A Generalization of Maximal Independent Sets

Arun Jagota * **Giri Narasimhan** † **Ľubomír Šoltés** ‡

Abstract

We generalize the concept of maximal independent set in the following way. For a nonnegative integer k we define a **k-insulated set** of a graph G as a subset S of its vertices such that each vertex in S is adjacent to at most k other vertices in S and each vertex not in S is adjacent to at least $k+1$ vertices in S. We show that it is NP-hard to approximate a maximum k-insulated set within a polynomial factor and describe a polynomial algorithm which approximates a maximum k-insulated set in an n-vertex graph to within the factor of $cnk/log^2 n$, for a constant $c > 0$. We also give an $O(kn^2)$ algorithm which finds an arbitrary k-insulated set.

1 Introduction

For a nonnegative integer k, define a **k-insulated set** of a graph $G(V, E)$ as a set $S \subseteq V$ of vertices such that each vertex in S is adjacent to at most k other vertices in S and each vertex not in S is adjacent to at least $k+1$ vertices in S. (If $S = V$ the second condition holds trivially.) The notion of a k-insulated set generalizes that of a maximal independent set, since 0-insulated sets are just the maximal independent sets. While it is obvious that each graph has a maximal independent set, even the existence of a k-insulated set for every positive k is not so obvious. It follows from Theorem 4.1 that each graph has a k-insulated set for each nonnegative k.

*Department of Computer Science, University of California, Santa Cruz, CA;

†Mathematical Sciences Department, University of Memphis, Memphis, TN 38152; Supported in part by NSF Grants INT-911-5870 and CCR-940-9752, and by a grant from Cadence Design Systems, Inc.; e-mail: `giri@msci.memphis.edu`;

‡Mathematical Sciences Department, University of Memphis, Memphis, TN 38152;

In this paper we present algorithms and complexity results on k-insulated sets. The maximum independent set (MIS) problem is a well-known NP-hard problem. It is also known that it is NP-hard to approximate the maximum independent set [FGLSS, ALMSS] to within a factor of n^ϵ. We prove an analogous result for maximum-cardinality k-insulated sets in Section 2.

While computing the largest k-insulated set is computationally intractable, finding an arbitrary k-insulated set turns out to be surprisingly non-trivial. Simple greedy algorithms exist to find a k-insulated set for $k = 0$, since this is a maximal independent set. However, these algorithms do not easily generalize to compute a k-insulated set for $k \geq 1$. It is interesting to note that, except for $k = 0$, a maximal vertex set inducing a subgraph of maximum degree at most k is not necessarily a k-insulated set. For example, by removing one endvertex from the $(k + 2)$-vertex star $K_{1,k+1}$ we obtain a maximal subgraph of maximum degree at most k, although its vertex set is not a k-insulated set.

In Section 3 we present an algorithm to compute a 1-insulated set in an n-vertex graph in $O(n^2)$ time. In section 4 we present an algorithm to compute an arbitrary k-insulated set in $O(kn^2)$ time. The energy function method used in the analysis of the time complexity of our algorithms is of independent interest. In Section 5 we describe a polynomial algorithm for computing a large k-insulated set. In Section 6 we study the performance of these algorithms on random graphs.

Finding large independent sets (or large cliques) has applications in various fields. Here are some. Object recognition in computer vision is often modeled as finding a large common induced subgraph between a pair of graphs [BY86]. The latter is usually solved by reduction to maximum clique. Maximum clique has also been used in experimental design, signal transmission [BY86], information retrieval [AM70], clustering in pattern recognition [DH73], labeled point pattern matching [Oga86], stereo vision correspondence [HS89], and PLA folding [1].

Finding k-insulated sets rather than independent sets would offer more flexibility, which in some applications might be welcome. Specifically, the quantity k could be used to trade-off the tightness of the constraint for the size of the set. (As k is increased, k-insulated sets generally become larger, and a large one becomes easier to find.) A somewhat similar trade-off, at its extremes, is widely used in graph-based clustering [DH73]. At one extreme a cluster is defined to be a clique. At the other, a connected component. While the former is more desirable than the latter, the latter is computationally more tractable.

It is interesting to note that a k-insulated set arises naturally when a discrete Hopfield neural network [Hop82] for computing maximal independent sets is modified slightly (by changing the threshold values). See [GJ93] for details about this connection. This connection facilitates a simple parallel implementation of the algorithm presented in Section 4 for finding a k-insulated set which runs in $O(kn)$ time on n processors. Via this connection this same algorithm may also be viewed as a self-stabilizing distributed algorithm for finding a k-insulated set. Self-stabilizing algorithms have in recent years attracted some interest in the distributed computing community [Sch93]. Hopfield neural networks are interpreted as self-stabilizing distributed algorithms in [Jag98].

2 Maximum k-insulated sets

Let $i_k(G)$ denote the number of vertices in a largest k-insulated set S in a graph G. Thus $i_0(G)$ denotes the independence number of G. Since the maximum degree in the subgraph induced by S is at most k, the set S can be partitioned into at most $k+1$ independent sets in G. Therefore $i_k(G) \leq (k+1)i_0(G)$. It follows from Lemma 5.1 proved later that $i_k(G)$ is at least the independence number of G for any k. Thus

$$i_0(G) \leq i_k(G) \leq (k+1)i_0(G)$$

for any graph G and any nonnegative integer k. Now we will show that these bounds cannot be improved. The left inequality is tight for the empty graph E_n on n vertices, since $i_k(E_n) = n$ for all integers $k \geq 0$ and $n \geq 1$. The right inequality becomes equality for complete graphs, since $i_k(K_n) = k+1$ if $k \leq n-1$ and $i_k(K_n) = n$ if $k \geq n$. Next we show that the inequality $i_k(G) \leq i_{k+1}(G)$ does not hold in general.

To see this, construct the complete bipartite graph $K_{2,k+1}$ with the parts $A = \{a_1, a_2\}$ and $B = \{b_1, b_2, ..., b_{k+1}\}$. Then extend this graph to a graph H_k by joining each vertex $b_i \in B$ $(i = 1, ..., k+1)$ to k new vertices $b_{i1}, b_{i2}, ..., b_{ik}$.

Lemma 2.1 *For each integer $k \geq 2$ we have $i_{k+1}(H_k) < i_k(H_k)$.*

Proof : Let n be the number of vertices in the graph H_k. First note that by removing the vertices a_1 and a_2 from the vertex set $V(H_k)$ we obtain a k-insulated set. Thus $i_k(H_k) \geq n - 2$. Now we will show that $V(H_k) - B$ is the only $(k+1)$-insulated set in the graph H_k. Since the degrees of the vertices a_1, a_2, $b_{11}, b_{12}, ..., b_{1k}, b_{21}, b_{22}, ..., b_{(k+1)k}$ are at most k, these vertices belong to any $(k+1)$-insulated set S in H_k. Since each vertex in B is adjacent to $k+2$ of these vertices, none of the vertices in B is in S. Thus $S = V(H_k) - B$. Hence we have $i_{k+1}(H_k) = n - |B| = n - (k+1) < n - 2 \leq i_k(H_k)$.

■

It would be of interest to characterize graphs G for which the sequence $i_0(G), i_1(G), \ldots,$ is monotone.

Now we focus on complexity issues. The complexity of the 0-insulated set problem is well-understood [ALMSS, Has96] (also see [Hal93]). We start with a result due to Arora, Lund, Motwani, Sudan, and Szegedy [ALMSS].

Theorem 2.2 [ALMSS] *There is an $\epsilon > 0$ such that, given a graph G with n vertices, it is NP-hard to compute an independent set in G of size at least $i_0(G)/n^\epsilon$.*

This theorem easily extends to the family of problems, indexed by k, of approximating $i_k(G)$ to within a factor of $n^{1-\epsilon}$ in a given graph G. This follows from the fact that $i_k(G) \leq (k+1)i_0(G)$ in a graph G. The extended result also follows from [LY, LuY]. Here we give another proof based on a reduction from the maximum independent set problem. This reduction has some additional value (see Corollary 2.4).

Theorem 2.3 *There is an $\epsilon > 0$ such that for each $k \geq 0$, given a graph G with n vertices, it is NP-hard to compute a k-insulated set in G of size at least $i_k(G)/n^\epsilon$.*

Proof : Given a graph G, we construct a new graph G' such that G contains an independent set of size at least c if and only if the graph G' contains a *k-insulated set* of size at least $(k+1)c$.

Construct the graph $G'(V', E')$ as follows. Assume that $V = \{v_1, v_2, \ldots, v_n\}$. Set $V' = \{v_{ij},\ 0 \leq i \leq k,\ 1 \leq j \leq n\}$, and $E' = \{(v_{ij}, v_{rs})|\ j = s \text{ or } (v_j, v_s) \in E\}$. Thus there are $k+1$ copies of V in G', and there are edges between two vertices in G', if and only if their images in G are adjacent or the same. Now we claim that G has an independent set of size at least c if and only if G' has a k-insulated set of size at least $(k+1)c$. The first part of the claim is straightforward, since an independent set along with its k other copies forms a k-insulated set in G'. The converse is also true because if G' has a k-insulated set of size at least $(k+1)c$, then this k-insulated set can be partitioned into $k+1$ independent sets (since the maximum degree in the subgraph induced by k-insulated set is at most k), at least one of which must be of size at least c. In particular, $i_k(G') = (k+1)i_0(G)$. Hence the result. ■

Corollary 2.4 *It is NP-hard to compute the value $i_{\sqrt{|V(G)|}}(G)$ in a given graph G.*

Proof : From the given graph G on n vertices construct the graph H on n^2 vertices via the reduction in the proof of Theorem 2.3, with the setting $k = n$. Compute the value $i_{\sqrt{|V(H)|}}(H) = i_n(H)$, and from it compute $i_0(G) := i_n(H)/(n+1)$. ■

In Section 5 we describe a polynomial algorithm which, given nonnegative integer k and a graph G, computes a k-insulated set in G of size at least $ci_k(G)\log^2 n/(kn)$, for some constant $c > 0$.

3 Computing Arbitrary 1-insulated sets

Below we present algorithm $\mathcal{A}$ (see Figure 1) for constructing a 1-insulated set in a given graph. We say that an independent set S is $(1,2)$–*extendible* if for some $x \in S$ and $u, v \notin S$, $S \cup \{u, v\} \backslash \{x\}$ is an independent set. In other words, S is $(1,2)$–extendible if a larger independent set can be obtained from S by deleting one vertex and adding two. Algorithm $\mathcal{A}$ consists of two phases. In phase I, the algorithm computes an independent set S_1 that is maximal and not $(1,2)$–extendible. In phase II, the algorithm computes a maximal independent set among all vertices that are adjacent to exactly one vertex in S_1. Figure 3 illustrates the main aspects of the algorithm.

Theorem 3.1 shows the correctness and running time of algorithm $\mathcal{A}$.

Algorithm $\mathcal{A}$

```
{ Given graph G(V,E), computes a 1-insulated set }
S_1 ← ∅
{ Phase I of algorithm }
repeat
        if (∃v ∉ S_1 such that S_1 ∪ {v} is independent) then
                S_1 ← S_1 ∪ {v}
        else if (S_1 is not (1,2)-extendible) then
                Find x ∈ S_1 and u,v ∉ S_1 that will extend it
                S_1 ← S_1\{x} ∪ {u,v}
        else go to Phase II of algorithm
forever
{ Phase II of algorithm }
Let L_1 be the set of vertices not in S_1 that are adjacent to exactly one vertex in S_1.
Find a maximal independent set in L_1. Call it S_2.
S ← S_1 ∪ S_2
Output S
end Algorithm A.
```

Figure 1: Algorithm to compute 1-insulated set

Theorem 3.1 *Algorithm $\mathcal{A}$ computes a $1-$insulated set in $O(n^2)$ time, given a graph G with n vertices.*

Proof : Algorithm $\mathcal{A}$ ensures that S_1 and S_2 are independent sets. Each vertex in S_2 is also in L_1 and hence is adjacent to at most one vertex from S_1. Each vertex in S_1 is adjacent to at most one vertex from S_2, since otherwise S_1 would be $(1,2)$–extendible.

Any vertex not in $S_1 \cup L_1$ is adjacent to at least 2 vertices in S_1. Any vertex in $L_1 \backslash S_2$ must be adjacent to exactly 1 vertex in S_1 and at least 1 vertex in S_2 (since S_2 is a maximal independent set in L_1). Hence $S_1 \cup S_2$ is a $1-$*insulated set*, proving the correctness of algorithm $\mathcal{A}$.

To bound the time complexity, we note first that the repeat loop is executed at most n times since the number of vertices in S_1 increases by 1 in every iteration. Furthermore, the algorithm will go through at most n iterations to construct S_2. Each of the iterations for S_1 can be implemented in $O(n)$ time as described below. For each vertex not in S_1, maintain the number of vertices of S_1 that it is adjacent to. Also maintain separate lists of vertices that are not in S_1 and that are adjacent to no vertices in S_1 (call this list L_0) and that are adjacent to exactly 1 vertex in S_1 (call this list L_1). For each vertex in S_1, keep a list of number of adjacencies and the adjacent vertices from L_1. Maintain a separate list of vertices from S_1 that are adjacent to more than one vertex in L_1 (call this list L_2). Phase I terminates when both lists L_0 and L_2 are empty. Clearly, these lists and the degree information can be updated after each iteration of the algorithm in $O(n)$ time, thus proving the required time complexity.

■

The algorithm described above is a generalization of the simple greedy algorithm for computing maximal independent sets. However, this algorithm does not generalize easily to an algorithm to compute k$-$*insulated set* when $k > 1$. In the next section we present one that finds a k-insulated set in any graph for arbitrary $k \geq 0$.

4 Computing Arbitrary k-insulated sets

In this section we present a local algorithm that takes a graph G and nonnegative integer k as input, and finds a k-insulated set in G. While the algorithm is simple, it is not even obvious that it terminates. We use the energy function method to prove its termination, and to bound its running time.

We need some further notation. Let $G[V']$ denote the subgraph of a graph $G(V, E)$ induced by a set of vertices $V' \subseteq V$. For any vertex $v \in V$, let $d_{V'}(v)$ be the number of vertices of V' that are adjacent to v. For any vertex $v \in V$, any set $S \subseteq V$, and any integer $k \geq 0$, let

$$\Delta_S^k(v) \equiv \begin{cases} d_S(v) - k & \text{if } v \in S \\ k + 1 - d_S(v) & \text{if } v \in V \setminus S \end{cases} \tag{1}$$

Note that, if $\Delta_S^k(v) > 0$ for some vertex v, then S is not a k-insulated set.

This algorithm starts from any given set $S_0 \subseteq V$ of vertices as its initial state, and transforms S_0 sequentially. When it terminates, the set S forms a k-insulated set.

This algorithm was introduced in [GJ93] in the context of neural nets, and is also implicit

Algorithm $\mathcal{B}(k, S_0)$
{ Given graph $G(V, E)$, finds a k-insulated set starting from set S_0}
`if` $(k \geq |V|)$ `then` output V and `stop`
$S \leftarrow S_0$
`while` $(\exists v \in V : \Delta_S^k(v) > 0)$ `do`
`if` $(v \in V \backslash S)$ `then`
$S \leftarrow S \cup \{v\}$
`else`
$S \leftarrow S \backslash \{v\}$
`endwhile`
Output S
`end` **Algorithm** $\mathcal{B}$.

Figure 2: Algorithm to compute k-insulated set

in [GLL88]. Its termination was noted in [GJ93] using the energy function approach of Hopfield [Hop82]. Here we obtain a tight running time bound on the operation of this algorithm, using the same energy function approach, but with a more refined analysis.

The energy function approach has been used in the past in amortized analysis of algorithms – mainly on-line algorithms (for many examples, see [CLR, Tar]), and in the analysis of neural networks (for example, see [Hop82]).

For $k = 0$, algorithm $\mathcal{B}(0, \emptyset)$ operates in a simple manner—start from the empty set and sequentially extend it to a maximal independent set—which terminates in $O(n)$ loop-iterations. For $k > 0$, the operation of $\mathcal{B}(k, \emptyset)$ is not so simple. The following example illustrates that vertices may not only be added but also deleted from S. Let G be a graph with vertex set $V = \{v_1, v_2, v_3, v_4\}$ and edge set $E = \{(v_1, v_3), (v_1, v_4)\}$. A run of algorithm $\mathcal{B}(1, \emptyset)$ can evolve according to the following sequence of states S:

$$\emptyset \rightarrow \{v_1\} \rightarrow \{v_1, v_2\} \rightarrow \{v_1, v_2, v_3\} \rightarrow \{v_1, v_2, v_3, v_4\} \rightarrow \{v_2, v_3, v_4\}$$

The next result shows that not only does the algorithm terminate, but does so reasonably efficiently.

Theorem 4.1 *For a graph $G = (V, E)$ with n vertices and m edges, the Algorithm $\mathcal{B}(k, \emptyset)$ terminates in at most $(2k+1)i_k(G)$ iterations of the while loop and the Algorithm $\mathcal{B}(k, V)$ terminates in at most $2m$ iterations of the while loop.*

Proof : The proof uses the energy function method of Hopfield [Hop82], specialized to this problem. For any vertex-set $S \subseteq V$, define the energy function:

$$E(S) = \epsilon(G[S]) - (k + 1/2)|S|$$

where $\epsilon(G[S])$ is the number of edges in the subgraph induced by S.

During the course of the proof, it will be shown that the function E is bounded from below, and its value decreases in every iteration by at least $1/2$. This will have proven termination. The rest will be refinements to yield the desired time bounds.

First we bound the number of iterations for the algorithm which starts from the empty set. Note that $E(\emptyset) = 0$. Further, since $\epsilon(G[S]) \geq 0$, the minimum value of $E(S)$ for a k-insulated set is at least $-(k+1/2)i_k(G)$. Consider an iteration of the while loop. Let v be such that $\Delta_S^k(v) > 0$.

Either

$$\begin{array}{llcl}
v \in V \backslash S & \Rightarrow \; d_S(v) \leq k & & \\
& \Rightarrow \; E(S \cup \{v\}) & \leq & E(S) + k - (k + 1/2) \\
& & = & E(S) - 1/2
\end{array}$$

or

$$\begin{array}{llcl}
v \in S & \Rightarrow \; d_S(v) > k & & \\
& \Rightarrow \; E(S \backslash \{v\}) & \leq & E(S) - (k+1) + (k + 1/2) \\
& & = & E(S) - 1/2
\end{array}$$

Thus in every iteration the energy decreases at least by $1/2$. Since the initial energy is zero and the energy of the terminal set is at least $-(k + 1/2)i_k(G)$, the while loop is executed at most $(2k+1)i_k(G)$ times.

Next we bound the number of iterations in the algorithm starting with the entire vertex set

V. Since the initial energy $E(V) = m - (k + 1/2)n$ and the terminal energy is again at least $-(k + 1/2)n$, the while loop is executed at most $2m$ times. ■

We now make several observations that are related to Theorem 4.1. The graph in Figure 4 shows that the first bound of Theorem 4.1 is tight for $k(k+1) \leq n$. Let k be odd. It can be shown that the following sequence of state changes in algorithm $\mathcal{B}$ is possible: starting from the empty set, first all vertices in B are added, then all vertices in $A_1, A_2, \ldots, A_k$ are added, then all vertices in $A_1, A_2, \ldots, A_{k-1}$ are deleted, then all vertices in $A_1, A_2, \ldots, A_{k-2}$ are added, then all vertices in $A_1, A_2, \ldots, A_{k-3}$ are deleted, and so on until the state is $A_1 \cup A_3 \cup \ldots A_{k-2} \cup A_k \cup B$. Next, all vertices in B are deleted. (Note that this is not yet a k-insulated set.)

Each iteration of the while loop can be made to run in $O(d)$ time, where d is the maximum degree in the graph G, as follows. With each vertex v store its (i) membership status in S and (ii) the value of $\Delta_S^k(v)$. Maintain $\{v : \Delta_S^k(v) > 0\}$ in a doubly-linked list L. Store the graph as an adjacency list. Each vertex v has a record, one of whose fields points to v's adjacency list. A second field in this record points to v's node in L if $\Delta_S^k(v) > 0$ and is null otherwise. (Thus an insertion or deletion on L takes $O(1)$ time.) Now the condition of the while loop can also be checked in $O(1)$ time. When adding or deleting a vertex v to S update (i) its membership status in S, (ii) the $\Delta_S^k(u)$ values of all vertices u that v is adjacent to, and (iii) the list L (insert or delete u into or from L if needed). Updates (i)–(iii) may be carried out in $O(d)$ time.

This yields a time complexity of $O(ki_k(G)d)$ for $\mathcal{B}(k, \emptyset)$ and $O(nd^2)$ for $\mathcal{B}(k, V)$. Note that the former bound is slightly better, since for $k \geq d$ the only k-insulated set is V.

The second part of Theorem 4.1 yields a better bound for sparse graphs. For example, a planar graph has at most $3n - 6$ edges, and thus the time-complexity of $\mathcal{B}(k, V)$ improves on these graphs to $O(nd)$.

The problem of finding a k-insulated set in a $(k+1)$-regular graph can be solved in $O(kn)$ time. An algorithm solving this problem is based on the following observation.

Theorem 4.2 *Let $G = (V, E)$ be a d-regular graph with $d \geq 1$ and let p and q be two nonnegative integers such that $p + q = d - 1$. Then a set S of vertices in G is a p-insulated set if and only if* *the set $V-S$ is q-insulated set.*

Proof : By symmetry it is enough to prove that if S is p-insulated than $V-S$ is q-insulated. Since S is p-insulated, every vertex not in S has at least $p+1$ neighbors in S, therefore it has at most $d-(p+1)=q$ neighbors in $V-S$. Every vertex in S has at most p neighbors in S, thus it has at least $d-p=q+1$ neighbors in $V-S$. Therefore $V-S$ is a q-insulated set. ■

In particular, a set S of vertices in a d-regular graph is $(d-1)$-insulated if and only if $V-S$ is a maximal independent set. Since a greedy algorithm finds a maximal independent set in a graph with m edges in $O(m+n)$ time, a k-insulated set in a $(k+1)$-regular graph can be found in $O(m)=O(kn)$ time.

5 A Polynomial Algorithm for Large k-insulated sets.

A simple example shows that algorithms $\mathcal{A}$ and $\mathcal{B}$ may retrieve constant-size 1-insulated sets in a graph that also contains ones of size $\Theta(n)$. Consider a graph with two identical components: the complete bipartite graphs $K_{p,2}$ and $K_{p,2}$. Let u,v,w,x denote the vertices of degree n and X the remaining ones. Then both $\{u,v,w,x\}$ and X are 1-insulated sets, and the algorithms can retrieve the former.

The next two results are more positive.

Lemma 5.1 *Let I be an independent set in a graph G and let S be the k-insulated set that algorithm $\mathcal{B}(k,I)$ finds. Then $|S|\geq|I|$.*

Proof : Let e be the number of edges in the subgraph induced by S. Now

$$E(S)=e-(k+1/2)|S|\leq E(I)=-(k+1/2)\ |I|$$

since the energy of S must be at most that of I. Since $e\geq 0$, we have $|S|\geq|I|$. ■

Let algorithm $\mathcal{C}$ be the algorithm for k-insulated set which first produces an independent set I using the Boppana-Halldórsson algorithm [BH90] and then extends I to a k-insulated set using algorithm $\mathcal{B}(k, I)$. We now show that algorithm $\mathcal{C}$ performs as well as the Boppana-Halldórsson algorithm.

Proposition 5.2 *Given an n-vertex graph G, the algorithm $\mathcal{C}$ finds a k-insulated set with at least $ci_k(G)\log^2 n/(nk)$ vertices, for some positive constant c.*

Proof : The Boppana-Halldórsson algorithm finds an independent set I with at least $(c\log^2 n)i_0(G)/n$ vertices [BH90]. The k-insulated set S which the algorithm $\mathcal{C}$ finds has at least $|I|$ vertices by Lemma 5.1. Using the inequality $i_k(G) \leq (k+1)i_0(G)$ we obtain

$$|S| \geq |I| \geq (c\log^2 n)i_0(G)/n \geq (c\log^2 n)i_k(G)/(n(k+1))$$

Hence the result. ■

6 k-Insulated Sets in Random Graphs

In this section, we discuss the performance of our algorithms on random graphs. Define a random graph G_n as a graph on n vertices with edges introduced independently in the $\binom{n}{2}$ edge-slots with probability 1/2. We say that *almost every* graph (or a typical graph) has a property P when the probability that a random graph G_n has property P tends to 1 as $n \to \infty$ [Bol85, p. ix]. Despite the fact that the MIS problem is NP-hard to approximate within some polynomial factor [ALMSS], it has long been known that any algorithm guaranteed to find a maximal independent set in a graph (such as the simple greedy one) will find one within a factor of two of the largest one, in almost every graph (see [BE76, Kar76] or [Pal85, Matula's theorem, p. 76]).

Theorem 6.1 *Given $\epsilon > 0$ and sufficiently large n, almost every graph G_n contains a maximal independent set of cardinality r, where $(1+\epsilon)\log_2 n < r < (2-\epsilon)\log_2 n$, but G_n does not contain a maximal independent set of cardinality less than $(1-\epsilon)\log_2 n$ or greater than $(2+\epsilon)\log_2 n$.*

One-half of this result easily extends to k-insulated set sets, for fixed k.

Proposition 6.2 *Given $\epsilon > 0$, fixed integer $k \geq 0$, and sufficiently large n, almost every graph G_n is such that it does not contain a k-insulated set of cardinality greater than $(2+\epsilon)\log_2 n$.*

Proof : The proof is a trivial extension of the simple probabilistic method proof of the case when $k = 0$ (see [BE76]). Define indicator random variables X_l^i to equal 1 when the i_{th} l-element subset of V is a k-insulated set and 0 otherwise. Define $X_l = \sum_i X_l^i$. Then

$$P(X_l \geq 1) \leq E(X_l) \leq \binom{kl/2}{2}(1/2)^{\binom{l}{2}}$$

Now setting $l = (2+\epsilon)log_2 n$ yields

$$\lim_{n\to\infty} P(X_l \geq 1) \leq lim_{n\to\infty}\binom{kl/2}{2}(1/2)^{\binom{l}{2}} = 0$$

■

We now have the following result.

Theorem 6.3 *Algorithm $\mathcal{A}$ finds a 1-insulated set within a factor of two of the largest 1-insulated set, in almost every graph.*

Proof : Recall that each 1-insulated set can be partitioned into two independent sets. Then by Proposition 6.2, in almost every graph, the size of the largest 1-insulated set is no more than $2\log_2 n$. Since the algorithm $\mathcal{A}$ finds an independent set S_1 that is maximal, by Theorem 6.1, in almost every graph, the size of S_1 is no less than $\log_2 n$. Note that the actual $(1,2)$–maximal independent set S_1 found by algorithm $\mathcal{A}$ can only be larger than the above bound. Algorithm $\mathcal{A}$ also finds another maximal independent set called S_2 to extend S_1 into a 1-insulated set. Hence the result. ■

Next we show that a similar result can be obtained for a slightly modified version of algorithm $\mathcal{B}(k,\emptyset)$ called $\mathcal{B}'(k,\emptyset)$ in which (i) adding a vertex has priority over deleting a vertex and (ii) the vertex to be added is picked in greedy fashion (i.e., the one that maximizes $\Delta_S^k(v)$ amongst those that can be added).

Proposition 6.4 *For fixed k, algorithm $\mathcal{B}'(k, \emptyset)$ outputs a k-insulated set which is within a factor of 2 of the largest k-insulated set, in almost every graph.*

Proof : It is easily checked that algorithm $\mathcal{B}'(k, \emptyset)$ finds a maximal independent set I in the first $|I|$-many iterations. By Theorem 6.1, $|I| \geq \log_2 n$ in almost every graph. By Proposition 6.2the size of the maximum k-insulated set is no more than $2 \log_2 n$ in almost every graph. By Lemma 5.1, the final k-insulated set S that the algorithm retrieves satisfies $|S| \geq |I|$. This concludes the proof.

∎

7 Conclusion

In this paper, we proposed an interesting generalization of maximal independent sets. We studied complexity of some problems associated with it, and designed and analyzed algorithms for them. Our analyses included those of worst-case running time, and worst-case and average-case approximation ratios.

The energy function approach seems to be a powerful tool to design and analyze algorithms. It would be interesting to explore other applications of this approach.

We pose the open problem of extending algorithm $\mathcal{A}$ to compute a k-insulated set for $k > 1$.

Acknowledgments

We thank an anonymous referee for helpful suggestions, especially on the complexity of finding k-insulated sets and on improvements to our average-case approximation results.

References

[ALMSS] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *The Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science*, pages 14–23, 1992.

[AM70] G. Augustson and J. Minker. An analysis of some graph-theoretical clustering techniques. *Journal of the Association of Computing Machinery*, 17(4):571–588, October 1970.

[BY86] E. Balas and C.S. Yu. Finding a maximum clique in an arbitrary graph. *SIAM Journal on Computing*, 15(4), November 1986.

[BE76] B. Bollobás and P. Erdös. Cliques in random graphs. *Proc. Camb. Phil. Soc.*, 80:419–427, 1976.

[Bol85] B. Bollobás. *Random Graphs.* Academic Press, New York, 1985.

[BH90] R. Boppana and M. Halldórsson. Approximating maximum independent sets by excluding subgraphs. In *Scandinavian Workshop on Algorithm Theory (SWAT)*, pages 11–25. Springer Verlag, 1990.

[CLR] T.H. Cormen, C.E. Leiserson, and R.L. Rivest; *Introduction to Algorithms*, McGraw Hill, 1990.

[DH73] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* Wiley, New York, 1973. Chapter 6: Unsupervised Learning and Clustering (Section 6.11).

[FGLSS] U. Feige, S. Goldwasser, L. Lovász, S. Safra, M. Szegedy; Interactive Proofs and the Hardness of Approximating Cliques, In *J. of the ACM*, 43(2):268-292, 1996.

[GLL88] G. H. Godbeer, J. Lipscomb and M. Luby, On the Computational Complexity of Finding Stable State Vectors in Connectionist Models (Hopfield Nets), Technical Report, Department of Computer Science, University of Toronto, vol. 208/88, 1988.

[GJ93] T. Grossman and A. Jagota. On the equivalence of two Hopfield-type networks. In *IEEE International Conference on Neural Networks*, pages 1063–1068. IEEE, 1993.

[Hal93] M. H. Halldorsson. Approximating the minimum maximal independence number. *Information Processing Letters*, **46**(4):169–172, 1993.

[Has96] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. In *The Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 627–636, 1996.

[Hop82] J.J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences, USA*, 79, 1982.

[HS89] R. Horaud and Skordas T. Stereo correspondence through feature grouping and maximal cliques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11):1168–1180, 1989.

[Jag98] A. Jagota. Hopfield neural networks and self-stabilization. *Chicago Journal of Theoretical Computer Science*, 1999.

[Kar76] R.M. Karp. The probabilistic analysis of some combinatorial search algorithms. In J.F. Traub, editor, *Algorithms and Complexity: New Directions and Recent Results*, pages 1–19. Academic Press, New York, 1976.

[1] J.E. Lecky, O.J. Murphy, and R.G. Absher. Graph theoretic algorithms for the PLA Folding problem. *IEEE Transactions on Computer Aided Design*, 8(9):1014–1021, 1989.

[LuY] C. Lund and M. Yannakakis. The approximation of maximum subgraph problems, In *Proceedings of 20th International Colloquium on Automata, Languages, and Programming*, Lecture Notes in Comput. Sci. 700, Springer-Verlag, 40-51.

[LY] M. Lewis and M. Yannakakis; The Node-Deletion Problem for Hereditary Properties is NP-Complete, *J. of Computer Systems and Sciences*, **20**, (1980).

[Oga86] H. Ogawa. Labeled point pattern matching by delaunay triangulation and maximal cliques. *Pattern Recognition*, 19(1):35–40, 1986.

[Pal85] E.M. Palmer. *Graphical evolution*. Wiley, New York, 1985.

[Sch93] M. Schneider. Self-stabilization. *ACM Computing Surveys*, 25(1):45–67, 1993.

[Tar] R.E. Tarjan; Amortized Computational Complexity, *SIAM J. on Alg. and Disc. Methods*, **6**(2):306-318, 1985.

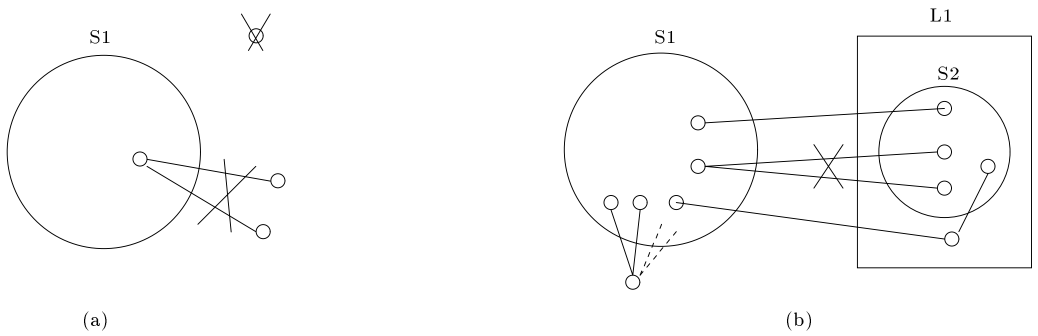

Figure 3: (a) illustrates set S_1 after phase I. (b) illustrates sets L_1 and S_2 in phase II.

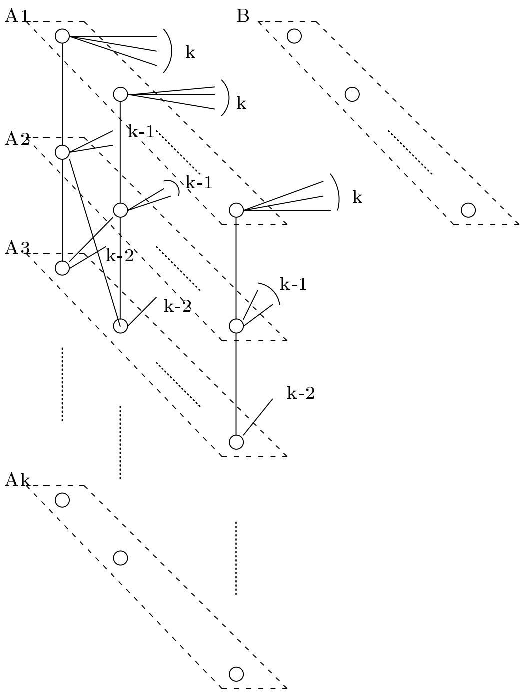

Figure 4: The graph is partitioned into $k+1$ independent sets $A_1, A_2, \ldots, A_k$, and B of equal cardinality $\geq k$. Each vertex in set A_i, $i = 1, \ldots, k$, is adjacent to $k-i+1$ vertices in set B. These vertices are chosen so that the degree in set A_i of every vertex of set B is also $k-i+1$. Each vertex in set A_i, $i = 1, \ldots, k-1$, is adjacent to i vertices in set A_{i+1}. These vertices are chosen so that the degree in set A_i of every vertex of set A_{i+1} is also i.