On extracting consistent graphs in wireless sensor networks

Murtuza Jadliwala,* Qi Duan, Jinhui Xu and Shambhu Upadhyaya

Department of Computer Science and Engineering,
The University at Buffalo,
State University at Newyork,
201 Bell Hall, Buffalo, NY 14260-2000, USA
E-mail: msj3@cse.buffalo.edu
E-mail: qiduan@cse.buffalo.edu
E-mail: jinhui@cse.buffalo.edu E-mail: shambhu@cse.buffalo.edu
*Corresponding author

Abstract: Robustness and security of services like localisation, routing and time synchronisation in Wireless Sensor Networks (WSNs) have been critical issues. Efficient mathematical (graph-theoretic) models for these services exist. Since, these services were not designed with robustness and security in mind, new mathematical models are needed to address these issues. In this paper, we propose a practical approach for modelling these services using weighted undirected graphs called Partially Consistent Grounded Graphs (PCGG). In such graphs, malicious behaviour or inaccurate information reporting is modelled as a function that assigns incorrect or inconsistent values (weights) to the edges connecting the corresponding nodes, called inconsistent edges. We formulate two optimisation problems, namely MAX-CON and LARGEST-CON. Given a PCGG, these are the problems of determining the largest induced subgraph (obtained by eliminating a subset of vertices) containing only consistent edges. MAX-CON maximises the number of vertices, while LARGEST-CON maximises the number of consistent edges of the resulting subgraph. We also discuss the practical implications of solving these problems, analyse their computational hardness and provide a comparitive analysis of heuristics-based algorithms for them.

Keywords: approximation algorithms; combinatorial optimisation; graph theory; wireless sensor networks; WSNs.

Reference to this paper should be made as follows: Jadliwala, M., Duan, Q., Xu, J. and Upadhyaya, S. (2007) 'On extracting consistent graphs in wireless sensor networks', *Int. J. Sensor Networks*, Vol. 2, Nos. 3/4, pp.149–162.

Biographical notes: Murtuza Jadliwala is a PhD Candidate in the Department of Computer Science and Engineering at The University at Buffalo, State University of New York since August 2006. He received an MS in Computer Science from the University at Buffalo, State University of New York in June 2004. He received a BE in Computer Science from the University at Mumbai, India. His primary research interests include secure and robust communication-based services (e.g. localisation and time synchronisation) in wireless data networks (wireless LANs, mobile ad hoc and sensor networks), operating system and network security, vulnerability analysis, cryptography, combinatorial optimisation and approximation algorithms, distributed artificial intelligence and multiagent systems.

Qi Duan is currently a PhD Candidate in the Department of Computer Science and Engineering of University at Buffalo, State University of New York. He received his BS from University of Science and Technology of China and Master's degree from University at Buffalo, all in Computer Science. His research interest includes security and reliability in wireless networks, applied cryptography and optimisation algorithms.

Jinhui Xu is currently an Associate Professor of Computer Science and Engineering at the State University of New York at Buffalo. He received a BS and an MS in Computer Science from the University of Science and Technology of China (USTC) and a PhD in Computer Science and Engineering from the University of Notre Dame in 2000. He joined the faculty of Computer Science and Engineering, SUNY at Buffalo, in 2000. His research interests are in the fields of algorithms (including both exact and approximation algorithms), computational geometry, combinatorial optimisation and geometric applications in biomedical imaging, treatment planning and diagnosis, networking, mobile computing and VLSI design. His research has been supported by National Science Foundation (NSF), National Institute of Health (NIH), NYSTAR, IBM and University at Buffalo. He is a recipient of the NSF CAREER Award and the IBM Faculty Partnership Award.

Shambhu J. Upadhyaya is an Associate Professor of Computer Science and Engineering at the State University of New York at Buffalo where he also directs the Center of Excellence in Information Systems Assurance Research and Education (CEISARE), designated by the National Security Agency. Prior to July 1998, he was a faculty member at the Department of Electrical and Computer Engineering. His research interests are information assurance, computer security, fault diagnosis, fault tolerant computing and VLSI testing. He has authored or co-authored more than 175 papers in refereed journals and conferences in these areas. His current projects involve intrusion detection, insider threat modelling, security in wireless networks, SoC test scheduling, analog circuit diagnosis and RF testing. His research has been supported by the National Science Foundation, Rome Laboratory, the US Air Force Office of Scientific Research, DARPA and National Security Agency.

1 Introduction

A Wireless Sensor Network (WSN) is an ad hoc network of minute devices with limited computation ability and battery power that communicate with each other using radio or ultra-sound signals. By attaching highly specialised sensors to these devices, WSNs can be used in a variety of applications including but not limited to environmental monitoring, healthcare, emergency response and military applications (Lorincz et al., 2004; Robinson, 2004; Shnayder et al., 2005; Tollefsen et al., 2004). An example of such a setup is a network of MICA2 or MICAz motes (processor modules) fitted with MDA/MTS series data acquisition modules, developed by Crossbow Technologies. The success of WSNs in these applications depends on the various services like routing, location discovery and time synchronisation. Extensive research on distributed algorithms for routing (Hong et al., 2002; Ko and Vaidya, 1998; Yu et al., 2001), location discovery (Bahl and Padmanabhan, 2000; Fang et al., 2005; He et al., 2003; Hightower and Borriello, 2001; Priyantha et al., 2000; Ray et al., 2003) and time synchronisation (Elson and Estrin, 2001; Ganeriwal et al., 2003) has already been done. Since the term 'mote' is still not an industry standard, we will refer to all such small, low power and autonomous sensing devices as 'nodes'.

As with any new technology, there are security and robustness issues associated with these services. Majority of the algorithms for these services use inter-mote communications and/or distance estimates (computed using techniques like Received Signal Strength, Angle of Arrival, Time Difference of Arrival, etc.) to neighbouring nodes, assuming that the neighbours are honest (non-malicious). This assumption is no longer valid in highly obstructive terrains and hostile environments like emergency situations, battlefields and enemy territories. Nodes can be captured and compromised or external conditions like obstructions, weather, environment, etc. may prevent the nodes from functioning normally. These factors introduce inaccuracy and errors in the measurement of physical properties like time difference, distance, etc. which we refer to as *inconsistencies*. Some research has already been done in securing location discovery and time synchronisation services for WSNs (Ganeriwal et al., 2005; Liu et al., 2005a,b). These techniques provide intelligent ways to overcome adversary. For example, Liu et al. (2005b) provides techniques for detecting malicious sensor nodes by deploying special nodes (their identities hidden) specifically for detection purposes. But, it rests on the assumption that these extra nodes will always be honest, which might not always be true. These special nodes can behave maliciously by reporting benign nodes in the network as malicious, thus rendering the entire scheme ineffective.

Due to such limitations, existing schemes are no longer viable when deployed in emergency situations and hostile environments. Researchers working in this direction are posed with important questions. Are there efficient techniques for detecting inconsistency-causing nodes with at least a high probability, if not with certainty? Do current modelling techniques for WSNs take into account inconsistencies introduced due to malicious behaviour of the nodes? Is it possible to efficiently eliminate these inconsistencies assuming they can be detected easily? In this paper, we attempt to provide answers to these questions. It may be noted that in this paper we do not intend to propose a new scheme for securing WSNs but aim to shed some light on problems that may arise when assumptions about the honest behaviour of sensor nodes are dropped.

1.1 Background and motivation

WSNs like any other network can be efficiently represented as a graph where each vertex of the graph corresponds to a sensor node and each edge represents certain association between the two nodes. In other words, an edge exists between two nodes if they are associated by some predefined, application dependent relation. One such relation, for example, is if two nodes are in the radio range of each other. Two nodes that are connected by an edge are called neighbours of each other. The relation that defines these edges varies from application to application. But in most cases, an edge exists between two nodes if they are in the radio range of each other. Eren et al. (2004) and Goldenberg et al. (2005) use a model similar to the one described above to formulate the problem of location discovery in WSNs. The graph-based model of WSN used by them is also referred to as *Grounded Graphs*. In the grounded graph model described by Eren et al. (2004), a function called the *Distance Function* assigns each edge a distance value indicating the separation between the two connecting nodes. It is similar to assigning weights or costs to edges. Depending on the application one is trying to model, this function may vary, quantifying different properties like time difference, wellness of routes, etc. For example, in routing services a routing function can assign probabilities to edges based on its usefulness to routes. Similarly, in a time synchronisation service a time function can assign each edge the time difference between the two connected

nodes. We refer to this function by a more generalised name, the *Property Function*, that assigns each edge an estimated property value. A subtle point over here is that the value of this function is only an estimate and not necessarily the true value of the property (parameter). This will become more clear later.

Eren et al. (2004) and Goldenberg et al. (2005) assume that the distance function is honest that is, it always assigns the correct or consistent distance to each edge. The main motivation for our work is to propose a more practical model by dropping this assumption. In other words, malicious or inconsistent behaviour of nodes can be modelled as a dishonest property function. This is also very intuitive as the property function for an edge is computed by the connecting nodes themselves or by using information obtained from these nodes. Thus, malicious behaviour by the nodes will be transformed into an incorrect or inconsistent value of the property function. The graph-based model with a dishonest property function, as described above, is referred by us as a Partially Consistent Grounded Graph (PCGG). To clarify how the above concepts map onto real world applications in WSNs, we give a few examples. Consider a location discovery service in which nodes determine their own locations by hearing from beacon nodes that know their own locations. By computing distances to these beacons using techniques like received signal strength, time difference of arrival, etc. nodes can compute their own location using multilateration or triangulation techniques. Beacon nodes can cheat by advertising incorrect self locations or by manipulating the power of the sent signal. This results in nodes receiving inconsistent information from these beacons and end up computing their locations incorrectly. The edges between the nodes and the malicious beacons, in this case, are the inconsistent edges. Similarly, in routing applications malicious nodes can advertise inconsistent routes to divert all the traffic through them. Such nodes eventually drop packets instead of forwarding them resulting in retransmissions and lower throughput. Thus, edges connecting the benign nodes to such malicious nodes can be labelled as inconsistent. Apart from malicious behaviour, low battery power, obstacles, extreme weather conditions, etc. can all result in inconsistencies. It can also be a result of malicious behaviour or inconsistencies propagated from remote parts of the network through communication channels. The existence of inconsistent edges implies that either or both nodes connected by these edges are responsible for the inconsistency. Either the nodes that compute the function values may have caused it or the ones that generate information for these functions may have caused it. Pires et al. (2004) and Sastry et al. (2003) present techniques for detecting and verifying such inconsistent information transmission by the nodes.

Given a PCGG, we focus on the problem of determining an induced subgraph, such that it has only consistent edges. In other words, we would like to eliminate vertices such that the resulting subgraph is fully consistent. Such a consistent subgraph is useful for various reasons. Firstly, the sparsity of the consistent subgraph can help the network administrator to make important decisions like redeployment, application abortion, etc. Secondly, with a high probability, the eliminated vertices can be assumed to be the problem causing nodes. This information is also useful during redeployment, as the deployment area previously occupied by such malicious or problem causing nodes can be avoided during the deployment of new nodes. Also, routing services can use this information to avoid particular routes while making routing decisions. Readers, please note that we will be using the terms nodes and vertices interchangeably. Depending on the context of its usage, it would either mean an actual sensor mote or its corresponding graph abstraction.

The first problem we study is that, given a PCGG, how to identify a *maximum* subset of vertices such that the subgraph induced by the vertices consists only consistent edges. We refer to this problem as the MAX-CON problem. Maximising the number of vertices is important as an administrator would like to replace or redeploy only a minimum number of new vertices. Moreover, if the sparsity (in terms of the vertices) of the resultant consistent subgraph is used to make redeployment decisions, one would like to know the maximum such consistent subgraph to justify a redeployment. The next optimisation problem we discuss is that, given a PCGG, how to identify a subset of vertices (need not be maximum) such that the subgraph induced by the vertices consists only consistent edges and the number of consistent edges is maximised. We refer to this problem as the LARGEST-CON problem. This problem is relevant in situations where quality is more important that quantity. More number of consistent edges would mean more number of accurate property (parameter) values which in turn results in better overall accuracy of applications. More importantly, the ideas and results presented in this paper not only apply to sensor networks but also can be used in any application (or domain) that can be modelled as a PCGG.

1.2 Contributions

This paper introduces PCGG as a practical modelling approach for WSNs. PCGG as discussed before, consists of an edge set which can be *partitioned* into a set of consistent and a set of inconsistent edges. We present two optimisation problems associated with PCGGs. We first provide a formal treatment of the MAX-CON problem and prove that it is NP-complete. We also show that an efficient approximation algorithm for VERTEX-COVER can be used to solve this problem.

Next, we prove that the LARGEST-CON problem is NP-complete. We give an elegant reduction from MAX-2SAT (Karp, 1972), known to be NP-hard, to this problem. We also give the inapproximability result for LARGEST-CON and provide two solution strategies for this problem. The first algorithm uses a *greedy* approach. The second one uses a technique called Local Solution Search (LSS). Currently, we are unable to bound the solution quality of these algorithms. We perform experiments to test these algorithms on randomly generated graphs and present a comparative analysis of their solution quality. In summary, we have the following main contributions.

- A formal technique for modelling WSN services and two graph-theoretic optimisation problems associated with this model.

- Analysis of the combinatorial hardness of these problems and testing the efficiency of heuristics-based algorithms for LARGEST-CON on randomly generated graphs with the help of computer simulations.

1.3 Paper organisation

Section 2 outlines the formal model of WSNs and introduces the concept of PCGG. In Section 3, we formulate and present hardness results for MAX-CON. In Section 4, we formulate and present hardness results for LARGEST-CON. Section 5 presents heuristics-based algorithms for LARGEST-CON and Section 6 provides experimentation results for these algorithms. Finally, we conclude this paper with a discussion of current contributions and proposed future work.

2 Mathematical formulation

2.1 Network model

Before introducing the graph model we define the *current state vector* of a node. The current state is the current value of a certain physical characteristic of the node and the current values of all the different physical characteristics (defined by the application) are represented in a current state vector. For example, in a location discovery service the current state vector $p_i = (x_i, y_i)$ is the current position of each node and $p_i \in \mathbb{R}^d$, where d is the dimension of the position coordinates. In a time synchronisation service the current state vector, $p_i = t_i$ s.t. $t_i \in \mathbb{R}$, is the current local time (crystal frequency) of each node. In most cases, a node i may not know p_i at the start when it is deployed, however, some special nodes may know their own p_i. For example, in a location discovery service, every beacon node knows its own position but other nodes may not and they use the location discovery service of the network to determine their own p_i. Using specialised algorithms and communications from other nodes, each node in the network determines its own current state vector p_i.

Let $N = \{1, 2, \ldots, n\}$ be the set of n nodes and let $P = \{p_1, p_2, \ldots, p_n\}$ be the set of their corresponding current state vectors. We now define the graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ for the network as follows. The set $\mathbb{V} = \{v_1, v_2, \ldots, v_n\}$ of vertices contains a vertex corresponding to each node in the network. An edge exists between two vertices i and j in the graph G if and only if the current states p_i and p_j of the corresponding nodes are associated in some way. One important assumption we make here is that this relationship is *symmetric*. For example, if two nodes are associated by the distance between them, then by symmetric we mean that if node i is in the radio range of node j, then node j is also in the radio range of node i. Thus, the edges in this graph model of the network are *undirected*. The set $\mathbb{E}$ is the set of all the edges as defined above. Two nodes are said to be neighbours if and only if there exists an edge connecting their corresponding vertices. In other words, $\mathbb{E}$ gives the neighbourhood relation for each node in the network. For simplicity we assume that the graph is a connected graph that is, every vertex is reachable from every other vertex through a sequence of edges.

The graph $\mathbb{G}$, as described above, is associated with a property function δ, $\delta : \mathbb{E} \rightarrow \mathbb{R}$, such that δ assigns a value to an edge signifying the *estimated* value of the associated property. In other words, it quantifies the associated property (defined by the edge) and is similar to assigning weights to the edges. The exact details of this function will vary from application to application and we do not discuss it further as it is out of scope of this paper. We can safely assume that such a function exists and can be efficiently computed. The *actual* property value is the true or correct property value. The difference between the property function value and the actual value is called the *estimation error*. For example, in location discovery schemes, a distance function assigns distance estimate values (distance between two nodes) to each edge. The actual property value in this case is the Euclidean distance between the two nodes. The actual property value between two nodes i and j with position coordinates $p_i(x_i, y_i)$ and $p_j(x_j, y_j)$ can be computed as shown in Equation (1)

$$\text{Euc}_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{1}$$

Similarly, in time synchronisation applications, the property function could assign an estimate of the time difference to each edge. As noted before, Eren et al. (2004) use a similar graph theoretical model called *Grounded Graphs* where the current state of each node is its position in d-dimensional space and the property function associates a value to each edge that is an estimate of the distance between the two nodes. What we have described above is a more generalised version of this model.

2.2 Partially consistent grounded graph

In the definition of grounded graphs proposed by Eren et al. (2004), they implicitly assume that the property function δ (distance function) that estimates the distance $d_{i,j}$ between the nodes i and j is honest that is,

$$\delta(i, j) = d_{i,j} \text{ s.t. } (\text{Euc}_{i,j} - \epsilon) \leq d_{i,j} \leq (\text{Euc}_{i,j} + \epsilon), \forall i, j$$

Here ϵ, called the error tolerance factor, is the maximum estimation error allowed by the application. Ideally, the value of ϵ should be zero but in most practical cases it is assumed to have a very small value. More generally, $\delta(i, j)$ is always assumed to be within the actual property value $\pm\ \epsilon$, $\forall\ i, j$ that is, the value of the property function is very close to the actual property value. We now introduce the notion of consistent and inconsistent edges where this assumption about the honest property function is dropped.

Definition 1: *Consistent edge: an edge (i, j) in the grounded graph is said to be consistent if and only if the estimated property value (value of the property function) associated with it is within some small error tolerance factor ϵ.*

An edge that is not consistent is said to be an *inconsistent edge* and the property function that assigns inconsistent values to edges is called a *dishonest property function*. At this point, there are three important observations we make.

- The implementation of the property function varies from application to application. But, property function computations normally use information collected from the nodes and other inter-node communications in the network. Thus, cheating or dishonest behaviour in part by the nodes would result in an incorrect value being assigned by the function to the corresponding edge. This incorrect value can also be due to other factors (low battery power, obstacles, extreme weather conditions, etc.) and not necessarily due to cheating by the nodes but it is not possible to differentiate between the two cases. The eventual goal of this research is to effectively eliminate the inconsistencies. We will not be concerned with how they are caused.
- Nodes do not behave maliciously all the time. Malicious behaviour is *random*. In other words, not all edges coming out of a particular node will be inconsistent. If they do, then such a behaviour is trivial to detect. Nodes will behave maliciously at random and intermittently to avoid easy detection. There will be some nodes that are an exception to this rule but we assume that their numbers are small.
- In this paper, we assume that inconsistent edge values for most applications in WSNs can be detected efficiently. We do not go into the details of how this can be acheived as it is out of scope of this paper. Some techniques for inconsistency/malicious behaviour detection are discussed by Sastry et al. (2003), Ray et al. (2003), Liu et al. (2005b) and Pires et al. (2004).

We are now ready to define a PCGG.

Definition 2: *PCGG: a PCGG $G' = (V', E' \cup E'')$ is a grounded graph associated with a dishonest or malicious property function. The set of vertices is denoted by V' and edge set can be partitioned into two disjoint subsets, namely the set of consistent edges (E') and the set of inconsistent edges (E'').*

Definition 3: *Consistent Subgrounded Graph (CSG): a CSG $G = (V, E)$ is an induced subgraph of a PCGG $G' = (V', E' \cup E'')$, where $E'' \neq \phi$, such that the vertex set $V \subset V'$ and the edge set E contains only consistent edges that is, $E \subseteq E'$.*

A CSG is obtained by eliminating vertices (and the corresponding edges) from a PCGG such that the resulting induced subgraph is consistent. The *size* of a CSG is the cardinality of its vertex set. The *edge size* of a CSG is the cardinality of its edge set. A CSG is *maximal* if its vertex set is not a proper subset of the vertex set of any other CSG. A *maximum* CSG is a maximal CSG with maximum size.

Definition 4: *Largest CSG: the Largest Consistent Subgrounded Graph (LCSG) of a PCGG is a CSG that has the maximum edge size, if more than one CSG exists.*

Figure 1(a) shows a PCGG $G' = (V', E' \cup E'')$, Figure 1(b) its corresponding maximum CSG and Figure 1(c) its largest CSG. In the next two sections, we formulate two optimisation problems for obtaining a CSG from a PCGG.

3 Maximum consistent subgrounded graph

3.1 Problem statement

The maximum CSG problem can be stated as follows. Given a PCGG $G' = (V', E' \cup E'')$, find the maximum CSG $G(V, E)$ of G'. This problem is denoted by MAX-CON. All the notations have the same meaning as discussed before. The problem can be alternatively stated as the problem of eliminating a minimum number of vertices from G' such that the subgraph induced by the remaining vertices consists of only consistent edges. MAX-CON is an optimisation problem and the decision version can be stated as:

MAX-CON
Input: A PCGG $G' = (V', E' \cup E'')$ and a positive integer k s.t. $k \leq \|V'\|$.
Question: Does G' contain a CSG of size k or more?

3.2 Hardness of MAX-CON

In this section, we show that MAX-CON is NP-complete. This result implies that MAX-CON $\in$ NP and the deterministic complexity of MAX-CON is as hard as any problem in NP. Thus, MAX-CON does not have a deterministic polynomial time solution. We prove this result by a polynomial time many-one reduction from VERTEX-COVER. VERTEX-COVER problem is a well known NP-complete problem. A vertex cover of an undirected graph $G = (V, E)$ is a subset of vertices $C \subseteq V$ that contains at least one vertex of every edge $e \in E$ and the VERTEX-COVER problem (also called minimum vertex cover problem) is to find such a subset C of the smallest cardinality. VERTEX-COVER, NP-completeness and polynomial time many-one reductions are explained in the seminal paper by Karp (1972). Before proceeding ahead we would like to state the decision version of VERTEX-COVER (Homer and Selman, 2001).

VERTEX-COVER
Input: A graph $G = (V, E)$ and a positive integer k s.t. $k \leq \|V\|$.
Question: Is there a vertex cover of size $\leq k$ for G?

Theorem 1: *MAX-CON is NP-complete.*

Proof: It is easy to see that MAX-CON $\in$ NP: Given a graph $G' = (V', E' \cup E'')$, guess a set of vertices V (s.t. $\|V\| \geq k$) and check whether the subgraph induced by V consists of only consistent edges (i.e. all the induced edges only belong to the set E'). This clearly can be done deterministically in polynomial time, provided it can be decided whether an edge is inconsistent or not in polynomial time. Now we show that VERTEX-COVER $\leq_m^P$ MAX-CON, that is, VERTEX-COVER many-one (m) reduces in polynomial time (P) to the MAX-CON problem.

Construction: we describe a polynomial time construction that maps an instance $G = (V, E)$ of the VERTEX-COVER problem to an instance $G' = (V', E' \cup E'')$ of the MAX-CON problem such that G has a vertex cover of size $\leq k$ ($k \leq \|V\|$) if and only if G' has a CSG of size $\geq \|V\| - k$. The construction is shown in Figure 2.

Figure 1 (a) PCGG, $G' = (V', E' \cup E'')$; (b) $\text{OPT}_{\text{MAX-CON}}$ and (c) $\text{OPT}_{\text{LARGEST-CON}}$

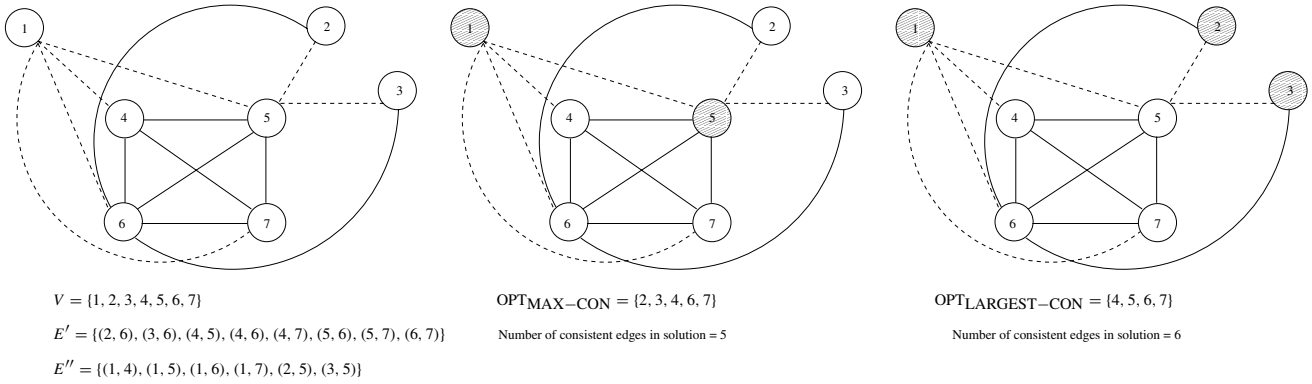
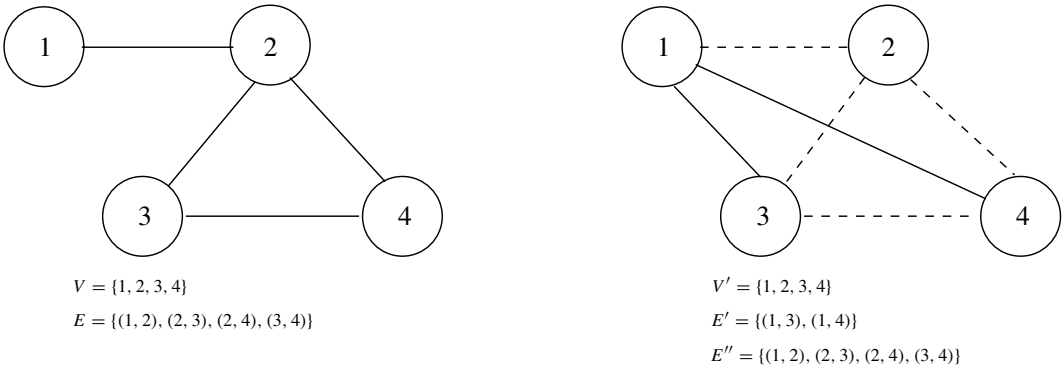


Figure 2 (a) Input graph for the VERTEX-COVER problem, $G = (V, E)$ and (b) input graph for the MAX-CON problem, $G' = (V', E' \cup E'')$



1 For each vertex v in the vertex set V of G, place a vertex v in the vertex set V' of G'.

2 For each edge $(u, v) \in E$ s.t. $u, v \in V$, add an edge (u, v) in the inconsistent edge set E'' of G'. These edges are shown as dotted lines in Figure 2(b).

3 For each edge $(u, v) \notin E$ s.t. $u, v \in V$, add an edge (u, v) in the consistent edge set E' of G'. These edges are shown as solid lines in Figure 2(b).

It is clear that the above construction can be completed in polynomial time. We now show that the graph G has a vertex cover of size k if and only if the graph G' has a CSG of size $\|V\| - k$.

Suppose the graph G in Figure 2 has a vertex cover C ($C \subseteq V$) of size k ($\|C\| = k$). Since C is a vertex cover, $\forall(u, v) \in E$, either u or v or both are in C. By our construction, $\forall(u, v) \in E$, $(u, v) \in E''$ (inconsistent edge set). Thus, C also covers all the inconsistent edges in G'. In other words, $V - C$ is a CSG. $\|V - C\| = \|V\| - k$. Thus, if G has a vertex cover of size k, G' has a CSG of size $\|V\| - k$.

Now we prove the other direction. Let C' be the CSG of G' of size m ($m \leq \|V'\|$). By definition of CSG, C' contains only consistent edges that is, for all edges (u, v) in C', $(u, v) \in E'$. Thus, $V' - C'$ covers all edges in the inconsistent edge set E''. If this was not true, that means there is an edge $(u, v) \in E''$ s.t. both u and v are not in $V' - C'$. Thus, both u and v are in C' and it is not a CSG which is a contradiction. Thus, $V' - C'$ covers all inconsistent edges. From our construction, $V' - C'$ is a vertex cover of the graph G (there is a one-one mapping of edges in G to inconsistent edges in G') and its size is $\|V'\| - m$ that is, $\|V\| - m$ since $\|V'\| = \|V\|$.

Thus, VERTEX-COVER many-one reduces in polynomial time to MAX-CON. Since VERTEX-COVER is NP-complete, MAX-CON is NP-complete.

3.3 *Approximation algorithm for MAX-CON*

Lemma 1: *MAX-CON many-one reduces in polynomial time ($\leq_m^P$) to the VERTEX-COVER Problem.*

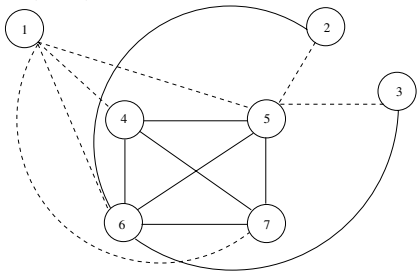
Proof: The proof of this lemma has a construction that is very similar to the one in Theorem 1. This construction maps an instance $G' = (V', E' \cup E'')$ of the MAX-CON problem to an instance $G = (V, E)$ of the VERTEX-COVER problem in polynomial time such that G' has a CSG of size k ($k \leq \|V'\|$) if and only if G has a vertex cover of size $\|V'\| - k$.

1 For each vertex v in the vertex set V' of G', place a vertex v in the vertex set V of G.

2 For each inconsistent edge $(u, v) \in E''$ add an edge (u, v) in the edge set E of G. These edges are shown as dotted lines in Figure 3(a) and as solid lines in Figure 3(b).

It is clear that the above construction can be completed in polynomial time. We now show that G' has a CSG of size k (for any $k \leq \|V'\|$) if and only if G has a vertex cover of size $\|V\| - k$.

Figure 3 (a) Input graph for the MAX-CON problem, $G' = (V', E' \cup E'')$ and (b) input graph for the VERTEX - COVER problem, $G = (V, E)$

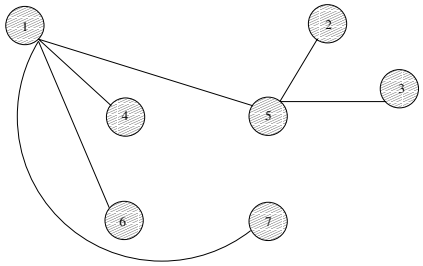


$V' = \{1, 2, 3, 4, 5, 6, 7\}$

$E' = \{(2, 6), (3, 6), (4, 5), (4, 6), (4, 7), (5, 6), (5, 7), (6, 7)\}$

$E'' = \{(1, 4), (1, 5), (1, 6), (1, 7), (2, 5), (3, 5)\}$

(a)



$V = \{1, 2, 3, 4, 5, 6, 7\}$

$E = \{(1, 4), (1, 5), (1, 6), (1, 7), (2, 5), (3, 5)\}$

(b)

Suppose G' has a CSG C of size k. This implies that C contains only consistent edges that is, edges from the edge set E'. Thus, $V' - C$ contains all the inconsistent edges (from E'') and the remaining consistent edges (from E'). Also, $\|V' - C\| = \|V'\| - k$. By our construction $E = E''$ and $V = V'$. Thus $V' - C$ covers all edges in E and is a vertex cover of size $\|V'\| - k$. Similarly, the other direction.

Lemma 1 implies that any efficient algorithm for solving the VERTEX-COVER problem can be used as a subroutine to solve the MAX-CON problem. The minimum VERTEX-COVER problem is a fundamental problem in graph theory and combinatorial optimisation and is a vastly studied problem with a large number of constant and fixed ratio approximation algorithms. Hastad (1997) has shown that VERTEX-COVER cannot be approximated within a factor of 7/6. It was further improved to $10\sqrt{5} - 21$ by Dinur and Safra (2005). Gavril introduced a 2-approximation algorithm for the VERTEX-COVER problem in Garey and Johnson (1979). This was improved to $2 - (\text{loglog}|V|)/(2\text{log}|V|)$ (Bar-Yehuda and Even, 1985; Monien and Speckenmeyer, 1985) and later to $2 - (\text{lnln}|V|)/(\text{ln}|V|)(1 - o(1))$ (Halperin, 2000) before it was eventually improved to $2 - \Theta(1/\sqrt{\log n})$ by Karakostas (2005). An interesting generalisation of the VERTEX-COVER problem is the weighted VERTEX-COVER problem in which positive weights are assigned to each vertex and the problem is to find the vertex cover with minimum cumulative weight. The first well-known 2-approximation algorithms for the weighted VC problem were discovered independently by Bar-Yehuda and Even (1981) and Hochbaum (1982). An important point to note here is that all the approximation results for the unweighted case also hold for the weighted case.

We are now ready to state the approximation algorithm for the MAX-CON problem. The approximation algorithm for MAX-CON is as shown in Algorithm 1. Let $A(V, E)$ be an algorithm for solving the VERTEX-COVER problem, where V and E are the set of vertices and edges respectively of the input graph G. Algorithm A returns the set of vertices that form the minimum vertex cover for the graph G. The approximation algorithm for MAX-CON is simple and uses the approximation algorithm A for the VERTEX-COVER problem as a subroutine. The for loop runs no more than $\binom{\|V'\|}{r}$ times. Also, the running time and solution quality of Algorithm 1 is bounded by the running time and solution quality of algorithm A.

Algorithm 1 Calculating the maximum CSG of the PCGG $G' = (V', E' \cup E'')$

```
E ⇐ E'' {place all inconsistent edges in E}
for all edge (u, v) ∈ E'' do
  if u ∉ V then
    V ← u {and corresponding vertices in V}
  end if
  if v ∉ V then
    V ← v
  end if
end for
C = A(V, E) {execute approx algorithm for VERTEX-COVER}
return V' − C {solution of MAX-CON}
```

4 Largest consistent subgrounded graph

4.1 Problem statement

The largest CSG problem can be stated as, given a PCGG $G' = (V', E' \cup E'')$, to find the largest CSG $G(V, E)$ (Definition 4) of G'. This problem is denoted by LARGEST-CON. The problem can be alternatively stated as the problem of eliminating vertices from G' in such a way that the subgraph induced by the remaining vertices consists of only consistent edges and the cardinality of these edges is maximised. From Figure 1, we can clearly see that an optimal solution for MAX-CON is not necessarily an optimal solution for LARGEST-CON. These two are different problems with different combinatorial hardness and approximation schemes. The decision version of the problem can be stated as:

LARGEST-CON
Input: A PCGG $G' = (V', E' \cup E'')$ and a positive integer k s.t. $k \leq \|E'\|$.
Question: Does G' contain a CSG of edge size k or more?

4.2 Hardness of LARGEST-CON

In this section, we show that LARGEST-CON is NP-complete. This result implies that LARGEST-CON $\in$ NP

and the deterministic complexity of LARGEST-CON is as hard as any problem in NP. Thus, LARGEST-CON does not have a deterministic polynomial time solution. We prove this result by a polynomial time many-one reduction from MAX-2SAT or Maximum 2-Satisfiability. MAX-2SAT is a known NP-complete problem (Garey and Johnson, 1979). MAX-2SAT is a restricted version of another NP-complete problem called the Maximum Satisfiability or MAX-SAT. MAX-SAT is the problem, given a set S of disjunctive form clauses, to find a truth assignment to the literals such that maximum number of clauses are satisfied (Garey and Johnson, 1979). MAX-2SAT is restricted to at-most two literals per clause. It can be formally stated as:

MAX-2SAT
Input: A Conjunctive Normal Form (CNF) formula F on Boolean variables $x_1, x_2, \ldots, x_n$ and m clauses $C_1, C_2, \ldots, C_m$, each containing at most two literals, where each literal is either Boolean variable x_i or its negation $\bar{x_i}$ ($\neg x_i$) and a positive integer k ($k < m$).
Question: Is there a truth assignment to the variables that satisfies k or more clauses?

Theorem 2: *LARGEST-CON is NP-complete.*

Proof: We use a technique similar to the polynomial time reduction from 3-SAT used to prove the NP-completeness of VERTEX-COVER problem (Homer and Selman, 2001). It is easy to see that LARGEST-CON $\in$ NP: Given a graph $G' = (V', E' \cup E'')$, guess a set of consistent edges E (s.t. $\|E\| \geq k$ and $E \subseteq E'$). Let V be the set of vertices of all these guessed edges. Check in polynomial time whether the other edges induced by V are consistent. This procedure clearly can be accomplished in polynomial time and thus LARGEST-CON $\in$ NP. Now we show that MAX-2SAT $\leq_m^P$ LARGEST-CON, that is, MAX-2SAT many-one reduces in polynomial time to LARGEST-CON. We can then claim that since MAX-2SAT is NP-complete, LARGEST-CON is NP-complete.

Construction of $G' = (V', E'UE'')$: we describe a polynomial time construction that maps an instance F of MAX-2SAT to an instance $G' = (V', E'UE'')$ of the LARGEST-CON problem such that F satisfies k clauses if and only if G' has a CSG of edge size k. Figure 4 shows the construction of a PCGG $G' = (V', E' \cup E'')$ from the MAX-2SAT formula $F = (x_1 \vee x_2) \wedge (x_2 \vee x_3) \wedge (x_1 \vee \bar{x_3}) \wedge (\bar{x_1} \vee \bar{x_2})$. The consistent edges are shown as solid lines and the inconsistent edges are shown as dotted lines.

The construction of G' consists of the following three steps, each adds a different component to the graph.

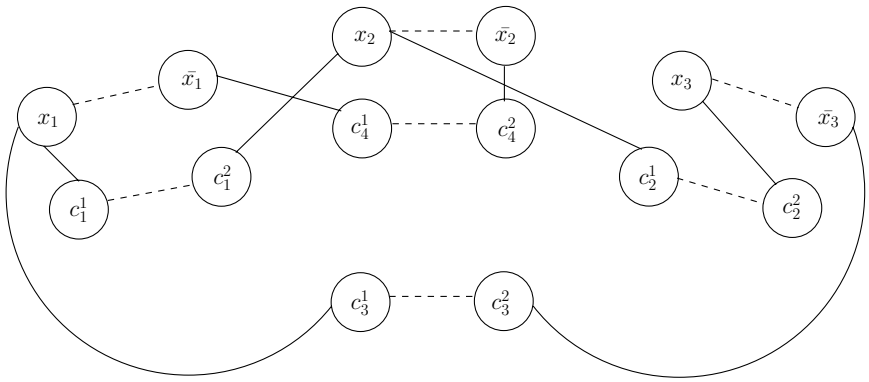
1 Let $U = \text{VAR}(F)$, be the set of variables in the Boolean formula F. For each variable $u_i \in U$, put u_i and $\bar{u_i}$ in the vertex set V' and put $(u_i, \bar{u_i})$ into the edge set E'' that is, the set of inconsistent edges in graph G'. This is the first component of the graph.

2 Let $C = \text{CLAUSE}(F)$ be the set of clauses in F that is, $F = \bigwedge_{c_j \in C} c_j$. For each clause c_j in the formula F put vertices c_j^1 and c_j^2 in V'. Put an edge (c_j^1, c_j^2) in the set E'' that is, the set of inconsistent edges. This is the second component of the graph G'.

3 In this step we create a new component by connecting components from the first two steps. This component depends on the literals that are contained in the clauses. As mentioned before, each clause $c_j \in C$ is a disjunction of two literals and literals are variables or their negations. Consider one such clause $c_j = (x_j \vee y_j)$, where x_j and y_j are literals. For each clause c_j, put edges (x_j, c_j^1) and (y_j, c_j^2) in E' that is, the set of consistent edges of G'. This forms the third set of components of the graph G'.

We need to show that PCGG G' has a CSG $G = (V, E)$ of edge size k that is, $\|E\| = k$ if and only if F has k satisfiable clauses. Suppose, there exists an assignment t s.t. exactly k clauses are satisfied. Then for each variable $u_i \in U$ either $t(u_i) = 1$ or $t(\bar{u_i}) = 1$ but both cannot be 1. Place u_i in the vertex set V of the subgraph G of the PCGG G' if $t(u_i) = 1$ or place $\bar{u_i}$ in V if $t(\bar{u_i}) = 1$. Thus, V contains one vertex of each edge in the first component. Now, for a clause $c_j = (x_j \vee y_j)$, c_j is satisfiable if either literals x_j or y_j or both are true. Thus, either x_j or y_j or both are in the set V based on their truth assignment. If both x_j and y_j are in V, randomly (with a probability 1/2) select vertex c_j^1 or c_j^2 and add it to V (never both). If only one of x_j or y_j is 1, pick the corresponding c_j^i (based on the construction of component 3) and place it V. One thing to note here is that when the clause c_j is satisfied, only one c_j^1 or c_j^2 is in the set V. When it is not satisfied none of them are in V. It follows that the vertex set V induces edges only from E'. Thus the graph induced by V is consistent and $G = (V, E)$ is a CSG. Also from the above

Figure 4 Construction of a PCGG $G' = (V', E' \cup E'')$ from the MAX-2SAT formula $F = (x_1 \vee x_2) \wedge (x_2 \vee x_3) \wedge (x_1 \vee \bar{x_3}) \wedge (\bar{x_1} \vee \bar{x_2})$



procedure, if k clauses are satisfied then exactly k consistent edges get induced in E. Thus $G = (V, E)$ is a CSG with edge size k. We now prove the other direction.

Suppose, $G = (V, E)$ is a CSG of the PCGG G' s.t. $\|E\| = k$, for some positive integer k. From the construction it is clear that all the consistent edges are of the form (u_i, c_j^i), where u_i is the ith ($i \in 1, 2$) literal in the jth clause c_j of the formula F. Also, if $u_i \in \{x_i, \bar{x_i}\}$, since the graph G is a CSG, the edges of the form $(x_i, \bar{x_i})$ and (c_j^1, c_j^2) cannot be in E, that is, both x_i and $\bar{x_i}$ or c_j^1 and c_j^2 cannot be in the vertex set V of the CSG G. Now define an assignment $t : U \rightarrow \{0, 1\}$ s.t. $t(u_i) = 1$ if $u_i \in V$ and $t(u_i) = 0$ if $u_i \notin V$. Similarly, $t(\bar{u_i}) = 1$ if $\bar{u_i} \in V$ and $t(\bar{u_i}) = 0$ if $\bar{u_i} \notin V$. We claim that this assignment is consistent and if there are k edges in E then there are k satisfied clauses by the above assignment. Since G is a CSG of G', none of the edges in the first two components of our construction can be present in G. Thus, for any variable x_i, both x_i and $\bar{x_i}$ cannot be in V. As a result the assignment t above will consistent. Similarly, for a clause c_i, both c_i^1 and c_i^2 cannot be in V. Thus, both edges in the third component of our construction above of the form (u, c_j^1) and (v, c_j^2) cannot be in G at the same time, where u and v are some literals. If this was not true, (c_j^1, c_j^2) would also be induced in G making it inconsistent. Thus, there is a one-one correspondence between an edge in G and the corresponding satisfied clause and since all these edges span distinct clauses there are exactly k satisfied clauses.

Lemma 2: *Let $OPT_{LAR}(G')$ and $OPT_{MAX}(G')$ be the optimal solutions of the LARGEST-CON and MAX-CON problems resp. on any input PCGG G'. Let $LAR(G')$ and $MAX(G')$ be the set of all the feasible solutions for the LARGEST-CON and MAX-CON problems resp. on G'. Then,*

1 *$Cost(OPT_{LAR}(G')) \leq cost(OPT_{MAX}(G'))$ that is, $\|OPT_{LAR}(G')\| \leq \|OPT_{MAX}(G')\|$.*

2 *$LAR(G') = MAX(G')$.*

3 *Let, $A \in LAR(G')$ ($MAX(G')$) and $B \in LAR(G')$ ($MAX(G')$) s.t. $C = A \cap B$ and $C \neq \phi$. Then, $A \cup B \notin LAR(x)$ ($MAX(x)$) $\Rightarrow \exists u \in A - C$ and $\exists v \in B - C$ s.t. $(u, v) \in E''$ that is, in the set of inconsistent edges of G'.*

Proof:

1 Assume that there exists a PCGG x such that the inequality 1 above is not true. This means, that there exists an optimal solution of LARGEST-CON that has more vertices than an optimal solution for MAX-CON. But, then the solution for MAX-CON is not a maximum CSG, thus not optimal and that is a contradiction. Thus, no such PCGG exists.

2 The proof of this point is trivial and follows directly from the definitions of MAX-CON and LARGEST-CON.

3 Since, A and B are both feasible solutions, there are no two vertices u and v both in A s.t. (u, v) is inconsistent. Similarly, there are no such vertices in either B or C. Also, since $A \cup B$ is not a feasible solution which implies that there exists two vertices $u, v \in (A \cup B)$ s.t (u, v) is inconsistent. The above two points imply that $\exists u \in A$ and $v \in B$ s.t. (u, v) is inconsistent.

4.3 Inapproximability of LARGEST-CON

In this section, we show the inapproximability of LARGEST-CON. CLIQUE is a well known problem and is one of the first problems shown to be NP-complete (Bomze et al., 1999). It can be stated as (Homer and Selman, 2001):

CLIQUE
Input: A graph $G = (V, E)$ and a positive integer $j \leq \|V\|$.
Question: Does G contain a clique of size j or more?

Bomze et al. (1999) have listed important combinatorial results for the CLIQUE problem. In summary, there is strong evidence that clique does not have a polynomial time approximation algorithm. In other words, unless $P = \text{NP}$, CLIQUE cannot be approximated with any ratio less than 1.

Thoerem 3: *If there exists an approximation algorithm that can approximate LARGEST-CON with an approximation ratio ε, then there exists an algorithm that approximates CLIQUE with ratio $1 - \sqrt{1 - \varepsilon/2}$.*

Proof: Suppose we have an instance of CLIQUE, $G = (V, E)$. We can construct a new graph $G' = (V, E')$ such that the new graph G' has the same vertex set as G and $E' = E \cup E^c$ where E^c contains all the edges that are not in E (in the complete graph induced by the vertex set). Now, if we take E to be the set of consistent edges and E^c to be the set of inconsistent edges, then G' is a PCGG. Also, it is easy to see that any CLIQUE in graph G corresponds to a CSG in G' and vice versa. Let A be the ε-approximation algorithm for solving the LARGEST-CON problem. We apply A on the graph G' to get the largest CSG in G'. Let this largest CSG be $\hat{G} = (\hat{V}, \hat{E})$. Also, let $|\hat{V}| = m$ and M be the vertex cardinality of the optimal solution.

Since A has an approximation ratio ε, we have

$$\left| \frac{\binom{M}{2} - \binom{m}{2}}{\binom{M}{2}} \right| = \frac{M^2 - M - m^2 + m}{M^2 - M} \leq \varepsilon$$

Then,

$$1 - \varepsilon \leq \frac{m^2 - m}{M^2 - M} < \frac{m^2}{M^2 - M}$$

Without loss of generality we can assume $M \geq 2$. Then we have,

$$1 - \varepsilon < 2\left(\frac{m}{M}\right)^2 \text{ which is, } \frac{M - m}{M} < 1 - \sqrt{\frac{1 - \varepsilon}{2}}$$

This means we have found an approximation algorithm for CLIQUE with ratio $1 - \sqrt{1 - \varepsilon/2}$.

Corollary 1: *Unless $P = NP$, the approximation threshold of LARGEST-CON is 1.*

Proof: This directly follows from the fact that CLIQUE cannot be approximated with any ratio less than 1 under the hypothesis $P \neq \text{NP}$.

The above corollary implies that LARGEST-CON cannot be approximated with any ratio less than 1 under the hypothesis $P \neq \text{NP}$.

5 Heuristics-based algorithms for LARGEST-CON

In this section, we present approximation algorithms for LARGEST-CON based on well known heuristics. Currently, we do not know if the quality of the solutions produced by these algorithms can be bounded.

5.1 Greedy algorithm

Let $G' = (V', E' \cup E'')$ be an instance of LARGEST-CON, where E' and E'' are the sets of consistent and inconsistent edges, respectively. For a vertex $v \in V'$, let $\text{con}(v)$ be the number of consistent edges of v and let $\text{incon}(v)$ be the number of inconsistent edges of v. A greedy approach for obtaining the largest CSG of G' is shown in Algorithm 2.

Algorithm 2 Greedy algorithm

```
C ⇐ φ; {Initialise the solution to empty set}
C = {v| v ∈ V' and incon(v) = 0}
V' ← V'\C;
while E'' ≠ φ do
   pick a vertex v ∈ V' of minimum con(v);
   V' ← V'\{v}
   E'' ← E''\{e|v ∈ e}
end while
C ← C + V';
return C {solution of LARGEST-CON}
```

This approach eliminates a vertex of inconsistent edge degree at least one and minimum consistent edge degree in each iteration. The greedy approach is pretty straightforward, with a running time bounded by the execution of the while loop that is $O(n^2)$ where $|V| = n$. The above algorithm can be modified slightly as shown in Algorithm 3. It picks a vertex v with minimum $\text{con}(v)\text{incon}(v)$ that is, eliminates vertices with the lowest ratio of consistent to inconsistent edges.

Algorithm 3 Modified greedy algorithm

```
C ⇐ φ; {Initialise the solution to empty set}
C = {v| v ∈ V' and incon(v) = 0}
V' ← V'\C;
while E'' ≠ φ do
   pick a vertex v ∈ V' of minimum con(v)/incon(v);
   V' ← V'\{v}
   E'' ← E''\{e|v ∈ e}
end while
C ← C + V';
return C {solution of LARGEST-CON}
```

5.2 Local solution search

LSS is an algorithm design technique for optimisation problems. Before giving details on this technique we introduce a few important concepts. Let U be an optimisation problem and x be an input problem instance for U. Let $M(x)$ be the set of feasible solutions of the problem U for the input instance x.

Definition 5: *Neighbourhood: For an optimisation problem U and for every input instance x, a neighbourhood on the set of feasible solutions ($M(x)$) is any mapping $f_x : M(x) \rightarrow Pot(M(x))$ (Pot denotes the power set) such that*

1 *$\alpha \in f_x(\alpha)$ for every $\alpha \in M(x)$*

2 *if $\beta \in f_x(\alpha)$ for some $\alpha \in M(x)$, then $\alpha \in f_x(\beta)$ and*

3 *for all $\alpha, \beta \in M(x)$ there exists a positive integer k and $\gamma_1, \ldots, \gamma_k \in M(x)$ such that $\gamma_1 \in f_x(\alpha)$, $\gamma_{i+1} \in f_x(\gamma_i)$ for $i = 1, \ldots, k-1$, and $\beta \in f_x(\gamma_k)$.*

If $\alpha \in f_x(\beta)$ for some $\alpha, \beta \in M(x)$, we say that α and β are neighbours in $M(x)$. The set $f_x(\alpha)$ is called the neighbourhood of the feasible solution α in $M(x)$ (Hromkovič, 2004).

We now introduce the concept of local optima.

Definition 6: *Let U be an optimisation problem and let for every input instance x, the function f_x be the neighbourhood on $M(x)$. Let cost be the cost function that assigns a positive real number to each feasible solution. A feasible solution $\alpha \in M(x)$ is a local optima for the input instance x of U according to f_x, if*

$$cost(\alpha) = (max) \text{ or } (min)\{cost(\beta) | \beta \in f_x(\alpha)\} \qquad (2)$$

We denote the set of all local optima for x according to the neighbourhood f_x by $LocOPT_U(x, f_x)$ (Hromkovič, 2004).

Neighbourhood definition for LARGEST-CON: The formalisms of functions and relations does not work when introducing neighbourhoods on $M(x)$ in practical problems like LARGEST-CON. The standard way to introduce a neighbourhood on $M(x)$ is to use a so-called *local transformation* on $M(x)$. Informally, a local transformation transforms a feasible solution α to a feasible solution β by some local changes of the specification of α. To define a neighbourhood for an instance of the LARGEST-CON problem, we introduce a transformation called a *n-neighbourhood* transformation. For simplicity, we first introduce a 1-*neighbourhood* transformation. Let $x = G'(V', E' \cup E'')$ be an instance of the LARGEST-CON problem. Let $M(x)$ be the set of feasible solutions for LARGEST-CON on input x. For $\alpha \in M(x)$, we define the 1-neighbourhood of α as follows.

To define a 1-neighbour of a feasible solution α, pick a vertex $v \in V' \backslash \alpha$ s.t. v has an inconsistent edge degree of exactly one and this inconsistent edge connects v to a vertex in α. Let this vertex in α be called w. If there are no such vertices then α has no 1-neighbours. Now to get a 1-neighbour of α, add v in α and remove w from α. It is clear that this resultant subgraph is also a feasible solution since the inconsistent edge which was covered by v previously is now covered by w. Also, addition of v does not induce any inconsistent edge in the resultant subgraph since its inconsistent edge degree is one and that edge is covered by w. We call this a 1-*neighbour* of the solution α.

The set of all the 1-neighbours of α is called the 1-neighbourhood of α and is represented as $\text{Neigh}_x^1(\alpha)$. Similarly, to define a 2-neighbourhood, a vertex $v \in V' \backslash \alpha$ with inconsistent edge degree of exactly two (to vertices in α) is selected. This vertex is added in α and the two vertices that v connects by inconsistent edges are removed from α. One thing to note here is that 1-neighbours of α have the same vertex set cardinality as α while its 2-neighbours have their vertex set cardinality reduced by 1. Similarly 3-*neighbourhoods* are defined.

LSS Algorithm for LARGEST-CON: Roughly speaking, a LSS algorithm starts off with an initial solution and then continually tries to find a better solution by searching neighbourhoods. If there is no better solution in the neighbourhood, then it stops. Having a structure on the set of feasible solutions $M(x)$ determined by a neighbourhood Neigh_x for every input instance x of an optimisation problem U, one can describe a general scheme of local search as shown in Algorithm 4.

Algorithm 4 Local Search Scheme according to a neighbourhood Neigh

Find a feasible solution $\alpha \in M(x)$
while $\alpha \notin LocOPT_U(x, Neigh_x)$ **do**
  find a $\beta \in Neigh_x(\alpha)$ such that $cost(\beta) < cost(\alpha)$ if U is a minimisation problem or $cost(\beta) > cost(\alpha)$ if U is a maximisation problem;
  If such a β is found, $\alpha = \beta$;
end while
return α

The success of a local search algorithm depends on the choice of the neighbourhood. If a neighbourhood Neigh_x has the property that $\text{Neigh}_x(\alpha)$ has a small cardinality for every $\alpha \in M(x)$, then one iterative improvement of the while loop of Algorithm 4 can be executed efficiently but the risk that there are many local optima (potentially with a cost that is very far from the optimal solution) can substantially grow. On the other hand, large $|\text{Neigh}_x(\alpha)|$ can lead to feasible solutions with costs that are closer to the optimal solution than smaller neighbourhoods can, but the complexity of the execution of one run of the while cycle can increase too much. Besides the choice of the neighbourhood there are two other factors that affect the execution of the local search algorithm. The first factor is the method by which the initial feasible solution is computed. The choice of the initial solution can essentially influence the quality of the resultant local optimum. The initial feasible solution can be either chosen randomly for problems in which the structure of the feasible solution is simple or it can be precomputed. In the LSS algorithm for LARGEST-CON the initial feasible solution is precomputed. From Lemma 2 we know that a solution for the MAX-CON problem is also a solution for the LARGEST-CON problem. Thus, any algorithm that produces an optimal solution for MAX-CON can be used as a good starting solution for the LARGEST-CON problem. Further improvement can be done by starting the LSS algorithm with multiple initial feasible solutions. The second factor affecting the performance of the LSS algorithm is the way in which a cost-improving feasible solution is selected inside the while loop. There are two strategies in doing this, namely, *first improvement* and *best improvement*. The first improvement strategy means that the current feasible solution is replaced by the first cost-improving feasible solution found by the neighbourhood search. The best improvement strategy replaces the current feasible solution by the best feasible solution in the neighbourhood. A LSS for solving LARGEST-CON is outlined in Algorithm 5.

Algorithm 5 Local Search Scheme for LARGEST-CON using $Neigh_x^1$

Let $x = G'(V', E' \cup E'')$ be a PCGG and an instance of LARGEST-CON and let A be an efficient algorithm for solving MAX-CON.
Let $\alpha = A(x)$ be the initial feasible solution.
while $\alpha \notin LocOPT_U(x, Neigh_x^1(\alpha))$ **do**
  Either by first improvement or best improvement, find a $\beta \in Neigh_x^1(\alpha)$ such that $cost(\beta) > cost(\alpha)$ {cost function outputs the edge count (consistent) of a solution}
  If such a β is found, $\alpha = \beta$;
end while
return α

One shortcoming of this approach is that in one iteration of the while loop only the 1-neighbourhood (Neigh_x^1) of the feasible solution α is checked. But α might not have 1-neighbourhoods at all or there might be better solutions in the 2-neighbourhoods and 3-neighbourhoods. The above algorithm can be further improved by also checking the 2-neighbourhoods and 3-neighbourhoods.

6 Experimental evaluation

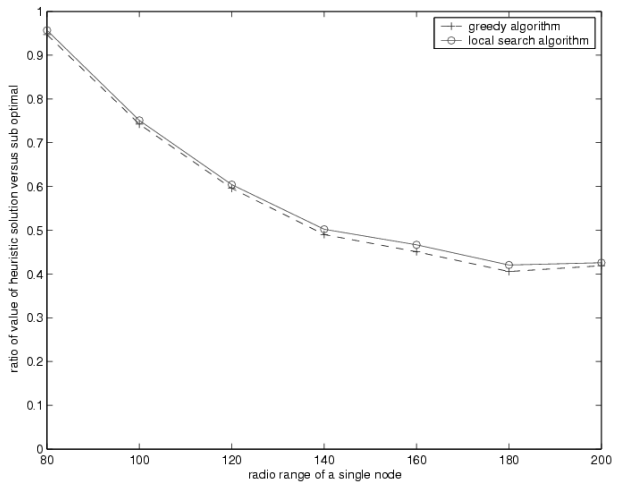
We implemented the greedy and LSS algorithms in C++ language and tested them on randomly generated graphs. The random graphs are generated in the following way: All vertices in the graph represent the nodes in the sensor network that are randomly distributed in a 500 m $\times$ 500 m region. If the distance between two nodes is less than or equal to the radio range (all nodes are assumed to have the same radio range), then the two corresponding vertices are connected by an edge in the graph.

The number of nodes (n) and the radio range (r) are adjustable parameters. We, then randomly assign one third of the nodes as malicious and for any edge between a malicious node and an honest node, we assign the edge to be inconsistent with a probability of 1/2. All other edges are assigned to be consistent. It is obvious that if we remove all the malicious nodes and the corresponding edges then the resulting subgraph becomes consistent. This subgraph may or may not be the optimal solution. Such a subgraph is called a *suboptimal solution* and the number of edges in a suboptimal solution is called its value.

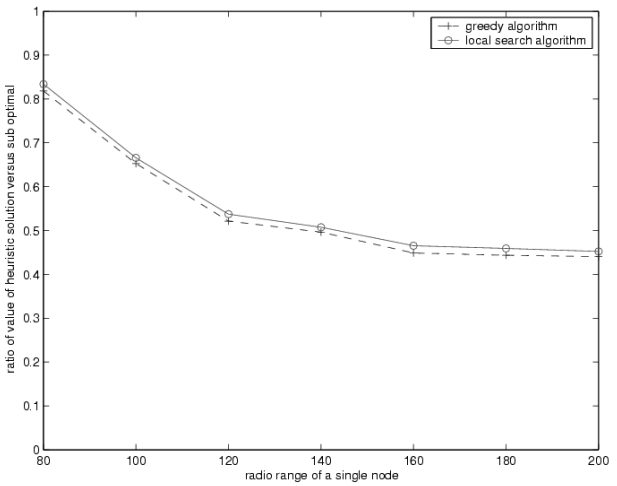
We tested the two greedy algorithms and the two local search algorithms with some fixed values for n and r. Since it is computationally infeasible to get the true optimal solution for large graphs, we measure the solution quality of the algorithms by evaluating the ratio of the value (number of edges in the solution) returned by these algorithms to the

value of the suboptimal solution. All data values are the average of 100 runs. The data diagrams in Figure 5(a)–(c) plot the data values of the algorithms with $n = 80$, $n = 100$ and $n = 120$, respectively. The radio range is along the x-axis and the ratio of the solution value to the suboptimal solution value is along the y-axis.

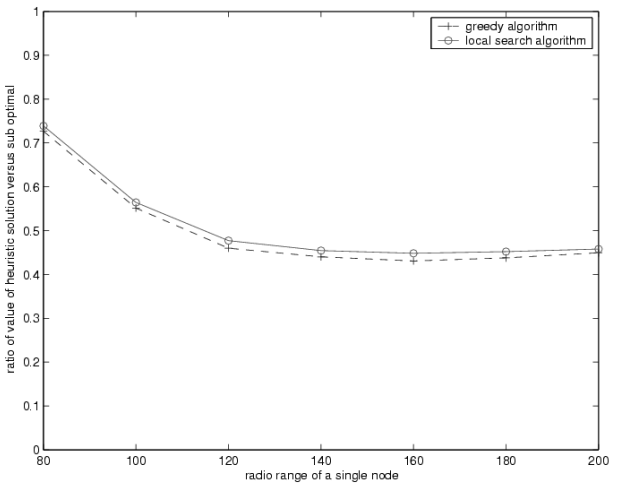
Figure 5 Plot of solution quality versus radio range for a network with (a) 80 nodes; (b) 100 nodes and (c) 120 nodes



(a)



(b)



(c)

We make the following observations:

1 None of the algorithms discussed above returns a solution that is better than the suboptimal solution.

2 The performance difference of the two greedy algorithms is negligible (for this reason we only indicate one data curve for greedy algorithms in the diagrams).

3 The performance difference of the two local search algorithms is negligible (for this reason we only indicate one data curve for local search algorithms in the diagrams).

4 The local search algorithm has some improvement over the greedy algorithm, but the improvement is not significant.

5 The performance of both the algorithms decreases as the number of nodes increases.

6 The performance of both the algorithms decreases as the radio range increases (i.e. the graph is more dense) and becomes stable after the radio range reaches a certain threshold value.

7 The solution quality does not deteriorate below 0.4 and the average solution quality is close to 0.5.

Despite the negative inapproximability result for LARGEST-CON, we can see that both the greedy and local search algorithms produce good solutions even for large graphs. Also, the solution quality is close to 0.5 even for highly sparse graphs. These results are encouraging. In our experiments, we found only a negligible difference between the two greedy algorithms and the local search algorithms. Moreover, the above algorithms and results are true only for general graphs. We have not investigated the LARGEST-CON problem in specific types of graphs like planar graphs. There is a possibility that we might be able to find a lower bound on the solution quality for such graphs.

7 Conclusion and future work

In this paper, a formal model for WSNs that takes into account the inconsistencies (inaccuracies) introduced due to malicious node behaviour or external factors is proposed. Using this model as a basis, we formulate two optimisation problems that aim to eliminate these inconsistencies in an efficient way, namely MAX-CON and LARGEST-CON. The hardness of these optimisation problems is indicative of the difficulty involved in eliminating inconsistency causing nodes in a WSN and is an important factor when considering redeployment, network termination, etc. We prove the above problems to be NP-complete and give the inapproximability result for LARGEST-CON. We also provide two approximation algorithms for LARGEST-CON based on popular heuristics like greedy choice and LSS. Finally, using experiments we show that LSS performs slightly better than greedy algorithms for randomly generated graphs and performance of both the algorithms deteriorates as the number of nodes and radio range increases.

As part of our future work in this direction, we would first like to get a lower bound on the solution

quality of LARGEST-CON. We are currently working on another solution strategy for solving LARGEST-CON. We would like to formulate, test and compare it against the existing strategies. Moreover, the combinatorial results for LARGEST-CON presented in this paper hold for graphs in general. As part of future research, we would like to investigate whether WSNs can be modelled as specific types of graph and whether or not the inapproximability result for LARGEST-CON holds for such graphs.

Acknowledgements

The authors would like to thank Dr. Hung Ngo and Mr. S. Vidyaraman for their valuable inputs and comments during the course of this work.

References

Bahl, P. and Padmanabhan, V.N. (2000) 'Radar: an in-building RF-based user location and tracking system', *IEEE INFOCOM Conference Proceedings*, *IEEE Communications Society*, pp.775–784.

Bar-Yehuda, R. and Even, S. (1981) 'A linear time approximation algorithm for the weighted vertex cover algorithm', *Journal of Algorithms*, Vol. 2, pp.198–210.

Bar-Yehuda, R. and Even, S. (1985) 'A local-ratio theorem for approximating the weighted vertex cover problem', *Analysis and Design of Algorithms for Combinatorial Problems, Annals of Discrete Mathematics*, Vol. 25, pp.27–46.

Bomze, I., Budinich, M., Pardalos, P. and Pelillo, M. (1999) 'The maximum clique problem', D-Z. Du and P.M. Pardalos, (Eds). *Handbook of Combinatorial Optimization*, Vol. 4, Boston, MA: Kluwer Academic Publishers.

Dinur, I. and Safra, S. (2005) 'On the hardness of approximating minimum vertex-cover', *Annals of Mathematics*, Vol. 162, No. 1.

Elson, J. and Estrin, D. (2001) 'Time synchronization for wireless sensor networks', *Proceedings of the 2001 International Parallel and Distributed Processing Symposium (IPDPS), Workshop on Parallel and Distributed Computing Issues in Wireless and Mobile Computing*, San Francisco, CA, pp.186–186.

Eren, T., Goldenberg, D., Whiteley, W., Yang, Y.R., Morse, A.S., Anderson, B. and Belhumeur, P. (2004) 'Rigidity, computation and randomization of network localization', *The IEEE INFOCOM 2004. Proceedings*, Hong Kong, China, IEEE Computer and Communications Society.

Fang, L., Du, W. and Ning, P. (2005) 'A beacon-less location discovery scheme for wireless sensor networks', *IEEE INFOCOM Conference Proceedings*, IEEE Communications Society.

Ganeriwal, S., Capkun, S., Han, C-C. and Srivastava, M.B. (2005) 'Secure time synchronization service for sensor networks', *WiSe '05: Proceedings of the Fourth ACM Workshop on Wireless Security*, New York, NY, USA: ACM Press, pp.97–106.

Ganeriwal, S., Kumar, R. and Srivastava, M.B. (2003) 'Timing-sync protocol for sensor networks', *SenSys '03: Proceedings of the First International Conference on Embedded Networked Sensor Systems*, New York, NY: ACM Press, pp.138–149.

Garey, M.R. and Johnson, D.S. (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman.

Goldenberg, D., Krishnamurthy, A., Maness, W., Yang, Y., Young, A., Morse, A., Savvides, A. and Anderson, B. (2005) 'Network localization in partially localizable networks', *The IEEE INFOCOM 2005. Proceedings*, Miami, FL.

Halperin, E. (2000) 'Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs', *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*.

Hastad, J. (1997) 'Some optimal inapproximability results', *Proceedings of 29th ACM Symposium on Theory of Computing*, pp.1–10.

He, T., Huang, C., Blum, B.M., Stankovic, J.A. and Abdelzaher, T. (2003) 'Range-free localization schemes for large scale sensor networks', *MobiCom '03: Proceedings of the Ninth Annual International Conference on Mobile Computing and Networking*, New York, NY: ACM Press, pp.81–95.

Hightower, J. and Borriello, G. (2001) Location systems for ubiquitous computing, *Computer*, Vol. 34, No. 8, pp.57–66.

Hochbaum, D.S. (1982) 'Approximation algorithms for the weighted set covering and node cover problems', *SIAM Journal on Computing*, Vol. 11, pp.555–556.

Homer, S. and Selman, A.L. (2001) *Computability and Complexity Theory*, chapter Nondeterminism and NP-Completeness, Springer-Verlag, pp.122–144.

Hong, X., Xu, K. and Gerla, M. (2002) 'Scalable routing protocols for mobile ad hoc networks', *IEEE Network Magazine*, Vol. 16, No. 4, pp.11–21.

Hromkovič, J. (2004) *Algorithms for Hard Problems*, Springer-Verlag.

Karakostas, G. (2005) 'A better approximation ratio for the vertex cover problem', *Proceedings of International Colloquium on Automata, Languages and Programming*.

Karp, R. (1972) *Complexity of Computer Computations*, chapter Reducibility Among Combinatorial Problems, Plenum Press, pp.85–104.

Ko, Y.B. and Vaidya, N. (1998) 'Location-aided routing(lar) in mobile ad hoc networks', *The Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking(MOBICOM)*, ACM SIGMOBILE/IEEE Communications Society, pp.66–75.

Liu, D., Ning, P. and Du, W. (2005a) 'Attack-resistant location estimation in sensor networks', *The Fourth International Symposium on Information Processing in Sensor Networks (IPSN '05)*, ACM SIGBED and IEEE Signal Processing Society, pp.99–106.

Liu, D., Ning, P. and Du, W. (2005b) 'Detecting malicious beacon nodes for secure location discovery in wireless sensor networks', *The 25th International Conference on Distributed Computing Systems (ICDCS '05)*, IEEE Computer Society, pp.609–619.

Lorincz, K., Malan, D., Fulford-Jones, T.R.F., Nawoj, A., Clavel, A., Shnayder, V., Mainland, G., Moulton, S. and Welsh, M. (2004) 'Sensor networks for emergency response: challenges and opportunities', *IEEE Pervasive Computing, Special Issue on Pervasive Computing for First Response*, Vol. 3, No. 4, pp.16–23.

Monien, B. and Speckenmeyer, E. (1985) 'Ramsey numbers and an approximation algorithm for the vertex cover problem', *Acta Information*, Vol. 22, pp.115–123.

Pires, W.R., de Paula Figueiredo, T.H., Wong, H.C. and Loureiro, A.A. (2004) 'Malicious node detection in wireless sensor networks', *Eighteenth International Parallel and Distributed Processing Symposium, 2004. Proceedings*, IEEE Computer Society, p. 24.

Priyantha, N., Chakraborty, A. and Balakrishnan, H. (2000) 'The cricket location-support system', *The Sixth Annual International Conference on Mobile Computing and Networking(MOBICOM)*, ACM SIGMOBILE, pp.32–43.

Ray, S., Ungrangsi, R., de Pellegrini, F., Trachtenberg, A. and Starobinski, D. (2003) 'Robust location detection in emergency sensor networks', *IEEE INFOCOM Conference Proceedings*, San Francisco, IEEE Communications Society, pp.1044–1053.

Robinson, C.A. (2004) 'Sensors bolster army prowess', *SIGNAL Magazine, AFCEA's International Journal*, Available at: http://www.afcea.org/signal/articles/anmviewer.asp? a =30.

Sastry, N., Shankar, U. and Wagner, D. (2003) 'Secure verification of location claims', *WiSe '03: Proceedings of the 2003 ACM Workshop on Wireless Security*, New York, NY: ACM Press, pp.1–10.

Shnayder, V., Rong Chen, B., Lorincz, K., Fulford-Jones, T.R.F. and Welsh, M. (2005) 'Sensor networks for medical care', *Technical Report*, Harvard University.

Tollefsen, W., Pepe, M., Myung, D., Gaynor, M., Welsh, M. and Moulton, S. (2004) 'iRevive, a pre-hospital mobile database for emergency medical services', *International Journal of Healthcare Technology and Management (IJHTM)*.

Yu, Y., Govindan, R. and Estrin, D. (2001) 'Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks', *Technical Report UCLA/CSD-TR-01-0023*.