An Efficient Interference Management Framework for Multi-hop Wireless Networks

Lei Shi*, Yi Shi†, Yuxiang Ye*, Zhenchun Wei*, Jianghong Han*
*School of Computer and Information, Hefei University of Technology, Hefei, Anhui, China
†Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA, USA

***Abstract*—Interference management is an important problem in wireless networks. In this paper, we focus on the successive interference cancellation (SIC) technique, and aim to design an efficient cross-layer solution to increase throughput for multi-hop wireless networks with SIC. We realize that the challenge of this problem is its mixed integer linear programming formulation, which has bunches of integer variables. In order to solve this problem efficiently, we propose an iterative framework to improve the solution for integer variables and use a linear programming to solve the problem for other variables. Our analysis indicates that the proposed algorithm is with polynomial-time complexity. Simulation results show that SIC can increase throughput of a multi-hop wireless network by around 300%.**

I. Introduction

Interference is a unique problem in wireless networks due to the shared wireless medium. In particular, collisions may occur when more than one nodes send data simultaneously. Traditional mechanisms, e.g. TDMA, CDMA, FDMA, CSMA, CSMA/CA, adopt the so-called "interference avoidance" technique to avoid collisions by transmitting data in different orthogonal channels (built in time-, space-, and/or code-domain, or combination of these domains). Although practical, these methods cannot achieve the maximum throughput theoretically.

Recently there is a growing interest in exploiting interference (rather than always avoiding it) to increase network throughput. Such interference management schemes can be categorized into three types [1]: interference cancellation [2], interference randomization [3] and interference coordination [3]. In this paper we focus on interference cancellation, where a receiver tries to decode strong interference and remove it from the received signal. The interference cancellation can be realized through several techniques, such as successive interference cancellation (SIC) [4], parallel interference cancellation (PIC) [5], iterative interference cancellation [6]. Among them, the SIC technique is preferred nowadays due to its simplicity and effectiveness [4]. Thus, we focus on SIC in this paper.

SIC is a powerful physical layer technique based on signal processing. However, most traditional techniques and protocols in upper layers (e.g., link layer, network layer) are developed according to interference avoidance schemes and thus may not work well with SIC. Some previous works have been done to develop upper layer techniques working with SIC [7], [8], [9], [13], [14]. But most of them are focused on the link layer (see e.g., [9]), or on the network layer with a relatively simple network model (see, e.g., [8], [13]). To date, results on how to apply SIC in a multi-hop network remain very limited due to the difficulty of designing a cross-layer (from physical layer to network layer) optimal solution with SIC. Jiang *et al.* [16] built a cross-layer optimization framework, including routing, scheduling, and SIC, with the goal of increasing the throughput. The formulated problem is a mixed integer linear program (MILP), which may not be solved efficiently when there are many integer variables.

In this paper, we develop an efficient cross-layer solution for multi-hop wireless networks. We first formulate the throughput maximization problem in consistent with the interference avoidance scheme. Although the formulated optimization problem is an MILP, it can be solved by some commercial solvers (e.g., [11]) to provide a performance benchmark. Secondly, we address the throughput maximization problem with SIC. We can also formulate this problem as an MILP. But this MILP has much more integer variables than the former one, and thus cannot be solved efficiently by existing solvers. This problem is mainly caused by a large number of integer variables for scheduling decisions. We build an iterative framework to overcome this problem. Once the scheduling problem is solved, the remaining problem can be formulated as a linear programming (LP), which can be solved in polynomial-time. Simulation results show that our algorithm can achieve about 300% performance improvement than the optimal solution acquired through the interference avoidance technique.

The rest of the paper is organized as follows. In Section II, we solve the throughput maximization problem for the interference avoidance scheme. Section III designs an algorithm for a multi-hop network through SIC. We also analyze the polynomial-time complexity of this algorithm. In Section IV, we present simulation results and show the efficiency of our algorithm. Section V concludes this paper.

II. Traditional Multi-Hop Wireless Networks

In this section we describe a throughput maximization problem for traditional multi-hop wireless networks using the interference avoidance scheme. Its solution will be used as a performance benchmark when comparing with the solution with SIC.

Consider a multi-hop wireless network with n nodes $s_1, s_2, \cdots, s_n$ and one base station (see Fig. 1). Each node transmits data to the base station via either single- or multi-hop transmissions. Suppose that all the nodes have the same

978-1-4673-5939-9/13/$31.00 ©2013 IEEE

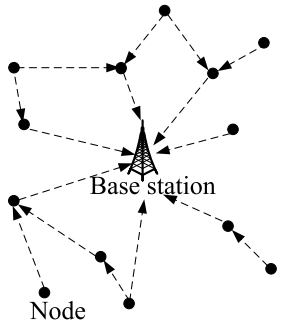

Fig. 1. The topological structure of a multi-hop random network.

transmission power P and bandwidth W. Denote N as the set of n nodes. To simplify discussion, the base station is denoted by s_0, as well.

A. Physical Layer and Link Layer Model

In this paper, we only discuss the scheduling problem in time domain, although our approach can be extended to space- or code domain. Denote t_k, $k = 1, 2, \cdots, h$, as time slots. Define a binary scheduling variable $x_{ij}[k]$ for link $s_i \to s_j$ in time slot t_k, which is one if node s_i transmits data to s_j in time slot t_k and is zero otherwise.

In a time slot t_k, a node s_i can transmit to (and receive from) if any, one node, i.e.,

$$\sum_{s_l \in T_i} x_{li}[k] + \sum_{s_j \in T_i} x_{ij}[k] \le 1 \quad (s_i \in N, 1 \le k \le h) \ , \quad (1)$$

where T_i is the set of all neighboring nodes in the transmission range R_T of node s_i.

In accordance with the protocol model, when node s_i transmits data to s_j, any node $s_l \in I_j$ cannot transmit simultaneously, where I_j is the set of all neighboring nodes in the interference range R_I of node s_j. Then we have

$$x_{ij}[k] + \frac{\sum_{s_l \in I_j}^{s_l \ne s_i} \sum_{s_m \in T_l} x_{lm}[k]}{|I_j| - 1} \le 1 \quad (s_i \in N, s_j \in T_i, 1 \le k \le h) \ , \quad (2)$$

where $|I_j|$ is the number of nodes in the set I_j.

B. Problem and Its Solution

We now introduce a set of unicast communication sessions in the network. Each session is from one node s_i to the base station and has a minimum data rate requirement $r(s_i)$. In this paper, we consider the problem of maximizing a common scaling factor K such that data with rate $Kr(s_i)$ can be transmitted from node s_i to the base station. Other objective functions can be treated similarly. Denote r_{ij}, $s_i \in N, s_j \in T_i$, as the average data rate from node s_i to node s_j. Then we have the following relationship for flow rates at a node s_i.

$$\sum_{s_l \in T_i} r_{li} + Kr(s_i) = \sum_{s_j \in T_i} r_{ij} \quad (s_i \in N) \quad (3)$$

Denote C as the data rate by a successful transmission. Since flow rate on any link cannot exceed the achievable average link rate, we have

$$r_{ij} \le \frac{1}{h} \sum_{k=1}^{h} (C \cdot x_{ij}[k]) \quad (s_i \in N, s_j \in T_i) \ . \quad (4)$$

The problem can be formulated as follows.

$$\begin{array}{ll} \max & K \\ \text{s.t.} & (1), (2), (3), (4) \\ & x_{ij}[k] \in \{0,1\}, r_{ij}, K \ge 0 \quad (s_i \in N, s_j \in T_i, 1 \le k \le h). \end{array} \quad (5)$$

Due to binary variables $x_{ij}[k]$, this is a mixed integer linear programming (MILP) problem, which is NP-hard in general [10]. Although we can obtain optimal solution by some commercial solver, e.g., CPLEX [11], the complexity is very high when the number of integer variables is large.

III. Multi-Hop Wireless Networks with SIC

In this section we consider the same throughput maximization problem while SIC is applied. In this case, a receiver can decode signals from multiple transmitters sequentially. That is, it tries to decode the strongest signal first. If the corresponding SINR (signal-interference-and-noise-ratio) is less than β, then even the strongest signal cannot be decoded. Otherwise (the strongest signal is decoded), the receiver can remove the strongest signal from its total received signal. The benefit of SIC is that by removing the strongest signal, we now have an improved SINR for the remaining interfered signals. Then the receiver tries to decode the second strongest signal. If the second strongest signal is decoded, the receiver can remove this signal and then tries to decode the third strongest signal. This process is repeated until all signals are decoded or no further signals can be decoded.

In general, when a node s_j tries to decode the signal received from node s_i in a time slot t_k, all stronger signals have already been decoded and removed according to SIC. Thus, the improved SINR can be calculated by

$$SINR_{ij}[k] = \frac{g_{ij}P}{\sum_{s_l \ne s_i}^{g_{lj} \le g_{ij}} (g_{lj} P \sum_{s_m \in T_l} x_{lm}[k]) + N_0} \ , \quad (6)$$

where $\sum_{s_m \in T_l} x_{lm}[k] = 1$ if s_l is transmitting data to some node in time slot t_k and $\sum_{s_m \in T_l} x_{lm}[k] = 0$ if s_l is not transmitting in time slot t_k. If this improved SINR is no less than β, node s_j can decode signal from node s_i.

From the process mentioned above, we can conclude that the problem formulation with SIC will be much more complex than (5). It is very challenging to find an optimal solution for multi-hop wireless networks with SIC. In this paper, we will propose a heuristic algorithm with much less complexity while good in performace.

A. Main Idea

We first analyze the difficulty to solve a problem with SIC. Similar to (5), binary scheduling variables $x_{ij}[k]$ can make the problem non-convex and thus NP-hard in general. If we can

```
1.  Let S = T_B, d = 1, and d(s_i) = d for each s_i ∈ S.
2.  while (S ≠ ∅) {
3.      Let d = d + 1 and Ŝ = ∅.
4.      while (S ≠ ∅) {
5.          Remove a node s_i from S.
6.          Add each node s_j ∈ T_i with undefined d(s_j) in Ŝ and
            let d(s_j) = d. }
7.      Let S = Ŝ. }
```

Fig. 2. Identify the minimum hop distance for all nodes.

find a way to determine these $x_{ij}[k]$ values, then the remaining problem is only on continuous variables r_{ij} and K, i.e.,

$$\begin{array}{ll} \max & K \\ \text{s.t.} & (3), (4) \\ & r_{ij}, K \geq 0 \quad (s_i \in N, s_j \in T_i) \ . \end{array} \tag{7}$$

This formulation is a linear programming (LP) problem, and can be solved in polynomial-time[12].

Now the question is how to determine the value of each $x_{ij}[k]$. The main idea and the basic steps of our algorithm is as follows.

I. Initialize $K = 0$ and all $x_{ij}[k] = 0$. For each node, establish an initial path to the base station and time slots assignment for each link on this path, such that (1) holds and each link has an improved SINR no less than β.
II. Under current $x_{ij}[k]$ values, calculate the maximum K and all r_{ij} values by (7). Note that by (7), routing solution (r_{ij} values) may be updated and thus K may be increased.
III. If calculated K is equal to the previous K, then our algorithm terminates. Otherwise, try to improve the current scheduling solution ($x_{ij}[k]$ values) and go to Step II.

There are two challenges in the above algorithm. First, how to establish an initial path for each node in Step I? Second, how to improve the current scheduling solution in Step III? These challenges will be addressed in Sections III-B and III-C.

B. Establish Initial Paths

Before we establish an initial path to the base station for each node, we first obtain the minimum hop distance to the base station for each node. For simplicity, we will use hop distance, instead of the minimum hop distance, in the following discussion. These hop distances can be calculated iteratively. In the first iteration, any neighboring node of the base station has hop distance 1. In the $(d-1)$-th iteration, we labeled all nodes with hop distance $d-1$. Then in the d-th iteration, we consider the neighboring nodes of those nodes with hop distance $d-1$. Each of these neighboring nodes, if its hop distance is not calculated yet, has hop distance d. This process terminates until all nodes have their hop distances calculated. To simplify discussion, we say that the base station B has hop distance 0. The pseudo-code to calculated the minimum hop distance for all nodes is given in Fig. 2.

One possible approach to establish an initial path to the base station for each node is using the minimum-hop path. The benefit of using the minimum-hop path is that time slots assignment can be easier due to the minimum number of links. We can obtain the minimum-hop path by letting a node with hop distance d choose one of its neighboring nodes with distance $d-1$ as its next hop node. Once a node s_i chooses node s_j as its next hop node, a new link is added in the network.

```
1.  Let minSINR = 0.9 · β and k* = 0.
    //Suppose time slots 1 to ĥ are used by some links.
2.  for (k = 1; k ≤ ĥ; k++) {
3.      Try to assign time slot t_k to link (s_i, s_j) (let x_ij[k] = 1).
4.      if SINR_ij[k] by (6) is less than β {
5.          Let x_ij[k] = 0.
6.          continue; }
7.      Let minSINR[k] = SINR_ij[k].
8.      for each link (s_l, s_m) with x_lm[k] = 1 {
9.          if SINR_lm[k] by (6) is less than β {
10.             Let x_ij[k] = 0 and minSINR[k] = 0.
11.             break; }
12.         if (SINR_lm[k] < minSINR[k])
13.             Let minSINR[k] = SINR_lm[k]. }
14.     if (minSINR[k] ≥ β) and (minSINR[k] > minSINR)
15.         Let minSINR = minSINR[k] and k* = k. }
16. if (k* > 0)
17.     Assign time slot t_k* to link link (s_i, s_j).
18. else
19.     Assign time slot t_ĥ+1 to link link (s_i, s_j).
```

Fig. 3. Time slot assignment for a new link (s_i, s_j).

Establishing a path also includes time slots assignment on each of its links, which is performed whenever a new link is added in the network.

- When a new link is added, we first try to assign a time slot already used by some link. If there is a conflict after this assignment (i.e., some existing links or the new link with $SINR < \beta$), then we should consider another time slot. Otherwise, an available time slot is found.
- It is possible that we find multiple time slots for a new link. We used the minimum improved SINR (see (6)) among all links in a time slot as a metric and choose the time slot that can maximize the minimum improved SINR as the most appropriate time slot.
- It is possible that all the used time slots cannot be chosen. In this situation, we allocate a new time slot for this link.

The pseudo-code to assign a time slot for a new link is given in Fig. 3.

Once we establish an initial path for each node and assign a time slot for each link, we can calculate the maximum K and r_{ij} values by (7).

C. Increase Bottleneck Node Throughput

To increase K value, we should identify bottleneck links and bottleneck nodes. A link $s_i \rightarrow s_j$ is a bottleneck link if $r_{ij} = \frac{1}{h}\sum_{k=1}^{h}(C \cdot x_{ij}[k])$, i.e., there is no residual capacity (defined as $\frac{1}{h}\sum_{k=1}^{h}(C \cdot x_{ij}[k]) - r_{ij}$) on this link. A node s_i is a bottleneck node if all its existing out-going links are bottleneck links. For a bottleneck node, if we can assign an additional time slot to one of its existing links or add a new out-going link, then the throughput at this node can be increased. Meanwhile, K value may also be increased.

1. Determine residual node capacity Z_j for each $s_j \in T_i$.
2. Check s_i's neighboring nodes from the largest residual capacity to the smallest residual capacity {
   //Suppose the current checked node is s_j.
3. if link (s_i,s_j) exists {
4. Try to assign an additional time slot by an algorithm similar to Fig. 3, where we skip the time slots already assigned to link (s_i, s_j).
5. if the assignment is success
6. return; }
7. else { //link (s_i,s_j) does not exist
8. Try to add link (s_i, s_j) and assign a time slot by the algorithm in Fig. 3.
9. if a new link is added
10. return; } }
11. We cannot improve node s_i's throughput. The entire algorithm terminates.

Fig. 4. Increase bottleneck node s_i's throughput.

Based on the above discussion, in each iteration, we can either assign one more time slot on the existing link of a bottleneck node or add a new link to increase K. Therefore, we have the following problems:

Problem 1: Should we use a new time slot (on an existing link) or a new link (with an assigned time slot) to increase K?

Problem 2: If we use a new time slot, which time slot should be assigned on which existing link?

Problem 3: If we use a new link, which new link (and which time slot) should be added?

To answer these problems, we need to analyze possible improvement after using a new time slot or a new link. We introduce the following definition.

Definition 1: The residual node capacity Z_i of node s_i is the maximum possible increase on node s_i's data rate to base station s_0 by considering all existing paths from s_i to base station s_0.

The residual node capacity can be determined by a maximum flow problem [10]. That is, we can define an auxiliary graph with the same set of nodes and links, where the capacity on each link is set as its residual link capacity. Then the maximum flow from s_i to base station s_0 in this auxiliary graph is the residual node capacity Z_i.

We now consider possible improvement under the following two cases.

- *Possible improvement by a new time slot.* For each out-going link (s_i, s_j), we can revise the algorithm in Fig. 3 to determine a suitable time slot. That is, instead of considering all time slots, we only consider time slots not used by link (s_i, s_j). Once a new time slot is found, possible improvement on link (s_i, s_j) is $\frac{C}{h}$ while possible improvement from node s_j to the base station is Z_j. Thus, possible improvement is $\min\{\frac{C}{h}, Z_j\}$. Note that for the case of $s_j = s_0$, possible improvement is $\frac{C}{h}$.
- *Possible improvement by a new link.* For a new link, we can apply the algorithm in Fig. 3 to determine a suitable time slot. Once a new time slot is found, possible improvement is again $\min\{\frac{C}{h}, Z_j\}$ if $s_j \neq s_0$ or $\frac{C}{h}$ if $s_j = s_0$.

Based on the above analysis, we can address Problems 1 – 3 by a uniform approach. That is, we check a bottleneck node s_i's neighboring nodes one by one, from the node with the largest residual capacity to the node with the smallest residual capacity. For the checked node s_j, if link (s_i, s_j) already exists, then we consider to assign a new time slot, otherwise we consider to add this link with an assigned time slot. This procedure terminates if an additional time slot is found for an existing link, a time slot is found for a new link, or we cannot improve node s_i's throughput. The pseudo-code to increase bottleneck node s_i's throughput is given in Fig. 4.

D. Complexity

We first show that the complexity in each iteration is polynomial. In the first iteration, we need to identify hop distance for all nodes, select a next hop node for each node and then assign a time slot for this link, and identify initial K.

- To identify hop distance for all nodes, we need to visit all neighboring nodes of each node. The complexity is $O(E) = O(n^2)$, where E is the number of all possible links.
- The complexity to select a next hop node for each node is $O(n)$.
- The complexity to assign a time slot to the first link is $O(1)$. To analyze the complexity to assign a time slot for the $(e+1)$-th link, we assume that previous e links use $\hat{h}$ time slots. When we consider a used time slot t_k for the $(e+1)$-th link, we need to check up to $e_k + 1$ SINR values, where e_k is the number of links in this time slot. In the worst case, we may check up to $\sum_{k=1}^{\hat{h}}(e_k+1) = e + \hat{h}$ SINR values and then decide to assign a new time slot. The complexity is $O(e + \hat{h} + 1) = O(e + h)$. Thus, the total complexity for time slot assignment is at most $O(1) + \sum_{e=2}^{E} O(e + h) = O(1) + O(E^2) + O(hE) = O(n^4 + hn^2)$.
- The complexity of solving an LP in (7) is $O(N_v^3)$ [10] where N_v is the number of variables in (7). Since $N_v = O(N^2)$, $O(N_v^3) = O(n^6)$.

The overall complexity in the first iteration is $O(n^2)+O(n)+O(n^4 + hn^2) + O(n^6) = O(n^6 + hn^2)$, which is polynomial.

In each of the subsequent iterations, we need to identify bottleneck node, calculate possible improvement of each of its neighbors, sort these neighbors, either assign a new time slot or add a new link, and update K.

- The complexity to identify bottleneck node is $O(E) = O(n^2)$.
- To calculate possible improvement for a neighboring node, we need to solve a maximum flow problem. Some algorithms (e.g., Push-relabel algorithm with FIFO vertex selection rule) for the maximum flow problem have complexity $O(n^3)$. The total complexity for a node s_i's $|T_i|$ neighbors is $|T_i|O(n^3) = O(n^4)$.
- Sorting $|T_i|$ neighbors has a complexity $O(|T_i| \ln |T_i|) = O(n \ln n)$.
- To analyze the complexity of assigning an additional time slot for an existing link, we assume that other $e-1$ links

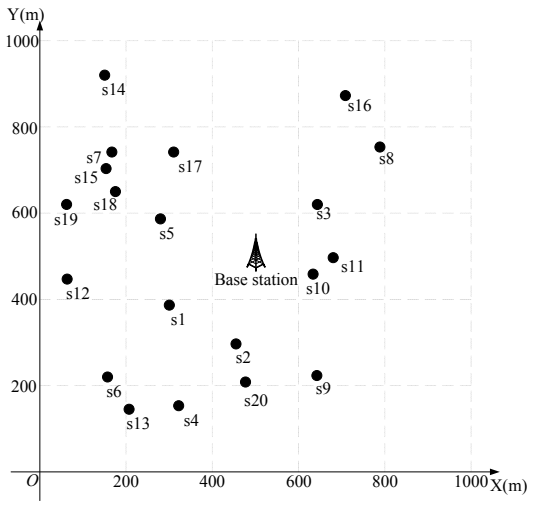

Fig. 5. A 20-node network.

use $\hat{h}$ time slots. We need to check up to e SINR values. In the worst case, we may find that $\hat{h}$ time slots are not available and then assign a new time slot. The complexity is $O(\hat{h}e+1) = O(hn^2)$.

To analyze the complexity of assigning a time slot for a new link, we assume that previous e links use $\hat{h}$ time slots. When we consider a used time slot, we need to check up to $e+1$ SINR values. In the worst case, we may find that $\hat{h}$ time slots are not available and then assign a new time slot. The complexity is $O(\hat{h}(e+1)+1 = O(hn^2)$.

Thus, the total complexity to assign a new time slot or add a new link is $|T_i|O(hn^2) = O(hn^3)$.

- The complexity of solving an LP in (7) is again $O(n^6)$.

The overall complexity in a subsequent iteration is $O(n^2) + O(n^4) + O(n\ln n) + O(hn^3) + O(n^6) = O(n^6 + hn^3)$, which is polynomial.

We then analyze the number of iterations. On one hand, in the first iteration, n $x_{ij}[k]$ values are set as one while in each subsequent iteration, one additional $x_{ij}[k]$ value is set as one. On the other hand, the total number of $x_{ij}[k]$ variables is $O(hE) = O(hn^2)$. Thus, the number of iterations is at most $1 + O(hn^2) - n = O(hn^2)$. The overall complexity of our algorithm is $O(n^6 + hn^2) + (O(hn^2) - 1)O(n^6 + hn^3) = O(hn^8 + h^2n^5)$, which is polynomial. Note that the purpose of this analysis is to show the complexity is polynomial. The obtained result is a loose upper bound on complexity while in practice, the complexity can be much less.

IV. Simulation Results

In this section we give simulation results to show the performance of our algorithm. We also compare results with and without SIC to show the advantage of SIC.

We consider multi-hop networks with 10 to 50 nodes randomly deployed in a square region of 1000×1000m. The base station is deployed at $(500, 500)$. Transmission power is $P = 1$W and noise power is $N_0 = 10^{-10}$W. The SINR threshold is $\beta = 3$. Channel bandwidth is $W = 22$MHz. The number of time slots is equal to the number of nodes.

TABLE I
The coordinates (in m) and the $r(s_i)$ value (in kbps) of each node in the 20-node network.

i	coordinates	$r(s_i)$	i	coordinates	$r(s_i)$
1	(301,394)	38	11	(685,492)	57
2	(453,309)	85	12	(79,445)	51
3	(639,614)	85	13	(203,162)	25
4	(335,181)	14	14	(166,919)	85
5	(278,596)	67	15	(175,713)	27
6	(177,224)	50	16	(719,839)	65
7	(183,737)	56	17	(321,772)	45
8	(790,770)	73	18	(194,643)	23
9	(645,268)	22	19	(73,627)	33
10	(644,450)	30	20	(445,205)	53

The required minimum data rate $r(s_i)$ is between 10kbps and 100kbps.

To perform a fair comparison between the interference avoidance scheme and SIC, we need to set an appropriate R_T value. We can identify R_T by considering the ideal case when there is no interference. Denote d_{ij} as the distance between node s_i and node s_j and g_{ij} as the channel gain between these nodes. We consider a gain model $g = d^{-\lambda}$, where d is the distance between two nodes and $\lambda = 4$ is the path loss index. When node s_i transmits data to s_j with power P, the SNR is $SNR_{ij} = g_{ij}P/N_0 = d_{ij}^{-\lambda}P/N_0 \geq \beta$. For the extreme case, we have $R_T^{-\lambda}P/N_0 = \beta$, i.e., $R_T = \left(\frac{P}{N_0\beta}\right)^{1/\lambda} \approx 240$m. The interference range R_I should be larger than R_T. We set $R_I = 320$m.

We first present detailed results of a multi-hop network with 20 nodes in Section IV-A. Then we provide complete results for all network instances with different number of nodes.

A. Results for a Multi-hop Network with 20 nodes

Consider a multi-hop network shown in Fig. 5. The coordinates and the required minimum data rate $r(s_i)$ are shown in Table I.

Under the interference avoidance scheme, we have $K = 15.6$ by (5). With SIC, we have $K = 63.9$ by our algorithm. Clearly, we can achieve a much better solution by using SIC.

The routing topologies under interference avoidance and SIC are given in Fig. 6 and Fig. 7, respectively. The numbers on each link represent the assigned time slots to this link. From which we can see that with SIC, links can be active in more time slots. For example, the link between nodes s_{14} and s_7 is active in only one time slot under interference avoidance while this link can be active in three time slots under SIC. As a result, throughput can be increased by using SIC.

B. Results for All Network Instances

We change the number of nodes n from 10 to 50, and generate 20 different network instances randomly for each n. Then we calculate the value K under the interference avoidance scheme and the SIC scheme, and show the average values for each n in Table II. All results can be calculated in seconds when using our algorithm. It can be seen that by using SIC, we can achieve about 300% improvement on throughput for multi-hop networks with various number of nodes.

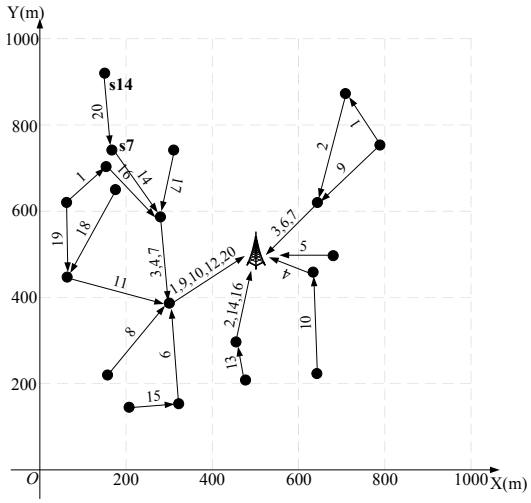

Fig. 6. Routing topology under interference avoidance.

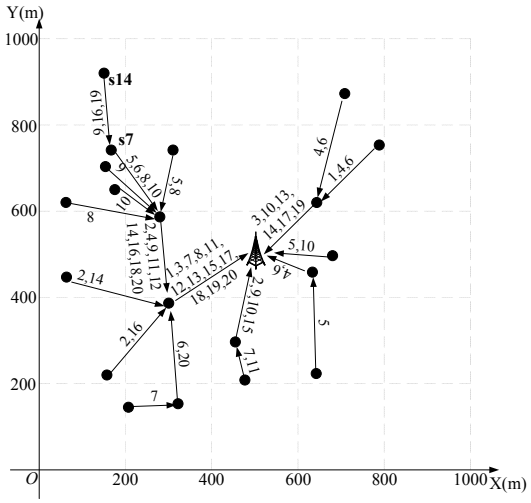

Fig. 7. Routing topology under SIC.

TABLE II
OPTIMIZATION RESULTS FOR MULTI-HOP NETWORKS.

n	Traditional K	SIC K	Improvement
10	37.8	156.7	314.55%
15	29.8	139.1	366.78%
20	23.2	100.0	331.03%
25	20.0	81.1	305.50%
30	16.7	70.8	323.95%
35	15.8	64.6	308.86%
40	14.9	59.2	297.32%
45	12.5	52.1	316.80%
50	11.7	48.3	312.82%

V. CONCLUSION

In this paper, we applied the SIC technique for interference management in multi-hop wireless networks. We considered a throughput maximization problem with a cross-layer design of SIC at the physical layer, time slot assignment at the link layer, and routing at the network layer. We identified that the challenge of this problem is time slot assignment under SIC, which yields a mixed integer linear programming (MILP) formulation with many integer variables. To overcome this challenge, we designed an iterative framework to efficiently determine time slot assignment and solved the remaining problem by a linear programming. We showed that the overall complexity of our algorithm is polynomial. Simulation results showed that throughput of a multi-hop wireless network can be increased by about 300% by using SIC. Given this significant performance improvement, we will further design distributed algorithms in our future work.

ACKNOWLEDGMENTS

This work was supported in part by the Research Fund for the Doctoral Program of Higher Education of China (20090111120004, 20100111110004), the International S&T Cooperation Program of Anhui Province of China (10080703001), and the Natural Science Foundation of Jiang-Su Province of China (BK2011236).

REFERENCES

[1] J.G. Andrews, "Further advancements for E-UTRA physical layer aspects," in *3GPP TR 36.814. v9.0.0.*

[2] S. Verdu, "Multiuser Detection," *Cambridge Univ. Press,* 1998.

[3] R. Bosisio, U. Spagnolini, "Interference Coordination Vs. Interference Randomization in Multicell 3GPP LTE System," in *Proc. IEEE WCNC,* pp. 824–829, Las Vegas, NV, March 31–April 3, 2008.

[4] J.G. Andrews, "Interference cancellation for cellular systems: A contemporary overview," in *IEEE Wireless Commun. Magazine,* pp. 19–29, April 2005.

[5] P. Patel, J. Holtzman, "Performance comparison of a DS/CDMA system using a successive interference cancellation (IC) scheme and a parallel IC scheme under fading," in *IEEE International Conference on Serving Humanity Through Communications,* pp. 510–514, New Orleans, LA, May 1–5, 1994.

[6] X. Wang, H.V. Poor, "Iterative (Turbo) soft interference cancellation and decoding for coded CDMA," in *IEEE Trans. on Communication,* vol. 47, no. 7, pp. 1046–1061, July 1999.

[7] J. Blomer and N. Jindal, "Transmission capacity of wireless ad hoc networks: Successive interference cancellation vs. joint detection," in *Proc. IEEE ICC,* 5 pages, Dresden, Germany, June 14–18, 2009.

[8] E. Gelal, K. Pelechrinis, T.S. Kim, I. Broustis, S.V. Krishnamurthy, and B. Rao, "Topology control for effective interference cancellation in multi-user MIMO networks," in *Proc. IEEE INFOCOM,* 9 pages, San Diego, CA, March 15–19, 2010.

[9] D. Halperin, T. Anderson, and D. Wetherall, "Taking the sting out of carrier sense: Interference cancellation for wireless LANs," in *Proc. ACM MobiCom,* pp. 339–350, San Francisco, CA, Sept. 14–19, 2008.

[10] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness, W.H. Freeman and Company,* New York, NY, 1979.

[11] IBM ILOG CPLEX Optimizer, http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.

[12] M.S. Bazaraa, J.J. Jarvis, and H.D. Sherali, *Linear Programming and Network Flows*, 4th edition, New York, NY, John Wiley & Sons Inc., 2010.

[13] S. Lv, X. Wang, and X. Zhou, "Scheduling under SINR model in ad hoc networks with successive interference cancellation," in *Proc. IEEE GLOBECOM,* 5 pages, Miami, FL, Dec. 6–10, 2010.

[14] S. Weber, J.G. Andrews, X. Yang, and G. de Veciana, "Transmission capacity of wireless ad hoc networks with successive interference cancellation," *IEEE Trans. on Information Theory,* vol. 53, no. 8, pp. 2799–2814, Aug. 2007.

[15] M.T. Thai and P. Pardalos, "The Handbook of Optimization in Complex Networks: Theory and Applications," *Springer Publisher,* 2011.

[16] C. Jiang, Y. Shi, Y.T. Hou, S. Kompella, S.F. Midkiff, "Squeezing the most out of interference: An optimization framework for joint interference exploitation and avoidance," in *Proc. IEEE INFOCOM,* pp. 424–432, Orlando, FL, March 25–30, 2012.