Fault-Tolerant Dual Power Management in Wireless Sensor Networks

Chen Wang∗ Myung-Ah Park† James Willson‡
Andras Farago‡ Ding-Zhu Du‡

Abstract—How to adjust the transmission power at each node to achieve global energy efficiency while maintaining the network connectivity, referred as power management problem, is the major target of various topology control technologies. Moreover, fault tolerance which is often modeled as 2-edge or 2-vertex connectivity is another desired feature in many applications. In this paper, we study the fault tolerant dual power assignment problem. With the assumption of dual universal transmission power levels, we aim to minimize the total number of nodes assigned to high power level such that the resultant network topology is 2-edge or 2-vertex connected. As the problems are NP-hard, we design a novel algorithm to compute nearly-optimal solutions. From the theoretical perspective, we prove that our algorithm can guarantee 3.67-approximation for both 2-edge connectivity and 2-vertex connectivity, which improves the existing best approximation algorithm. We also conduct some numerical experiments which show that results of our algorithm are at most 2 times of optimal solutions in average and have significant improvements compared to that of existing algorithm.

I. INTRODUCTION

Wireless sensor networks (WSN) have a broad range of applications such as environmental monitoring, military operation and health applications and therefore obtain more and more attentions in past decades. Especially, new advances of MEMS technology enable the development of micro sensors with low-power radios and multi-functions in a low-cost way and broaden the potential WSN applications.

In many applications, e.g. a battlefield sensor network, sensor nodes need to be deployed in some unmanned area. In such scenarios, environmental conditions are unpredicted. Failures of wireless connections and sensor nodes occur from time to time. In order to supply reliable QoS, node-redundancy as well as fault-tolerant internetworking technologies are desired as extra requirements of energy saving technologies.

Fault tolerance of a network is often modeled as k-edge (k-vertex, resp.) connectivity. A k-edge (k-vertex, resp.) connected network has the property that, under up to k − 1 simultaneous edge (vertex, resp.) failures, the network still keeps connected.

Topology control is among the various approaches to achieve the conflicting goals of saving energy while maintaining fault tolerance. The basic problem of topology control, namely power assignment problem, is to adjust transmission power at each node to minimize the global power cost, subject to desired network features such as connectivity, k-connectivity or low interference among neighbors.

In most used radio propagation models, required transmission power from a node u to another node v is proportionate to \(d(u, v)^{\alpha}\), where \(d(u, v)\) is the Euclidean distance between u and v and \(c\) is called the power attenuation exponent, typically between 2 and 4, and depends on environmental conditions. For the sake of simplification, we assume a uniform environment which implies that each node can realize the same effective transmission range with the same transmission power. In other words, given the attenuation exponent, a power assignment is equivalent to a transmission range assignment.

A topology or communication graph induced by a power adjustment is usually a directed graph due to the asymmetricity of transmission powers among neighbor nodes. However, it has been shown in the literature that a unidirectional link is detrimental to the performance of a network. So, it is profitable to symmetrize edges among neighbor nodes. In our paper, we assume that there exists a (symmetric) link between nodes u and v if \(d(u, v) \leq r_u\) and \(d(u, v) \leq r_v\) where \(r_u\) and \(r_v\) are the transmission radius of u and v, respectively. In other words, we prune all unidirectional links induced by a power assignment to obtain a symmetric topology.

Extensive researches have been done on topology control in literature. Most of these optimization problems are proved NP-hard and therefore nearly optimal algorithms are proposed under different models. Among these works, they often assume a continuous power assignment: a node can set its transmission range to \(r\) where \(r\) can be any value in \([r_{\text{min}}, r_{\text{max}}]\). And the optimization object is the total power cost over all nodes.

In reality, however, only discrete power levels are available, i.e., each sensor chooses one from \(t\) different transmission powers where \(t\) is a small constant. As a fundamental case, we consider the dual power assignment (\(t = 2\), i.e., each node can use either a high power (HP) or a low power (LP) to achieve radio transmission. Note that this actually imply a dual transmission range assignment according to our radio propagation model, and we use \(r_h\) (\(r_l\), resp.) to denote the high (low resp.) power transmission radius in context. In this scenario, the total power cost is minimal when the number of high power nodes is minimized. However, their optimization objectives are different from each other, if we focus on an approximation algorithm. The approximation of latter problem is more difficult in terms of relative ratio of an
approximation outcome to optimal solution. Note that an \( \alpha \)-ratio approximation of minimum number of high power costs less than \( \alpha \) times optimal total power.

For a given set of nodes \( V \), a dual power assignment is defined as a range assignment \( A \) where \( A(v) = r_h \) or \( A(v) = r_l \), respectively. The low-power graph is defined as the graph induced by the low-power assignment where \( A(v) = r_l \) for all \( v \in V \). On the other hand, the high-power graph is defined as the graph induced by the high-power assignment where \( A(v) = r_h \) for all \( v \in V \).

In addition, less number of HP nodes is also important for reducing communication interferences. \cite{14} and \cite{15} showed that low-power nodes suffer from high-power nodes in terms of successful data transmission, especially when a 802.11-like MAC protocol is used. By decide a high-interfering power threshold according to applications, we can actually transform this problem as a dual power problem if we can generalize dual power assignment to a graph theoretical definition without metric constraint.

**Definition 1:** Given an graph \( G_h = (V, E_h) \) and its spanning subgraph \( G_i = (V, E_i) \) where \( E_i \subseteq E_h \), we define dual power assignment problem (DPA) to find an edge set \( E \subseteq E_h \setminus E_i \), such that \( G = (V, E_i \cup E) \) satisfies some required property \(^1\), while the size of \( V(E) \) is minimized, where \( V(E) \) denotes the set of all vertices which have at least one edge in \( E \).

By definition, we can have different versions of dual power assignment problem if different network properties are required. In this paper, we study the dual power assignment problem for 2-edge connectivity (DPA-2EC) and 2-vertex connectivity (DPA-2VC). One can prove that both of them are NP-hard by using reduction from the minimum set cover problem which implies exact algorithms with polynomial time complexity can not be expected.

The remainder of this paper is organized as follows. In Section II, we investigate previous works related to our paper. In Section III, some preliminaries about 2-edge (2-vertex) connectivity are supplied. Then we present our algorithm to solve the DPA-2EC problem with approximation ratio of 3.67 in Section IV and use the similar idea to develop an algorithm to solve DPA-2VC problem in Section V. We also show that it computes a 3.67 approximation for DPA-2VC. To evaluate real performance of our algorithms, experimental results and some numerical analysis are given in Section VI.

## II. RELATED WORKS

The power-optimal continuous range assignment problem has been studied extensively. As this problem is NP-hard even in the Euclidean plane \cite{11}, some approximation algorithms are developed. Kirousis et al. \cite{16} give a 2-approximation algorithm by constructing the minimum spanning tree of the network graph. Calinescu et al. \cite{3} improve the approximation factor to 1.69 with a steiner tree based algorithm. As a natural generalization of connectivity, \( k \)-connectivity or \( k \)-edge connectivity power range assignment problem is also studied in previous works. For the special case of biconnectivity, Lloyd et al. \cite{9} present a 8-approximation algorithm. For general \( k \)-connectivity and \( k \)-edge connectivity problem, an \( O(k) \)-approximation centralized algorithm is designed by Hajiaghayi et al. \cite{5}. Calinescu et al. \cite{3} improve the approximation result for \( k \)-edge connectivity to 2\( k \), and for biconnectivity to 4.

As our best known, the dual power assignment problem is first studied by Rong et al. \cite{4}. The authors define the asymmetric version of the problem, prove the NP-hardness of strongly connected dual power assignment problem, and design a 2-approximation algorithm based on graph theoretic facts. Chen et al. \cite{2} improved the approximation factor of asymmetric version to 1.75. In \cite{1}, Lloyd and Liu study the minimum number of maximum power users problem, which is actually the symmetric version DPA problem, prove the NP-hardness of the symmetric version and present a 1.67-approximation algorithm.

The first work with fault tolerance taken in account is \cite{12}. The authors proposed an algorithm for 2-edge connectivity and 2-vertex connectivity with dual power assignment. The algorithms are quite simple and achieve approximation ratio of 6 for 2-edge connectivity and 5 for 2-vertex connectivity respectively. In \cite{13}, another approximation algorithm is designed to achieve an improved approximation of 4 for both 2-edge connectivity and 2-vertex connectivity. In the algorithm, they assign a priority for each high-power edge according to its contribution to biconnectivity, and iteratively add high power nodes.

## III. PRELIMINARIES

In this section, we give the definition of 2-edge(\( k \)-vertex) connectivity and list some lemmas which describe some basic properties of 2-edge(\( k \)-vertex) connectivity and will be used for our algorithms. Some of them can be found in textbook of graph theory, e.g., \cite{17} and others can be proved without difficulties. We just list them here without proof due to the page limit.

**Definition 2:** An undirected graph \( G \) is called 2-edge (2-vertex, resp.) connected if, \( G \) remains connected after removing any one edge (vertex, resp.).

The following lemmas give some characteristics of 2-edge(\( k \)-vertex) connectivity. We will use them to prove the correctness as well as an approximation ratio of our algorithms in Section IV and Section V.

**Lemma 1:** Given a 2-edge connected graph \( G = (V,E) \) and a non-trivial subset of nodes, \( U \), there are at least two edges between \( U \) and \( V \setminus U \).

**Lemma 2:** Given a 2-vertex connected graph \( G = (V,E) \) and a non-trivial subset of nodes, \( U \), where \(|U| > 1\), there are at least two nodes in \( U \) connecting to \( V \setminus U \).

**Lemma 3:** Given a connected graph \( G = (V,E) \) and a spanning tree \( T = (V,E_T) \) of \( G \), \( G \) is 2-edge connected iff for each edge \( e \in E_T \) of \( T \), there is an edge \( e' \in E \), such that the new graph \( (V,E_T \setminus \{e\} \cup \{e'\}) \) is still connected.

---

\(^1\)Since we consider only symmetrical links, \( G_h, G_i \) and \( G \) are undirected graphs.

\(^2\)\( G_h \) is supposed to satisfy the property.
Lemma 4: Given a connected graph \( G = (V,E) \) and a spanning tree \( T = (V,E_T) \) of \( G \), \( G \) is 2-vertex connected iff for each vertex \( v \in V \) of \( T \), there are an edge set \( E' \in E - E_T \), such that the new graph \( (V,E_T - E_T(v) + E') \) is still connected, where \( E_T(v) \) is the set of all edges in \( E_T \) attached to \( v \).

Our algorithm is motivated by Lemma 3 and 4. The basic idea is to construct a spanning tree of \( G_h \) and then add more high power nodes to make it 2-edge or 2-vertex connected.

In the first step, a straightforward method is to compute a minimum weighted spanning tree \( T \) based on the following weight assignments: for edge \( e \in E_h \), \( w(e) = 1 \) if \( e \) is a high power edge, and otherwise set \( w(e) = 0 \). If we denote \( U_1 \) as the number of high power nodes used to construct \( T \), we can see
\[
w(T) = \sum_{e \in T} w(e) = \frac{|U_1|}{2}.
\]
Moreover, if we assume that \( G_t \) has \( c \) different connected components, we know\[
w(T) = c - 1.
\]

Let us use \( opt_e \) and \( opt_v \) to denote the optimal solutions of DPA-2EC and DPA-2VC, respectively. According to Lemma 1, it can be shown that \( c \leq opt_e \), and \( c \leq opt_v \). Consequently, we know \( U_1 \leq 2 \cdot opt_e \) and \( U_1 \leq 2 \cdot opt_v \). In fact, this 2-ratio can be improved to 1.67 by using the greedy algorithm in [1].

IV. APPROXIMATION ALGORITHM FOR DPA-2EC

In this section, we propose an algorithm called Candidate Sets Filtering Algorithm (CSFA) to solve the DPA-2EC problem.

The algorithm includes two steps. In the first step, we construct a rooted spanning tree \( T \) of the network by using the pervious algorithm in [1]. In the second step, for each edge in the tree, we will check if there is another low power edge connecting the two separated parts formed by removing the edge from \( T \), otherwise we use a high power edge to connect them. It is sufficient to guarantee the 2-edge connectivity of the resultant graph according to Lemma 3.

Before we give more details, we need introduce some basic terminologies and notations.

- The rooted spanning tree constructed in step 1 is denoted by \( T \). The root is denoted by \( r \).
- The parent of a node \( v \) is denoted by \( p(v) \);
- A tree edge is an edge in the rooted tree \( T \).
- A GREEN edge is an edge which is in \( E_t \) but not in \( T \), i.e., a low-power non-tree edge;
- A RED edge is an edge which is in \( E_h \setminus E_t \) but not in \( T \), i.e., a high-power non-tree edge;
- The rooted subtree of a node \( v \), denoted by \( T_v \), is the subtree of \( T \) rooted at \( v \).
- For a node \( v \) other than the root, the inside part of \( v \) is defined as \( T_v \) and the outside part of \( v \) is defined as \( T - T_v \).
- A node \( v \) other than \( r \) is covered by a GREEN or RED edge \( e \) if \( e \) connects the insider part and outside part of \( v \).

- The candidate set of a node \( v \), denoted by \( CS(v) \) is a set of RED edges which is created during the algorithm runs.

Given a rooted tree, the second step computes an 2-edge connected graph. As initialization, for each node other than \( r \), we mark it WHITE and create an empty candidate set. Then the algorithm explores nodes bottom-up along \( T \). For each leaf \( v \), if it has another low-power edge other than \( (v,p(v)) \), i.e., a GREEN edge, we mark \( v \) BLACK to indicates this node is already explored. Otherwise, we will fulfill its candidate set with all of its RED edges and then mark it BLACK.

For each non-leaf WHITE node \( v \), it is explored after all of its children are already BLACK. In this iteration, our purpose is to check if there is either a GREEN edge covering \( v \). If such a GREEN edge exists, we do nothing but color \( v \) BLACK. Otherwise, we check whether there is some edge in any inside candidate set covering \( v \) for the candidate set of each node in \( T_v \), if it has at least one edge covering \( v \), we “filter” this candidate set by removing any other edges which is an edge inside \( T_v \). If neither such a GREEN edge nor a candidate set exists, we will fulfill \( CS(v) \) with all of RED edges covering \( v \). Eventually, we mark \( v \) BLACK to indicate it has been processed.

The algorithm runs iteratively on WHITE nodes and the number of WHITE nodes decreases by one in a single iteration. So our algorithm always terminates with an output.

The correctness of the algorithm is implied by the following lemma.

Lemma 5: CSFA outputs a power assignment which results in a 2-edge connected graph.

Proof: It is easy to see that the resulting graph of output high power nodes has all edges in \( T \) and exact one edge in each non-empty candidate set.

Since \( T \) is connected, according to Lemma 3, we only need to prove that for each tree edge \( (v,p(v)) \), there is at least another edge in the resulting graph covering \( v \). As we known, if \( CS(v) \) is non-empty, all the edges in \( CS(v) \) can cover \( v \). Note that \( CS(v) \) keeps non-empty even though some of its edges may be removed. On the other hand, if \( CS(v) \) is eventually empty, it means that there is either a GREEN edge or an edge in some existing candidate set \( CS(v') \) covering \( (v,p(v)) \). Moreover, if the latter case happens, we will filter \( CS(v') \) and remove all edges in it which does not cover \( v \). So this guarantees that there is eventually either a GREEN edge covering \( v \) or a candidate set \( CS(v') \) where all edges in it covering \( v \). So the lemma holds.

In addition, we can show that the CSFA has a 3.67-approximation ratio. This result is proved in the following theorem.


Proof: The output set of high power nodes consists of two parts, \( U_1 \) and \( U_2 \). The approximation result from [1] directly implies that \( |U_1| \leq 1.67 \cdot opt_e \). It is sufficient to prove \( |U_2| \leq 2 \cdot opt_v \). This is done based on Lemma 1.

First of all, as we arbitrarily choose an edge from non-empty
Algorithm 1 Candidate Sets Filtering Algorithm for DPA-2EC
1: INPUT: \( (V, G_h, G_t) \)
2: OUTPUT: Power assignment such that the resulting symmetric graph is 2-edge connected.

Step 1.
3: Construct a spanning tree \( T \) of \( G_h \) rooted at \( r \) by the algorithm in [1], and assume the set of used high power nodes is \( U_1 \).

Step 2.
4: for each node \( v \) except \( r \) do
5: \( \text{Mark } v \text{ WHITE; } \)
6: \( CS(v) \leftarrow \emptyset; /* \text{Initialize all candidate sets */} \)
7: end for
8: while there exists at least a WHITE node do
9: \( v \) is processed, it will not has attached edges added when \( v \) is processed. This follows two observations: 1. a non-empty candidate set \( CS(v) \) is created only if there is no GREEN edge or other edges in some existing candidate set to cover the underlying edge (between \( v \) and its parent); 2. assume that \( v' \) in \( T_v \) has an attached edge added into \( CS(v) \) when \( v \) is processed, it will not has attached edges added when any ancestors of \( v \) is processed.

The first observation ensures that when a non-empty candidate set \( CS(v) \) is created, there is at least one high power node inside \( T_v \) which is also in \( OPT \) according to Lemma 1. In addition, such a high power node in \( OPT \) has at least an attached edge added into \( CS(v) \) when \( v \) is processed.

The second observation can be shown by contradiction. That is, if \( v' \) has at least an edge in \( CS(v) \) and also an edge in \( CS(u') \) which \( u' \) is some ancestor of \( v', \) then let us consider the iteration of processing \( u' \). As \( CS(u') \) is non-empty, it means \( CS(v) \) has no edge to cover \( u' \). However, if \( v' \) has an attached edge in \( CS(u') \), this edge should also be added into \( CS(v) \) and has not been removed when \( CS(u') \) is fulfilled. This contradicts that \( CS(v) \) has no edge to cover \( u' \).

At last, we define the candidate node set of \( v \) is the set of all nodes in \( T_v \) which has at least one attached edge added into \( CS(v) \) in the iteration when \( v \) is processed. From two observations, we actually prove the optimal solution \( OPT \) has at least one high power node for each non-empty candidate set node set and two candidate node set are disjoint. So we see

\[
|U_2| \leq 2 \cdot |\{\text{non-empty candidate sets}\}| \quad (1)
\]

\[= 2 \cdot |\{\text{non-empty candidate node set}\}| \quad (2)
\]

\[\leq 2 \cdot opt_e. \quad (3)
\]

From all above, we can conclude our algorithm computes a 3.67-approximation. \( \blacksquare \)

V. APPROXIMATION ALGORITHM FOR DPA-2VC

In this section, we modify the Candidate Set Filtering Algorithm to solve the DPA-2VC problem. To avoid confusion, we will use CSFA-2VC to denote the modified algorithm. All terminologies of CSFA except the following one can be applied immediately:

- A node \( v \) other than \( r \) is covered by a GREEN or RED edge \( e \) if \( e \) connects \( T \setminus T_v \) and \( T_v \), and \( p(v) \) is not attached by \( e \).

CSFA-2VC also starts with a rooted spanning tree by the algorithm in [1]. However, the second step need to be modified. Roughly speaking, according to Lemma 4, we need to consider each node of the tree in a bottom-up order. After check if the removal of current node disconnects the graph by looking up existing non-empty candidate sets, we update candidate sets accordingly.

An observation is, for a node \( v \) with at most 2 children, if there is at least one edges covering \( v \) and at least one covering each of its children respectively, the removal of \( v \) cannot disconnected the graph. This will be proved in Lemma 6. Based on the observation, we locally transform a node with more than 2 children to a binary tree, and use the similar process on this subtree as that of CSFA.

In concrete, when we process a node \( v \) with \( k \) children \( w_1, w_2, ..., w_k \), the subroutine splits \( v \) into \( k - 1 \) nodes \( v_1, v_2, ..., v_{k-1} \), and construct a binary tree consisting of \( v_1, v_2, ..., v_{k-1}, T_{w_1}, T_{w_2}, ..., T_{w_{k-1}}, \) and \( T \setminus T_v \). This operation is illustrated in Figure 1. Note that we require that each \( v_i \) has at most two children regardless of other parts of the tree in this iteration. We call the graph obtained from \( T \) by splitting \( v \) the local binary tree of \( v \), denoted by \( BT_v \). A subtree of \( BT_v \) rooted at node \( u \) is denoted by \( BT_u(u) \).

After we got \( BT_v \), we consider each \( v_i \) as a real WHITE node and check if there is either a GREEN edge or a candidate

candidate set and add the two endpoints into \( U_2 \),

\[|U_2| \leq 2 \cdot |\{\text{non-empty candidate sets}\}|. \]

So it is sufficient to prove that the number of non-empty candidate sets is at most \( opt_e \). This follows two observations:

1. a non-empty candidate set \( CS(v) \) is created only if there is no GREEN edge or other edges in some existing candidate set to cover the underlying edge (between \( v \) and its parent); 2. assume that \( v' \) in \( T_v \) has an attached edge added into \( CS(v) \) when \( v \) is processed, it will not has attached edges added when any ancestors of \( v \) is processed.
set which has a edge covering \( v_i \). If so, we filter all the candidate sets of his children in the same way as CSFA. Otherwise, we fulfill \( CS(v_i) \) with all edges from \( BT_u(v_i) \setminus \{v_i\} \) to any of \( v_i \)'s ancestors. At the end of the iteration when \( v \) is processed, we mark \( v \) BLACK in \( T \).

Note that in CSFA, if node \( v \) is split into \( k \) nodes in its iteration, all \( CS(v_k) \) are regarded as independent candidate sets in following iterations where \( v \)'s ancestor is processed. Eventually, after all WHITE nodes are done, we choose an edge from each candidate set and output the set of high power nodes accordingly.

The correctness of CSFA-2VC is proved in the following lemma.

**Lemma 6:** The CSFA-2VC outputs a power assignment which results in a 2-vertex connected graph.

**Proof:** According to Lemma 4, we need only check that, for each node \( v \) in \( T \), all subtrees rooted at \( v \)'s children and \( T \setminus T_v \) are connected. We prove this by induction.

As induction basis, a single leaf apparently satisfies this property. Then, we assume that all children of a node \( v \) have satisfied this property. Here we need consider two cases. In the first case that \( v \) has at most 2 children, say \( u_1 \) and \( u_2 \). It implies the removal of \( v \) divides \( T \) into 3 parts: \( T_{u_1}, T_{u_2} \) and \( T \setminus T_v \). According to induction hypothesis, \( T_{u_1} \) is connected to either \( T_{u_2} \) or \( T \setminus T_v \), and \( T_{u_2} \) is connected to either \( T_{u_1} \) or \( T \setminus T_v \) by some candidate sets. Moreover, we guarantee that \( T_{u_1} \cup T_{u_2} \) are connected to \( T \setminus T_v \) if \( v \) is covered by some GREEN edge or candidate set. Therefore, all of \( T_{u_1}, T_{u_2} \) and \( T \setminus T_v \) are interconnected after the iteration on \( v \).

If the second case happens, i.e., \( v \) has more than 2 children. Since we omit all edges attached to \( v \) in this iteration, it is easy to see that \( BT_v \) is connected even though all of \( v_i \)'s are removed. So we can conclude the correctness of CSFA-2VC.

We have mentioned that CSFA-2VC can achieve 3.67-approximation for DPA-2VC. The approximation ratio is concluded as follows.

**Theorem 2:** The CSFA-2VC computes a 3.67-ratio approximation of DPA-2VC problem.

**Proof:** (sketch) The idea is almost the same as the proof of Theorem 1. The key is to prove \( |U_2| \leq 2opt_v \). It follows: 1. each candidate node set has at least one node appeared in an optimal solution; 2. two candidate node sets are disjoint. One can check all arguments for CSFA still hold for CSFA-2VC. We omit details here due to page limit.

### VI. NUMERICAL RESULTS

We evaluate the performance of CSFA and CSFA-2VC through numerical simulations in this section. As a comparison, we also simulation the PESA Algorithm proposed in [13]. Algorithms and experiments are implemented in Matlab.

We consider sensor networks in Euclidean plane. Each sensor is uniformly distributed in a \( 500m \times 500m \) area. We investigate the results under different network densities by varying the number of nodes \( N \) from 50 to 200 with the increment of 50. For a given deployment of a sensor nodes, the high(low) power transmission range \( r_h(r_l) \) need to be preset. According to our requirement of 2-edge and 2-vertex connectivity, a good choice for \( r_h \) would be the smallest transmission range which makes the network 2-edge(vertex) connected. By using binary search strategy, we iteratively compute \( r_h \) until there is at most 5m larger than minimum \( r_h \). Then we compute the value of \( r_l \) by varying the ratio \( \rho \) where \( \rho = r_l/r_h \) from 0.1 to 0.9 with the increment of 0.05. We repeat experiments of each pair of \( N \) and \( \rho \) for 50 times and compute the average.

As mentioned in Section IV and V, the number of non-empty candidate sets is a lower bound of the optimal solution. Therefore, we also compute the lower bound for each network setting as a comparison.

![Figure 2](image.png)

**Fig. 2:** Number of High-Power Nodes for DPA-2EC

Figure 2(a)-(d) illustrate the performance of CSFA algorithm. The compared algorithm is labeled as PESA in legends. Apparently, the lower ratio \( \rho \) is, the more high-power nodes we need, which is supported by experimental results. It is also shown that the solution of our algorithm has less high power nodes than that of the compared algorithm. The improvement is most significant when \( \rho \) is in the middle range.

In addition, when we compare the output of CSFA and the lower bound, we can find that CSFA are nearly optimal in terms of the ratio of outputs to lower bounds, which is even
better than our theoretical expectation. The relative ratio is no more than 2 and is even less while $\rho$ is not too large. This may imply a possibly better approximation ratio.

One can also observe that absolute differences from optimum are very close to 0 while $\rho$ approaching 0 or 1. This is because in the first case, almost every node must be assigned high power in order to achieve 2-edge connectivity, and only few nodes need to be assigned high power if $\rho$ is close to 1.

For DPA-2VC problem, we illustrate results for CSFA-2VC algorithm. The results have very similar behaviors compared with CSFA algorithm.

![Fig. 3: Number of High-Power Nodes for DPA-2VC](image)

**REFERENCES**


