Greedy Approximations of Independent Sets in Low Degree Graphs

Magnús M. Halldórsson[*] Kiyohito Yoshihara[†]

Abstract

We investigate the power of a family of greedy algorithms for the independent set problem in cubic graphs and graphs of maximum degree three. These algorithm iteratively select vertices of minimum degree, but differ in the secondary rule for choosing among many candidates. We study three such algorithms, and prove tight performance ratios, with the best one being $9/7 \approx 1.28$. All of these algorithms are practical and run in linear time, in contrast with the algorithm with the best performance ratio known of 1.2.

We also show certain inherent limitations in the power of this family of algorithm: any algorithm that greedily selects vertices of minimum has a performance ratio at least 1.25 on degree-three graphs, even if given an oracle to choose among candidate vertices of minimum degree.

1 Introduction

An *independent set* of a graph G is a subset of vertices in which no two are adjacent. The Max Independent Set problems is that of finding an independent set of maximum cardinality. It is one of the core $\mathcal{NP}$-hard problems [4], and thus, polynomial time exact algorithms are unlikely to exist. It is therefore interesting to explore algorithms that produce solutions that are not always optimal but are close to optimal. The quality of an approximation algorithm is generally measured by the *performance ratio*, or the maximum ratio of the size of an optimal solution (the size of the maximum independent set) to the size of the solution found by the algorithm.

The independent set problem is known to be hard to approximate on general graphs. Arora *et al.* [1] showed that it is $\mathcal{NP}$-hard to obtain a performance ratio of less than n^δ for some $\delta > 0$, where n is the number of vertices. On special classes of graphs, however, the problem does admit constant factor approximations.

One important such class is that of bounded-degree graphs. After a decade of non-activity following a paper of Hochbaum [9], there has been a flurry of results on the approximation of independent sets in bounded-degree graphs [3, 6, 7, 2]. The currently best ratios known are $(\Delta + 3)/5$ for maximum degree $\Delta \leq 613$ [3, 2], $\Delta/6 + O(1)$ for intermediate values of Δ, and $O(\Delta/\log\log\Delta)$ [7, 8] for large values of Δ..

In this paper we focus on a central case of bounded-degree graphs, namely when the maximum degree is at most three. Since the independent set problem is polynomial solvable when maximum degree is two, this problem can be thought of as the initial frontier of $\mathcal{NP}$-hardness of the problem. Also, many of the results for higher degrees use reductions to lower degree cases, in

[*]Contact author. Science Institute, University of Iceland, IS-107 Reykjavik, Iceland. Research partly performed at Japan Advanced Institute of Science and Technology – Hokuriku, IBM Tokyo Research Lab, and Max Planck Institut fuer Informatik.

[†]Department of Computer Science, Tokyo Institute of Technology

which the degree-three plays the role of the basis case [3, 7, 5, 2], and improvements for that case translate to improvements for all odd degrees.

We additionally consider cubic graphs, i.e. 3-regular graphs, where all vertices are of degree three. The problem remains $\mathcal{NP}$-hard and MAX SNP-hard (hard to approximate within some fixed constant greater than one) even under these strong restrictions.

Let us review the known results about approximating independent sets in degree-three graphs. Hochbaum [9] presented an algorithm with a 1.5 ratio, that runs in time proportional to bipartite matching or $O(n^{1.5})$. Berman and Fürer [3] gave a powerful local search approach that attains a performance ratio of 1.25. This has recently been brought down to 1.2 by Berman and Fujito [2] using additional tricks. The disadvantage of this approach is a phenomenally high time complexity: the analysis of [3, 2] yields a bound of at least $n^{2^{100}}$, while even with a tighter analysis [7] the complexity appears to be no less than n^{50}. In response to this, Halldórsson and Radhakrishnan [7] gave a scaled-down version of the local search approach of [3] which runs in linear time with a performance ratio of 1.4. Generalizations [5] lead to a $1.33 + \epsilon$ ratio in time $O(\exp(1/\epsilon)n)$.

The algorithm paradigm that we consider in this paper is that of greedy algorithms: select a vertex of minimum degree, add the vertex to the solution, remove the vertex and its neighbors, and repeat until the graph is empty. This approach is non-deterministic in the choice of a particular vertex of minimum degree. The basic algorithm, selecting an arbitrary minimum degree vertex, was analyzed in detail by Halldórsson and Radhakrishnan [6]. In particular, the performance ratio on degree-three graphs was shown to be 5/3.

We consider here greedy algorithms with more goal-directed selection rules. The algorithms always choose minimum-degree vertices, but with different rules to decide among candidate vertices. Typically, the algorithms attempt to eliminate more than the minimum number of edges in each reduction, or prefer reductions that compare well with the optimal solution. In summary, our results are as follows:

1. The basic greedy algorithm attains a performance ratio of 3/2 on cubic graphs. Further restricting the input to graphs of high odd girth yields no further improvements.

2. A modified greedy algorithm is presented that attains a ratio of 3/2 on (general) degree-three graphs. The ratio improves to a ratio that approaches 4/3 on cubic graphs with high odd girth.

3. A second modified greedy algorithm is presented that attains a ratio of $9/7 \approx 1.28$ on degree-three graphs.

4. Any greedy algorithm is shown to have a performance ratio at least 1.25 on degree-three graphs. Thus, the whole family has limitations which are nearly matched by our second algorithm.

The paper is organized into sections following the above list.

2 Preliminaries

2.1 Notation

We use standard symbols and notations. The input graph $G = (V, E)$ is assumed to be of maximum degree three, with further restrictions explicitly stated when in place. Let n denote the number of vertices, m the number of edges, α the independence number (i.e. size of the optimal independent set). For a vertex v, $N(v)$ denotes the neighborhood of v, or the set of adjacent vertices.

For an algorithm A for MAX INDEPENDENT SET, the size of the solution produced on G is denoted by $A(G)$, and the *approximation ratio* $\rho_A(G)$ is defined as $\rho_A(G) \stackrel{\text{def}}{=} \alpha(G)/A(G)$. The *performance ratio* of A is defined as the maximum approximation ratio over all input graphs, or $\rho_A \stackrel{\text{def}}{=} \max_G \rho_A(G)$. We are primarily interested in the limit of this value as n goes to infinity.

We let I denote a fixed but arbitrary maximum independent set in G. Let Out denote the number of edges with both endpoints in $V - I$.

2.2 Upper bounding the optimal solution

In order to analyze the relative value of a heuristic solution compared with an optimal solution, it is essential to have at ones disposal a good upper bound of the optimal solution. The number Out of edges outside some maximum cardinality solution I plays a crucial role.

Lemma 2.1 *For a degree-three graph G,*

$$\alpha(G) \le n - m/3 - Out/3. \tag{1}$$

Proof. Each edge has either one endpoint in I or both endpoints in $V - I$. The total number of endpoints in $V - I$ is at most $3(n - |I|)$. Thus,

$$m \le 3(n - |I|) - Out,$$

which, when rearranged, yields the claim. ∎

In a cubic graph, $m = 3n/2$, and the inequality becomes $\alpha \le n/2 - Out/3$.

3 Greedy Algorithm on Cubic Graphs

We first consider the well-known Greedy algorithm, which we label here as Greedy. The algorithm proceeds along a sequence of iterations or *reductions*, each of which consists of the following two steps: some vertex of minimum degree is added to the solution, and the vertex, its neighbors, and all incident edges are removed from the graph. The algorithm terminates when the all vertices have been deleted from the graph. Since neighbors of a selected vertex are immediately deleted, the solution consisting of the selected vertices form a proper independent set. By maintaining track of the degrees of the vertices, the algorithm can be implemented in $O(n + m)$ time.

Greedy was analyzed for bounded-degree graphs in [6], where its approximation performance on degree-three graphs was shown to be $5/3 \approx 1.66$. A better ratio is possible in the case of cubic graphs.

Theorem 3.1 *The performance ratio of* Greedy *on cubic graphs is* $3/2$.

We first argue the upper bound. Assume without loss of generality that the graph is connected. Observe that Greedy picks a vertex of maximum degree at most once, since no proper induced subgraph can be regular. Thus, at most three vertices are deleted in all but the first step. That is,

$$Min_0 \;\ge\; (n-1)/3.$$

Consider the last reduction made. The deleted vertices must form a clique on 1 to 4 vertices. If 1 or 2, then Greedy $\geq n/3$; if 3 or 4, then $\alpha \leq n/2 - 1/3$ by Lemma 2.1. In either case, the performance ratio is at most 3/2.

Remark. It can be argued that Greedy finds an optimal solution in regular bipartite graphs. By applying Lemma 2.1, that implies a performance ratio of at most $(n/2 - 1/3)/((n-1)/3) = 3/2 - 3/(2n-2)$.

We now construct a hard graph for Greedy that shows that the above ratio is tight. The graph is constructed from three units: *front unit*, *back unit*, and multiple copies of *repetition units*. The repetition units are in the form of a 12-cycle with three cords, and are connected in a chain with three edges between adjacent copies. The chain is flanked on the ends by the front and rear units, both in the form of a complete bipartite graph $K_{2,3}$, with the three vertices in one partition connected to the ends of the repetition unit chain. The graph, G_0, is best described by picture, in Figure 1.

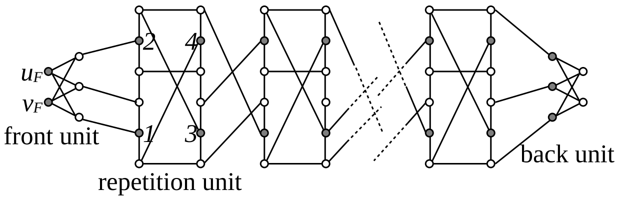

Figure 1: A hard graph for Greedy.

We indicate the worst-case behavior of Greedy by presenting a particular sequence of symmetry-breaking choices. The algorithm starts by choosing v_F and u_F of the front unit. On the first repetition unit it chooses vertices 1 through 4 in that order. This leaves an identical graph less a single repetition unit. Hence, the algorithm picks the four shaded vertices of each repetition unit, ending with three vertices from the rear unit.

If ℓ is the number of repetition units, the algorithm finds $4\ell + 5$ vertices while the optimal solution contains $6\ell + 4$. The total number of vertices is $12\ell + 10$. Hence, the approximation ratio of Greedy on G_0 is:

$$\rho_0(G_0) = \frac{6\ell + 4}{4\ell + 5} = \frac{3}{2} - \frac{21}{2(n+5)}.$$

We conclude that the performance ratio of Greedy on cubic graphs asymptotically equals 3/2. Observe that the same holds even if the graphs are required to be triangle-free.

4 A Modified Greedy Algorithm

The worst case behavior of Greedy, as seen when applied to the hard graphs G_0 in the previous section, suggests a direction for modifying the strategy of the algorithm. A situation where Greedy appears to be weak is when there are many vertices of minimum degree. In this section, we propose a modified version of Greedy, named MoreEdges, which considers the degrees of vertices adjacent to a vertex as a criteria for selecting the vertices. The criteria is:

> When minimum degree is two, select – whenever possible – a vertex with a neighbor of degree three.

If no such vertex exists and the minimum degree is two, then we can show that the graph is composed of several disjoint cycles. On that remaining portion, MoreEdges obtains an optimal solution.

We find that this modified algorithm yields an improvement over the 5/3 ratio of Greedy on degree-three graphs.

Theorem 4.1 *The performance ratio of* MoreEdges *on degree-three graphs is* 3/2.

Upper bound

The operation of the algorithm can be broken up into *reductions*, each of which consists of the addition of a single vertex to the current solution and the deletion of this and neighboring vertices along with the incident edges. If the recursive description of the algorithm is made iterative, a reduction corresponds to a single iteration. An (i, j)*-reduction* refers to one where $i - 1$ vertices and j edges are deleted.

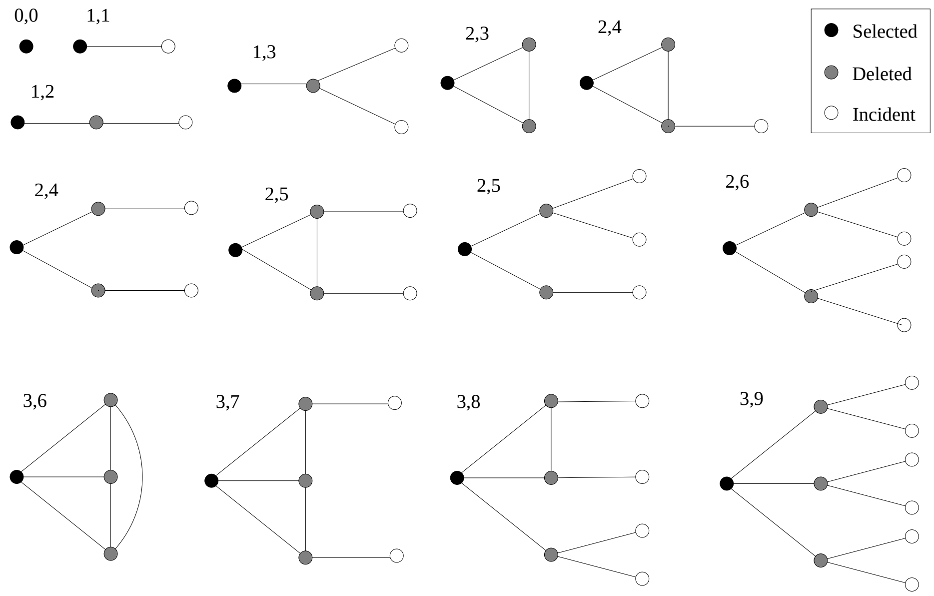

Figure 2: The forms of the various reductions.

The form of the possible reductions are given in Figure 2. The selected vertices are in black, their neighbors (which are also deleted) are in grey, and other incident vertices are in white. $(2, 4)$ and $(2, 5)$ reductions appear in two different guises.

We consider the following measures of each reduction r. Here, we fix some maximum cardinality independent set I for comparison.

$n(r)$	Number of vertices deleted
$e(r)$	Number of edges deleted
$\alpha(r)$	Number of the deleted vertices that belong to I
$Out(r)$	Number of deleted edges with both endpoints in $V - I$.

Our primary *cost* measure of each reduction r is given by:

$$f(r) \;=\; 3n(r) - e(r) - Out(r) + \alpha(r).$$

Table 1 gives conservative bounds for these measures on each type of reduction. The values for $Out(r)$, $e(r)$ are lower bounds, while $\alpha(r)$ and $f(r)$ are upper bounds.

r	$n(r)$	$e(r)$	$Out(r)$	$\alpha(r)$	$f(r)$
$(0,0)$	1	0	0	1	4
$(1,x)$	2	1	0	1	6
$(2,3)$	3	3	1	1	6
$(2,4)$	3	4	0	1†	6
$(2,5+)$	3	5	1	2	6
$(3,6)$	4	6	3	1	4
$(3,7)$	4	7	1	2	6
$(3,8)$	4	8	1	2	5
$(3,9)$	4	9	0	3	6

Table 1: Bounds on measures of the reductions performed by MoreEdges

The values in Table 1 are easily verified from Figure 2, with the exception of the α value of $(2,4)$. When a $(2,4)$-reduction occurs, the graph necessarily consists of disjoint cycles. The algorithm will add the same number of vertices to the solution from a given cycle, no matter which vertex it starts with. Thus, it causes no harm if we assume that it chooses a vertex for which at most one neighbor belongs to a given maximum independent set.

The claim of the upper bound now follows easily from the f-values of Table 1, and Lemma 2.1:

$$6t \;=\; \sum_r 6 \;\geq\; \sum_r f_r \;=\; 3n - e - Out + \alpha \;\geq\; 4\alpha.$$

Lower bound

We construct a hard graph for MoreEdges as in Figure 3. It is a chain of simple units with six vertices each. Each unit forms a six-cycle with one cord between the third and the fifth vertex. The last vertex in each unit is also adjacent to the first vertex in the subsequent unit. Formally, we construct a family of graphs G_q, with vertices vertices $v_{i,1},\ldots,v_{i,6}$ and edges $(v_{i,j},v_{i,j+1}),(v_{i,3},v_{i,5}),(v_{i',6},v_{i'+1,1})$, where $i=1,\ldots q$, $j=1,\ldots 6$ and $i'=1,\ldots q-1$.

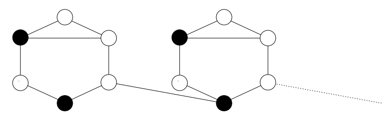

Figure 3: Initial portion of a hard graph for MoreEdges.

MoreEdges may be assumed to select the first and the third vertex of each unit, while the optimal solution will contain the second, fourth and sixth. Only on the last unit will MoreEdges also find three vertices. Hence, the performance ratio of MoreEdges is no better than $3q/(2q+1) = 3/2 - \theta(1/n)$.

Further results

We have further analyzed the algorithm for classes of cubic graphs. In particular, the algorithm attains performance ratios of $17/12 \approx 1.42$ on cubic graphs, $29/21 \approx 1.38$ on cubic triangle-free graphs, and in general $4/3 + 1/(9k+3)$ on cubic graphs of odd girth $2k+1$. Those values are tight as there are graphs where these ratios occur. We omit the descriptions for reasons of space.

5 A Second Improved Algorithm

We consider in this section a still stronger member of the greedy paradigm. In particular, the algorithm performs the following two types of transformations whenever possible.

Branchy reduction When two vertices v and u of degree two are adjacent, some optimal solution will contain exactly one of these vertices. We can transform the graph into a graph G' that contains all vertices but v and u and has the other neighbors of v and u adjacent. To ensure that multi-edges do not appear, we insist that no third vertex be adjacent to both v and u. The solution of the heuristic will contain the heuristic solution on G' along with one of v and u. This is a case of a *delayed-commitment* reduction, whose effect is optimal.

Simplicial reduction A *simplicial* vertex is one whose neighborhood forms a clique. An optimal solution can contain at most one vertex from this open neighborhood, hence selecting a simplicial vertex is always optimal. In particular, when minimum degree is two, we select one whose two neighbors are adjacent, whenever possible.

Simplicial reductions appear as $(1, 1-3)$, $(2, 3-5)$ and $(3, 6)$-reductions. Branchy reductions appear as $(1, 2)$-reductions. These tricks have earlier been used in [2].

```
Simplicial(G)
  repeat
    perform reductions in the following order of preference:
      1. branchy, simplicial, (2,6)
      2. (3,8)
      3. (3,9)
  until done
end
```

By keeping track of the shape of the neighborhood of each vertex in the current graph, the algorithm can be implemented in linear time.

This algorithm attains a ratio of $9/7 \approx 1.28$ on degree-three graphs. In comparison, the previous best ratio claimed for an algorithm with low-polynomial time complexity [6, 7, 5] was $4/3 + 1/\epsilon$ [5]. Further, by using this algorithm as the subroutine for degree three graphs in the schema of [7] (originating in [3]), we obtain similar improvements for the independent set problem in other classes of bounded-degree graphs.

We devote the rest of this section for proving our main result.

Theorem 5.1 *The performance ratio of* Simplicial *on graphs of maximum degree three is exactly* $9/7$.

We first give a simple construction for the lower bound in Figure 4. First build a unit with seven vertices, $v_1, \ldots v_7$ forming a cycle with chords between second and fourth, third and sixth, and fifth and seventh. All vertices are of degree three, except v_1. The hard graph is obtained by adding one vertex u connected to the v_1's of three units.

For the initial choice, the algorithm will prefer a $(3, 8)$-reduction over a $(3, 9)$-reduction, and may choose any vertex but u and v_1 vertices. One tie-breaking choice is to select v_2, followed by v_7, u, and two vertices from each of the two remaining units, for a total of seven. On the other hand, the optimal solution consists of the first, third and fifth vertex of each unit, for a total of nine.

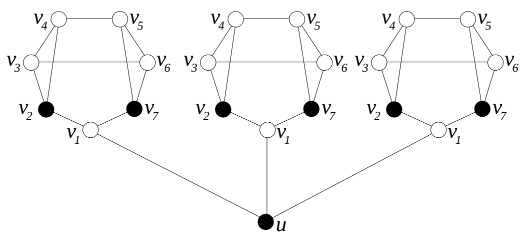

Figure 4: A hard graph for Simplicial.

5.1 Upper bound

Using the same measures of the reductions as for MoreEdges, our *cost measure* for this algorithm is given by:

$$g(r) \;=\; 6n(r) - 2e(r) - 2Out(r) + \alpha(r).$$

Unfortunately, this measure is too large on $(1,1)$ and $(2,3)$ reductions. We alleviate this problem by considering short sequences of reductions, or *idioms* as we call them, and showing that the cost measure on these combinations behave as desired.

Let us view the execution of the algorithm as a string of reductions. We argue that that this string can be lexically partitioned into strings from a restricted class.

Claim 5.2 *The following is an alphabet for the reduction sequence of the algorithm:*

$\{(0,0),\ (1,2),\ (1,3),\ (2,4),\ (2,5),\ (2,6),\ (3,8),\ (3,9),$
$[\{(2,6),(3,8),(3,9)\}\{(1,2),(2,4)\}^*(1,1)],\ [\{(1,3),(2,5),(2,6),(3,9)\}\{(1,2),(2,4)\}^*(2,3)],$
$[(3,8)\{(1,2),(2,4)\}^*(2,3),\{(2,5),(2,6)\}].$

The following observations have bearing on Claim 5.2.

1. A $(3,7)$-reduction is impossible, since some two of the vertices in the reduction would give rise to a $(3,8)$-reduction.
2. The possibility of the idiom $[(3,9),(2,3),(2,3)]$ is eliminated by the third case of the algorithm.
3. The only reductions that can precede a $(2,3)$ reduction are: $(1,3)$, $(2,5)$, $(2,6)$, or $(3,9)$. This ignores $(1,2)$ and $(2,4)$-reductions which may be interspersed in various ways.
4. Following the sequence $[(3,8),(2,3)]$, the remaining graph will be cubic except for a single vertex. Hence only $(2,5)$ or $(2,6)$ reductions can immediately follow. Further, $[(3,8),(2,3),(2,5),(2,3)]$ is not possible.

We generalize the measures of reductions to measures of idioms, by taking the sum of the measure of each reduction in the idiom. Further, for an idiom π, let $t(\pi)$ denote the number of reductions within π.

Table 2 gives lists lower bounds for $Out(\pi)$ and upper bounds for $\alpha(\pi)$ and $g(\pi)$ for the fundamental idioms π in the alphabet. For the idioms that include $(2,3)$, we have omitted the interspersed $(1,2)$ and $(2,4)$ reductions, and counted them as individual idioms. The idioms involving $(1,1)$ have also been compacted.

The following values are different from Table 1 or are different from the sums of the values of the individual reductions in the idiom.

π	$n(\pi)$	$e(\pi)$	$Out(\pi)$	$\alpha(\pi)$	$g(\pi)$	$g(\pi)/t(\pi)$
$(0,0)$	1	0	0	1	7	7
$(1,1)$	2	1	1	1	9	9
$(1,2)$	2	2	0	1	9	9
$(1,3)$	2	3	0	1	7	7
$(2,4)$	3	4	1	1	9	9
$(2,5)$	3	5	1	1	7	7
$(2,6)$	3	6	0	2	8	8
$(3,8)$	4	8	1	2	8	8
$(3,9)$	4	9	0	3	9	9
$\{(2,6),(2,3)\}$	6	9	2	2	16	8
$\{(3,9),(2,3)\}$	7	12	2	3	17	8.5
$\{(1,3),(2,3)\}$	5	6	1	2	18	9
$\{(2,5),(2,3)\}$	6	8	2	2	18	9
$\{(3,8),(2,3),(2,5)\}$	10	16	3	4	26	8.66
$\{(3,8),(2,3),(2,6)\}$	10	17	2	5	27	9

Table 2: Bounds on measures of the reductions performed by Simplicial

1. α and Out for $(2,4)$ and $\alpha(2,5)$: Notice that only the latter form of these reductions in Figure 2 can now appear due to the preference to the delayed-commitment reduction.

2. Out for $(1,1)$: Whichever idiom a $(1,1)$-reduction appears in, an additional edge must be outside of I.

3. The Out values of $(2,5)(2,3)$ and $(3,9)(2,3)$ and α values of $(2,6)(2,3)$ and $(3,9)(2,3)$: The reasons are clear when we look at the subgraphs induced by these pairs of reductions. $(1,2)$ and $(2,4)$-reductions yield optimal results on the subgraph induced by the deleted vertices, thus, having interspersed within an idiom does not affect any argument about the independence number or outside edges in the subgraph induced by the idiom.

The theorem now follows from Lemma 2.1 along with the fact that $g(\pi)/t(\pi)$ in Table 2 is always at most 9.

$$9t \;=\; 9\sum_{\pi} t(\pi) \;\geq\; \sum_{\pi} g(\pi) \;\geq\; 6n - 2e - 2Out + \alpha \;\geq\; 7\alpha.$$

6 Algorithm with Advice, Ultimate

In the previous sections, we considered the performance ratios of three greedy algorithms: Greedy, MoreEdges and Simplicial. These two algorithms have a common basic strategy of removing a vertex with the minimum degree in a current graph at each stage. We can easily see by their hard graphs that the weaknesses of the greedy algorithms appear when they have several ways of choosing a minimum-degree vertex. If we could give these algorithms some advice such that they could proceed optimally whenever they face a branch road, how much would the algorithms improve? Or would an algorithm that was given perfect advice necessarily find an optimal solution?

In this section, we study the power of algorithms that are given the additional benefit of an oracle for selecting among alternatives. The only requirement is that the algorithm must choose one of the minimum degree vertices at any step. We refer to this ultimate algorithm as Ultimate, indicating that the algorithm has infinite "visibility" (or arbitrary distance from the given node) for choosing among minimum degree vertices.

We shall show that even by employing Ultimate, there remains graphs for which Ultimate cannot find an optimal solution. In fact, it cannot guarantee a much better performance ratio than the algorithm of the previous section. This reveals a limitation on the power of the family of greedy algorithms.

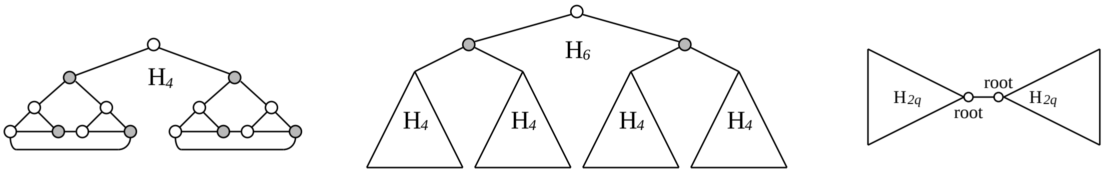

Figure 5: Construction of H_4, H_6, and G_∞.

We first form a subgraph H_4 as on the left of Figure 5. The size of the maximum independent set of H_4 is six, whereas Ultimate finds only five vertices. One optimal solution consists of the shaded vertices in Figure 5. We construct a pseudo binary tree H_6 with four H_4's as leaves. It is illustrated in the center of Figure 5. We form a pseudo binary tree H_{2q} with $2q (q \geq 2)$ levels by repeating the same operation $q-2$ times.

The size of the solution found by the algorithm is:

$$HEU(H_{2q}) = 1 + 4 \cdot HEU(H_{2(q-1)}) = 1 + 4 + 4^2 + \cdots + 4^{q-2} + 4^{q-1}5 = 4^{q-1}16/3 - 1/3,$$

while the size of the optimal solution is:

$$OPT(H_{2q}) = 2 + 4 \cdot OPT(H_{2(q-1)}) = 2(1 + 4 + 4^2 + \cdots + 4^{q-2}) + 4^{q-1}6 = 4^{q-1}20/3 - 2/3.$$

As q grows, the ratio of HEU to OPT approaches $5/4$.

We can also obtain a hardness result for regular (i.e. cubic) graphs. Join two H_{2q}s as on the right of Figure 5 in order to make the entire graph cubic, and call the resulting graph G_∞.

An optimal solution of G_∞ contains all vertices on the even levels of each H_{2q}. Suppose that Ultimate picks any vertex in a H_{2q} at the first step. It is easy to verify that Min_∞ then proceeds optimally on that half of the graph. That leaves the other H_{2q} left, for which the algorithm will, by induction, be non-optimal.

Thus,

$$\rho_\infty(G_\infty) = \frac{2 \cdot OPT(H_{2q})}{OPT(H_{2q}) + HEU(H_{2q})} = \frac{20+20}{20+16} = \frac{10}{9} = 1.\overline{1}.$$

We can obtain similar hardness results for triangle-free graphs (or more generally graphs of high odd girth), by replacing the triangles at the bottom of H_5 by a five-cycle (or an appropriately large odd cycle) and connecting the pairs together as needed.

Theorem 6.1 *Any greedy algorithm that selects vertices of minimum degree must have performance ratios at least:* 1.25, *for degree-three;* $1.1\overline{1}$, *for cubic graphs; and* $16/15 \approx 1.06$, *for triangle-free cubic graphs.*

Acknowledgments

We are much indebted to Professor Osamu Watanabe and Professor Jaikumar Radhakrishnan for informative comments and discussions.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. *FOCS 1992*.

[2] P. Berman and T. Fujito. On the approximation properties of independent set problem in degree 3 graphs. *WADS 1995*.

[3] P. Berman and M. Fürer. Approximating maximum independent set in bounded degree graphs. *SODA 1994*.

[4] M. R. Garey and D. S. Johnson. *Computers and Intractibility: A Guide to the Theory of NP-completeness*. Freeman, 1979.

[5] M. M. Halldórsson. Approximating discrete collections via local improvements. *SODA 1995*.

[6] M. M. Halldórsson and J. Radhakrishnan. Greed is good: Approximating independent sets in sparse and bounded-degree graphs. *STOC 1994*. To appear in *Algorithmica*.

[7] M. M. Halldórsson and J. Radhakrishnan. Improved approximations of independent sets in bounded-degree graphs. *SWAT 1994*.

[8] M. M. Halldórsson and J. Radhakrishnan. Improved approximations of independent sets in bounded-degree via subgraph removal. *Nordic J. Computing*, 1(4):475–492, 1994.

[9] D. S. Hochbaum. Efficient bounds for the stable set, vertex cover, and set packing problems. *Disc. Applied Math.*, 6:243–254, 1983.