denote \( \mathcal{R}(s_i) = \emptyset \) and the capacity is the direct transmission capacity, i.e.,
\[ C_R(s_i, \emptyset) = C_D(s_i, d_i) \].

Combining both cases, we have
\[ C_R(s_i, \mathcal{R}(s_i)) = \begin{cases} W R_{\text{DF}}(\text{SNR}_{s_i,d_i}, \text{SNR}_{s_i,r_i}, \mathcal{R}(s_i), \text{SNR}_{\mathcal{R}(s_i),d_i}) & \text{if } \mathcal{R}(s_i) \neq \emptyset, \\ W \log_2 (1 + \text{SNR}_{s_i,d_i}) & \text{if } \mathcal{R}(s_i) = \emptyset. \end{cases} \] (6)

Note that we do not list \( d_i \) in function \( C_R(s_i, \mathcal{R}(s_i)) \) since for each source node \( s_i \), the corresponding destination node \( d_i \) is deterministic.

Denote \( C_{\text{min}} \) the minimum capacity among all source nodes. That is,
\[ C_{\text{min}} = \min \{ C_R(s_i, \mathcal{R}(s_i)) : s_i \in \mathcal{N}_s \}. \]

Our objective is to find an optimal relay node assignment for all the source-destination pairs such that \( C_{\text{min}} \) is maximized.

### 4. AN OPTIMAL RELAY ASSIGNMENT ALGORITHM

We can formulate the relay node assignment problem as an integer linear program. It is important to note here that an integer linear programming problem is NP-hard in general, i.e., there does not exist a general polynomial-time solution procedure to solve every integer linear program. However, as we show in this paper, we can exploit problem specific properties, and design a polynomial-time solution for our specific problem. The main contribution of this paper is a polynomial-time algorithm to the relay node assignment problem, which we will present in this section.
4.1 Basic Idea

The algorithm we will present is called Optimal Relay Assignment (ORA) algorithm. Figure 3 shows the flow chart of ORA algorithm.

Initially, ORA algorithm starts with a random feasible relay node assignment. By feasible, we mean that each source-destination pair can be assigned at most one relay node and that a relay node can be assigned only once. Such initial feasible assignment is easy to construct, e.g., direct transmission between each source-destination pair (without the use of a relay) is a special case of feasible assignment.

Starting with this initial assignment, ORA adjusts the assignment during each iteration, with the goal of increasing the objective function $C_{\text{min}}$. Specifically, during each iteration, ORA identifies the source node that corresponds to $C_{\text{min}}$. Then, ORA helps this source node to search a better relay such that this “bottleneck” capacity can be increased. In the case that the selected relay is already assigned to another source node, further relay node adjustment on that source node is necessary (so that its current relay can be released). Such adjustment may have a chain effect on a number of source nodes in the network. It is important that for any adjustment on a relay node, the affected source node should always maintain a capacity larger than $C_{\text{min}}$. There are only two outcomes from such search in an iteration: (i) a better assignment is found, in which case, ORA moves on to the next iteration; or (ii) a better assignment cannot be found, in which case, ORA terminates.

There are two key technical challenges we aim to address in the design. First, for any non-optimal solution, the algorithm should be able to find a better solution. As a result, upon termination, the final assignment is optimal. Second, its running time must be polynomial. We will show that ORA addresses both problems successfully. Specifically, we show the complexity of ORA algorithm is polynomial in Section 4.4. We will also give a correctness proof of its optimality upon termination in Section 5.

4.2 Algorithm Details

In the beginning, ORA algorithm performs a “preprocessing” step. In this step, for each source-destination pair, the source node $s_i$ considers each relay node $r_j$ in the network and computes the corresponding capacity $C_R(s_i, r_j)$ by (6). Each source node $s_i$ also computes the capacity $C_R(s_i, \emptyset)$ by (6) under direct transmissions (i.e., without the use of a relay node). After these computations, each source node $s_i$ can identify those relay nodes that can offer an increase of its capacity compared to direct transmissions, i.e., those relays with $C_R(s_i, r_j) > C_R(s_i, \emptyset)$. Obviously, it only makes sense to consider these relays for cooperative communications. In the case that no relay can offer any increase of capacity compared to direct transmissions, we will just employ direct transmissions for these source nodes.

After the preprocessing step, we enter the initial assignment step. The objective of this step is to obtain an initial feasible solution for ORA algorithm so that it can start its iteration. In the preprocessing step, we have already identified, for each source node, the list of relay nodes that can increase capacity compared to direct transmissions. We can randomly assign a relay node from this list to a source node. Note that once a relay node is assigned to a source node, it cannot be assigned again to another source node. Thus, if the selected relay node is already assigned to another source node, then this source node will simply employ direct transmissions, i.e., without the use of a relay. Upon the completion of this assignment, each source node will have a capacity no less than that under direct transmissions.

The next step in ORA algorithm is finding a better assignment, which represents an iteration process. This is the key step in ORA algorithm. The detail of this step is shown in the right hand side of Fig. 3. As a starting point of this step, ORA algorithm identifies the smallest capacity $C_{\text{min}}$ among all sources. ORA algorithm aims to increase this minimum capacity for the corresponding source node, while keeping all other source nodes to have their capacities stay above $C_{\text{min}}$. Without loss of generality, we use Fig. 4 to illustrate a search process.
Suppose ORA identifies that $s_1$ has the smallest capacity $C_{\text{min}}$ under the current assignment (with relay node $r_1$). Then $s_1$ examines other relays with a capacity larger than $C_{\text{min}}$. If it cannot find such a relay, then no better solution is found and the ORA algorithm is completed.

Otherwise, i.e., there are better relays, we consider these relays in the non-increasing order in terms of achieved capacity (should it be assigned to $s_1$). That is, we try the relay that can offer the maximum possible increase in capacity first.

Suppose that source node $s_1$ considers relay node $r_2$. If this relay node is not yet assigned to any other source node, then $r_2$ can be immediately assigned to $s_1$. In this simple case, we find a better solution and the current iteration is completed.

Otherwise, i.e., $r_2$ is already assigned to a source node, say $s_2$, we mark $r_2$ to indicate that $r_2$ is "under consideration" and check whether $r_2$ can be released by $s_2$.

To release $r_2$, source node $s_2$ needs to find another relay (or use direct transmissions) while making sure that such new assignment still makes its capacity larger than $C_{\text{min}}$. This process is the same as what we have done at $s_1$, with the only (but important) difference that $s_2$ will not consider a relay that is already "marked", as that relay node has already been considered by a source node encountered earlier in the search process of this iteration.

Suppose that source node $s_2$ now considers relay $r_3$. If this relay node is not yet assigned to any source node, then $r_3$ can be assigned to $s_2$; $r_2$ can be assigned to $s_1$; and the current iteration is completed. If the relay being considered is $r_1$ (or $\emptyset$), then a better solution, where $r_1$ (or $\emptyset$) is assigned to $s_2$ and $r_3$ is assigned to $s_1$, is found and the current iteration is completed. Otherwise, we mark $r_3$ and check further to see whether $r_3$ can be released by its corresponding source node, say $s_3$.

Suppose that $s_3$ cannot find any "unmarked" relay that offers a capacity larger than $C_{\text{min}}$ and its capacity under direct transmissions is not larger than $C_{\text{min}}$. Then $s_2$ cannot use $r_3$ as its relay.

If any "unmarked" relay that has a capacity larger than $C_{\text{min}}$ cannot be assigned to $s_2$, then $s_1$ cannot use $r_2$ and will move on to consider the next relay on its non-increasing capacity list, say $r_4$.

The search continues, with relay nodes being marked along the way, until a better solution is found or no better solution can be found. For example, in Fig. 4, $s_6$ finds a new relay $r_7$. As a result, we have a new assignment, where $r_7$ is assigned to $s_6$; $r_6$ is assigned to $s_4$; and $r_4$ is assigned to $s_1$.

Note that the "mark" on a relay node will not be cleared throughout the search process in the same iteration. We call this "linear marking" mechanism. These marks will only be cleared when the current iteration terminates and before the start of the next iteration. The pseudo-code of ORA algorithm is shown in Fig. 5.

It should be clear that ORA works regardless of whether $N_r \geq N_s$ or $N_r < N_s$. For the latter case, i.e., the number of relay nodes in the network is less than the number of source nodes, it is only necessary to consider relay node assignment for a reduced subset of $N_s$ source nodes, where the capacity of each source in this subset under direct transmissions is less than the capacity of those $(N_s - N_r)$ source nodes not in this subset. As a result, in the case of $N_s > N_r$, ORA will run even more efficiently due to a smaller problem size.

### 4.3 Caveat on the Marking Mechanism

We now re-visit the marking mechanism in ORA algorithm. Although different marking mechanisms may be designed to achieve the optimal objective, the algorithm complexity under different marking mechanisms may differ significantly. In this section, we first present a marking mechanism, which appears to be a natural approach but leads to exponential complexity for each iteration. Then we re-examine our marking mechanism and show that it leads to a linear complexity for each iteration.

A natural approach is to perform both marking and unmarking within an iteration. This approach is best explained with an example. Again, let’s look at Fig. 4. Source node $s_1$ first considers $r_2$. Since $r_2$ will be used by $s_1$ in the new solution, $r_2$ is marked. Source node $s_2$ considers $r_3$, which is already assigned to $s_3$. Since $s_3$ cannot release $r_3$ without reducing its capacity below the current $C_{\text{min}}$, this branch of search is futile and $s_1$ now considers a different relay node $r_4$. Since $r_4$ is currently assigned to $s_4$, we try to find a new relay for $s_4$. Now the question is: shall we remove those
marks on $r_2$ and $r_3$ that we put earlier in the process within this iteration? Under this natural approach, $r_2$ and $r_3$ should be un-marked so that they can be considered as candidate relay nodes for $s_4$ in its search. Similarly, when we try to find a relay for $s_6$, relay nodes $r_2$, $r_3$, and $r_5$ should be un-marked so that they can be considered as candidate relay nodes for $s_6$, in addition to $r_7$. In summary, under this approach, each relay node that has been considered earlier in the search process by a source node should be un-marked when this source node considers the next relay node, so that this relay node can remain in the pool of candidate relay nodes to be considered in the search process. It is not hard to show that such marking/unmarking mechanism considers all possible assignments and can guarantee to find a better solution (if it exists). However, the complexity of such approach is exponential for each iteration.

In contrast, under ORA algorithm, there is no unmarking mechanism within an iteration. That is, relay nodes that are marked earlier in the search process by some source nodes will remain marked. For example, in Fig. 4, when $s_2$ tries to find another relay, it will no longer consider $r_2$ and $r_3$ that have been marked earlier. Similarly, when $s_6$ tries to find another relay, it will not consider $r_2$, $r_3$, $r_4$, and $r_5$. As a result, any relay node will be considered at most once in the search process, which leads to a linear complexity for each iteration of ORA algorithm. Unmarking for all nodes is performed only upon the termination of an iteration.

An immediate question on our marking mechanism is how such a “linear marking” can lead to an optimal solution, as it appears that many possible assignments that may increase $C_{\text{min}}$ are not considered. This is precisely the question that we will address in Section 5, where we will prove that ORA can guarantee that its final solution is optimal (Theorem 1).

### 4.4 Complexity Analysis

We now analyze the computational complexity of ORA algorithm. Most computations in ORA are for iteratively finding a better solution. During each iteration, due to the “linear marking” mechanism in our algorithm, a relay node is checked for its availability at most once. Thus, the complexity of each iteration is $O(N_s)$. Now we examine the number of iterations that ORA will execute. For each source node, the number of possible capacities is $(N_c + 1)$. Thus, the total number of possible capacities (i.e., objective values) among all the source nodes in the network is $O(N_s(N_c + 1))$. Since the objective value is increased at each iteration (except the last iteration), the number of iterations is $O(N_s(N_c + 1))$. So the overall complexity of all the iterations is $O(N_s(N_c + 1) \cdot N_c) = O(N_s N_c^2)$.

### 4.5 An Example

We now use an example to illustrate the operation of the ORA algorithm, in particular, its “linear marking” mechanism. Suppose that there are four source-destination pairs and six relay nodes in the network. Table 1(a) shows the capacity for each source node $s_i$ when relay node $r_j$ is assigned to it. The symbol $\emptyset$ indicates direct transmissions, i.e., without the use of a relay node. Also shown in Table 1(a) is an initial relay node assignment, which is indicated by an underscore on the intersecting row ($s_i$) and column ($r_j$). That is, the initial assignment is $r_5$ for $s_1$, $r_4$ for $s_2$, $r_3$ for $s_4$, $r_2$ for $s_3$. Note that the preprocessing step before the initial assignment ensures that the capacity for each source-destination pair by the initial assignment is no less than direct transmissions.

Under the initial relay node assignment in Table 1(a), source $s_3$ is identified as having the smallest capacity of 13, which is the current value of $C_{\text{min}}$. Since consideration of relay nodes is performed in the order of non-increasing capacity for the source node under consideration, $r_4$ is therefore considered for $s_3$ (as it offers the largest capacity among all relay nodes for $s_3$). But $r_4$ is already assigned to source node $s_2$, so $r_4$ is “marked”. Now $s_2$ needs to find another relay. But, any other relay (or direct transmissions) will result in a capacity no greater than the current objective value $C_{\text{min}} = 13$. This means that $r_1$ cannot be taken away from $s_2$. Since $r_1$ does not work out for $s_3$, $s_3$ will then consider the relay node that offers the second largest capacity, i.e., relay node $r_3$. Since $r_3$ is already assigned to sender $s_3$, $r_3$ will be “marked”. Now, ORA algorithm checks to see if $s_4$ can find another relay.

Now $s_4$ checks relay nodes in non-increasing order of capacity. Since both $r_4$ (with the largest capacity) and $r_3$ (with the second largest capacity) are marked, they will not be considered. The relay with the third largest capacity is $r_2$, which is unmarked. Relay $r_2$ offers a capacity of 16, which is greater than $C_{\text{min}} = 13$. So $s_4$ will choose $r_2$. The new assignment after the first iteration is shown in Table 1(b).

Now the objective value, $C_{\text{min}}$, is updated to 15, which corresponds to $s_1$. Before the second iteration, all markings done in the first iteration are cleared. In the second iteration, ORA algorithm will perform a new search of relay node for $s_1$ with the aim that after some relay node re-assignment on other source nodes, they all have a capacity larger than 15.

The iteration continues and the final assignment upon termination of ORA algorithm is shown in Table 1(c), with the optimal (maximum) value of $C_{\text{min}}$ being 16.

### 5. PROOF OF OPTIMALITY

In this section, we give a correctness proof of ORA algorithm, that is, upon the termination of ORA algorithm, the solution (i.e., objective value and the corresponding relay node assignment) is optimal.

Our proof is based on contradiction. Denote $\psi$ the final solution obtained by the ORA algorithm, with the objective value being $C_{\text{min}}$. For $\psi$, denote the relay node assigned to source node $s_i$ as $R(s_i)$. Conversely, for $\psi$, denote the source node that uses relay node $r_j$ as $S(r_j)$.

### Table 1: An example illustrating the operation of ORA algorithm.

#### (a) Initial relay node assignment.

<table>
<thead>
<tr>
<th>$s_1$</th>
<th>$r_1$</th>
<th>$r_2$</th>
<th>$r_3$</th>
<th>$r_4$</th>
<th>$r_5$</th>
<th>$r_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>7</td>
<td>24</td>
<td>5</td>
<td>15</td>
<td>15</td>
<td>25</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>10</td>
<td>11</td>
<td>20</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>$s_3$</td>
<td>11</td>
<td>10</td>
<td>13</td>
<td>17</td>
<td>18</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>16</td>
<td>19</td>
<td>24</td>
<td>9</td>
<td>13</td>
</tr>
</tbody>
</table>

#### (b) Assignment after the first iteration.

<table>
<thead>
<tr>
<th>$s_1$</th>
<th>$r_1$</th>
<th>$r_2$</th>
<th>$r_3$</th>
<th>$r_4$</th>
<th>$r_5$</th>
<th>$r_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>7</td>
<td>24</td>
<td>5</td>
<td>15</td>
<td>15</td>
<td>25</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>10</td>
<td>11</td>
<td>20</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>$s_3$</td>
<td>11</td>
<td>10</td>
<td>13</td>
<td>17</td>
<td>18</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>16</td>
<td>19</td>
<td>24</td>
<td>9</td>
<td>13</td>
</tr>
</tbody>
</table>

#### (c) Final assignment upon algorithm termination.

<table>
<thead>
<tr>
<th>$s_1$</th>
<th>$r_1$</th>
<th>$r_2$</th>
<th>$r_3$</th>
<th>$r_4$</th>
<th>$r_5$</th>
<th>$r_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>7</td>
<td>24</td>
<td>5</td>
<td>15</td>
<td>15</td>
<td>25</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>10</td>
<td>11</td>
<td>20</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>$s_3$</td>
<td>11</td>
<td>10</td>
<td>13</td>
<td>17</td>
<td>18</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>16</td>
<td>19</td>
<td>24</td>
<td>9</td>
<td>13</td>
</tr>
</tbody>
</table>
Assume there exists a better solution \( \hat{\psi} \) than \( \psi \). That is, the objective value by \( \hat{\psi} \), denoted as \( C_{\text{min}}^{\hat{\psi}} \), is greater than that by \( \psi \), i.e., \( C_{\text{min}}^{\hat{\psi}} > C_{\text{min}}^{\psi} \). For \( \psi \), denote the relay node assigned to source node \( s_i \) as \( R(s_i) \). Conversely, for \( \hat{\psi} \), denote the source node that uses relay node \( r_j \) as \( S(r_j) \).

The key idea in the proof is to exploit the marking status at the end of the last iteration of ORA algorithm. Now let’s take a close look at this last iteration. During this iteration, ORA attempts to find a better solution but concludes that it cannot find any, and thus the algorithm terminates. So this last iteration is the only “non-improving” iteration for the objective value. At the end of this last iteration, assume that \( s_i \), with its assigned relay node \( R(s_i) \), is the “bottleneck” source node, i.e., \( C_{\text{min}} = C_{R(s_i), R(s_i)} \). Then we have the following fact for the marking status of \( R(s_i) \).

**FACT 1.** For the bottleneck source node \( s_i \) under \( \psi \), its relay node \( R(s_i) \) is not marked at the end of the last iteration of ORA algorithm.

**PROOF.** In the last iteration of ORA algorithm, a relay node \( r_j \) is marked only if it has been checked for its availability and it is not \( R(s_i) \) (see Check Relay Availability() in Fig. 5). Thus, \( R(s_i) \) cannot be marked at the end of the last iteration of the ORA algorithm. \( \square \)

Fact 1 will be a basis for contradiction in our proof for Theorem 1, the main result of this section. But first, we present the following three claims, which {	extit{reversely}} examine relay node assignment under \( \psi \).

First, for the relay node assigned to \( s_i \) in \( \hat{\psi} \), i.e., \( \hat{R}(s_i) \), we have the following claim.

**CLAIM 1.** Relay node \( \hat{R}(s_i) \) cannot be \( \emptyset \) and must be assigned to some source node under solution \( \psi \). Further, it must be marked at the end of the last iteration of the ORA algorithm.

**PROOF.** The proof for the first statement is based on contradiction. Suppose that \( \hat{R}(s_i) = \emptyset \) or relay node \( \hat{R}(s_i) \) is not assigned to any source node under solution \( \psi \). Since \( \hat{\psi} \) is a better solution than \( \psi \), we have \( C_{R(s_i), \hat{R}(s_i)} \geq C_{\text{min}}^{\psi} \). Thus, in the last iteration of the ORA algorithm, we should check \( \hat{R}(s_i) \)'s availability and a better solution should be found. However, we know that this last iteration is a non-improving iteration and the ORA algorithm cannot find a better solution. So, \( \hat{R}(s_i) \) cannot be \( \emptyset \) and must be assigned to some source node \( s_i \).

We now prove the second statement. Since \( C_{R(s_i), \hat{R}(s_i)} > C_{\text{min}} \), we should check \( \hat{R}(s_i) \)'s availability in the last iteration of the ORA algorithm. Since ORA algorithm cannot find a better solution in this last iteration, \( \hat{R}(s_i) \) should be marked and then the result for checking \( \hat{R}(s_i) \)'s availability must be unavailable. Thus, \( \hat{R}(s_i) \) must be marked at the end of ORA algorithm. \( \square \)

Claim 1 states that in solution \( \psi \), relay node \( \hat{R}(s_i) \) must be assigned to some source node. By the definition of \( S(\cdot) \), we have that \( \hat{R}(s_i) \) is assigned to source node \( S(\hat{R}(s_i)) \) in solution \( \psi \). To simplify notation, define function \( G(\cdot) \) as

\[
G(\cdot) = S(\hat{R}(\cdot)) .
\]

Thus, relay node \( \hat{R}(s_i) \) is assigned to source node \( G(s_i) \) in \( \psi \) (see the top portion of Fig. 6).

Now we recursively investigate the relay node assigned to \( G(s_i) \) under solution \( \hat{\psi} \), i.e., \( \hat{R}(G(s_i)) \). We have the following claim (also see Fig. 6). Its proof follows the same token as that for Claim 1 and is omitted to conserve page length.

**CLAIM 2.** Relay node \( \hat{R}(G(s_i)) \) cannot be \( \emptyset \) and must be assigned to some source node under solution \( \psi \). Further, it must be marked at the end of the last iteration of the ORA algorithm.

Claim 2 states that in solution \( \psi \), relay node \( \hat{R}(G(s_i)) \) must be assigned to some source node. By the definition of \( S(\cdot) \), we have that \( \hat{R}(G(s_i)) \) is assigned to source node \( S(\hat{R}(G(s_i))) \) in solution \( \psi \). By (7), we have \( S(\hat{R}(G(s_i))) = G(s_i) \). To simply the notation, denote function \( G^2(\cdot) \) as

\[
G^2(\cdot) = G(\cdot) .
\]

Thus, relay node \( \hat{R}(G(s_i)) \) is assigned to source node \( G^2(s_i) \) in \( \psi \).

Following the same token for Claims 1 and 2, we can obtain a similar claim for the relay node assigned to \( G^2(s_i) \) under \( \psi \), i.e., \( \hat{R}(G^2(s_i)) \). This recursive investigation of source node of \( \hat{R}(\cdot) \) under \( \psi \) and relay node assigned to \( G(\cdot) \) under \( \hat{\psi} \) continues and will terminate at some \( n \)-th step since the numbers of source and relay nodes are finite (see Fig. 6).

Denote

\[
G^k(s_i) = \begin{cases} s_i, & 1 \leq k \leq n \end{cases}
\]

Thus, we have that in \( \hat{\psi} \), the corresponding relay nodes for \( s_i, G(s_i), \ldots, G^n(s_i) \) are \( \hat{R}(s_i), \hat{R}(G(s_i)), \ldots, \hat{R}(G^n(s_i)) \), respectively (see Fig. 6). We can prove one claim for each of these relay nodes. In summary, we have the following claim.

**CLAIM 3.** Relay node \( \hat{R}(G^k(s_i)) \) cannot be \( \emptyset \) and must be assigned to some source node under solution \( \psi \), \( k = 0, 1, 2, \ldots, n \). Further, it must be marked at the end of the last iteration of the ORA algorithm.

Note that we already have Claims 1 and 2 for \( k = 0 \) and \( k = 1 \), respectively. The proof for the general case in this claim follows the same token and is omitted to conserve page length.

We are now ready to prove the following theorem, which is the main result of this section.

**THEOREM 1.** Upon the termination of the ORA algorithm, the obtained solution \( \psi \) is optimal.

**PROOF.** Referring to Fig. 6, we have Claim 3 for a set of relay nodes \( R(s_i), \hat{R}(G(s_i)), \ldots, \hat{R}(G^n(s_i)) \).

Under Claim 3, \( \hat{R}(G^n(s_i)) \) is assigned to a source node in solution \( \psi \). We now investigate to which source node it is assigned in \( \psi \). This source node must be a node among \( \{ s_i, G(s_i), \ldots, G^n(s_i) \} \), otherwise the recursive process will not terminate at \( \hat{R}(G^n(s_i)) \).
But under $\psi$, each of $G(s_1), G^2(s_1), G^3(s_1), \ldots, G^n(s_1)$ has its own relay $R(s_i), R(G(s_i)), R(G^2(s_i)), \ldots, R(G^n(s_1))$, respectively. Thus, $R(G^n(s_1))$ can only be assigned to $s_i$ in solution $\psi$. On the other hand, relay $R(s_i)$ is assigned to $s_i$ in solution $\psi$. So we must have $R(G^n(s_1)) = R(s_i)$.

However, by Claim 3, we have $R(G^n(s_1))$ is marked, while by Fact 1, we have $R(s_i)$ is not marked. This is a contradiction. Thus the assumption that there exists a better solution $\psi$ than $\psi$ does not hold and the proof is complete.

Note that the proof of Theorem 1 does not depend on the initial assignment in the ORA algorithm. So we have the following important property for the ORA algorithm.

**Corollary 1.1.** Under any initial relay node assignment, the ORA algorithm can find an optimal relay node assignment.

### 6. NUMERICAL RESULTS

In this section, we present some numerical results to demonstrate the properties of the ORA algorithm.

#### 6.1 Simulation Setting

We consider a 50-node ad hoc network. For this network, we consider both the cases of $N_r \geq N_s$ and $N_r < N_s$. In the first case, we have 15 source-destination pairs and 20 relay nodes. While in the second case, we have 20 source-destination pairs and only 10 relay nodes. The role of each node (either as a source, destination, or relay) for each case is shown in Figs. 7 and 9, respectively.

For the simulations, we assume $W = 22$ MHz bandwidth for each channel. The maximum transmission power at each node is set to 1 W. Each relay works on AF mode. For simplicity, we assume that $h_{sd}$ only includes the path loss component between nodes $s$ and $d$ and is given by $|h_{sd}| = |s - d|^{-\alpha}$, where $|s - d|$ is the distance (in meters) between these two nodes and 4 is the path loss index. For the AWGN channel, we assume the variance of noise is $10^{-10}$ W at all nodes.

#### 6.2 Results

**Case 1: $N_r \geq N_s$.** In this case (see Fig. 7), we have 15 source-destination pairs and 20 relay nodes.

Under ORA, after preprocessing, we start with an initial relay node assignment in the first iteration. Such initial assignment is not

<table>
<thead>
<tr>
<th>Sender</th>
<th>Direct Transmission Capacity (Mbps)</th>
<th>Initial</th>
<th>Final</th>
<th>Final Capacity (Mbps)</th>
<th>Initial</th>
<th>Final</th>
<th>Final Capacity (Mbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>6.1</td>
<td>$r_6$</td>
<td>$r_2$</td>
<td>15.3</td>
<td>$r_1$</td>
<td>$r_2$</td>
<td>15.3</td>
</tr>
<tr>
<td>$s_2$</td>
<td>10.6</td>
<td>$r_6$</td>
<td>$r_7$</td>
<td>21.8</td>
<td>$r_7$</td>
<td>$r_7$</td>
<td>21.8</td>
</tr>
<tr>
<td>$s_3$</td>
<td>8.8</td>
<td>$r_1$</td>
<td>$r_1$</td>
<td>16.6</td>
<td>$r_8$</td>
<td>$r_3$</td>
<td>18.9</td>
</tr>
<tr>
<td>$s_4$</td>
<td>6.3</td>
<td>$r_5$</td>
<td>$r_4$</td>
<td>10.7</td>
<td>$r_2$</td>
<td>$r_4$</td>
<td>10.7</td>
</tr>
<tr>
<td>$s_5$</td>
<td>7.3</td>
<td>$r_{10}$</td>
<td>$r_{14}$</td>
<td>14.9</td>
<td>$r_{13}$</td>
<td>$r_{12}$</td>
<td>15.0</td>
</tr>
<tr>
<td>$s_6$</td>
<td>9.6</td>
<td>$r_4$</td>
<td>$r_6$</td>
<td>21.3</td>
<td>$r_5$</td>
<td>$r_8$</td>
<td>11.0</td>
</tr>
<tr>
<td>$s_7$</td>
<td>4.2</td>
<td>$r_2$</td>
<td>$r_8$</td>
<td>11.0</td>
<td>$r_{10}$</td>
<td>$r_8$</td>
<td>11.0</td>
</tr>
<tr>
<td>$s_8$</td>
<td>6.9</td>
<td>$r_{16}$</td>
<td>$r_{12}$</td>
<td>16.6</td>
<td>$r_6$</td>
<td>$r_{10}$</td>
<td>11.0</td>
</tr>
<tr>
<td>$s_9$</td>
<td>11.3</td>
<td>$r_7$</td>
<td>$r_{10}$</td>
<td>22.6</td>
<td>$r_9$</td>
<td>$r_9$</td>
<td>11.9</td>
</tr>
<tr>
<td>$s_{10}$</td>
<td>11.1</td>
<td>$r_{19}$</td>
<td>$r_{17}$</td>
<td>19.9</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>11.1</td>
</tr>
<tr>
<td>$s_{11}$</td>
<td>9.5</td>
<td>$r_{17}$</td>
<td>$r_{20}$</td>
<td>21.0</td>
<td>$r_{16}$</td>
<td>$r_{20}$</td>
<td>21.1</td>
</tr>
<tr>
<td>$s_{12}$</td>
<td>7.4</td>
<td>$r_{18}$</td>
<td>$r_{18}$</td>
<td>12.8</td>
<td>$r_{19}$</td>
<td>$r_{19}$</td>
<td>13.6</td>
</tr>
<tr>
<td>$s_{13}$</td>
<td>8.5</td>
<td>$r_{20}$</td>
<td>$r_{16}$</td>
<td>11.9</td>
<td>$r_{18}$</td>
<td>$r_{18}$</td>
<td>12.2</td>
</tr>
<tr>
<td>$s_{14}$</td>
<td>9.7</td>
<td>$r_{15}$</td>
<td>$r_{15}$</td>
<td>12.2</td>
<td>$r_{17}$</td>
<td>$r_{16}$</td>
<td>18.1</td>
</tr>
<tr>
<td>$s_{15}$</td>
<td>6.1</td>
<td>$\emptyset$</td>
<td>$r_{19}$</td>
<td>13.5</td>
<td>$r_{20}$</td>
<td>$r_{17}$</td>
<td>11.2</td>
</tr>
</tbody>
</table>
unique. But regardless of initial relay node assignment, the objective value can always converge to the optimum (by Corollary 1.1). To validate this point, in Table 2, we show the results of running the ORA algorithm under two different initial relay node assignments, denoted as I and II (see Table 2).

In Table 2, the second column shows the capacity for each source-destination pair under direct transmissions. Note that the minimum capacity among all pairs is 4.2 Mbps, which is associated with $s_7$. The third to fifth columns are results under initial relay node assignment I and sixth to eighth columns are results under initial relay node assignment II. The symbol $∅$ denotes direct transmissions. Note than the initial relay node assignments I and II are different. As a result, the final assignment is different under I and II. However, the final objective value (i.e., $C_{\text{min}}$) under I and II is identical (10.7 Mbps).

Figure 8 shows the objective value $C_{\text{min}}$ at each iteration under initial relay node assignments I and II. Under either initial relay node assignments I or II, $C_{\text{min}}$ is a non-decreasing function of iteration number. The increase of $C_{\text{min}}$ by cooperative communications over direct transmissions is significant (from 4.2 Mbps to 10.7 Mbps).

Case 2: $N_r < N_s$. In this case (see Fig. 9), we have 20 source-destination pairs and 10 relay nodes.

Table 3 shows the results of this case under two different initial relay node assignments I and II. The second column in Table 3 lists the capacity among all pairs is 4.2 Mbps, it is only necessary to consider relay node assignment for $N_r = 10$ source nodes corresponding to the 10 smallest capacities, i.e., nodes $s_1$, $s_3$, $s_4$, $s_5$, $s_7$, $s_8$, $s_{11}$, $s_{12}$, $s_{13}$, and $s_{15}$. As a result, the problem size can be reduced.

Figure 9: A 50-node network topology for Case 2 $(N_r < N_s)$, with $N_s = 20$ and $N_r = 10$.

Again in Table 3, the objective value $C_{\text{min}}$ is identical (9.0 Mbps) regardless of different initial relay node assignments (I and II). Note that the final relay node assignment under I and II is not identical, although the objective value $C_{\text{min}}$ is the same. The increase of

<table>
<thead>
<tr>
<th>Sender</th>
<th>Direct Transmission Capacity (Mbps)</th>
<th>Initial Relay Assignment I</th>
<th>Final Relay Assignment I</th>
<th>Initial Relay Assignment II</th>
<th>Final Relay Assignment II</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>6.1</td>
<td>$r_1$</td>
<td>10.2</td>
<td>$r_1$</td>
<td>10.3</td>
</tr>
<tr>
<td>$s_2$</td>
<td>10.6</td>
<td>$∅$</td>
<td>10.6</td>
<td>$∅$</td>
<td>10.6</td>
</tr>
<tr>
<td>$s_3$</td>
<td>8.8</td>
<td>$r_2$</td>
<td>20.1</td>
<td>$r_2$</td>
<td>20.1</td>
</tr>
<tr>
<td>$s_4$</td>
<td>6.3</td>
<td>$∅$</td>
<td>10.7</td>
<td>$r_3$</td>
<td>10.7</td>
</tr>
<tr>
<td>$s_5$</td>
<td>7.3</td>
<td>$r_5$</td>
<td>9.0</td>
<td>$r_5$</td>
<td>9.0</td>
</tr>
<tr>
<td>$s_6$</td>
<td>9.6</td>
<td>$∅$</td>
<td>9.6</td>
<td>$∅$</td>
<td>9.6</td>
</tr>
<tr>
<td>$s_7$</td>
<td>4.2</td>
<td>$r_3$</td>
<td>9.3</td>
<td>$r_1$</td>
<td>9.3</td>
</tr>
<tr>
<td>$s_8$</td>
<td>6.9</td>
<td>$r_9$</td>
<td>9.3</td>
<td>$r_9$</td>
<td>9.3</td>
</tr>
<tr>
<td>$s_9$</td>
<td>9.9</td>
<td>$∅$</td>
<td>9.3</td>
<td>$∅$</td>
<td>9.3</td>
</tr>
<tr>
<td>$s_{10}$</td>
<td>11.1</td>
<td>$∅$</td>
<td>11.1</td>
<td>$∅$</td>
<td>11.1</td>
</tr>
<tr>
<td>$s_{11}$</td>
<td>3.5</td>
<td>$r_{10}$</td>
<td>20.1</td>
<td>$r_{10}$</td>
<td>20.1</td>
</tr>
<tr>
<td>$s_{12}$</td>
<td>7.5</td>
<td>$∅$</td>
<td>12.8</td>
<td>$r_8$</td>
<td>12.8</td>
</tr>
<tr>
<td>$s_{13}$</td>
<td>8.5</td>
<td>$∅$</td>
<td>16.9</td>
<td>$r_8$</td>
<td>16.9</td>
</tr>
<tr>
<td>$s_{14}$</td>
<td>9.8</td>
<td>$∅$</td>
<td>11.1</td>
<td>$∅$</td>
<td>11.1</td>
</tr>
<tr>
<td>$s_{15}$</td>
<td>6.1</td>
<td>$r_8$</td>
<td>12.9</td>
<td>$r_7$</td>
<td>12.9</td>
</tr>
<tr>
<td>$s_{16}$</td>
<td>20.5</td>
<td>$∅$</td>
<td>20.5</td>
<td>$∅$</td>
<td>20.5</td>
</tr>
<tr>
<td>$s_{17}$</td>
<td>49.6</td>
<td>$∅$</td>
<td>49.6</td>
<td>$∅$</td>
<td>49.6</td>
</tr>
<tr>
<td>$s_{18}$</td>
<td>36.0</td>
<td>$∅$</td>
<td>36.0</td>
<td>$∅$</td>
<td>36.0</td>
</tr>
<tr>
<td>$s_{19}$</td>
<td>33.5</td>
<td>$∅$</td>
<td>33.5</td>
<td>$∅$</td>
<td>33.5</td>
</tr>
<tr>
<td>$s_{20}$</td>
<td>21.2</td>
<td>$∅$</td>
<td>21.2</td>
<td>$∅$</td>
<td>21.2</td>
</tr>
</tbody>
</table>
Table 4: An example illustrating the importance of preprocessing.

<table>
<thead>
<tr>
<th>Sender</th>
<th>Direct Transmission Capacity (Mbps)</th>
<th>Without Preprocessing</th>
<th>Final Capacity (Mbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>6.1</td>
<td>r2</td>
<td>15.3</td>
</tr>
<tr>
<td>s2</td>
<td>10.6</td>
<td>r12</td>
<td>21.8</td>
</tr>
<tr>
<td>s3</td>
<td>8.8</td>
<td>r1</td>
<td>16.6</td>
</tr>
<tr>
<td>s4</td>
<td>6.4</td>
<td>r14</td>
<td>10.7</td>
</tr>
<tr>
<td>s5</td>
<td>7.3</td>
<td>r13</td>
<td>15.0</td>
</tr>
<tr>
<td>s6</td>
<td>9.6</td>
<td>r16</td>
<td>21.3</td>
</tr>
<tr>
<td>s7</td>
<td>4.2</td>
<td>r5</td>
<td>11.0</td>
</tr>
<tr>
<td>s8</td>
<td>6.9</td>
<td>r6</td>
<td>11.0</td>
</tr>
<tr>
<td>s9</td>
<td>11.3</td>
<td>r11</td>
<td><strong>11.0</strong></td>
</tr>
<tr>
<td>s10</td>
<td>11.0</td>
<td>r15</td>
<td>22.6</td>
</tr>
<tr>
<td>s11</td>
<td>9.5</td>
<td>r17</td>
<td>21.1</td>
</tr>
<tr>
<td>s12</td>
<td>7.5</td>
<td>r18</td>
<td>12.8</td>
</tr>
<tr>
<td>s13</td>
<td>8.5</td>
<td>r3</td>
<td>16.9</td>
</tr>
<tr>
<td>s14</td>
<td>9.8</td>
<td>r10</td>
<td>12.2</td>
</tr>
<tr>
<td>s15</td>
<td>6.0</td>
<td>r19</td>
<td>12.9</td>
</tr>
</tbody>
</table>

$C_{\text{min}}$ by cooperative communications over direct transmissions is significant (from 4.2 Mbps to 9.0 Mbps).

Figure 10 shows the objective value $C_{\text{min}}$ at each iteration under initial relay node assignments I and II. Again, we observe that in Fig. 10, $C_{\text{min}}$ is a non-decreasing function of iteration number under either initial relay node assignments I or II.

### Importance of Preprocessing

Now we use a set of simulation results to show the significance of preprocessing in our ORA algorithm. We consider the same network in Fig. 7 with 15 source-destination pairs and 20 relay nodes. Now we remove the preprocessing step in ORA algorithm. As an example, the third column of Table 4 shows an initial assignment without first going through the preprocessing step. Although the objective value $C_{\text{min}}$ also reaches the same optimal value (10.7 Mbps) as that in Table 2, the final capacity for some non-bottleneck source nodes could be worse than direct transmissions. For example, for $s_9$, its final capacity is 11.0 Mbps, which is less than direct transmissions (11.3 Mbps). Such event is undetectable without the preprocessing step, as 11.0 Mbps is still greater than the optimal objective value (10.7 Mbps).

On the other hand, when the preprocessing step is employed, ORA can ensure that the final capacity for each source-destination pair is no less than direct transmissions.

### 7. CONCLUSION

Cooperative communications is a powerful communication paradigm to achieve spatial diversity. However, the performance of such communication paradigm hinges upon the choice of relay node in the network. In this paper, we studied the relay node assignment problem in a network environment, where multiple source-destination pairs compete for the same pool of relay nodes in the network. The main contribution of this paper is a polynomial time algorithm to solve this problem. A key idea in this algorithm is a “linear marking” mechanism, which is able to achieve linear complexity at each iteration. We gave a formal proof of optimality for the algorithm and used numerical results to demonstrate its capability. There are several attractive properties associated with this algorithm, such as its robustness to the number of relay nodes in the network, its guarantee for each source-destination pair to have capacity no less than direct transmissions, and its ability to find the optimal objective regardless of initial assignment.

### Acknowledgments

The work of Y.T. Hou, Y. Shi, and S. Sharma has been supported in part by the National Science Foundation (NSF) under Grant CNS-0721421 and Office of Naval Research (ONR) under Grant N00014-08-1-0084. The work of S. Kompella has been supported in part by the ONR.

### 8. REFERENCES


