On provably best construction heuristics for hard combinatorial optimization problems*

Sera Kahruman-Anderoglu Austin Buchanan
Sergiy Butenko Oleg A. Prokopyev

March 2, 2016

Abstract

In this paper, a heuristic is said to be *provably best* if, assuming $\mathcal{P} \neq \mathcal{NP}$, no other heuristic always finds a better solution (when one exists). This extends the usual notion of "best possible" approximation algorithms to include a larger class of heuristics. We illustrate the idea on several problems that are somewhat stylized versions of real-life network optimization problems, including the maximum clique, maximum k-club, minimum (connected) dominating set, and minimum vertex coloring problems. The corresponding provably best construction heuristics resemble those commonly used within popular metaheuristics. Along the way, we show that it is hard to recognize whether the clique number and the k-club number of a graph are equal, yet a polynomial-time computable function is "sandwiched" between them. This is similar to the celebrated Lovász function wherein an efficiently computable function lies between two graph invariants that are $\mathcal{NP}$-hard to compute.

1 Introduction

Researchers and practitioners often rely upon metaheuristics for $\mathcal{NP}$-hard combinatorial optimization problems [18]. Their usefulness is twofold. They find good feasible solutions that can be implemented in practice. Second, even if they provide no guarantee about solution quality, they can still be

*Published as S. Kahruman-Anderoglu, A. Buchanan, S. Butenko, O.A. Prokopyev. On provably best construction heuristics for hard combinatorial optimization problems. *Networks*, to appear. http://dx.doi.org/10.1002/net.21620

used to prune the search tree of a branch-and-bound algorithm. Metaheuristics rely on construction heuristics to find initial solutions [17] that are hoped to provide a good starting point for further search [2]. It is important that these construction heuristics run very quickly and, as such, one often uses greedy choices which are also easy to understand and implement [14]. On the other hand, there is a fair amount of skepticism towards such approaches due to a lack of supporting theoretical foundations. Indeed, in most cases there is no provable approximation guarantee for the performance of a given construction heuristic, meaning that the initial solution may be arbitrarily far away from optimal. Then a reasonable question to ask is, *what is the reason a particular construction heuristic is chosen? Is it really the "best" choice?*

For problems that admit a constant-factor approximation, questions of this type have been answered as follows [27, 51]. Assume that we know that a problem is hard to approximate to within a factor of $c - \epsilon$ for any $\epsilon > 0$, where c is a given constant. Then any polynomial-time algorithm that gives a c-approximate solution can be claimed to be "best possible." An example is the k-center problem when the edge weights satisfy the triangle inequality, which, assuming $\mathcal{P} \neq \mathcal{NP}$, has no $(2 - \epsilon)$-approximate algorithm; however, there is a simple 2-approximate algorithm [28]. For problems that are not believed to admit constant-factor approximation algorithms, claiming that an approximation algorithm is best possible may be trickier. For example, assuming $\mathcal{P} \neq \mathcal{NP}$, the maximum clique problem cannot be approximated to within a factor of $n^{1-\epsilon}$ for any constant $\epsilon > 0$ [25, 52]. However, selecting a single vertex gives an n-approximation. Should this be considered best possible?

Another approach for evaluating the quality of a heuristic is through *domination analysis* as introduced by [19] and later surveyed by [22]. In domination analysis, a solution's quality is noted with respect to that of other feasible solutions. For example, there exist n-city instances of the asymmetric traveling salesman problem (ATSP) for which a nearest neighbor heuristic produces the worst of the $(n-1)!$ possible tours. In contrast, a particular construction heuristic *always* builds a tour not worse than $(n-2)!$ of the tours [23]. In this case, its *domination number* $(n-2)!$ is close to the number of feasible solutions. Thus, this ATSP construction heuristic may be deemed to be of good quality with respect to domination analysis, despite the fact that the ATSP is inapproximable within any polynomially-computable function [47]. The nearest neighbor heuristic for ATSP would be deemed poor.

In this paper, we propose an alternative approach to justify the use of certain heuristics. It extends the common definition of the "best possible"

approximation algorithm to the problems for which a constant-ratio approximation algorithm is unlikely to be found. Informally, a heuristic is "provably best" if it is difficult to beat. For brevity, when we refer to a heuristic, we will implicitly mean polynomial-time heuristic.

Definition 1. *A heuristic is said to be* provably best *for an optimization problem if, assuming $\mathcal{P} \neq \mathcal{NP}$, there is no polynomial-time algorithm that always finds a better solution (when one exists).*

A few points should be made. As a consequence of Definition 1, only exact algorithms can be provably best for problems that admit polynomial-time algorithms. Second, as we will see, to prove that a heuristic is provably best, it suffices to show that it is $\mathcal{NP}$-hard (with respect to Cook reductions[1]) to recognize whether there is a gap between the optimal objective function value and the value of the solution output by the heuristic. Finally, provably best heuristics are not unique, just as "best possible" approximation algorithms are not unique. For example, one can always perform local search after the initial solution has been found. The analysis just implies that, in some cases, this will be futile.

We focus on some $\mathcal{NP}$-hard graph problems, and show that some simple, greedy heuristics are provably best. Specifically:

- choosing a vertex of maximum degree and its neighbors is provably best for the maximum k-club problem for any fixed $k \geq 2$;
- a greedy best-in heuristic is provably best for the maximum clique problem (and, consequently, for maximum independent set);
- a coloring algorithm that finds a Brooks coloring (i.e., with $\Delta(G)$ colors) is provably best for the vertex coloring problem;
- a greedy worst-out heuristic is provably best for the minimum vertex cover problem; and
- a greedy worst-out heuristic is provably best for the minimum (connected) dominating set problem.

[1]Recall that showing $\mathcal{NP}$-hardness of a problem using a Cook reduction (i.e., polynomial-time Turing reduction) involves demonstrating that, if the problem were solvable in polynomial time, then $\mathcal{P} = \mathcal{NP}$. In many cases, a Karp reduction (i.e., polynomial-time many-one reduction) is preferred, since it is a more restrictive type of reduction that allows for a finer analysis. However, for our purposes, Cook reductions will be enough, as we are only attempting to discern polynomial-time solvability. For more information about Cook reductions, Karp reductions, and other types of reductions, consult [34].

We note that the basic ideas of these construction heuristics have already been employed for many of these problems. Indeed, the first has been used as a basis for maximum k-club heuristics [4, 48]. A greedy best-in construction heuristic (usually with some randomness incorporated) is a staple for maximum clique heuristics [15, 29]. Finally, a worst-out heuristic for minimum connected dominating set has been proposed by [9].

The problems considered in this paper are somewhat stylized versions of real-life network optimization problems (cf. [10, 44, 45] and references therein). The k-club definition originates with [42] and has been used to model cohesive subgroups in the analysis of social and biological networks. For more on k-clubs and related clique relaxation models consult [44, 48]. Cliques were introduced by [39] in the context of social network analysis and represent the 'perfect' cluster. For more information about cliques and the maximum clique problem see [3]. Vertex coloring has applications in frequency assignment in wireless networks, register allocation in compiler optimization, and scheduling [41]. (Connected) dominating sets have applications in the design and analysis of communication networks [13, 26].

In Section 2.1, we show that, for any fixed $k \geq 2$ it is hard to recognize whether the clique number and the k-club number of a graph are equal, yet a linear-time computable function is "sandwiched" between them. This is analogous to the Lovász function (cf. [33, 38]), which is polynomial-time computable, but lies between two difficult-to-calculate graph invariants – the clique partitioning number $\bar{\chi}(G)$ and the independence number $\alpha(G)$ of graph G. Moreover, it is $\mathcal{NP}$-hard to recognize whether $\bar{\chi}(G) = \alpha(G)$ [8].

In Section 3, we consider a heuristic for the maximum subgraph satisfying property Π problem, where Π is any graph property exhibiting the Lewis-Yannakakis conditions [35]. These conditions state that property Π should be hereditary on induced subgraphs, and Π should be satisfied by infinitely many graphs and unsatisfied for infinitely many graphs. Examples of such properties include: planar, bipartite, acyclic, degree-constrained, and complete. The remarkable theorem of [35] implies that this maximization problem is $\mathcal{NP}$-hard for any such property. The heuristic that we provide finds a maximal such subgraph in polynomial time under the assumption that checking whether a graph satisfies Π can be determined in polynomial time. Whether this heuristic is provably best is left as an open question.

1.1 Notation and terminology

We consider a simple, undirected graph $G = (V, E)$ that has vertex set V and edge set E. We usually let $n = |V|$. The induced subgraph of $S \subseteq V$

is denoted $G[S] := (S, E_S)$, where $E_S = \{\{u, v\} \mid u, v \in S, \{u, v\} \in E\}$. The open neighborhood of a vertex $v \in V$ is denoted $N(v) := \{u \in V \mid \{u, v\} \in E\}$ and the degree of v is given by $|N(v)|$. The closed neighborhood of $v \in V$ is denoted $N[v] := N(v) \cup \{v\}$. The largest degree of a vertex in G is denoted $\Delta(G)$. Let $\mathrm{diam}(G)$ denote the diameter of graph G, that is $\mathrm{diam}(G) := \max\{\mathrm{dist}(i, j) \mid i, j \in V\}$, where $\mathrm{dist}(i, j)$ is the length of a shortest path between vertices i and j in G (measured by the number of edges). A subset of vertices $C \subseteq V$ is called a k-club if $\mathrm{diam}(G[C]) \leq k$. A clique is a subset of pairwise-adjacent vertices, i.e., a 1-club. A subset $D \subseteq V$ of vertices is called a dominating set if every vertex from $V \setminus D$ has a neighbor in D. A dominating set that induces a connected subgraph is called a connected dominating set.

2 Some Provably Best Heuristics

We consider the following $\mathcal{NP}$-hard combinatorial optimization problems defined on a graph $G = (V, E)$. Each problem is well studied in the literature.

- **maximum k-club,** $k \geq 2$. Find a largest subset $S \subseteq V$ of vertices such that $\mathrm{diam}(G[S]) \leq k$.
- **maximum clique**. Find a largest subset $S \subseteq V$ of vertices such that $\mathrm{diam}(G[S]) = 1$.
- **minimum vertex coloring**. Color the vertices of G using a minimum number of colors such that adjacent vertices receive different colors.
- **minimum vertex cover**. Find a smallest subset $S \subseteq V$ of vertices such that each edge in E has an incident vertex in S.
- **minimum dominating set**. Find a smallest subset $S \subseteq V$ of vertices such that every vertex in $V \setminus S$ has a neighboring vertex in S.
- **minimum connected dominating set**. Find a smallest dominating set that induces a connected subgraph.

Before moving on to the heuristics and $\mathcal{NP}$-hardness reductions, we will need the following lemma which will be used implicitly throughout the paper. Essentially, it says that a search problem is at least as hard as the corresponding decision problem.

Lemma 1. *Assume $\mathcal{P} \neq \mathcal{NP}$. A heuristic is provably best if no polynomial-time algorithm determines whether its output is optimal.*

Proof. By contradiction. Suppose there is no polynomial-time algorithm to test whether the output of heuristic $\mathcal{A}$ is optimal, but that $\mathcal{A}$ is not provably best. Since $\mathcal{A}$ is not provably best, there is another heuristic $\mathcal{A}'$ that always finds a better solution than $\mathcal{A}$ (when one exists). Thus, $\mathcal{A}'$ can be used to determine whether $\mathcal{A}$ outputs an optimal solution, a contradiction. □

2.1 The maximum k-club problem

For a simple undirected graph and a given positive integer k, a k-club is a subset of vertices that induces a subgraph of diameter at most k, and the k-club number $\bar{\omega}_k(G)$ is the cardinality of a largest k-club in G. The maximum k-club problem, which is to find a largest k-club in a graph, is $\mathcal{NP}$-hard for any fixed k [5], even if restricted to graphs of diameter $k+1$ [1]. Even worse, the problem of determining whether a k-club is maximal (by inclusion) is co$\mathcal{NP}$-complete [40] for any fixed $k \geq 2$; in contrast, when $k = 1$ (i.e., clique), verifying maximality is trivial.

Denote by $\bar{\omega}_k(G)$ the k-club number of G, which is the size of a largest k-club in G. Clearly, for $j < k$ we have

$$\bar{\omega}_j(G) \leq \bar{\omega}_k(G).$$

Note that a 1-club is equivalent to a clique, so $\bar{\omega}_1(G) \equiv \omega(G)$, where $\omega(G)$ is the clique number of G. For $k \geq 2$, we have two obvious inequalities:

$$\omega(G) \leq \Delta(G) + 1 \leq \bar{\omega}_k(G), \tag{1}$$

in which, similarly to the famous Sandwich Theorem [33], a polynomially-computable value is sandwiched between two values that are $\mathcal{NP}$-hard to compute. In an interesting related paper, [8] have shown that it is $\mathcal{NP}$-hard to check whether $\bar{\chi}(G) = \alpha(G)$, where $\bar{\chi}(G)$ denotes the cardinality of a minimum clique partitioning in G and $\alpha(G)$ is the independence number of G. This is in contrast to the fact that checking whether a graph G is perfect (i.e., for every subset S of vertices we have $\bar{\chi}(G[S]) = \alpha(G[S])$) is polynomial-time solvable [12]. The Busygin-Pasechnik result implies that any polynomially-computable parameter that lies between $\alpha(G)$ and $\bar{\chi}(G)$ will provide a provably best upper bound on the independence number in the sense that no other polynomially-computable bound can be provably better for all graphs where this bound can be improved. In particular, the Lovász

theta is one such polynomially-computable bound [21]. Analogous to the Busygin-Pasechnik result, we prove that recognizing whether $\bar{\omega}_j(G) = \bar{\omega}_k(G)$ cannot be done in polynomial time assuming $\mathcal{P} \neq \mathcal{NP}$.

Proposition 1. *Let j and k be fixed positive integers with $j < k$. Assuming $\mathcal{P} \neq \mathcal{NP}$, there is no polynomial-time algorithm to determine whether $\bar{\omega}_j(G) = \bar{\omega}_k(G)$ for a graph G.*

Proof. Let $\mathcal{A}$ be an algorithm that, given a graph G, determines whether $\bar{\omega}_j(G) = \bar{\omega}_k(G)$. We provide a Cook reduction that shows that if $\mathcal{A}$ runs in polynomial time then $\bar{\omega}_k(G)$ can be calculated in polynomial time. Recall that for any fixed positive integer k it is $\mathcal{NP}$-hard to calculate $\bar{\omega}_k(G)$ [5]. Algorithm $\mathcal{A}$ is used as a subroutine in the last three if-statements of Algorithm 1.

In the following algorithm, B_i^k denotes the (k, i)-*broom graph*, which consists of a path of k vertices and i more vertices connected to the same endpoint of this path. The (k, i)-broom is illustrated in Figure 1. Recognize that for $k \geq 2$, we have $\bar{\omega}_k(B_i^k) = i + k$ and $\bar{\omega}_j(B_i^k) < i + k$ for any $1 \leq j < k$.

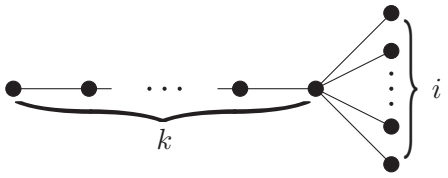

Figure 1: The (k, i)-broom graph.

If every connected component has fewer than k vertices then $\bar{\omega}_k(G) < k$, and we can use a naive algorithm to calculate $\bar{\omega}_k(G)$ in polynomial time since k is a fixed constant. Otherwise, we can assume that $\bar{\omega}_k(G) \geq k$, which will be important for the next step of the algorithm.

If the next if-statement is satisfied, then we have established that $\bar{\omega}_j(G) = \bar{\omega}_k(G)$ and the subsequent operations are performed to obtain the k-club number. In this case, we take the disjoint union of G with a steadily larger broom graph until the broom graph provides a k-club that is larger than the j-club number. Then the conclusion is that the previous broom was an optimal k-club for G', thus showing that $\bar{\omega}_k(G) = i + k - 1$.

Otherwise, we know that $\bar{\omega}_j(G) < \bar{\omega}_k(G)$. We obtain the k-club number by taking the disjoint union of G with a steadily larger complete graph until $\omega(G') = \bar{\omega}_j(G') = \bar{\omega}_k(G') = \bar{\omega}_k(G)$. □

Data: An undirected graph $G = (V, E)$ and fixed positive integers $j < k$.
Result: The k-club number $\bar{\omega}_k(G)$ of G.
if *every component of G has fewer than k vertices* **then**
  calculate $\bar{\omega}_k(G)$ naively (check all subsets of $< k$ vertices);
  return $\bar{\omega}_k(G)$
end
if $\bar{\omega}_j(G) = \bar{\omega}_k(G)$ **then**
  for $i = 1, \ldots, n$ **do**
    $G' \leftarrow G \cup B_i^k$; `// disjoint union of` G `with` (k, i)`-broom graph`
    if $\bar{\omega}_j(G') \neq \bar{\omega}_k(G')$ **then**
      return $i + k - 1$
    end
  end
else
  for $i = 1, \ldots, n$ **do**
    $G' \leftarrow G \cup K_i$; `// disjoint union of` G `with` K_i`, the complete graph on` i `vertices`
    if $\bar{\omega}_j(G') = \bar{\omega}_k(G')$ **then**
      return i
    end
  end
end

Algorithm 1: Calculating the k-club number using an oracle that checks if $\bar{\omega}_j(G) = \bar{\omega}_k(G)$.

Theorem 1. *Let $k \geq 2$ be a fixed integer. Assuming $\mathcal{P} \neq \mathcal{NP}$, no polynomial-time algorithm determines whether $\bar{\omega}_k(G) = \Delta(G) + 1$. Accordingly, the greedy heuristic of selecting a maximum degree vertex and its neighborhood is provably best for the maximum k-club problem.*

Proof. Assuming $\mathcal{P} \neq \mathcal{NP}$, no polynomial-time algorithm determines whether $\omega(G) = \bar{\omega}_k(G)$ for a graph G (by Proposition 1). Recall that $\omega(G) \leq \Delta(G) + 1 \leq \bar{\omega}_k(G)$. Note that it is easy to check if $\omega(G) = \Delta(G) + 1$; check if the neighborhood of a maximum degree vertex is a clique. Thus, a polynomial-time algorithm that determines whether $\Delta(G) + 1 = \bar{\omega}_k(G)$ could be used to check if $\omega(G) = \bar{\omega}_k(G)$ in polynomial time. □

It is perhaps interesting to note the ways in which the clique/k-club sandwich is different or similar to the Lovász theta sandwich. In both cases, the "meat" of the sandwich is polynomially computable, and the "bread" is $\mathcal{NP}$-hard to compute. However, the former provides a feasible solution, while the latter only provides a bound. Second, in the clique/k-club sandwich, both slices of bread are the result of maximization problems, whereas the Lovász sandwich involves both maximization and minimization.

2.2 The maximum clique problem

For any fixed $\epsilon > 0$, the maximum clique problem is inapproximable within a factor of $n^{1-\epsilon}$, unless $\mathcal{P} = \mathcal{NP}$ [25, 52]. This would then seem to suggest that the best one can hope for is to pick a single vertex since this gives an n-approximation. This, however, would not be provably best. This is because one can easily check if a graph has a clique of size two. In fact, no algorithm that searches for a clique of a constant size can be deemed provably best. However, there is a simple, linear-time heuristic that is provably best. It relies on a greedy, degree-based ordering, which is common in exact algorithms for the maximum clique problem [7, 11, 43].

Data: An undirected graph $G = (V, E)$.
Result: A maximal clique $C \subseteq V$ of G.
initialize $C \leftarrow \emptyset$ and sort vertices $(v_1, \ldots, v_n)$ by nonincreasing degree;
for $j = 1, \ldots, n$ **do**
  if $C \cup \{v_j\}$ *is a clique* **then**
    $C \leftarrow C \cup \{v_j\}$;
  end
end
return C

Algorithm 2: A provably best heuristic for the maximum clique problem.

Proposition 2. *Algorithm 2 is provably best for the maximum clique problem.*

Proof. Let $\mathcal{A}$ be an algorithm that determines whether the output of Algorithm 2 is optimal. We provide a Cook reduction that shows that if $\mathcal{A}$ runs in polynomial time, then it could be used to determine the clique number in polynomial time. Specifically, algorithm $\mathcal{A}$ is used in Algorithm 3 to determine whether $|C_i| = \omega(G')$.

Data: An undirected graph $G = (V, E)$.
Result: The clique number $\omega(G)$ of G.
initialize $G' = (V', E') \leftarrow G$ and $i \leftarrow 1$;
do
  $C_i \leftarrow$GreedyClique(G'); `// using Algorithm 2`
  if $|C_i| = \omega(G')$ **then**
    return $|C_i|$
  else
    $V' \leftarrow V' \cup \{v'_i\}$;
    $E' \leftarrow E' \cup \{\{v'_i, u\} \mid u \in C_i\}$;
    $G' \leftarrow (V', E')$;
    $i \leftarrow i + 1$;
  end
loop

Algorithm 3: Calculating the clique number using an optimality-checking oracle.

There are two key points to the correctness of Algorithm 3:

1. $C_{i+1} = C_i \cup \{v'_i\}$ for any $i \geq 1$, and
2. $\omega(G')$ does not change between iterations of the loop.

These two points would then imply that the number of iterations of the loop is at most $|V|$ and in each iteration $|V'| \leq 2|V|$, so the reduction is polynomial. The first point is clear by construction of G' and the ordering used in Algorithm 2. Now, see that the new vertex v'_i that is added to G' has $|C_i| < \omega(G')$ neighbors, so it cannot belong to a clique of size $> \omega(G')$. Thus, point 2 follows by induction on the number i of loop iterations. □

2.3 The minimum vertex coloring problem

The coloring problem is well known to be computationally intractable. It asks: given a graph G, color its vertices using a minimum number $\chi(G)$ of colors such that no pair of adjacent vertices take the same color. The problem of testing if a graph can be properly k-colored was one of Richard Karp's original 21 $\mathcal{NP}$-complete problems [32]. Within just a few years, coloring was shown to be $\mathcal{NP}$-complete in ever more restrictive cases; $\mathcal{NP}$-completeness holds for any fixed $k \geq 3$ [36] even in planar graphs [50] of

maximum degree four [16]. In contrast, 1-colorability is trivial to determine, and breadth-first search determines 2-colorability in linear time. The problem of approximating the chromatic number is also hard – finding an $(n^{1-\epsilon})$-approximate solution is $\mathcal{NP}$-hard for any constant $\epsilon > 0$ [52]. This might seem to suggest that the best one can do is to give each vertex its own color since this gives an n-approximation. However, this would not be provably best, as one can quickly check if the graph has an $(n-1)$-coloring.

Due to the very restrictive cases of coloring that remain hard, it is easy to show that several known algorithms are provably best. For example, since 3-colorability remains $\mathcal{NP}$-hard in planar graphs, the quadratic-time four-coloring algorithm of [46] is provably best for planar graphs. However, this algorithm is specifically designed to accept planar graphs as input. So, we require a heuristic that will properly color an arbitrary graph. In this case, recall Brooks' theorem [6] that states that a simple, connected graph G satisfies $\chi(G) \leq \Delta(G)$ unless it is complete or an odd cycle. Thus, since it is $\mathcal{NP}$-hard to determine if $\chi(G) < \Delta(G)$ (since testing 3-colorability of a graph with $\Delta(G) = 4$ is hard), we have the following result.

Proposition 3. *Any heuristic that finds a $\Delta(G)$-coloring (i.e., a Brooks coloring) of graph G is provably best for the vertex coloring problem.*

We note that there are many ways to find a Brooks coloring (when one exists), see e.g., [20, 24, 30, 31, 37, 49].

2.4 Grab bag: Independent Set, Clique Cover, Vertex Cover, (Connected) Dominating Set

Using common $\mathcal{NP}$-hardness reductions, we can establish provably best heuristics for the problems of finding: a maximum independent set, minimum clique cover, minimum vertex cover, and minimum (connected) dominating set.

Recall that a maximum independent set (resp., minimum clique cover) of a graph G can be found by finding a maximum clique (resp., minimum vertex coloring) of $\overline{G}$, the complement of G. Thus, in a provably best heuristic for the clique cover problem, take the complement graph and find a Brooks coloring. Also, Algorithm 4 is provably best for the maximum independent set problem, since it finds the same independent set that would be found by complementing the graph and finding a maximal clique via Algorithm 2.

Then recall that the complement $V \setminus C$ of a maximum independent set C is a minimum vertex cover. This implies that Algorithm 5 is provably

Data: An undirected graph $G = (V, E)$.
Result: A maximal independent set $C \subseteq V$ of G.
initialize $C \leftarrow \emptyset$ and sort vertices $(v_1, \ldots, v_n)$ by nondecreasing degree;
for $i = 1, \ldots, n$ **do**
  if $C \cup \{v_i\}$ *is an independent set* **then**
    $C \leftarrow C \cup \{v_i\}$;
  end
end
return C

Algorithm 4: A provably best heuristic for the maximum independent set problem.

best for the minimum vertex cover problem, since it finds the same minimal vertex cover that would be found using Algorithm 4.

Data: An undirected graph $G = (V, E)$.
Result: A minimal vertex cover $C \subseteq V$ of G.
initialize $C \leftarrow V$ and sort vertices $(v_1, \ldots, v_n)$ by nondecreasing degree;
for $i = 1, \ldots, n$ **do**
  if $C \setminus \{v_i\}$ *is a vertex cover* **then**
    $C \leftarrow C \setminus \{v_i\}$;
  end
end
return C

Algorithm 5: A provably best heuristic for the minimum vertex cover problem.

Finally, the same heuristic is provably best for the minimum (connected) dominating set problem. This follows by a standard reduction from the minimum vertex cover problem in a graph $G = (V, E)$ to the minimum (connected) dominating set problem in a graph $G' = (V', E_1 \cup E_2)$, where $V' = V \cup E,\ E_1 = \{\{u, v\} \mid u, v \in V,\ u \neq v\}$, and $E_2 = \{\{e, v\} \mid e = \{u, v\} \in E\}$. Algorithm 6 is provably best since it finds a minimal (connected) dominating set that maps to the minimal vertex cover found in Algorithm 5.

Data: A (connected) undirected graph $G = (V, E)$.
Result: A minimal (connected) dominating set $D \subseteq V$ of G.
initialize $D \leftarrow V$ and sort vertices $(v_1, \ldots, v_n)$ by nondecreasing degree;
for $i = 1, \ldots, n$ **do**
  if $D \setminus \{v_i\}$ *is a (connected) dominating set* **then**
    $D \leftarrow D \setminus \{v_i\}$;
  end
end
return D

Algorithm 6: A provably best heuristic for the minimum (connected) dominating set problem.

3 The Maximum Subgraph Satisfying Property Π Problem

After seeing the provably best heuristics in Section 2 for clique, independent set, vertex cover, and dominating set, one may notice that they are very similar. For the maximization problems, the strategy is a best-in greedy strategy, and the minimization problems use a worst-out strategy. Moreover, the problems themselves are very similar as the class of feasible solutions is closed under taking either supersets or subsets. However, it can be seen that these heuristics are not provably best for all *independence systems*, as evidenced by the maximum matching problem, where the best-in heuristic does not always find an optimal solution, but the problem is indeed polynomial-time solvable. But what happens if we restrict ourselves to Lewis-Yannakakis style graph problems?

Recall that a Lewis-Yannakakis graph property Π is hereditary on induced subgraphs, is satisfied by infinitely many graphs, and is not satisfied for infinitely many graphs. Examples of such properties include: planar, bipartite, acyclic, degree-constrained, and complete. The remarkable theorem of [35] implies that the problem of finding a largest subset S of vertices such that $G[S]$ satisfies Π is $\mathcal{NP}$-hard.

We describe a heuristic for this very general problem under the assumption that there is a known polynomial-time algorithm for determining whether a graph satisfies property Π. Note that, by Lewis and Yannakakis, Π will either be satisfied by all complete graphs or it will be satisfied by all empty (i.e., edgeless) graphs.

```
Data: An undirected graph G = (V, E) and a Lewis-Yannakakis
      property Π.
Result: A maximal subset C ⊆ V of vertices such that G[C] satisfies
        Π.
initialize C ← ∅;
if all complete graphs satisfy Π then
    sort vertices (v_1, ..., v_n) by nonincreasing degree;
else
    sort vertices (v_1, ..., v_n) by nondecreasing degree;
end
for i = 1, ..., n do
    if G[C ∪ {v_i}] satisfies Π then
        C ← C ∪ {v_i};
    end
end
return C
```

Algorithm 7: A heuristic for the maximum Π-subgraph problem.

Open Question. Is Algorithm 7 provably best for the maximum Π-subgraph problem for any Lewis-Yannakakis property Π?

4 Discussion and Concluding Remarks

We have shown that simple heuristics are *provably best* for some $\mathcal{NP}$-hard graph problems. The heuristics are "provably best," in the sense that they are hard to beat in the worst case; assuming $\mathcal{P} \neq \mathcal{NP}$, no polynomial-time heuristic always finds a better solution (when one exists). However, this should not prevent practitioners from designing more sophisticated approaches, since the analysis only shows that these heuristics are hard to beat in the worst case. Nevertheless, it does provide some theoretical justification for their use as a subroutine within a larger metaheuristic approach. Indeed, variants of these heuristics are commonly used within popular metaheuristics.

A notable contribution of this paper is a "sandwich theorem" that states that the quantity $\Delta(G) + 1$ lies between the clique number $\omega(G)$ and the k-club number $\bar{\omega}_k(G)$ of a graph G, yet it is difficult to tell whether $\omega(G) = \bar{\omega}_k(G)$ when $k \geq 2$. Our work shows that $\Delta(G) + 1$ is a provably best upper bound for the maximum clique problem, as well as a provably best

lower bound for the maximum k-club problem. This is analogous to the Lovász function's "sandwich" relationship with the independence and clique partitioning numbers of a graph.

Admittedly, there are potential drawbacks to a provably best analysis presented in this paper. A provably best heuristic may perform very well on a subset of hard instances, but poorly on others. As an example, consider the four-coloring algorithm of [46] for planar graphs. A modified version of this algorithm provides a provably best heuristic for the coloring problem in arbitrary graphs that performs rather poorly for most instances. It operates as follows: if the input graph is planar, find a four-coloring; otherwise give every vertex its own color. However, one could argue that *any* heuristic that attempts to solve the coloring problem in arbitrary instances is doomed to perform poorly (due to the infamous inapproximability of coloring). Thus, it may be fruitful to explore provably best heuristics for restricted classes of input instances. This may be more satisfying when particular types of instances are more likely to occur in practice.

On a positive note, provably best heuristics can be used to rule out inferior approaches. There is little use in a heuristic $\mathcal{A}$ that has been shown to *not* be provably best. In this case, there is another polynomial-time heuristic $\mathcal{A}'$ that returns optimal solutions anytime that $\mathcal{A}$ does. Furthermore, $\mathcal{A}'$ returns a strictly better solution on all other instances. A possible caveat is that $\mathcal{A}'$ might be significantly slower than $\mathcal{A}$. However, most of the provably best heuristics discussed in this paper can be implemented to run in linear time.

We note that a slight modification to our provably best definition renders it worthless. In this paper, we say that a heuristic is provably best if no other heuristic *always* outperforms it (when possible). One may want to relax "always" to "at least once." Under this modification, no heuristic can be provably best for $\mathcal{NP}$-optimization problems that do not admit polynomial-time algorithms. Indeed, suppose that a polynomial-time heuristic $\mathcal{A}$ is provably best (for a hard problem) under the modified, relaxed definition. Since the problem does not admit polynomial-time algorithms, $\mathcal{A}$ must return a suboptimal solution on at least one instance x. Note that x is of finite size and thus can be solved in constant time. Then the new heuristic $\mathcal{A}'$ that returns an optimal solution when it encounters instance x and simulates $\mathcal{A}$ otherwise demonstrates that $\mathcal{A}$ cannot be provably best, a contradiction.

Many important problems that are tackled with metaheuristics were not considered in this paper. A provably best analysis can be performed to justify the use of existing as well as future construction heuristics (or to rule out inferior ones). The relative simplicity of our analysis in this paper

suggests that it should not be hard to apply to other problems.

Acknowledgments

This material is based upon work supported by the AFRL Mathematical Modeling and Optimization Institute. Partial support by AFOSR under grants FA9550-12-1-0103, FA8651-12-2-0011, and FA9550-11-1-0037 is also gratefully acknowledged. The research of Oleg A. Prokopyev was also supported by a US Air Force Summer Faculty Fellowship. The authors thank two anonymous referees and the guest editors, whose comments have improved the paper.

References

[1] B. Balasundaram, S. Butenko, and S. Trukhanov, Novel approaches for analyzing biological networks, J Combinatorial Optim 10 (2005), 23–39.

[2] C. Blum and A. Roli, Metaheuristics in combinatorial optimization: Overview and conceptual comparison, ACM Comput Surveys 35 (2003), 268–308.

[3] I.M. Bomze, M. Budinich, P.M. Pardalos, and M. Pelillo, "The maximum clique problem," Handbook of combinatorial optimization, D.Z. Du and P.M. Pardalos (Editors), Springer, 1999, pp. 1–74.

[4] J.M. Bourjolly, G. Laporte, and G. Pesant, Heuristics for finding k-clubs in an undirected graph, Comput & Oper Res 27 (2000), 559–569.

[5] J.M. Bourjolly, G. Laporte, and G. Pesant, An exact algorithm for the maximum k-club problem in an undirected graph, Eur J Oper Res 138 (2002), 21–28.

[6] R.L. Brooks, On colouring the nodes of a network, Math Proc Cambridge Philosophical Soc, Vol. 37, Cambridge Univ Press, 1941, pp. 194–197.

[7] A. Buchanan, J.L. Walteros, S. Butenko, and P.M. Pardalos, Solving maximum clique in sparse graphs: An $O(nm + n2^{d/4})$ algorithm for d-degenerate graphs, Optim Lett 8 (2014), 1611–1617.

[8] S. Busygin and D.V. Pasechnik, On NP-hardness of the clique partition – independence number gap recognition and related problems, Discr Math 304 (2006), 460–463.

[9] S. Butenko, X. Cheng, C.A. Oliveira, and P.M. Pardalos, "A new heuristic for the minimum connected dominating set problem on ad hoc wireless networks," Recent developments in cooperative control and optimization, S. Butenko, R. Murphey, and P.M. Pardalos (Editors), Kluwer Academic Publishers, 2004, pp. 61–73.

[10] S. Butenko and W.E. Wilhelm, Clique-detection models in computational biochemistry and genomics, Eur J Oper Res 173 (2006), 1–17.

[11] R. Carraghan and P.M. Pardalos, An exact algorithm for the maximum clique problem, Oper Res Lett 9 (1990), 375–382.

[12] M. Chudnovsky, G. Cornuéjols, X. Liu, P. Seymour, and K. Vuskovic, Recognizing Berge graphs, Combinatorica 25 (2005), 143–186.

[13] D.Z. Du and P.J. Wan, Connected dominating set: Theory and applications, Springer, 2013.

[14] T.A. Feo and M.G.C. Resende, Greedy randomized adaptive search procedures, J Global Optim 6 (1995), 109–133.

[15] T.A. Feo, M.G.C. Resende, and S.H. Smith, A greedy randomized adaptive search procedure for maximum independent set, Oper Res 42 (1994), 860–878.

[16] M.R. Garey, D.S. Johnson, and L. Stockmeyer, Some simplified NP-complete graph problems, Theoret Comput Sci 1 (1976), 237–267.

[17] M. Gendreau and J.Y. Potvin, Metaheuristics in combinatorial optimization, Ann Oper Res 140 (2005), 189–213.

[18] F. Glover and G.A. Kochenberger (Editors), Handbook of metaheuristics, Kluwer Academic Publishers, 2003.

[19] F. Glover and A.P. Punnen, The travelling salesman problem: New solvable cases and linkages with the development of approximation algorithms, J Oper Res Soc 48 (1997), 502–510.

[20] D.A. Grable and A. Panconesi, Fast distributed algorithms for Brooks-Vizing colourings, Proc Ninth Ann ACM-SIAM Symp Discr Algorith, Society for Industrial and Applied Mathematics, 1998, pp. 473–480.

[21] M. Grötschel, L. Lovász, and A. Schrijver, Geometric algorithms and combinatorial optimization, Second edition, Springer-Verlag, Berlin, 1993.

[22] G. Gutin and A. Yeo, "Domination analysis of combinatorial optimization algorithms and problems," Graph theory, combinatorics and algorithms, M. Golumbic and I. Ben-Arroyo Hartman (Editors), Springer, 2005, pp. 145–171.

[23] G. Gutin, A. Yeo, and A. Zverovich, Traveling salesman should not be greedy: Domination analysis of greedy-type heuristics for the TSP, Discr Appl Math 117 (2002), 81–86.

[24] P. Hajnal and E. Szemerédi, Brooks coloring in parallel, SIAM J Discr Math 3 (1990), 74–80.

[25] J. Håstad, Clique is hard to approximate within $n^{1-\varepsilon}$, Acta Mathematica 182 (1999), 105–142.

[26] T.W. Haynes, S.T. Hedetniemi, and P.J.B. Slater, Fundamentals of domination in graphs, Vol. 208 of *Pure and Applied Mathematics*, Marcel Dekker, Inc, 1998.

[27] D.S. Hochbaum (Editor), Approximation algorithms for NP-hard problems, PWS Publishing Co., 1996.

[28] D.S. Hochbaum and D.B. Shmoys, A best possible heuristic for the k-center problem, Math Oper Res 10 (1985), 180–184.

[29] A. Jagota and L.A. Sanchis, Adaptive, restart, randomized greedy heuristics for maximum clique, J Heuristics 7 (2001), 565–585.

[30] M. Karchmer and J. Naor, A fast parallel algorithm to color a graph with Δ colors, J Algorithms 9 (1988), 83–91.

[31] H.J. Karloff, An NC algorithm for Brooks' theorem, Theoret Comput Sci 68 (1989), 89–103.

[32] R.M. Karp, "Reducibility among combinatorial problems," Complexity of computer computations, R.E. Miller and J.W. Thatcher (Editors), Plenum Press, 1972, pp. 85–103.

[33] D.E. Knuth, The sandwich theorem, Electronic J. Combin 1 (1994), 1–48.

[34] R.E. Ladner, N.A. Lynch, and A.L. Selman, A comparison of polynomial time reducibilities, Theoret Comput Sci 1 (1975), 103–123.

[35] J.M. Lewis and M. Yannakakis, The node-deletion problem for hereditary properties is NP-complete, J Comput System Sciences 20 (1980), 219–230.

[36] L. Lovász, Coverings and colorings of hypergraphs, Proc. Fourth Southeastern Conference Combin, Graph Theory, Comput, Utilitas Mathematica Publishing, Winnipeg, 1973, pp. 3–12.

[37] L. Lovász, Three short proofs in graph theory, J Combinatorial Theory, Ser B 19 (1975), 269–271.

[38] L. Lovász, On the Shannon capacity of a graph, IEEE Trans Informat Theory 25 (1979), 1–7.

[39] R.D. Luce and A.D. Perry, A method of matrix analysis of group structure, Psychometrika 14 (1949), 95–116.

[40] F. Mahdavi Pajouh and B. Balasundaram, On inclusionwise maximal and maximum cardinality k-clubs in graphs, Discr Optim 9 (2012), 84–97.

[41] E. Malaguti and P. Toth, A survey on vertex coloring problems, Int Trans in Oper Res 17 (2010), 1–34.

[42] R.J. Mokken, Cliques, clubs and clans, Qual & Quantity 13 (1979), 161–173.

[43] P.R.J. Östergård, A fast algorithm for the maximum clique problem, Discr Appl Math 120 (2002), 197–207.

[44] J. Pattillo, N. Youssef, and S. Butenko, On clique relaxation models in network analysis, Eur J Oper Res 226 (2013), 9–18.

[45] M.G.C. Resende and P.M. Pardalos (Editors), Handbook of optimization in telecommunications, Springer, 2008.

[46] N. Robertson, D.P. Sanders, P. Seymour, and R. Thomas, Efficiently four-coloring planar graphs, Proc Twenty-Eighth Ann ACM Symp Theory Computi, ACM, 1996, pp. 571–575.

[47] S. Sahni and T. Gonzalez, P-complete approximation problems, J ACM 23 (1976), 555–565.

[48] S. Shahinpour and S. Butenko, "Distance-based clique relaxations in networks: s-clique and s-club," Models, algorithms, and technologies for network analysis, B.I. Goldengorin, V.A. Kalyagin, and P.M. Pardalos (Editors), Springer, 2013, Vol. 59 of Springer Proceedings in Mathematics and Statistics, pp. 149–174.

[49] S. Skulrattanakulchai, Δ-list vertex coloring in linear time, Informat Process Lett 98 (2006), 101–106.

[50] L. Stockmeyer, Planar 3-colorability is polynomial complete, ACM SIGACT Ne 5 (1973), 19–25.

[51] V.V. Vazirani, Approximation algorithms, Springer, 2001.

[52] D. Zuckerman, Linear degree extractors and the inapproximability of max clique and chromatic number, Proc Thirty-Eighth Ann ACM Symp Theory Computi, ACM, 2006, pp. 681–690.