# We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

**4,800**
Open access books available

**122,000**
International authors and editors

**135M**
Downloads

**154**
Countries delivered to

Our authors are among the

**TOP 1%**
most cited scientists

**12.2%**
Contributors from top 500 universities

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

## Interested in publishing with us? Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Chapter 3

# Graph-Based Routing, Broadcasting and Organizing Algorithms for Ad-Hoc Networks

Li Liu, Xianyue Li, Jiong Jin, Zigang Huang,
Ming Liu and Marimuthu Palaniswami

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/54146

## 1. Introduction

The development of networks of low-cost, low-power, multi-functional devices has received increasing attention over the last ten years. These devices are small in size and able to process data, communicate with each other, typically over a radio channel, and even sense. Each device particpates in a self-configuring infrastureless network connected by wireless, called ad-hoc network. Since most of the individual node in ad-hoc networks is inherently resource constrained: limited processing speed, storage capacity, and communication range and energy, it is impossible to achieve application requirements by individual device or unattached devices. A number of devices within a network have to combine as an aggregate collaborating to achieve application requirements. However, such massive devices cooperation must be achieved by the necessary organizational structures without requiring human intervention.

An ad-hoc network is able to arrange itself to achieve the application requirements according to the present situations. Hence, wireless communication has to be the primary means to enable information exchange among these devices. In a wired network like the Internet, each router connects to a specific set of other routers, forming a routing graph. In ad-hoc networks, each device has a radio that provides a set of communication links to nearby devices. Multi-hop communication is expected to overcome some of the signal propagation effects experienced in long distance wireless communication.

In a wide array of disciplines, an ad-hoc network can be intuitively casted into the format of a graph which is a set of vertex and a set of edges that might connect pairs of the nodes. The ad-hoc network consists of devices (vertex or nodes) and the communication links (edges) between them. Graphs are seemingly ubiquitous in ad-hoc network field. The problems of

designing multi-hop routing, broadcasting and organization algorithms for ad-hoc networks have received great considerable attention [22][23][24][25][26][27]. All are tightly coupled to the problem of the distinguished graphs. In this chapter, we discuss the routing, broadcasting and organization algorithms and protocols that can be formulated by three types of graph.

- Connected Dominating Set: Connected dominating set is useful in the computation of routing, broadcasting and organization for mobile ad-hoc networks. In mobile ad-hoc networks, each device is free to move independently, and therefore change its communications links to other devices frequently. A small connected dominating set is used as a backbone for communications. Nodes that are not in the connected dominating set communicate by passing messages through neighbors that are in this set.
- Disjoint Sets: Disjoint sets are used in the implementation of energy efficent routing and organization, e.g., scheduling nodesą̋ status between running and sleeping, as well as in the aspect of fault tolerant routing. In mobile ad-hoc networks, several disjoint sets every pair of which have no nodes in common can gurantee multiple choices of message passing paths and nodes organization.
- Minimum Spanning Subgraph and Steiner Minimum Tree: Mininum spanning subgraph and Steiner minimum tree represent a spanning subgraph or a tree with the lowest total costs. The generation of subgraphs and Steiner trees has applications in mobile ad-hoc routing and organization design. Several varieties of the minimum spanning subgraph problem and steiner tree problem are proposed for the sake of describing the issues on the fault tolerant, topology control and constrained routing protocol design in mobile ad-hoc networks.

Most of these problem are either *NP-hard*. Several approximate and near-approximate algorithms are proposed to solve these issues based on the combinatorial optimization and graph theory. In real mobile ad-hoc networks, there are some restricted conditions to be achieved in various applications, which will make the problems more difficult to solve. In this chapter, we attempt to give a preliminary review of the design and implementation of the heursitic or approxiamte algorithms on routing, broadcasting and organization by using the combinatorial optimization and graph theory. Note that we only focused on the three problems, all of which were our previous research works. The interested reader is also referred to some excellent works on other topics of combinatorial optimization and graph theory[20][21].

The organization of this chapter is as follows. In Section 2, we give some basic definitions of graph theory that appear in ad-hoc network formulation. We also give the notations used throughout this chapter. In Section 3, we provide the main ideas and approaches of formulating the ad-hoc network issues into several versions of connected dominating set problems. We present our previous research works of proposed algorithms and results related to graph theory. Then in Section 4, we consider the ad-hoc network issues which can be formulated to find disjoint sets. We also present a method of converting disjoint sets issues to network flow and combinatorial optimization problems. In Section 5, we present the minimum spanning subgraph and minimum steiner tree problem applied in fault-tolerant algorithm design. Finally, Section 6 concludes this chapter.

http://dx.doi.org/10.5772/54146

## 2. Basic definitions and notations

An ad-hoc network topology could be represented by a graph $G$ that is an ordered triple $(V(G), E(G); C(G))$ or $G(V, E; C)$ consisting of a nonempty set $V$ of vertices $v_1, v_2, ..., v_n$, and a set $E$ of edges, and $C$ is the set of weights on the nodes or the edges. Generally, an edge denotes that the two nodes belong to it can communicate. Therefore,$E = \{(v_i, v_j) : dist(v_i, v_j) \leq r_i\}$, where $dist$ is the Euclidean distance function and $r_i$ represents the transmission range of node $i$. The edge $(v_i, v_j)$ denotes $v_i$ is able to communicate with $v_j$. An unweighted graph $G$ is also presented as $(V(G), E(G))$ or $G(V, E)$.

The graph $G$ could be a directed graph if the network is heterogeneous that nodes have various transmission ranges, or be an undirected graph if any two nodes can communicate with each other where an edge $(v_i, v_j)$ indicates that there must be an edge $(v_i, v_j)$ existing in $E$. The weight on a node or an edge could denote the metrics of the network. In a power aware application, the weight might be the remaining power of a node. It might be a vector containing transmission speed and power consumption on an edge for the application that aims to find an energy-efficient delay-constraint routing path. In some applications, the weights on all nodes or edges are the same, e.g, in a fault tolerant network that needs achieve no requirement except finding routing paths between two nodes.

A graph $H$ is a subgraph of $G$ (written $H \subseteq G$) if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. An induced subgraph of $G, G[V']$, contains a vertex set $V'$, where $V'$ is a nonempty subset of $V(G)$, and an edge set $E'$, where $E' \subseteq E(G)$ that have both ends in $V'$. The induced subgraph $G[V(G) \backslash V']$ is denoted by $G - V'$. If $V' = \{v\}$, we write $G - v$. Similarly, an edge-induced subgraph of $G, G[E']$, contains a vertex set $V'$ and an edge set $E'$, where $E'$ is a nonempty subset of $E(G)$ and ends of edges in $E'$ belong to $V'$. The spanning subgraph of $G$ with edge set $E(G) \backslash E'$ is written simply as $G - E'$. The graph obtained from $G$ by adding a set of edges $E'$ is denoted by $G + E'$. If $E' = \{e\}$, we write $G - e$ and $G + e$ instead of $G - \{e\}$ and $G + \{e\}$.

Let $G_1$ and $G_2$ be subgraphs of $G$. $G_1$ and $G_2$ are disjoint if they have no vertex in common, and edge-disjoint if they have no edge in common. The union $G_1 \cup G_2$ of $G_1$ and $G - 2$ is the subgraph with the vertex set $V(G_1) \cup V(G_2)$ and the edge set $E(G_1) \cup E(G_2)$; if $G_1$ and $G_2$ are disjoint, their union can be also denoted by $G_1 + G_2$.

The degree of a vertex $v$ in $G$, $d(v)$, is the number of edges of $G$ incident with $v$. $\delta(G)$ and $\Delta(G)$ represent the minimum and the maximum degrees of vertices of $G$ respectively.

A path in $G$ is a finit non-null sequence $P = v_0 e_1 v_1 e_2 v_2 ... e_k v_k$, whose terms are alternately vertices and edges, such that, for $l \leq i \leq k$, the ends of $e_i$ are $v_{i-1}$ and $v_i$. In addition, the vertices $v_0, v_1, ..., v_k$ are distinct, $P$ is called a path. Usually, denote the section $v_i v_{i+1} v_j$ of the path $P = v_0 v_1 ... v_k$ by $P[v_i, v_j]$. Two vertices $u$ and $v$ of $G$ are connected if there is a $P[u, v]$ in $G$. $u$ and $v$ are directly connected or adjacent if $(u, v) \in E(G)$. Connection is an equivalence relation on the vertex set $V(G)$. Therefore, there is a partition of $V(G)$ into nonempty subsets $V_1, V_2, ..., V_k$ such that every pair of the vertices $u$ and $v$ is connected if and only if both $u$ and $v$ belong to the same set $V_i$. The subgraphs $G[V_1], G[V_2], ..., G[V_k]$ are called the components of $G$. If $G$ has exactly one component, $G$ is connected; otherwise, $G$ is disconnected.

A flow network is a directed graph $G(V, E; C, f, s, t)$, where every edge $(u, v) \in E$ has a non-negative capacity $c(u, v)$, $f$ is a flow function $f : V \times V \rightarrow \Re^+$, $s$ is a source and $t$ is a sink. A flow network must contain the properties for all nodes $u$ and $v$: (1) The flow along an

edge $(u, v)$ cannot exceed its capacity $f(u, v) \leq c(u, v)$; (2) The flow to a node is zero, except for the source $s$, which produces flow, and the sink, which consumes flow, $\sum_{w \in V} f(u, w) = 0$, where $u \neq s$ or $t$.

## 3. Connected Dominating Set

A Connected Dominating Set (*CDS*) of an ad-hoc network is a subset of nodes in the network, where the nodes in *CDS* are responsible for maintaining routing information, and other nodes have to rely on these nodes in *CDS* for transmission. Exploring *CDS* problem is frequently used to model the problem of computing a minimum number on the set. *CDS* plays a very important role in routing, broadcasting and connectivity management in wireless ad-hoc and sensor networks where there is no pre-defined physical backbone infrastructure to support routing and topology control that makes routing-related tasks or hierarchical organizations are very complicated.

The *CDS* problem can be formulated as follows: a graph $G(V, E)$, a Dominating Set(*DS*) ia a subset $U \subseteq V$ such that for every vertex $v \in V$, either $v \in V$, or there exists an edge $(u, v) \in E$ and $u \in U$. If the induced subgraph $G[U]$ is connected, then $U$ is called a *CDS*. The *CDS* problem is to find a *CDS* with minimum size. In this chapter, we will give three classes of this problem. Minimum Connected Dominating Set, which is the complementary problem of all *CDS* related problems, finds a set with minimum number of nodes to construct a virtual backbone or elect cluster heads in practice. Minimum Weighted Connected Dominating Set, where the graph is weighted on node that represents energy, cost, or neighbor size in real applications, finds the minimum sum of the weighted nodes to achieve better power consumption requirement. Fault Tolerant Connected Dominating Set finds a minimum set of nodes such that it remains a connected dominating set after any part of nodes leave, to guarantee the stability and robust of a backbone or a cluster-based network upon the node failure that occurs frequently in ad-hoc networks.

### 3.1. Minimum Connected Dominating Set

It is well-known that to find a Minimum Connected Dominating Set (*MCDS*) in a general graph is *NP-complete*. In wireless ad-hoc and sensor networks, if all nodes are homogeneous, Unit Disk Graph (*UDG*) is used to represent their geometrical structures. A *UDG* can be formally defined as follows: Given an undirected graph $G(V, E)$, each vertex $v$ has a transmission range with radius 1. Two vertices $u$ and $v$ adjacent if their Euclidean distance is less than or equal to 1. Clark et al. [1] show that computing *MCDS* is also *NP-hard* in *UDG*, and a lot of approximation algorithms for *MCDS* can be found in the literature.

To find an approximated *MCDS*, the most popular method is as follows. Firstly, find a maximal independent set(*MIS*) in given graphs. Given a graph $G(V, E)$, an Independent Set(*IS*) is a subset $I \subset V$ such that for any two vertex $u, v \in I$, they are not adjacent, say, $(u, v) \notin E$. An *IS* is called a Maximal Independent Set if any other arbitrary vertex is added to this set, the new set will not be an *IS* any more. Compared with *CDS*, *MIS* is much easier to be constructed. Usually, we use $mis(G)$ to denote the size of the constructed *MIS*. The second step is to make this *MIS* connected. We donote the number of the added vertices in this step by $conn(G)$. Let $mcds(G)$ be the size of minimum *CDS*. Then, the approximation

http://dx.doi.org/10.5772/54146

ratio for such algorithm is

$$\frac{mis(G) + conn(G)}{mcds(G)} = \frac{mis(G)}{mcds(G)} + \frac{conn(G)}{mcds(G)}$$

*3.1.1. CDS in UDG*

For the connecting part, the best-known algorithm is a Steiner tree based algorithm with $conn(G) \leq 3mcds(G)$ till now[7]. On the other hand, for selecting *MIS* part, there exist many results. Let $M$ be the set of *MCDS*. Based on the geometry structure on *UDG*, if we increase$V \backslash M$ from 1 to 0.5, then we can construct a new graph $G'$. It is easy to see that all the disks in $V$ are located insides the area formed by $M$. Then we can get a conclusion that the sum of maximum area for MIS should be less than the area of MCDS, which is a rough bound for $\frac{mis(G)}{mcds(G)}$.The following theorem gives this bound.

**Theorem1.**[2] The rough bound for $mis(G)$ and $mcds(G)$ is

$$mis(G) \leq 3.748mcds(G) + 5.252$$

Next, because the above result is rough, we analyzed the relationship between $mis(G)$ and $mcds(G)$ more specifically. Firstly, we used Voronoi Division to divide the whole area into some small Voronoi cell. The following Fig.1. gives an example. We also analyzed the area for each kind of polygons under densest situations. Then we can have a better bound for $\frac{mis(G)}{mcds(G)}$.

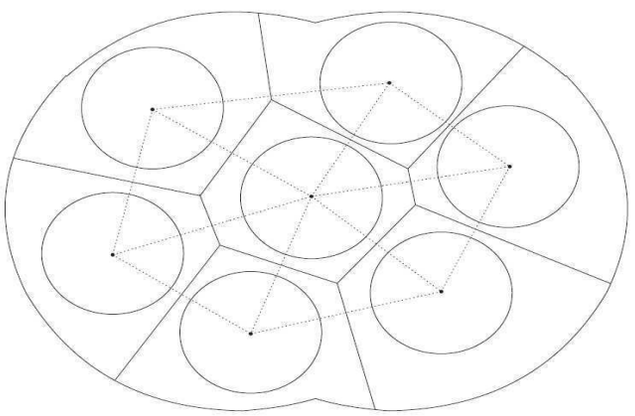


**Figure 1.** An example of Voronoi cell.

**Theorem2.**[2]$mis(G) \leq 3.453mcds(G) + 4.839$

Finally, we apply graph theory to consider the problem. By modifying the Voronoi division 3-regularization and combining classical Euler's Formula, we obtain the following result.

**Theorem3.**[2]$mis(G) \leq 3.399mcds(G) + 0.0790k + 4.874$, *where k is the number of the holes in the whole area*.

By these theoritical results, if we use two-step method to find a *CDS*, the size of the *CDS* is at most 6.4 times optimal solution. Now, we go back to Theorem 3, it has a parameter $k$. Then, the future work of this problem is to decide the parameter $k$. There exists litter rusults about it and the following theorem gives a basic solution.

**Theorem4.**[2]*For any unit disk graph G, let MCDS be a minimum connected dominating set. To form a hole, there need at least 6 connect vertices in MCDS.*

*3.1.2. CDS in UBG*

*UDG* is widely used to abstract the homogeneous wireless networks. However, sometimes, this assumption is far from the reality. In this case, to abstract homogeneous wireless networks in three-dimensional space, Unit Ball Graphs (*UBGs*) model is used. Since *UDGs* are special instances of *UBGs* in which the altitude of every node is the same, every *NP-hard* problem in *UDGs* is also *NP-hard* in *UBGs*. Naturally, *MCDS* in a *UBGs* is still *NP-hard*.

Like the *CDS* problem on *UDGs*, we use two-step method to find an approximated *MCDS*. The first step is to construct a *MIS* and to give a bound for $\frac{mis(G)}{mcds(G)}$ on *UBGs*. Recall the famous Gregory-Newton Problem concerning about kissing number, the kissing number $k(S_3) = 12$, that is, there all at most 12 independent unit balls that can simultaneously touch the surface of one unit ball. Based this result, there is a trivial bound $mis(G) \leq 11mcds(G) + 1$. In order to get a better result, we consider the problem: how many independent unit balls can simultaneously touch the surface of two adjacent unit balls. Through some accurate computation, we obtain the following lemma.

**Lemma 1.***The number of independent nodes in the union of two adjacent unit balls is at most 22.*

Since the result in Lemma 1 is better than kissing number, we can improve the above ratio.

**Theorem 5.**[5]$mis(G) \leq 10.917mcds(G) + 1.083$

Let $M$ be a *MIS* in a *UBG* such that for any partition $(M_1, M_2)$ of $M$, $dist(M_1, M_2) = 2$. Next, we present two differernt greedy algorithms to connect $M$ and give the approximation ratios.

**Algorithm 1.**[10]*Greedy Algorithm for CDS on UBGs.*
1: $H = G[M]$, *that is, H is the subgraph of G induced by M;*
2: ***WHILE*** *H is disconnected* ***DO***
3: *Choose the vertex v which connected the maximum component of H;*
4: $M = M \cup v$ *and* $H = G[M]$;
5:***END WHILE***
6:***RETURN*** *M;*

**Theorem 6.**[10]*The Algorithm 1 outputs a CDS in the unit ball graph G. And the size is up-bounded by* $(13 + ln10)opt + 1$, *where opt is the size of MCDS.*

Before introducing the Algorithm 2, some useful notations are presented. For any vertex $x$, let $N(x)$ be the set of vertices adjacent to $x$. For any vertex set $U$, let $N(U) = (\cup_{x \in U} N(x)) \backslash U$ and $M_{v,U}$ be the set of vertices which is adjacent to $v$ and belong to $M \backslash U$.

**Algorithm 2.**[5]*CDS Computation Algorithm on UBGs*
1: $U = r$, *and* $M' = M - r$
2: ***WHILE*** $M' \neq \emptyset$ ***DO***

*3: Choose the vertex v such that* $M_{v,U} = \max\{|M_{x,U}| \, | x \in N(U)\}$;
*4:* $U = U \cup v \cup M_{v,U}$ *and* $M' = M' \backslash M_{v,U}$;
*5:**END WHILE***
*6:**RETURN** U;*

**Theorem 7.**[5]*The Algorithm 2 outputs a CDS in the unit ball graph G. And* $|U| \leq 14.937opt + 1.083$.

Using above two algorithms, we can get an approximated *CDS* in any *UBG G* with linear performance.

*3.1.3. Multi-hop CDS in UDG*

If we further consider the architecture of wireless networks, we can separate the network into many clusters and the selected *CDS* are cluster heads and gateways. Each node will send message to its local cluster head, and information is exchanged among those cluster heads through more steady and responsible channels, which makes the whole network more reliable.

For a *CDS*,each cluster is really small, including nodes only one hop away from the corresponding cluster head. Therefore, some researchers enlarged the size of clusters, such that the super cluster head can be at most *d*-hop away from the nodes within its dominating range. The set of such super cluster head is called *d-CDS*. Given a graph *G*, an *d-CDS* is a subset $U \subset V$such that for any vertex $v \in V \backslash U$, there exists a path form $v$to some vertex of $U$ with length at most $d$. Furthermore, the subgraph induced by $U$ is connected. *d*-hop *CDS* problem is also NP-complete for general graphs and *UDGs*. The Fig.2. gives an example for 2-*CDS (TCDS)*.

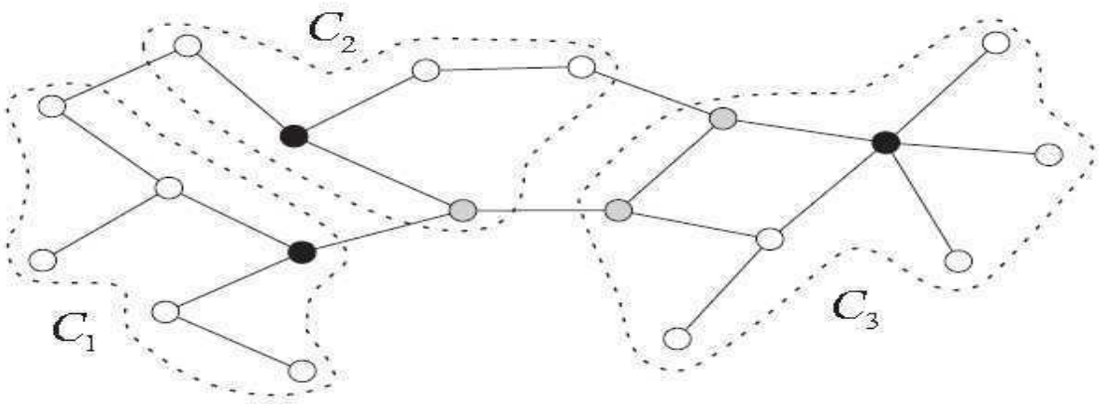



**Figure 2.** The black and gray vertices form a *TCDS* of *G* with three subgraphs $C_1$,$C_2$, $C_3$.

As the *CDS* problem, approximation algorithms for *d-CDS* problem usually divide into two steps. The first step, find a *d-DS* (usually *d-MIS*) in the graph. The second step, connect the *d-DS* into *d-CDS*. A *TCDS* algorithm can be generalized to *d-CDS* algorithm directly. Table 1 shows the local variables presented in Algorithm 3.

Then, we presented a distributed approximation algorithm for*TCDS* problem.

| Name | Explanation |
|---|---|
| level | Number of hops from root to this node. The rootąŕs level=0. |
| id | The ordering number of this node. |
| rank | The list of neighbors within 2 hops ordered by (level, id). |
| color | Totally for colors, black, brown, grey and white. |
| blacklist | The path from one black node to another black node which actives it. |

**Table 1.** Local Variables in Algorithm 3.

**Algorithm 3.**[3]*4-Coloring Algorithm for TCDS.*
***Phase 1***
*1. Choose an arbitrary root r, set r's level as 0 and build a spanning tree T for r, such that each node will get its level information.*
*2. Each node exchange information to get a rank list, which records its neighbors within 2-hops.*
***Phase 2***
*1. Root r colors itself as black, and broadcasts a black message with its id.*
*2. When a node receives a black message, it broadcasts a brown message with path as* $(id_1, id_2)$*, where* $id_1$ *is the sender's id and* $id_2$ *is the receiver's id. Besides, if this node is white or grey, mark itself as brown.*
*3. When a node receives a brown message with* $(id_1, id_2)$*, it broadcasts a grey message with a path as* $(id_1, id_2, id_3)$*, where* $id_3$ *is the receiver's id. If it is a white node, mark itself as grey.*
*4. When a node receives a grey message, if the node has been colored before, it will transfer this GREY message as T-grey message with sender's id. If the node is white, it gives labels to its colored neighbors in rank list. After update, if every of its lower-level neighbors have been colored already, record path as blacklist, mark itself as black, and broadcast a black message with its id.*
*5. When a white node receives a T-grey message with an id, it gives label to corresponding neighbor with same id in rank list as colored neighbor.*
*6. If a leaf node has been colored, it transmits a colored message to its parent. If a node has been colored and it receives colored message from all its children, it sends a colored message to its parent. Phase 2 will terminate until the root receives colored messages from all its children.*
***Phase 3***
*1. All black nodes join themselves into TCDS list, and send Join message with blacklist.*
*2. When receive a Join message, if a node's id is in blacklist, mark itself into TCDS set, and transmit this message until it reaches the black node with the first id in this list. The algorithm will terminate when all the nodes in blacklist has been inserted into TCDS set.*

Using the same method in *CDS* problem on *UDG* to analysis the algorithm, we can obtain the approximation ratio of Algorithm 3.

**Theorem 8.**[3]*The Algorithm 3 outputs a TCDS in the UDG G with time complexity* $O(n)$ *and message complexity* $O(nlgn)$*. Furthermore, the result from Phase 2 has size at most* $5.807opt + 17.152$*, and Algorithm 3 has approximation ratio* $17.421$*.*

Algorithm 3 can be easily modified into a distributed algorithm for $d$-hop *CDS* problem on *UDG* with approximation ratio $0.225r^3 + 1.337r^2 + 0.585r$, where $r = d + 0.5$. Hence, for any *UDG* and any fixed parameter $d$, we can get a $d$-hop *CDS* with constant approximation ratio.

http://dx.doi.org/10.5772/54146

### 3.2. Node-Weighted Connected Dominating Set

The Node-Weighted Connected Dominating Set (*NWCDS*) problem is a generalization of the *CDS* problem. Given a graph $G(V, E)$ with node weight function $f : V \to R^+$, the *NWCDS* problem is to find a *CDS* of $G$ such that its total weight is minimum. For convenience, the weight function $f$ such that $f(V) \geq 1$ is normalized for any vertex $v$ in $G$. If the weights on all vertices are the same, the *NWCDS* problems are equal to the *CDS* problem. Hence, the *NWCDS* problems are also *NP-complete* on general graphs and *UDGs*.

To deal with this problem, we firstly considered the Min-Weight Chromatic Disk Cover (*MWCDC*) problem and used dynamic program to obtain a polynomial algorithm for *MWCDC*. Then, comparing the two problems and studying the relationship between them, we had the Lemma 2.

**Lemma 2.**[11]*If there exists an $\rho$-approximation algorithm for the MWCDC and for any fixed $\varepsilon$, there is a polynomial $(4\rho + \varepsilon)$-approximation algorithm for the NWDS.*

Based on Lemma 2 and the exact algorithm for *MWCDC*, we can get a $(4 + \varepsilon)$-approximation algorithm for *NWDS*. Then, using the $(1 + \varepsilon)$ -approximation algorithm for the Node-Weighted Steiner tree (*WST*) problem to connect the *NWDS*, we can obtain an approximation algorithm for *NWCDS* on *UDG*.

**Theorem 9.**[11]*There is a $(5 + \varepsilon)$-approximation algorithm for the MWCDS by using a node-weighted Steiner tree to interconnect all nodes of the MWDS.*

### 3.3. Fault-Tolerant Connected Dominating Set

In wireless ad-hoc and sensor networks, nodes are mobile and thus the topology of such networks can be changed frequently. As a result, a virtual backbone (*VB*) induced by a*CDS* can be broken easily and thus it should be re-computed repeatedly. Hence, to construct a fault-tolerance *VB* is important. Here, a $k$-connected $m$-dominating set is introduced as a generalized abstraction of a fault tolerant *VB*. Given a graph $G(V, E)$, a subset $U \subset V$ is a $m$-dominating set ($m$-DS) if for any vertex $v \in V \backslash U$,$v$ has at least $m$ neighbors in $U$ . Furthermore, if $U$ is $k$-connected, we call $U$ is a $k$-connected $m$-dominating set ($(k, m)$-*CDS*).

To obtain a $(k, m)$-*CDS*, the main idea is as follows. The first step is to get a basic *CDS*, that is, $(1, 1)$-*CDS*. Next, add $m - 1$ *MISs* in the rest graph to make the $(1, 1)$-*CDS* into $(1, m)$-*CDS*. Finally, by adding some new vertices to increase the connectivity of the *CDS*, the $(k, m)$-*CDS* is obtained. The first and second steps are easy to complete, but the final step is very hard for $k \geq 3$. When $k = 2$, there are some approximation algorithms, and the best one is given by Shang et. al. [8] with approximation ratio $5 + \frac{25}{m}$ for $2 \leq m \leq 5$ and 11 for $m > 5$.

In the following, we introduce a $(3, m)$-*CDS* approximation algorithm. The key idea about this algorithm is to become the "bad-points" in the $(2, m)$-*CDS* to good. Given a 2-connected graph $G$, a vertex $v$ is called a "good-point" if $G - v$ is also 2-connected; otherwise, $v$ is called a "bad-point". Based on above definition, we can get Lemma 3.

**Lemma 3.**[4]*A 2-connected graph without any bad-point is 3-connected.*

**Algorithm 4.**[4]*Algorithm for $(3, 3)$-CDS on UDGs.*
*1: Computer a $C_{2,3}$, and Set $Y = C_{2,3}$;*
*2: **WHILE** $Y$ has bad-points **DO***

*3: Choose an arbitrary bad-point $v$ and Set $B = Y$;*
*4: Construct the leaf-block tree $T$ of $B - \{v\}$ with blocks $\{B_1, B_2, ..., B_s\}$ and cut-vertices $\{c_1, c_2, ..., c_t\}$;*
*5:* ***WHILE*** *there exist a non cut-vertex $w$ in some block $B_i$ and a cut-vertex $c_j$ such that $(w, c_j)$ is a saparetor of $Y$* ***DO***
*6: Set $B = B_i$ and $v = c_j$;*
*7: Construct the leaf-block tree $T$ of $B - \{v\}$ with blocks $\{B_1, B_2, ..., B_s\}$ and cut-vertices $\{c_1, c_2, ..., c_t\}$;*
*8:* ***END WHILE***
*9: Find a path $H$ to make a bad-point $c_j \in \{c_1, c_2, ..., c_t\}$ to be a good-point such that all other new vertices of $H$ are good;*
*10: Set $Y = Y \cup H$;*
*11:* ***END WHILE***
*12:* ***RETURN*** *$Y$;*

**Theorem 10.**[4]*The Algorithm 4 output a $(3, 3)$-CDS with approximation ratio* $520/3$.

Easy to see, if we start the Algorithm 4 with a $C_{2,m}$ with $m \geq 3$, the algorithm will return a $(3, m)$-*CDS*. And we can get the following result.

**Theorem 11.**[4]*There exists a constant ratio approximation algorithm for $(3, m)$-CDS problem in UDG for any $m$.*

As above, given a *UDG* and an integer $m$, we can get a $(3, m)$-*CDS* as a fault-tolerance *VB* with constant ratio.

## 4. Disjoint Sets

Disjoint Sets (*DS*) of an ad-hoc network is a collection of disjoint sets of nodes that each set is capable of achieving application requirements. For instance in a wireless sensor network where coverage is an important demand, the Connect Disjoint Set divides the nodes into a number of disjoint sets, such that every set completely covers all the target points. Connected Disjoint Set problem is frequently used to formulate the problem into finding a maximum number of sets.

*DS* problem plays an important role in ad-hoc networks, especially in wireless sensor networks. Disjoint Set Covers (*DSC*) problem is one of the classical problems that aims to determine a maximum number of disjoint covers, where every cover is a set of sensors which together monitor all the target points. It can be formulated as a graph, and solved by combinatorial optimization method-mixed integer programming. Besides, we studied further to find maximum disjoint sets for maintaining not only coverage but also connectivity. This class of Disjoint Set problem could be used for node scheduling methods to conserve energy, topology control methods to tolerant failure, and routing protocol design.

The disjoint set covers (*DSC*) problem was addressed by M. Cardei and D.Z. Du [12] in order to solve the problem of energy efficiency for surveillance of a set of targets. Let $T = \{t_1, t_2, ..., t_m\}$ be a set of $m$ targets. Each node covers a subset of targets. A collection of nodes $S = \{s_1, s_2, ..., s_n\}$ are defined as that each set $S_i = \{t_{i_1}, t_{i_2}, ..., t_{i_l}\}$ if node $S_i$ covers targets $t_{i_1}, t_{i_2}, ..., t_{i_l}$. *DSC* problem aims to find a maximum number of disjoint sets of nodes, where every set is able to cover all the targets. They presented a heuristic algorithm based on

the flow network since the *DSC* problem was proved *NP-complete*. First, a bipartite directed graph $G(V, E)$ is constructed with the vertex set $V = S \cup T$ and $s_i t_{i_j} \in E$ with a capacity 1 if and only if $t_{i_j} \in s_i$. Then, draw $k$ copies of $G$, namely $G_1, G_2, ..., G_k$, where $k$ is the maximum number of disjoint sets. A vertex $s_i$ in $G$, is presented $S_1 i, S_2 i, ..., s_k i$ in $G_1, G_2, ..., G_k$. A source node $S$ n the flow network is constructed. For each $s_i$ in $S$, a vertex $s_o i$ is created to connect $s$ with an edge with the capacity equals to the degree of $s_i$ in $G$. Also, edges connecting $s_o i$ with $s_j i$ are constructed with the capacity equals to the degree of $s_i$ in $G$. A vertex $X_i$ is is created to connect every vertex $t_{i_j}$ in $G_i$ with a capacity 1. Two ends $Y_1$ and $Y_2$ are created in the flow network. Each vertex $X_i$ is connected to $Y_2$ with a capacity $m$. Every vertex $t_{i_j}$ is connected to $Y_1$ with a capacity $n$. The *DSC* problem turns to a maximum-flow problem that is to maximize the flow received in $Y_2$. Finally, the *DSC* problem was formulated and computed by using the mixed integer programming (*MIP*) heuristic.

We considered not only the coverage optimization, but also the connectivity issue. We proposed the Multiple Disjoint Sets with Maintaining Coverage and Connectivity (*MDS-MCC*) problem [13] that given a wireless network with $w$ sinks (or base stations) and $n$ nodes each of which has its respective transmission range and sensing range, and $m$ targets in territory, determine a maximum number of disjoint sets of nodes such that (1) all nodes of each set together cover the whole $m$ targets; (2) for every node in each set, it is connected to a sink via nodes within the same set.

We presented two graph-based models to formulate the *MDS-MCC* problem.

**Model 1**:*Given integers $n, m$ and $w$, a directed graph $G = (V, E; f)$, where $R = \{v_{n+1}, ..., v_{n+w} \subset V\}$ is a set of sinks, and $T = \{t_1, t_2, ..., t_m\}$ is a set of targets. Find the maximum integer $k$ such that there exist pairwise disjoint subgraphs $H_1, H_2, ..., H_k$ of $G \backslash R$, and for each supergraph $H_i'(V_i', E_i')$ of $H_i$, where $V_i' = V(H_i) \cup R$ and $E_i' = \{(v_i, v_j) : (v_i, v_j) \in E, v_i, v_j \in V_i'\}$, satisfying (1) $f(V_i') = T$; (2) for each $v_j in V(H_i)$, $v_j$ is connected to a vertex $u \in R$. $f(v)$ is a labeling function that denotes the set of targets node covers.*

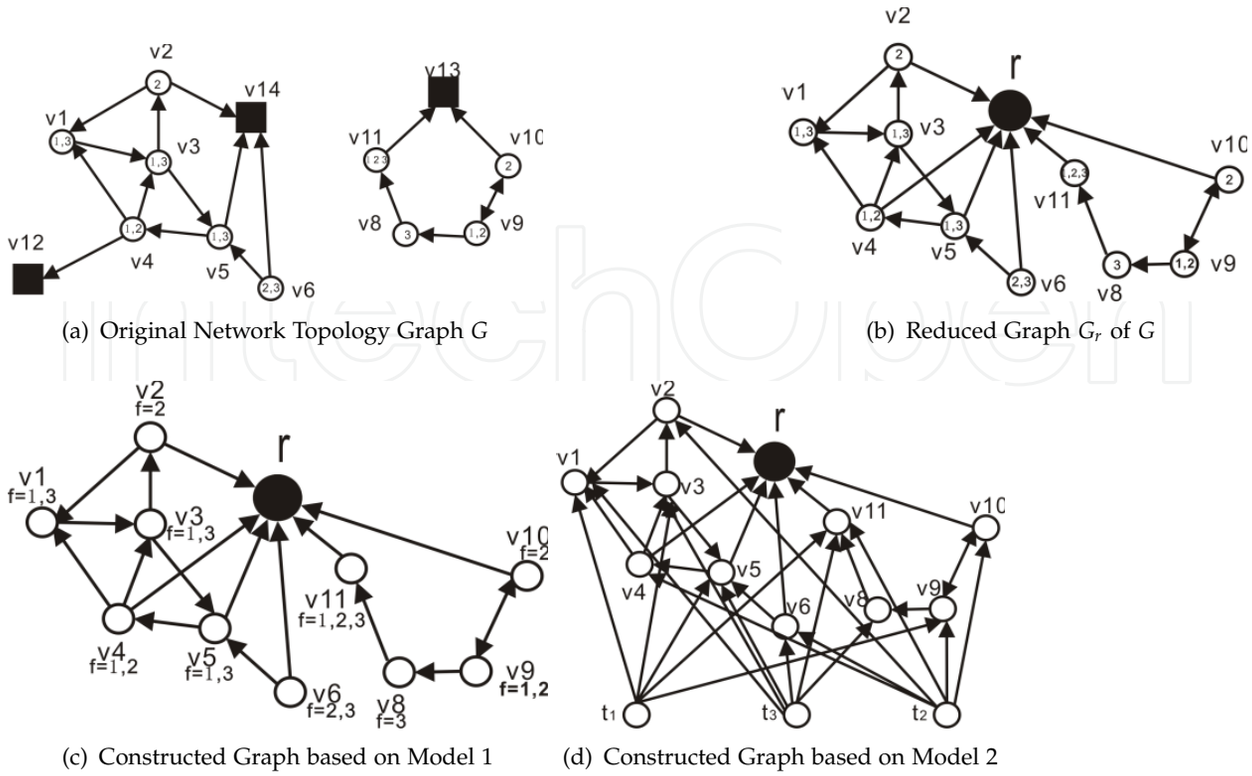
$$f(v_i) = \begin{cases} \{t_j : t_j \in s_i\}, i \leq n \\ \emptyset, i > n \end{cases}$$

*And $f(V') = \cup_{v \in V'} f(v)$ where $V' \subseteq V$, $f(H) = \cup_{v \in V(H)} f(v)$ where $H \subseteq G$.*

**Model 2**: *Given integers $n, m$ and $w$, a directed graph $G = (V, E)$, where $R = \{v_{n+1}, ..., v_{n+w}\} \subset V$ is a set of sinks, and $T = \{t_1, t_2, ..., t_m\}$ is a set of targets. Find the maximum integer $k$ such that there exist pairwise disjoint subgraphs $H_1, H_2, ..., H_k$ of $G \backslash (R \cup T)$, and for each $t_j \in T, t_j$ is connected to $u$ in each supergraph $H_i'(V_i', E_i')$ of $H_i$, where $V_i' = V(H_i) \cup R \cup T, E_i' = \{(v_i, v_j) : (v_i, v_j) \in E, v_i, v_j \in V_i'\}$ and $u \in R$.*

All sinks can be reduced to only one node $r$ called root by using Lemma 4. Therefore, given either the graph $G = (V, E; f)$ based on Model 1 or the graph $G = (V, E)$ based on Model 2, the corresponding reduced graph can be constructed as $G_r = (V_r, E_r; f, r)$ or $G_r = (V_r, E_r; r)$ where $V_r = \{v_1, v_2, ..., v_n, r\}$, the first $n$ vertex are the nodes, and $r$ is root. There is no more changes expect that an edge in $E$ from a node to a sink becomes an edge in $E_r$ from the node to the root. If a node is connected to more than one sink, only one edge is added in $G_r$.

**Lemma 4.**[13]*A node is connected to a sink if and only if it is connected to the root in the corresponding reduced graph.*




(a) Original Network Topology Graph $G$ (b) Reduced Graph $G_r$ of $G$

(c) Constructed Graph based on Model 1 (d) Constructed Graph based on Model 2

**Figure 3.** Example of original network topology (a) and its corresponding reduced graph (b). (c) and (d) shows the constructed graph based on the Model 1 and Model 2 respectively.

We found that *MDS-MCC* problem is *NP-complete*. Two algorithms, heuristic and network flow were proposed to solve *MDS-MCC* based on the two models. Heuristic algorithm was designed to find the maximal number of disjoint subgraphs of $G$ based on Model 1. The algorithm first initializes that $H = \emptyset$, which presents the set of vertex having been found. Then the algorithm finds out all the paths whose ends belongs to $H$ and put the vertex in these paths into $H$ until $H$ covers all of the targets. The algorithm deletes the redundant nodes from $H$ when $H$ still covers all of the targets after delete it. The algorithm finds one subgraph $H$ and repeats to find other subgraphs.

**Algorithm 5.**[15]*Heuristic Algorithm for MDS-MCC.*
*1:Construct reduced graph* $G_r = (V_r, E_r; f, r)$ *from* $G = (V, E; f)$ *and T;*
*2:k=0;*
*3:**WHILE**still has the subgraph of G that covers all the targets **DO***
*4:* $k = k + 1$;
*5:* $H_k = \{r\}$;
*6:* ***WHILE*** $|f(H_k)| \neq m$ ***DO***
*7:* $PS = \{v_0 v_1 ... v_e : v_e \in H_k, v_i \in V_r - H_k, f(v_j) \subseteq f(H_k), f(v_0) \notin f(H_K)\}$;
*8:* *Select one path* $P \in PS$;
*9:* $H_k = P \cup H_k$;
*10:* ***END WHILE***
*11:* *Delete redundant nodes from* $H_k$;
*12:* $G_r = G_r \backslash H_k$;

*13:**END WHILE***
*14:**RETURN** $H_1, H_2, ..., H_k$*

In Model 2, the problem is converted to find the maximal $k$ disjoint sets such that there exists a path from every target node $t$ to the root node $r$ for each set. For any two paths from a target node $t$ to the root $r$ belonging to any two disjoint sets, the two paths are disjoint. Therefore, there are $k$ pairwise disjoint paths from $t$ to $r$.

The network flow algorithm first finds a set of paths $P_i = \{p_i^1, p_i^2, p_i^3, ..., p_i^i, \}$ for every target node $t_i$ to $r$ such that any two paths are disjoint. We converted the problem of obtaining the related paths with maximum $l_i$ from $t_i$ to $r$ to the maximum flow problem like *DSC*. We presented a method to construct the flow network from the network of Model 2. An example is shown in Fig.4. The problem that finds the maximum number of paths is converted to solve the problem of computing the maximum flow from $s$ to $Y_2$. For each copy in the flow network, the flow network algorithm chooses the vertex for each disjoint result set if the flow from the vertex $v_0j$ to the vertex $v_ij$ is greater than 0.

**Algorithm 6.**[13]*Network Flow Algorithm for MDS-MCC.*
*1:Construct $G_r = (V_r, E_r; r)$ from $(G = (V, E))$ and $T$;*
*2:**FOR EACH** $t_i \in T$ **DO***
*3: Find $l_i$ pairwise disjoint paths, $p_i^1, p_i^2, ..., p_i^i$, from $t_i$ to $r$;*
*4:**END FOR***
*5:Find the maximum $k$ pairwise disjoint sets $H_1, H_2, ..., H_k$ such that each set $H_i = \cup_{j=1}^{m}\{V(P_j^{i_j}) - \{t_j, r\}\}$;*
*6:**RETURN** $H_1, H_2, ..., H_k$;*

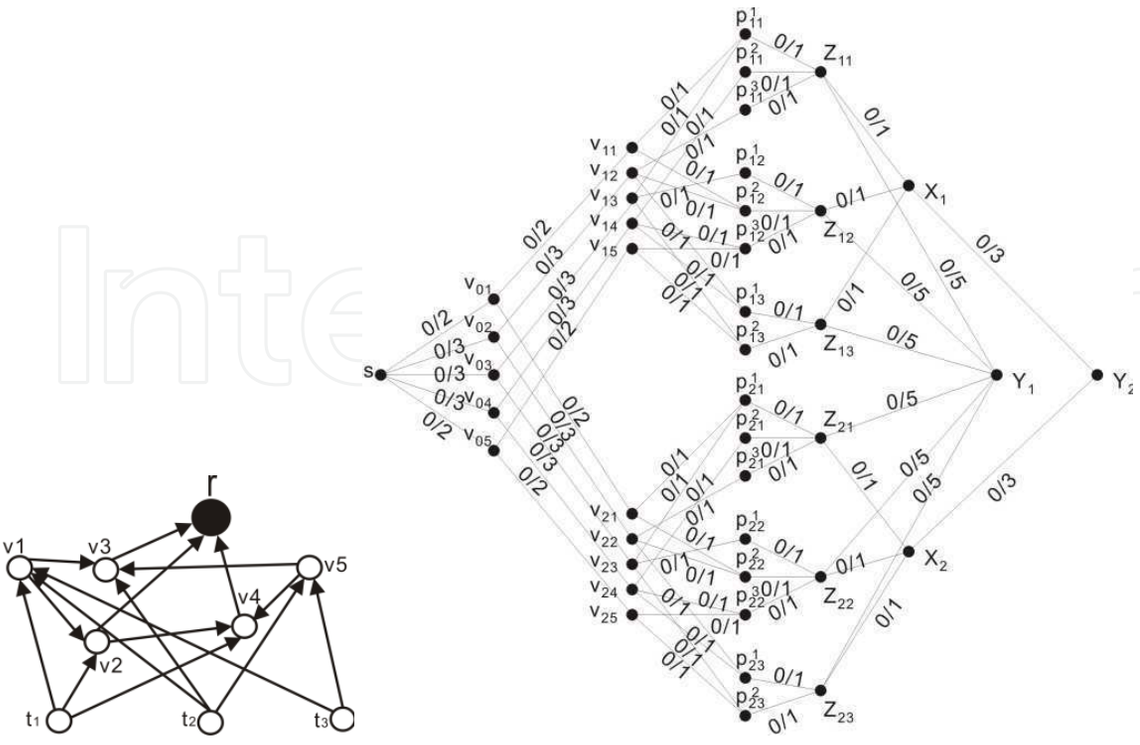
A special case was studied in wireless sensor networks that each node covers at most one target. We assumed there are $m$ targets that each is exactly monitored by $k$ sensor nodes. There must be $k$ disjoint sets each of which completely covers all the targets and is connected to one of sinks. And $k$ is the theoretical maximum number.

**Theorem 12.**[14]*Given integers $n$ and $m$, a directed graph $G = (V, E; f, r)$ and a target set $T = \{t_1, t_2, ..., t_m\}$.$A_1, A_2, ..., A_m$ be $m$ disjoint sets with $|A_i| = k$, where $A_i \subset V(G)$. If $k = 2$ and $G$ is $(m + max\{1, m - 4\})$-connected, or $k \geq 3$ and $G$ is $(m(k-1)+1)$-connected, then there exist $k$ connected subgraphs $H_1, H_2, ..., H_k$, satisfying (1) $f(H_i) = T$; (2) for each $v_j \in V(H_i), v_j$ is connected to $r$.*

## 5. Minimum Spanning Subgraph and Minimum Steiner Tree

The minimum spanning subgraph is used to reduce the cost of algorithms in underlying wireless ad-hoc networks that are modeled as graphs. For instance, a spanning tree is used as a backbone to reduce the cost of broadcast, or to cluster the hierarchical structure. A spanning tree of a graph is a subgraph that is a tree and connects all the vertices together. Many research works on ad hoc related networks exploited minimum spanning subgraph to design energy-efficient distributed protocols, multicast routing protocols, fault tolerant topology control protocols and etc. To construct spanning subgraph has the advantage of low time and message complexity.

The Steiner tree problem is superficially similar to the minimum spanning tree problem. The minimum Steiner tree problem is a problem in combinatorial optimization, which may




(a) Network Topology (b) Constructed Flow Network

**Figure 4.** [13] Example of converting the network topology based on Model 2 to a constructed flow network.

be formulated in a number of settings, with the common part being that it is required to find the shortest interconnect for a given set of objects. Most of the Steiner Tree problems are NP-complete. Similar with spanning tree, Minimum Steiner Tree problem has been applied to design minimum energy broadcasting protocols, delay constrained protocols, range assignment topology control protocols in ad hoc networks.

## 5.1. Minimum Spanning Tree and Subgraph

Given a graph $G(V, E; C)$, a spanning tree of $G$ is a subgraph that is a tree and connects all the vertices together. A minimum spanning tree (*MST*) is the spanning tree with the minimum total weight in all spanning trees. There are many extended concepts of *MST*. The $k$-minimum spanning tree ($k$-*MST*) is the tree that spans some subset of $k$ vertices in the graph with minimum weight. A set of $k$-smallest spanning trees is a subset of $k$ spanning trees such that no spanning tree outside the subset has smaller weight. The Euclidean minimum spanning tree is a spanning tree of a graph with edge weights corresponding to the Euclidean distance between vertices that are points in the plane. The degree constrained minimum spanning tree is a minimum spanning tree with each vertex $v$ is connected to no more than $d(v)$ other vertices. If the graph $G$ is not connected, the graph has minimum spanning forest. Similarly, a minimum spanning subgraph is the spanning subgraph with the minimum total weight in all spanning subgraphs.

M. Cardei et al. [16] addressed the topology control issue of power assignment in ad-hoc networks by using minimum spanning subgraph techniques. They aimed to minimize the total transmission power assigned for all nodes while building $k$-vertex fault tolerant

communication paths from each node to the sinks. They modeled the network topology with an undirected weighted graph. The issue was formulated as that given a directed graph $G$ and a root $r$, find a directed spanning subgraph of $G$ such that: (1) the sum of the weight of the edges is minimized, (2) there are $k$-vertex disjoint paths between $r$ and every vertex in $G$. The $FT$ algorithm [18] was applied to solve this problem optimally to reach $k$-approximate.

We addressed the issue of $k$-vertex fault-tolerant many-to-many routing power assignments in ad-hoc networks [17] that given an ad-hoc network consisting of $n$ nodes with the various transmission ranges. For each node $V_i$, it can adjust the transmission ranges up to its maximum value $R_max$. Determine the power $p_i$ of node $V_i$ such that 1) there exist $k$-vertex disjoint data routing paths between any pair of nodes; 2) the total power consumed over all sensor nodes is minimized, namely $\sum_{i=1}^{n} p_i$ is minimized.

A directed weighted graph $G(V, E; C)$ was represented to model the network topology, where $V = \{v_1, v_2, ..., v_n\}$ is the set of nodes and $E = \{(v_i, v_j) : dist(v_i, v_j) \leq R_max\}$ is the set of edges. For each edge $(u, v) \in E$, there exists a weight $C(u, v)$ associated with it. $C(u, v)$ represents the power consumption needed by $u$ to communicate with $v$. It aimed to construct a minimum $k$-vertex connected subgraph of $G$ by finding a set of power assignments for each node.

Two algorithms were proposed to find such minimum $k$-vertex connected subgraph of $G$ by using spanning subgraph technology. The first algorithm produces a $k$-vertex connected spanning subgraph and assigns to each vertex the minimum transmission range to reach all of its neighbors. The algorithm removes the edges in decreasing order of their weights if and only if the graph keeps $k$-vertex connected after the removal. Theorem 13 guarantees that the final remaining subgraph is $k$-vertex connected. The algorithm assigns the transmission power to each node according to the subgraph.

**Theorem 13.**[17] *A graph $G(V, E)$ is a k-vertex connected directed graph. If $(u, v) \in E$ and there are at least $k + 1$ disjoint paths from $u$ to $v$, namely $\lambda(u, v) = k + 1, G - (u, v)$ is a k-vertex connected graph.*

**Algorithm 7.**[17] *Heuristic Algorithm for k-vertex fault-tolerant power assignments.*
*1:Sort all edges in E in decreasing order of the weights;*
*2:**FOR EACH** edge $(u, v)$ in the sorted order **DO***
*3: **IF** $\lambda(u, v) = k + 1$ **THEN***
*4: $G = G - (u, v)$;*
*5: **END IF***
*6:**END FOR***
*7:**FOR** $i = 1$ **TO** $n$ **DO***
*8: $p_i = max\{C(v_i, v_j) : (v_i, v_j) \in E\}$;*
*9:**END FOR***

Another algorithm is an $O(\sqrt{n/\varepsilon})$-approximation algorithm [17] by using the solution of the minimum-cost $k$-vertex connected spanning subgraph problem proposed by Cheriyan $k$-vertex connected, for any $\varepsilon > 0$ and $k \leq (1 - \varepsilon)n$.

### 5.2. Minimum Steiner Tree

Steiner tree problem ($STP$) is a classical combinatorial optimization problem. This problem has a lot of versions. The graph version of $STP$ is that: Given a edge-weighted graph $G =$

$(V, E; C)$ with edge-weight function $C : E \to \Re^+$ and a subset $U \subset V$ called a terminal set, the *STP* is to find a subtree $T$ of $G$ interconnecting the terminal set $U$ with minimum total weight. The graph version of *STP* is *NP-complete* and the best approximation ratio is $\rho = 1 + \frac{\ln 3}{2} = 1.55$ till now [28].

We considered a variant of *STP* – Node-Weighted Steiner Tree (*NWST*) problem, that is, the weight function is from vertex set to positive real set now. Given a node-weighted graph $G = (V, E; C)$ with node-weight function $C : V \to \Re^+$ and a subset $U \subset V$ called a terminal set, the *STP* is to find a subtree $T$ of $G$ interconnecting the terminal set $U$ with minimum total weight. As an application, *NWST* can be used on *NWCDS* problem to interconnect the node-weighted dominating set.

To deal with the *NWST* problem, the first idea is to convert this problem to classical *STP*. We constructed a new graph $G'$ with the same vertex set, edge set and terminal set. The difference is the weighted function of $G'$ is on edges.

**Algorithm 8.**[9]*Approximation Algorithm for NWST on UDGs*
*1:Construct an edge-weighted graph $G' = (V, E; C')$ with the same vertex set, edge set and terminal set of G;*
*2:**FOR EACH** edge $u, v$ in graph $G'$ **DO***
*3: Assign the weight of this edge $C'(u, v) = (C(u) + C(v))/2$;*
*4:**END FOR***
*5:$T = SMT(G', U)$, where $SMT(G', U)$ is the best-known approximation algorithm on graph $G'$ and terminal U;*
*6:**RETURN** T;*

**Theorem 14.**[9]*Algorithm 8 is a $2.5\rho$-approximation for node-weighted Steiner tree problem in unit disk graph.*

Furthermore, we should give a theoretical result, to show that the *NWST* has polynomial-time approximation scheme (*PTAS*) on *UDGs* if the terminal set $U$ is $c$-local, that is, in the minimum node-weighted spanning tree for $U$, the Euclid distance of the longest edge is at most some constant $c$. A *PTAS* is a family of approximation algorithm with ration $1 + \varepsilon$ for any $\varepsilon > 0$.

The main idea of the *PTAS* for *NWST* is based on the partition and shifting strategy. Firstly, we partitioned the whole area containing all vertices into some small cells. Then, we divided every cell into interior area and boundary area. Secondly, for each cell, we constructed a local optimal Steiner forest on terminal vertices in the interior area of this cell. Then, we combined all these forests to obtain a local optimal Steiner forest. Thirdly, we added all the crossing edges to get a Steiner tree on terminal set $R$. We call the resulting graph $G_p$ for a specific partition $P$. In order to get a better node-weighted Steiner tree, we shifted the partition and choose the minimum output among all of partitions.

**Theorem 15.**[6]*Node-weighted Steiner tree problem has PTAS on unit disk graphs.*

## 6. Conclusion

Many problems in mobile ad-hoc networks can be formulated by using graph theory. However, in real ad-hoc network applications, there are many constraints that make the

issues difficult to be tractable. The methods that might convert these practical issues into graph-based problems are important for the design and implementation of routing, broadcasting and organization algorithms. In this chapter, we present three essential graph-based issues casted from practicle ad-hoc network issues: Connected Dominating Set, Disjoint Sets, and Minimum Spanning Subgraph and Minimum Steiner Tree. Theoritical analysis are described to verify the correctness of these proposed algorithms that are either heuristics or approximationg.

While much efforts have been made to solve the graph-based issues, still much progress needs to be done. For instance, some clustering issues in ad-hoc networks can be casted into graph labeling or graph coloring problem that assigns of labels to the nodes subject to certain constraints. In other mobile ad-hoc applications, many problems involve graph classification, graph subsumption, and even the description and implementation of graph data structure, querying and database. More approaches for ad-hoc network applications should be discussed from the aspectes of the applicability and the utility by using graph theory.

## Acknowledgements

This work was partially supported by the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry, the Gansu Provincial Science & Technology Department (grant no. 1007RJYA010), the National Natural Science Foundation of China (grant nos. 61003240, 11201208, 10905026, 11275003) and the Fundamental Research Funds for the Central Universities (grant no. lzujbky-2011-44).

## Author details

Li Liu[1], Xianyue Li[2], Jiong Jin[3], Zigang Huang[4],
Ming Liu[5] and Marimuthu Palaniswami[3]

1 School of Information Science and Engineering, Lanzhou University, P.R.China
2 School of Mathematics and Statistics, Lanzhou University, P.R.China
3 Department of Electrical and Electronic Engineering, The University of Melbourne, Australia
4 School of Physical Science and Technology, Lanzhou University, P.R.China
5 School of Electrical and Information Engineering, The University of Sydney, Australia

## References

[1] B.N. Clark, C.J. Colbourn and D.S. Johnson, Unit Disk Graphs, Discrete Mathematics, 86 165-177 (1990).

[2] X. Gao, Y. Wang, X. Li and W. Wu, Analysis on Theoretical Bounds for Approximating Dominating Set Problems, Discrete Mathematics, Algorithms and Applications, 1 71-84 (2009).

[3] X. Gao, X. Li and W. Wu, A Constant-Factor Approximation for d-Hop Connected Dominating Set in Unit Disk Graph, The 10th International Conference on Information and Management Sciences (IMS 2011), 263-272 (2011).

[4] D. Kim, W. Wang, X. Li, Z. Zhang and W. Wu, A New Constant Factor Approximation for Computing 3-Connected m-Dominating Sets in Homogeneous Wireless Networks, The 29th IEEE Conference on Computer Communications (INFOCOM 2010), 1-9 (2010).

[5] D. Kim, Z. Zhang, X. Li, W. Wang, W. Wu and D.-Z. Du, A Better Approximation Algorithm for Computing Connected Dominating Sets in Unit Ball Graphs, IEEE Transactions On Mobile Computing, 9 1108-1118 (2010).

[6] X. Li, X.-H. Xu, F. Zou, H. Du, P.-J. Wan, Y. Wang and W. Wu, A PTAS for Node-Weighted Steiner Tree in Unit Disk Graphs, The 3rd Annual International Conference on Combinatorial Optimization and Applications (COCOA 2009), 36-48 (2009).

[7] M. Min, H.-W.Du, X.-H. Jia, C.-X. Huang, S.-C. Huang and W. Wu, Improving Construction for Connected Dominating Set with Steiner Tree in Wireless Sensor Networks, Journal of Global Optimization, 35 111-119 (2006).

[8] W. Shang, F. Yao, P. Wan and X. Hu, On Minimum m-Connected k-Dominating Set Problem in Unit Disc Graphs, Journal of Combinatorial Optimization, 16 99-106 (2008).

[9] F. Zou, X. Li, S. Gao and W. Wu, Node-weighted Steiner tree approximation in unit disk graphs, Journal of Combinatorial Optimization, 18 342-349 (2009).

[10] F. Zou, X. Li, D. Kim and W. Wu, Construction of Minimum Connected Dominating Set in 3-Dimensional Wireless Network, The 3rd International Conference on Wireless Algorithms, Systems and Applications (WASA 2008), 134-140 (2008).

[11] F. Zou, Y. Wang, X.-H. Xu, X. Li, H. Du, P. Wan and W. Wu, New approximations for minimum-weighted dominating set and minimum-weighted connected dominating sets on unit disk graphs, Theoretical Computer Science, 412 198-208 (2011).

[12] M. Cardei, D.-Z. Du, Improving wireless sensor network lifetime through power aware organization, Wireless Networks, 11(3) 333-340 (2005).

[13] L. Liu, B. Hu, L. Li, Energy Conservation Algorithms for Maintaining Coverage and Connectivity in Wireless Sensor Networks, IET Communications, 4(7) 786-800 (2010).

[14] H. Li, H. Miao, L. Liu, L. Li, H. Zhang, Energy conservation in wireless sensor networks and connectivity of graphs, Theoretical Computer Science, 393(1-3) 81-89 (2008).

[15] L. Liu, B. Hu, H.F. Miao, H. Li, L. Li, Q.L. Zhao, Achieving Energy Conservation, Coverage and Connectivity Requirements in Wireless Sensor Networks, The 29th IEEE International Conference on Distributed Computing Systems Workshops (ICDCS Workshop 2009), 227-232 (2009).

[16] M. Cardei, S. Yang, J. WU, Algorithms for fault-tolerant topology in heterogeneous wireless sensor networks, IEEE Transaction on Parallel Distributed Systems, 19(3) 545-558 (2008).

[17] L. Liu, L. Li, B. Hu, Algorithms for k-fault Tolerant Power Assignments in Wireless Sensor Networks, Science China-Information Sciences, Springer, 53(12) 2527-2537 (2010).

[18] N. Li, J.-C. Hou, FLSS: A fault-tolerant topology control algorithm for wireless networks, The 10th Annual International Conference on Mobile Computing and Networking (MobiCom 2004), 275ÍC286 (2004).

[19] J. Cheriyan, S. Vempala, A. Vetta, Approximation algorithms for minimum-cost k-vertex connected subgraphs, The 34th Annual ACM Symposium on the Theory of Computing (STOC 2002), 306-312 (2002).

[20] D. Chakrabarti, C. Faloutsos, Graph mining: laws, generators, and algorithms, ACM Computing Surveys, 38(1) (2006).

[21] A.-L. Barabási, Linked: the new science of networks, Perseus Books Group (2002).

[22] L. Liu, B. Hu, L. Li, Algorithms for Energy Efficient Mobile Object Tracking in Wireless Sensor Networks, Cluster Computing, 13(2) 181-197 (2010).

[23] J. Tian, J. Hähner, C. Becker, I. Stepanov, K. Rothermel, Graph-based mobility model for mobile ad hoc network simulation, The 35th Annual Simulation Symposium, (2002).

[24] A. Casteigts, S. Chaumette, Dynamicity aware graph relabeling systems (da-grs), a local computation based model to describe manet algorithms, The 17th International Conference on Parallel and Distributed Computing and Systems (PDCS 2005), 231-236 (2005).

[25] S. Bittner, W.-U. Raffel, and M. Scholz, The area graph-based mobility model and its impact on data dissemination, The 3rd IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops 2005), 268-272 (2005).

[26] M. Jahnke, C. Thul, P. Martini, Graph based metrics for intrusion response measures in computer networks, The 32nd IEEE Conference on Local Computer Networks (LCN 2007), 1035-1042 (2007).

[27] W. Ke, P. Basu, T.D.C. Little, A task graph based application framework for mobile ad hoc networks, The 5th IEEE International Conference on Communications (ICC 2002), 3279-3283 (2002).

[28] G. Robins and A. Zelikovsky, Improved Steiner tree approximation in graphs, The 11th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, 770?779 (2000).

IntechOpen