Greedy local improvement and weighted set packing approximation

Barun Chandra* Magnús M. Halldórsson†

Abstract

Given a collection of weighted sets, each containing at most k elements drawn from a finite base set, the k-set packing problem is to find a maximum weight subcollection of disjoint sets. A greedy algorithm for this problem approximates it to within a factor of k, and natural local search has been shown to approximate it to within a factor of roughly $k-1$. However, neither paradigm can yield approximations that improve on this.

We present an approximation algorithm for the weighted k-set packing problem that combines the two paradigms by starting with an initial greedy solution and then repeatedly choosing the best possible local improvement. The algorithm has a performance ratio of $2(k+1)/3$, which we show is asymptotically tight. This is the first asymptotic improvement over the straightforward ratio of k.

1 Introduction

We consider the following problem:

Weighted k-set packing Given a collection of sets, each of which has an associated real weight and contains at most k elements drawn from a finite base set, find a collection of disjoint sets of maximum total weight.

Set packing is a fundamental combinatorial problem that underlies a range of practical and theoretical problems. The restriction to sets of size at most k properly includes multi-dimensional matching, which is a generalization of the ordinary graph matching problem.

k-Set Packing can be further generalized to a certain independent set problem. The intersection graph of a set system of k-sets has the property that no induced subgraph contains a $k+1$-*claw*, which is a set of $k+1$ independent (i.e. mutually non-adjacent) vertices that share a common neighbor. A set packing corresponds to an independent set in the intersection graph. The class of $(k+1)$-claw free graphs, however, properly includes these intersection graphs.

These problems are NP-hard for any $k \geq 3$, even in the unweighted case [3], thus we seek heuristics with guaranteed solution quality. The most natural heuristic for the Weighted k-Set Packing problem is the *greedy* algorithm: Add to the current solution a set of maximum weight, remove it and all sets that intersect it, and repeat until all sets have been removed. It is easy to see that this solution is within a factor of k from optimal: from the sets removed in each iteration, the optimal solution can contain at most k sets (at most one for each element of our chosen set) all of which are of weight at most that of our chosen set. It is also easy to construct examples that show that this factor of k cannot be improved.

Another natural strategy is local search: Attempt to replace a small subset of the solution by some collection of greater total weight that does not intersect the remainder of the solution. For such a search to be polynomially bounded, it has to be restricted in some way, such as that either the number of sets added or the number of sets removed should be constant. In the case of unweighted sets, Hurkens and Schrijver [5] showed that a local search algorithm leads to an approximation of $k/2+\epsilon$, where $\epsilon > 0$ depends on the size of the change set. This is the best performance ratio known to date for the unweighted k-Set Packing problem. A restricted form of this local search was considered in [4] with the same performance but decreased complexity.

Local search for the weighted case was analyzed by Arkin and Hassin [1] and Bafna, Narayan and Ravi [2], independently. They showed that a local search algorithm (which bounds the number of sets added) yields an approximation of $k-1+\epsilon$, and showed this to be a tight bound.

The results proven for local search apply to a very general situation that contains much non-determinism: any locally optimal solution achieves the given bound, independent of the starting solution or the particular sequence of feasible improvements. It leaves open the question if there exist easily computable rules for choosing a starting solution and for deciding among candidate improvements such that the resulting locally optimal solutions are guaranteed to be better. In particular, can

*University of New Haven, Connecticut, U.S.A. barun@charger.newhaven.edu

†Science Institute, University of Iceland, Reykjavik, Iceland. mmh@hi.is

one get an improved performance ratio by starting with a greedy solution and choosing improvements that yield bigger gains to the solution? Notice that the latter is necessary, since it is possible to show that even if starting with a greedy solution, indiscriminate choices of improvements will lead to a solution no better than for pure local search.

We answer this question in the positive, and obtain the first asymptotic improvement in the approximability of the problem. We present a natural heuristic BestImp that combines the greedy and local search paradigms by starting with an initial greedy solution and then repeatedly choosing the best possible local improvement. Its performance ratio is at most $2(k+1)/3$, which is asymptotically tight.

In order to further examine the effect of the choice of the improvement (when more than one local improvement applies) on the quality of the solution, we consider another algorithm, AnyImp, that also combines the greedy and local search paradigms. The difference is that AnyImp just looks for an improvement that leads to a gain bigger than a specified threshold, instead of looking for the best improvement. The proof technique we use to obtain upper bounds on performance ratio can be understood in a simpler setting with AnyImp. We illustrate it by deriving an asymptotically tight bound on the performance ratio of AnyImp as a function of the threshold, which for the best choice of a threshold is at most $(4k+2)/5$. It is interesting to find that picking the best improvement instead of a good enough improvement leads to a substantially better performance ratio.

The organization of the rest of the paper is as follows. Section 2 contains formal definitions of the problem, statement of the algorithms, and bounds on their time complexity. The key concepts of the analysis are considered in Section 3 along with some of their properties. Section 3 also contains the analysis of the upper bound of AnyImp, while the analysis of the upper bound of BestImp is presented in Section 4. A construction is given in Section 5 that establishes a lower bound on the performance ratio of BestImp. Section 6 ends with conclusions and open issues.

2 Definitions, Notation, Problem Statements

The *weighted independent set* problem is as follows: Given a vertex-weighted graph $G = (V, E)$, find a maximum weight subset of mutually non-adjacent vertices; i.e. a subset $V' \subseteq V$ such that $v_i, v_j \in V'$ implies $(v_i, v_j) \notin E$.

The intersection graph of a set system contains a vertex for each set with edges between vertices whose corresponding sets intersect. Such graphs have the property that they contain no $k+1$-*claw*, i.e. a $k+1$-independent set in the neighborhood of some vertex. A *set packing* of a set system is a collection of mutually disjoint sets, and it corresponds to an independent set of the associated intersection graph. In the weighted version, sets in the set system have weights, which translate to vertex weights in the intersection graph. Since the independent set problem in claw-free graphs generalizes the set packing problem, we state our algorithms and results in terms of the former.

The approximation ratio $\rho(G)$ of a heuristic algorithm on a given graph G is the ratio between the size of the optimal solution and the size of the algorithm's solution on G. The *performance ratio* ρ of the algorithm is the maximum approximation ratio over all instances.

Let $G = (V, E)$ be a graph, $U, W \subseteq V$, $x \in V$. The neighborhood of x, $N(x)$, is the set of all vertices from V which are adjacent to x. $N_W(x)$ is defined to be $N(x) \cap W$. These definitions are extended to neighborhoods of sets of vertices: $N(U) = \cup_{u \in U} N(u)$, $N_W(U) = N(U) \cap W$. $deg(x) = |N(x)|$ and $deg_W(x) = |N_W(x)|$. Note that if G is $k+1$-claw free and W is an independent set, $deg_W(x) \leq k$.

The Algorithms AnyImp$_\alpha$ **and** BestImp Greedy is the natural greedy algorithm for the independent set problem which works by repeatedly picking the heaviest vertex from among the remaining vertices and eliminating it and adjacent vertices. Let Gr denote the independent set selected by Greedy on input graph G.

Our algorithms are based on the following type of a local improvement. Let I be an independent set, x a member of I, and $Q \subseteq N(x)$ be an independent set. Form a new independent set I' by adding Q to I and removing those vertices of I which are adjacent to any vertex of Q: $I' = (I \cup Q) - N_I(Q)$.

We define the *payoff factor* of the improvement to be $w(Q)/w(N_I(Q))$. For $\alpha > 1$, an α-*improvement* is an improvement with payoff factor α, and an α-*good* improvement is an improvement with payoff factor at least α. A solution is α-locally optimal if it has no α-good improvement, and *locally optimal* if no local improvement exists.

Both of our algorithms, AnyImp$_\alpha$ and BestImp, start with the initial greedy independent set Gr, and repeatedly make local improvements until they reach a local optima. In each iteration they find a improvement I' to the current solution I, and then arbitrarily extend it to a maximal solution I''. The difference lies in which improvement is made: BestImp makes the improvement with the highest payoff factor, while AnyImp$_\alpha$ makes any α-good improvement.

Let us view the addition of vertices (to a nonmaximal independent set) as improvements of nearly infinite payoff factor, and further ordered according to the weight of the vertex added. Then, BestImp can be more succinctly stated as follows:

```
BESTIMP(G)
 I ← ∅
 while I is not locally optimal do
   Let I′ be improvement of maximum payoff factor
   I ← I′
 od
 output I
```

Complexity analysis The algorithms as presented are not polynomially bounded, since the improvements made can be arbitrarily small and the length of the improvement sequence can be exponential in the length of the input. In fact, the theory of PLS-completeness indicates that finding locally optimal solutions to many weighted problems may be not much easier than finding optimal solutions. We are, however, not interested in locally optimal solutions *per se*; rather, they are our vehicle for finding solutions with good performance bounds.

One solution to this problem is to truncate the weight of the vertices to integer multiples of $\frac{1}{n^2}w(Gr)$. Since we know that Gr is within a factor k from optimal, this limits the number of improvements to n^2k. This may underestimate the size of the optimal solution, but only by an additive $w(Gr)/n \leq OPT/n$ term which is negligible.

The cost of each improvement step depends on the time it takes to explore all independent sets within the neighborhood of each vertex in the solution. This is bounded by $O(nk^2\Delta^k)$, where Δ is the maximum degree of the graph.

3 Proof technique

In this section we describe the proof technique which we use to prove upper bounds on the performance ratios of AnyImp and BestImp, and apply it to AnyImp. A central task in the analysis of an approximation algorithm is to identify the computational structure that provides a bound on the optimal solution and its relationship with the heuristic solution. E.g. Christofides' TSP heuristic uses both a minimum spanning tree and a minimum weight matching. In some cases, as in Christofides' heuristic, the structure is a part of the heuristic solution; in other cases, it is an entirely different beast, that may not even be polynomial computable, but in terms of which both the optimal and the heuristic solution can be bounded.

The Projection We introduce the concept of a *projection* of the optimal solution OPT to a given solution I. Each vertex v of the graph is assigned a *representative*, which is the maximum weight vertex in I that is adjacent to v (or identical to v, when v is also in I). Since we may assume I to be maximal, each vertex is assigned a representative. We are only interested in the representatives of vertices in OPT. The projection of OPT onto I is the multiset of all the representatives of OPT. We will overload the term to also denote its weight, i.e. the (weighted) sum of the representatives' weights.

The projection is crucial to the analysis, acting as a stored potential or unused capability of the algorithm's solution. We will show that the projection has certain properties, which can be intuitively thought of as follows:

Property 1: The projection starts high compared to the optimal solution value.

Property 2: If the value of the projection goes down, the weight of the algorithm's current independent set goes up.

Property 3: The weight of the locally optimal solution (i.e. the final independent set of the algorithm) is large compared to the projection.

We will use these to argue that the weight of the algorithm's solution is large compared to the optimal solution. The intuition is that if the projection does not decrease much from beginning to end, then by Property 3 the algorithm's solution is large compared to the projection which, by Property 1, is large compared to the optimal solution. On the other hand, if the value of the projection decreases a lot as the algorithm progresses, then by Property 2 the algorithm's solution improves a lot. In other words, either the initial greedy solution is already a good one, or the final solution will be a significant improvement on the greedy solution.

Let OPT be a particular independent set of maximum weight. We overload OPT to also refer to the weight of the set. We associate with each maximal independent set I a function $f_I : OPT \rightarrow I$ which maps elements in OPT to vertices in I. $f_I(b)$, the representative of b, is the vertex of maximum weight among neighbors of b in I, or b itself if b is also in I.

For an element v in I, the *pre-image* of v is the set of elements that map to v.

$$S_I(v) = \{b \in OPT : f_I(b) = v\}.$$

We omit the subscript when I is clear from context.

DEFINITION 3.1. *For an independent set I, the* projection *of OPT to I is defined as*

$$proj(I) = \sum_{b \in OPT} w(f_I(b)) = \sum_{v \in I} |S_I(v)| w(v).$$

Properties of the projection If we consider the greedy solution Gr, we see that the weight of any element of OPT is at most the weight of its representative, as otherwise greedy would have chosen it instead of the representative. It follows that

$$proj(Gr) \geq OPT. \tag{3.1}$$

Since the graph is $k+1$-claw free, any preimage contains at most k elements, and thus we have for any I, $proj(I) \leq k \cdot w(I)$. This combines with (3.1) to yield a simple bound on the greedy solution.

$$w(Gr) \geq OPT/k. \tag{3.2}$$

LEMMA 3.1. *Let I' be obtained from I via α-good improvements. Then,*

$$\frac{k}{\alpha - 1} \cdot w(I') + proj(I') \geq \frac{k}{\alpha - 1} \cdot w(I) + proj(I).$$

Proof. Let us split OPT into OPT_0, containing those b such that $f_I(b) \in I \cap I'$ (though $f_{I'}(b)$ may not be in I), and the rest $OPT_1 = OPT - OPT_0$, of those b where $f_I(b) \in I - I'$.

First observe that for $b \in OPT_0$, $w(f_{I'}(b)) \geq w(f_I(b))$. We then bound the difference in the projections.

$$\begin{aligned}
&(3.3)\quad proj(I) - proj(I') \\
&\qquad \leq \sum_{b \in OPT_1} w(f_I(b)) - w(f_{I'}(b)) \\
&\qquad \leq \sum_{b \in OPT_1} w(f_I(b)) = \sum_{v \in I - I'} |S_I(v)| w(v) \\
&(3.4)\quad \leq k \cdot w(I - I').
\end{aligned}$$

Since I' is obtained from I through α-good improvements, $I_0 = I, I_1, \ldots, I_n = I'$, we have that

$$\begin{aligned}
w(I' - I) &= w(I_n - I_{n-1}) + \cdots + w(I_1 - I_0) \\
&\geq \alpha w(I_{n-1} - I_n) + \cdots + \alpha w(I_0 - I_1) \\
&= \alpha w(I - I').
\end{aligned}$$

Thus,

$$\begin{aligned}
w(I') - w(I) &= w(I' - I) - w(I - I') \\
&\geq (\alpha - 1) w(I - I'). \qquad (3.5)
\end{aligned}$$

Combining (3.4) with (3.5) establishes the lemma.

Finally, the projection also has a meaning in the context of a locally optimal solution.

LEMMA 3.2. *For a α-locally optimal solution I,*

$$(k+1) w(I) \geq \frac{1}{\alpha} OPT + proj(I).$$

Proof. First note that the set of preimages of elements of a maximal independent set partitions the optimal solution. Thus,

$$OPT = \sum_{v \in I} w(S(v)). \tag{3.6}$$

Since, for any vertex $v \in I$, the independent set $I \cup S(v) - N(S(v))$ is not an α-improvement over I,

$$w(S(v)) < \alpha \cdot w(N_I(S(v))).$$

Adding over all vertices in I, applying (3.6), we have that

$$\frac{1}{\alpha} OPT < \sum_{v \in I} w(N_I(S(v))).$$

The right hand side can be rewritten as

$$\sum_{v \in I} |\{u : v \in N(S(u)\}| w(v).$$

Each vertex v is adjacent to all of $S(v)$, but at most k vertices in OPT in total, thus it is adjacent to at most $k - |S(v)|$ vertices in $OPT - S(v)$. It follows that

$$\begin{aligned}
\frac{1}{\alpha} OPT &< \sum_{v \in I, |S(v)| > 0} w(v) + \sum_{v \in I} (k - |S(v)|) w(v) \\
&\leq (k+1) w(I) - proj(I).
\end{aligned}$$

The AnyImp **algorithm** Given the tools we have proved for the projection, we are now ready to prove an upper bound on ρ_α, the performance ratio of AnyImp_α.

THEOREM 3.1. $\rho_\alpha \leq \dfrac{k + 1 - \frac{1}{\alpha}}{1 + \frac{1}{\alpha} - \frac{1}{\alpha^2}}$. *In particular,* $\rho_2 \leq \dfrac{4k+2}{5}$.

Any solution I of AnyImp_α, whether initial, intermediate, or final, satisfies

$$\frac{k}{\alpha - 1} \cdot w(I) + proj(I) \geq \frac{\alpha}{\alpha - 1} OPT. \tag{3.7}$$

This is true for the initial greedy solution from (3.1) and (3.2), and follows for the other solutions from Lemmas 3.1.

The solution ALG of the algorithm ANYIMP$_\alpha$ satisfies the conditions of both Lemma 3.2 and (3.7). Combining them, we get

$$\left(k+1+\frac{k}{\alpha-1}\right) w(ALG) \geq \left(\frac{1}{\alpha}+\frac{\alpha}{\alpha-1}\right) OPT,$$

from which Theorem 3.1 follows.

4 Performance analysis of the BESTIMP algorithm

We start by noting that as BESTIMP makes successive local improvements, the payoff factors of these improvements need not be monotonically decreasing. Since this monotonically decreasing property is needed in our analysis, we simulate it as follows. Among the improvements made by the algorithm, let X_i, $i = t, t-1, \ldots, 1$, be the first improvement where the payoff factor drops to a new low. That is, X_t is the first improvement made on the greedy solution, X_{t-1} the first improvement whose payoff factor is less than that of X_t, etc. Let d_i be the payoff factor of X_i, and define $d_0 = 1$ and $d_{t+1} = d_t$. Let I_{i+1} be the independent set obtained by the algorithm before X_i is applied, and let I_1 be the final solution.

We analyze the weight of the sucessively improved solutions using the following potential function.

DEFINITION 4.1. *Let* $\Phi(I, d) = \dfrac{k+1}{d-1} w(I) + proj(I)$.

Applying Lemmas 3.1 and 3.2, and (3.1), we obtain the following relationships, for $i = 1, \ldots, t$, $j = 1, \ldots, t+1$.

$$\text{(4.8)} \qquad \Phi(I_i, d_i) \geq \Phi(I_{i+1}, d_i),$$

$$\text{(4.9)} \qquad (k+1)w(I_j) \geq \frac{1}{d_{j-1}} OPT + proj(I_j),$$

$$\text{(4.10)} \qquad proj(I_{t+1}) \geq OPT$$

For (4.8), we used, in addition to Lemma 3.1, the fact that all local improvements leading from I_{i+1} to I_i are d_i-good and that $w(I_i) \geq w(I_{i+1})$.

Consider the function

$$h(i) = \sum_{j=i}^{t-1} \frac{d_i}{d_j} \left(\frac{1}{d_j} - \frac{1}{d_{j+1}}\right) + \frac{d_i}{d_t}\frac{1}{d_t}, \quad i = 1, \ldots, t.$$

Also, define $h(t+1) = h(t) = 1/d_t$. Observe that h satisfies the recurrence relation

$$h(i) = h(i+1)\frac{d_i}{d_{i+1}} + \frac{1}{d_i} - \frac{1}{d_{i+1}}, \quad i = 1, \ldots, t.$$

Further, we can simplify it using that $\frac{1}{d_j} \geq \frac{1}{x}$, for $x \in [d_j, d_{j+1}]$:

$$\begin{aligned} h(i) &= d_i \left[\sum_{j=i}^{t-1} \frac{1}{d_j}\left(\int_{d_j}^{d_{j+1}} \frac{1}{x^2}dx\right) + \frac{1}{d_t^2}\right] \\ &\geq d_i \left[\int_{d_i}^{d_t} \frac{1}{x^3}dx + \frac{1}{d_t^2}\right], \\ &= d_i \left(\frac{1}{2d_i^2} - \frac{1}{2d_t^2} + \frac{1}{d_t^2}\right) > \frac{1}{2d_i}. \end{aligned}$$

LEMMA 4.1. *The following two inequalities hold for* $i = 1, 2, \ldots, t+1$*:*

$$\text{(4.11)} \qquad \Phi(I_i, d_i) \geq \frac{d_i + h(i)}{d_i - 1} OPT,$$

$$\text{(4.12)} \qquad (k+1)w(I_i) \geq \left[1 + \frac{h(i)}{d_i} + \frac{d_i - 1}{d_i d_{i-1}}\right] OPT.$$

Proof. The proof proceeds by induction.

Consider the base case, $i = t+1$. Recall that $d_{t+1} = d_t$ and $h(t+1) = h(t) = 1/d_t$. Then, by applying (4.9) and (4.10),

$$\begin{aligned} \Phi(I_{t+1}, d_{t+1}) &= \frac{k+1}{d_t - 1} w(I_{t+1}) + proj(I_{t+1}) \\ &\geq \left(\frac{1+1/d_t}{d_t - 1} + 1\right) OPT = \frac{d_{t+1} + h(t+1)}{d_{t+1} - 1} OPT. \end{aligned}$$

Also, the same equations imply that

$$\begin{aligned} (k+1)w(I_{t+1}) &\geq (1 + \frac{1}{d_t}) OPT \\ &= \left[1 + \frac{h(t+1)}{d_{t+1}} + \frac{d_{t+1} - 1}{d_{t+1} d_t}\right] OPT. \end{aligned}$$

Suppose the claim holds for $i = q+1$. We show that it then also holds for $i = q$. From this, the claim follows by induction. By (4.8),

$$\begin{aligned} \Phi(I_q, d_q) &\geq \Phi(I_{q+1}, d_q) \\ &= \Phi(I_{q+1}, d_{q+1}) + (k+1)w(I_{q+1}) \left(\frac{1}{d_q - 1} - \frac{1}{d_{q+1} - 1}\right). \end{aligned}$$

By applying (4.11) and (4.12) inductively on both terms of the right hand side, we obtain, after considerable algebraic simplification, (4.11) for the case $i = q$.

Furthermore, using the above, along with (4.9) (with $i = q$), we have

$$\left(1 + \frac{1}{d_q - 1}\right)(k+1)w(I_q) \geq \left[\frac{1}{d_{q-1}} + \frac{d_q + h(q)}{d_q - 1}\right] OPT,$$

which, when rearranged, establishes (4.12).

THEOREM 4.1. *The performance ratio of* BESTIMP *is at most* $2(k+1)/3$.

Proof. By (4.12), the final solution I_1 obtained by the algorithm satisfies

$$\begin{aligned} (k+1)w(I_1) &\geq \left[1 + \frac{1}{2d_1^2} + \frac{1}{d_0} - \frac{1}{d_1 d_0}\right] OPT \\ &= \left[2 - \frac{1}{d_1} + \frac{1}{2d_1^2}\right] OPT. \end{aligned}$$

The right hand term is minimized when $d_1 = 1$, for the bound claimed.

5 Lower bound constructions

The focus of this section is on showing the following lower bound on the performance of BESTIMP.

THEOREM 5.1. *The performance ratio of* BESTIMP *is at least* $(2/3)k - o(k)$.

We first illustrate the idea behind the lower bound by constructing a graph for which the approximation ratio of BESTIMP is $(4/7)k - o(k)$. The stronger $(2/3)k - o(k)$ lower bound construction is a generalization of this simpler construction.

Simpler construction The intuition behind the construction is as follows: the graph is "almost bipartite" with most of the edges being between V, the optimal independent set, and the other vertices. V has a large number of vertices of large weight. The set of other vertices can be partitioned into two subsets: one subset $(A_0 \cup A_1)$ has a small number of vertices of large weight, while the other subset $(B \cup C)$ has a large number of vertices of small weight. The greedy algorithm will first pick $A_0 \cup A_1$, which will eliminate V from the greedy solution, and then pick B. Subsequently, when the local improvements are being made, vertices from V will not be picked because equally good improvements result in replacing A_1 by C. After this replacement, the independent set $A_0 \cup B \cup C$ will be locally optimal.

More formally, let X be an appropriate large number and $Y = \frac{1}{4(k-1)}X$. The graph G has vertex sets A_0, A_1, B, C, V, of cardinalities $\frac{1}{k}X - Y$, Y, $(\frac{1}{2} + \frac{1}{k})X$, $kY = \frac{1}{4}X + Y$, and X, respectively. The weight of vertices is 1 in A_0, A_1 and V, $1/k$ in B, and $2/k$ in C. Split V into V_0 and V_1, of sizes $X - kY$ and kY.

Edges are either between A_1 and C, or have one end point in V. A_0 and V_0, as well as A_1 and V_1, form a $(k,1)$-regular bipartite graph, A_1 and C form a $(k,1)$-regular bipartite graph, B and V form a $(k, k/2+1)$-regular bipartite graph, and $A_0 \cup C$ and V form a $(k, k/4+1)$-regular bipartite graph. We additionally require that there be a perfect matching between C and V_1.

Finally, we require the input graph to contain no 4-cycles, i.e. that any two vertices have at most one common neighbor. This can be achieved by transforming the graph G into a graph G', where each vertex in G has multiple copies in G', with edges chosen carefully. We omit the details here, but refer to the literature [5, 1]. This completes the specification of the graph.

We first verify that the graph constructed is $k+1$-claw free. It clearly holds for vertices in A_0, B and V, since they are of degree at most k. The vertices can be so ordered that for any vertex x in A_1, its neighbors in C have a complete matching with its neighbors in V_1. It follows, that the independence number of the neighborhood of any vertex in A_1 or C is exactly k.

The algorithm initially greedily chooses all of A_0 and A_1, followed by B. We now argue that the algorithm then repeatedly performs 2-improvements that replace a single vertex of A_1 of weight 1 by k vertices of C, each of weight $2/k$, since at every stage the best improvement possible is a 2-improvement. We can argue this by induction.

Suppose $I = A_0 \cup B \cup I'$ is the current independent set, where $I' \subset A_1 \cup C$. Consider an $x \in I - A_1$ and a $V' \subset N_V(x)$. Any vertex in V' has $k/2+1$ neighbors in B, of which at most one is in common with any other vertex in V', since no four-cycles exist. Thus,

$$w(N_I(V')) \geq |V'|\frac{k}{2} \cdot \frac{1}{k} + \frac{1}{k} > \frac{1}{2}w(V').$$

Thus, the only 2-good improvements are those replacing a node in A_1 with a set in C.

The resulting solution, $A_0 \cup B \cup C$, is locally optimal, and of weight

$$ALG = |A_0| + |B|\frac{1}{k} + |C|\frac{2}{k} = \left(\frac{3}{2k} + \frac{1}{4(k-1)}\right)X,$$

while the optimal solution is V, of weight X. This yields an approximation ratio of $(4/7)k + O(1) \approx 0.57k$.

General construction We indicate how to generalize the simpler construction to get a stronger lower bound. We would like to reduce the weight of the vertices in C, so that the ratio between the weight of V and the algorithm's solution becomes bigger. However, if we just reduce the weight of all the vertices in C, then the payoff from replacing the A_1 vertices by the C vertices will become less than the payoff of bringing in the V vertices. So we instead do this in stages, bringing in successively smaller weight vertices from C into the independent set, with smaller payoffs. What will prevent vertices from V being chosen is that at the later stages, the the heavier vertices from the earlier stages of C will also be in the neighborhood of the V vertices.

Consider a sequence of breakpoints, $\alpha_t > \alpha_{t-1} > \cdots > \alpha_1 > \alpha_0 = 1$, to be determined later. The graph contains as before: V, $A = A_0 \cup A_1 \cup \ldots \cup A_t$ minimally dominating V, and B, an additional set of greedy vertices, forming a $(k, k/\alpha_t)$-regular graph with V. The weights of the vertices are unchanged. C is now partitioned into $C_1 \cup C_2 \cup \ldots C_t$. Each C_i forms a $(k, \lceil g(i)k \rceil + 1)$-regular graph with V, (which fixes

$|C_i| = \lceil (g(i)k\rceil + 1)OPT/k)$ where

$$g(i) = \frac{1}{\alpha_i}(\frac{1}{\alpha_{i-1}} - \frac{1}{\alpha_i}).$$

The weight of a vertex in C_i is α_i/k. Also, each A_i and C_i pair forms a $(k,1)$-regular graph, which fixes $|A_i| = \frac{1}{k}|C_i|$.

We also require that for any vertex x in A_i, its k neighbors in C_i are perfectly matched with its k neighbors in V. This ensures that the graph contains no $(k+1)$-claw. Also, we continue to require that the graph contain no four-cycles. This completes the specification of the graph.

The algorithm will greedily choose A followed by B. We claim that it will then make α_t-improvements, replacing A_t with C_t, followed by α_{t-1}-improvements, replacing A_{t-1} with C_{t-1}, and so on.

This can be proved inductively, and holds initially for t by the property of B we have already seen. Suppose it has completed replacing $A_t, \ldots, A_q$ by $C_t, \ldots, C_q$, respectively, and possibly part of A_{q-1} by C_{q-1}. Consider any vertex x outside V and its neighborhood $N_V(x)$ in V. Since the graph contains no 4-cycles, any pair of vertices in $N_V(x)$ share only x as a neighbor. Thus, we can partition the remainder of the neighborhood $N(N_V(x))$ of $N_V(x)$ in the current solution onto the vertices in $N_V(x)$. Each vertex in $N_V(x)$ thus gets assigned at least $g(i)k$ vertices in C_i, $i = t, \ldots, q$ and k/α_t vertices in B. The sum of the weight of those vertices is at least

$$\frac{1}{\alpha_t} + \sum_{i=q}^{t} g(i)k \cdot \frac{\alpha_i}{k} = \sum_{i=q}^{t}(\frac{1}{\alpha_{i-1}} - \frac{1}{\alpha_i}) = \frac{1}{\alpha_{q-1}}.$$

Thus, any improvement involving vertices in $N_V(x)$ will be less profitable than α_{q-1}. Since replacing vertices in A_{q-1} by vertices in C_{q-1} is an α_{q-1}-improvement, it follows that all of A_{q-1} will be replaced by C_{q-1}.

The resulting solution, $ALG = A_0 \cup B \cup C$, is locally optimal. The optimal solution is given by V, whose size and weight we denote by OPT. Observe that $|C_i| \leq (g_i + 2/k)OPT$, $|C_i| = k|A_i|$, $w(C_i) = \alpha_i w(A_i) = \alpha_i |A_i|$, and $|A_0| = \frac{1}{k}OPT - \sum_{i=1}^{t}|A_i|$. The weight of ALG is then

$$\begin{aligned}
ALG &\\
&= |A_0| + |B|\frac{1}{k} + \sum_{i=1}^{t}|A_i|\alpha_i \\
&= \frac{1}{k}OPT + |B|\frac{1}{k} + \sum_{i=1}^{t}(\alpha_i - 1)|A_i| \\
&\leq \left[1 + \frac{1}{\alpha_t} + \sum_{i=1}^{t}(\alpha_i - 1)(g_i + 2/k)\right]\frac{OPT}{k} \\
&= \left[1 + \frac{1}{\alpha_t} + \sum_{i=1}^{t}(\frac{1}{\alpha_{i-1}} - \frac{1}{\alpha_i}) - \sum_{i=1}^{t}\frac{1}{\alpha_i}(\frac{1}{\alpha_{i-1}} - \frac{1}{\alpha_i}) \right. \\
&\qquad \left. + \sum_{i=1}^{t}(\alpha_i - 1)2/k\right]\frac{OPT}{k} \\
&= \left[2 - \sum_{i=1}^{t}\frac{1}{\alpha_i}(\frac{1}{\alpha_{i-1}} - \frac{1}{\alpha_i}) + \sum_{i=1}^{t}(\alpha_i - 1)2/k\right]\frac{OPT}{k}.
\end{aligned}$$

Let W be $\lfloor k^{1/4} \rfloor$ and $t = W^2$. We set $\alpha_i = (W+i)/W$, and note that $\alpha_i - \alpha_{i-1}$ is $1/W$. Observe that the second sum is at most $2t^2/(kW) \leq 2/W$, while the first sum simplifies to

$$W^2 \sum_{i=W+1}^{t+W} \frac{1}{i^2(i-1)} \geq W^2 \int_{i=W+1}^{t+W} \frac{1}{x^3}dx \geq \frac{1}{2} - \frac{2}{W}.$$

Hence, we have that

$$ALG \leq (3/2 + 4/W)\frac{OPT}{k},$$

and the performance ratio is at least $(2/3)k - O(k^{3/4})$.

The AnyImp **algorithm** We also have a construction (omitted here), parametrized by α, which shows that our upper bound for AnyImp$_\alpha$ is asymptotically tight:

Theorem 5.2. *The performance ratio of* AnyImp$_\alpha$ *algorithm is at least*

$$\rho \geq \frac{k-1}{1 + \frac{1}{\alpha} - \frac{1}{\alpha^2} + \frac{\alpha-2}{k}} \geq \frac{4(k-1)}{5} - O(1/k^2).$$

6 Conclusions

Our initial success with AnyImp led us to conjecture that the performance of BestImp matched the bound it achieves on unweighted graphs, $(k+1)/2$. Instead, it turned out to be halfway between the greedy bound and the unweighted bound. A logical open issue is whether this is a natural separation between the approximability of the weighted and the unweighted cases.

Considering neighborhoods of sets of vertices in the local improvement phase is likely to reduce the additive term in the performance ratio. A $(2k-1)/3 + \epsilon$ would be a natural guess, improving the $k - 1 + \epsilon$ bound of pure local search for all $k \geq 3$.

The time complexity of the algorithms, like the previous local search algorithms, is $O(nk^2\Delta^k)$, where Δ is the maximum degree of the graph. It would be interesting to determine if similar performance ratio can be obtained by an algorithm whose time complexity depends less on k, e.g. $2^{O(k)}n^{O(1)}$.

Finally, it would be interesting to see the proof technique applied to other problems.

References

[1] E. Arkin, R. Hassin. Approximating weighted set packing by local search. *ESA '97.*

[2] V. Bafna, B. Narayan, R. Ravi. Nonoverlapping local alignments (Weighted independent sets of axis-parallel rectangles). To appear in *Disc. Appl. Math.*

[3] M. R. Garey and D. S. Johnson, *Computers and Intractability*, Freeman, New York, 1979.

[4] M. M. Halldórsson. Approximating discrete collections via local improvements. *SODA '95*, 160–169.

[5] C. A. J. Hurkens and A. Schrijver. On the size of systems of sets every t of which have an SDR, with an application to the worst-case ratio of heuristics for packing problems. *SIAM J. Disc. Math.*, 2(1):68–72, Feb. 1989.