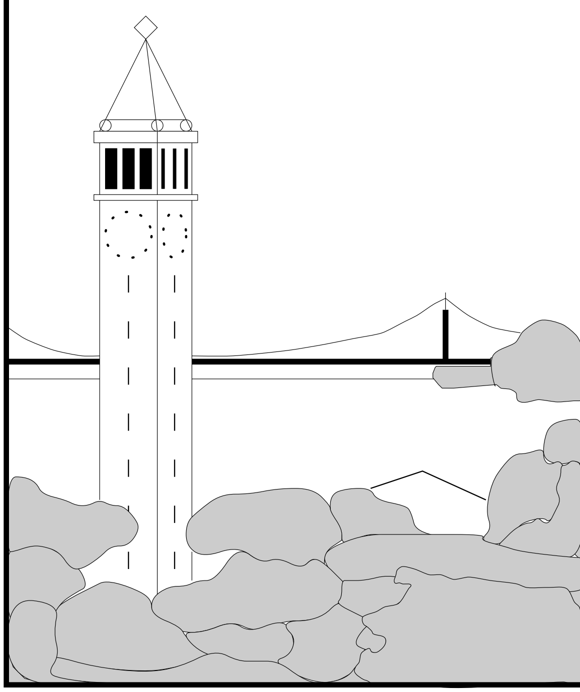
Approaches to Bin Packing with Clique-Graph Conflicts

Bill McCloskey
`billm@cs.berkeley.edu`

AJ Shankar
`aj@cs.berkeley.edu`



Report No. UCB/CSD-5-1378

April 2005

Computer Science Division (EECS)
University of California
Berkeley, California 94720

Approaches to Bin Packing with Clique-Graph Conflicts

Bill McCloskey
billm@cs.berkeley.edu

AJ Shankar
aj@cs.berkeley.edu

April 2005

Department of Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, CA 94720

Abstract

The problem of bin packing with arbitrary conflicts was introduced in [3]. In this paper, we consider a restricted problem, bin packing with clique-graph conflicts. We prove bounds for several approximation algorithms, and show that certain on- and off-line algorithms are equivalent. Finally, we present an optimal polynomial-time algorithm for the case of constant item sizes, and analyze its performance in the more general case of bounded item sizes.

1 Introduction

The standard bin packing problem (without conflicts) has been studied for some time. Coffman, Garey, and Johnson [1] present a survey of current results. A number of simple approximation algorithms exist for standard bin packing. A generalization of this problem, called bin packing with arbitrary conflicts (BPAC) in this paper, was treated by Jansen and Öhring [3]. In BPAC, certain items cannot be packed into the same bin. Conflicts are represented by edges in a graph over the items. In this paper, we analyze a special case of BPAC involving conflict graphs consisting only of cliques (called clique-graphs here); we call this problem bin packing with clique-graph conflicts (BPCC), or, informally, the *mix-tape problem.*

Formally, the inputs to the problem are a set of items V with weight function w, a conflict clique-graph $G = (V, E)$, and a number m. A valid solution is a partition of items into bins $B_1, \ldots, B_m$, with bin B_i containing items $B_{i,1}, \ldots, B_{i,b_i}$, satisfying:

$$\begin{aligned} \forall i. \qquad & \sum_{j=1}^{b_i} w(B_{i,j}) \leq 1 \\ \forall i, j, \ell. \qquad & \{B_{i,j}, B_{i,\ell}\} \notin E \end{aligned}$$

1.1 Motivations

The BPCC problem has many interesting applications. For example, distributed systems such as SETI@Home must dispense work items to clients. Clients can be considered bins that contain work items. Each work item requires some amount of processing time. A client will contribute only a fixed number of processor cycles per day. Assume that work items that are correlated (such as signals from the same region of the sky) can be verified against each other. To avoid tampering, the server would like to distribute the signals so that no client processes more than one signal from the same region.

Or, consider the problem of seating guests at a large corporate gathering. In order for a guest to make as many new acquaintances as possible, members of the same office should not be seated together. Of course, each table can only seat a certain number of people. In this case, the problem includes not only clique-graph conflicts, but also constant item sizes. We give an exact polynomial-time solution to this problem later in this paper.

A third problem concerns the use of digital music, which is exploding in popularity. It is becoming common for users to store tens or hundreds of gigabytes of music on their computers. But many disc-based music players require songs to be apportioned into 650 megabyte CDs (or mix-tapes). Users would like to create CDs with a varied selection of songs, where no two songs from the same album appear on a CD. Combined with the 650 megabyte limitation, this problem fits nicely into the BPCC formulation.

Finally, BPCC is applicable to the problem of sharing music over a peer-to-peer network. For legal reasons, no one user can store more than a single fragment (say, 1 minute) of a song. Thus, songs must be distributed among the clients so that every song is stored somehwere, and no two fragments of the same song are stored on the same client. Here, clients are bins, song fragments are items, and songs are cliques. As in the corporate gathering problem, since fragments are of fixed size, we can solve this problem exactly in polynomial time.

1.2 NP-completeness

BPCC is in NP. A valid certificate is an assignment of items to bins. A polynomial-time verifier can check that the assignment uses at most m bins, that no bin overflows, and that no bin contains conflicting items. A reduction from standard bin packing suffices to show that BPCC is NP-hard. On input S, a set of items for bin packing, the reduction creates an empty conflict graph, $G = (S, \emptyset)$, since naturally BPCC without conflicts is equivalent to bin packing.

2 Naive Approximations

We present the analysis of several naive algorithms adapted from standard bin packing algorithms. We let k denote the size of the largest clique in the conflict graph.

2.1 Naive Next Fit

Next fit is the most elementary bin packing algorithm. In this algorithm, one bin is designated the "active bin." New items are always placed in the active bin. If the current item does not fit in the active bin, a new bin is created and designated the active bin. According to [1], the number of bins used by the naive next fit algorithm, ω^{NF}, satisfies

$$\omega^{NF} \leq 2 \cdot \omega^* - 1$$

where ω^* is the optimal number of bins.

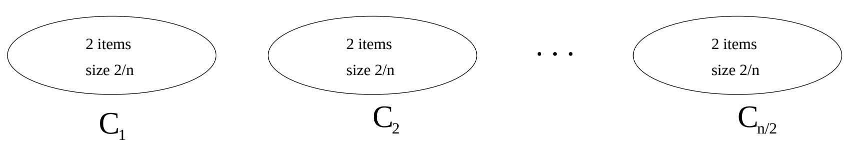
This algorithm generalizes to BPCC. We call the generalization *naive next fit* (NNF). The condition that tests if an item fits into a bin must additionally check that the item does not conflict with any item in the bin. Unfortunately, Jansen shows that the algorithm has arbitrarily bad behavior (i.e. $O(n)$) for the BPAC problem; we prove that the same fact holds true even for BPCC.

Theorem 1 *Let ω^{NNF} be the number of bins required by the naive next fit algorithm on an instance I of BPCC. Let n be the number of items in I. Let ω^* be the optimal number of bins required by I. Then*

$$\omega^{NNF} = O(n) \cdot \omega^*$$

and the bound is tight.

Proof. Our notation is as follows. A circle denotes a clique. The j^{th} element of clique C_i is written $c_{i,j}$. A packing of items into bins, such that the i^{th} bin contains elements B_i, is written $[B_1, \ldots, B_k]$. For a given n, consider the input instance below.



Let the items be processed in order of the bins they are in: $c_{1,1}, c_{1,2}, c_{2,1}, c_{2,2}, \ldots, c_{n/2,1}, c_{n/2,2}$. The optimal packing uses two bins, each with one item from each clique:

$$[\{c_{1,1}, \ldots, c_{n/2,1}\}, \{c_{1,2}, \ldots, c_{n/2,2}\}].$$

However, next fit creates a new bin every time it hits a conflict, resulting in the following assignment

$$[\{c_{1,1}\}, \{c_{1,2}, c_{2,1}\}, \ldots, \{c_{n/2-1,2}, c_{n/2,1}\}, \{c_{n/2,2}\}].$$

This assignment uses $\frac{n}{2}+1$ bins. Thus $\omega^{NNF} = \frac{n}{2}+1$, while $\omega^* = 2$, which yields the ratio $\frac{n}{4}+\frac{1}{2} = O(n)$. $\square$

2.2 Naive First Fit

First fit is another elementary bin packing algorithm. For each item to be processed, the algorithm iterates through the list of existing bins. It places the item in the first bin in which it will fit. If the item does not fit in any existing bins, then the item is placed in a new bin. As before, we generalize the algorithm to skip bins with conflicts, and we call it *naive first fit* (NFF). Like NNF, NFF has $O(n)$ behavior for BPAC; however, we show below that it performs much better for BPCC.

Theorem 2 *Let ω^{NFF} be the number of bins required by this naive first fit algorithm on an instance I of BPCC where the largest clique has k items. Let ω^* be the optimal number of bins required by I. Then*

$$\omega^{NFF} \leq 2 \cdot \omega^* + k$$

Proof. Let L be the sum of the sizes of the items. Assume that all but k bins produced by the NFF algorithm are at least half full (proved below). Since they each contribute at least $1/2$ to the sum L, $L \geq \frac{1}{2}(\omega^{NFF} - k)$. In addition, $\omega^* \geq L$, as there must be enough bins to hold all of the items. Therefore,

$$\begin{aligned} \omega^{NFF} &\leq 2 \cdot L + k \\ &\leq 2 \cdot \omega^* + k \end{aligned}$$

Now it is necessary to prove that at most k of the bins are less than half full. Assume, for purposes of finding a contradiction, that there are at least $k+1$ bins, $B_1, \ldots, B_{k+1}$, less than half full, ordered as they were created by the NFF algorithm. Let b_i be the number of elements in bin B_i. Let B_f be a bin with the fewest number of elements. Consider an element in the bins $B_{f+1}, \ldots, B_{k+1}$. It could fit into B_f, since all of the bins are less than half full; therefore it must conflict with some element in B_f, or else NFF would have placed it there. So, the total number of conflict edges between elements in $B_{f+1}, \ldots, B_{k+1}$ and elements in B_f is $\sum_{i=f+1}^{k+1} b_i$. Let x be an element of B_f that has at least the average number of conflicts with elements in $B_{f+1}, \ldots, B_{k+1}$,

$$\frac{\sum_{i=f+1}^{k+1} b_i}{b_f}.$$

As x was placed in B_f, NFF could not have placed it in any of the previous bins. Since all B_i are at most half full, x must have conflicted with an element in each of the bins $B_1, \ldots, B_{f-1}$. This accounts for $f-1$ additional items in x's clique. Therefore, including x itself, the size of x's clique must be at least

$$\begin{aligned} & 1 + (f-1) + \frac{\sum_{i=f+1}^{k+1} b_i}{b_f} \\ = \quad & f + \sum_{i=f+1}^{k+1} \frac{b_i}{b_f} \\ \geq \quad & f + \sum_{i=f+1}^{k+1} 1 && \text{since } b_f \text{ is minimal} \\ = \quad & f + (k+1) - (f+1) + 1 \\ = \quad & k+1 \end{aligned}$$

However, the largest clique has k nodes. This is a contradiction, meaning that the number of bins that are less than half full is at most k. $\square$

Oh and Son [4] prove an alternative bound for NFF:

Theorem 3 $\omega^{NFF} \leq 1.7 \cdot \omega^* + 2.19 \cdot k$

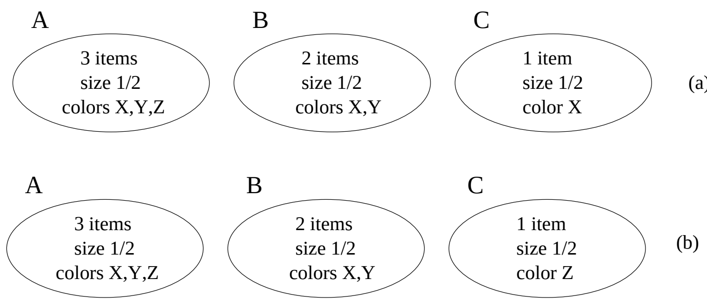
Depending on the value of k, either Theorem 2 or Theorem 3 gives a tighter bound.

3 Graph Coloring-Based Approximations

In bin packing with conflicts, two nodes connected by an edge in the conflict graph cannot be placed in the same bin. Analogously, in the graph coloring problem, two nodes connected by an edge cannot share the same color. In fact, a correspondence exists between colors and bins. Any k-coloring of the conflict graph reduces the problem to k standard bin packing instances, since items of the same color can be packed without constraint [3].

In fact, the conflict graph coloring approach is comprehensive: any arbitrary packing is achievable by first coloring the conflict graph in a particular way and then independently solving conflict-free bin packing for each color. If the arbitrary packing uses bins $B_1, \ldots, B_m$, then the items of bin B_i are given color i. The conflict-free algorithm may then pack the items of color B_i using one bin. This implies that, for any optimal bin packing, a coloring exists such that when each color set is independently solved optimally, the result is no worse.

It is interesting to note that, among *minimal* colorings of k colors, some colorings result in better packings than others even when item sizes are the same. Consider a graph with three cliques, A, B, C, where A has three items, B has two items, and C has one item. In this case, a minimal coloring uses $k = 3$ colors. Two possible minimal colorings are shown here.



In coloring (a), four bins are required to pack all of the items: $[X : \{a_1, b_1\}, X : \{c_1\}, Y : \{a_2, b_2\}, Z : \{a_3\}]$ (the notation has been extended to allow each bin to have a color). In coloring (b), however, only three bins are required: $[X : \{a_1, b_1\}, Y : \{a_2, b_2\}, Z : \{a_3, c_1\}]$. Hence, finding a minimal coloring is not sufficient to solve the BPCC problem.

Furthermore, there may be no minimal coloring such that each item in the same bin of an optimal packing has the same color. For example, consider the conflict graph containing three copies of K_2, called A, B, and C, each with items of size $1/2$. There is a packing $[\{a_1, b_1\}, \{a_2, c_1\}, \{b_2, c_2\}]$ using three bins. However, any 2-coloring of the conflict graph splits the items into two groups of three items each. Each group requires two bins, so any packing needs a total of four bins. Therefore, any 2-coloring of the conflict graph results in a suboptimal packing.

Further investigation of the effect of different coloring strategies on BPCC is future work.

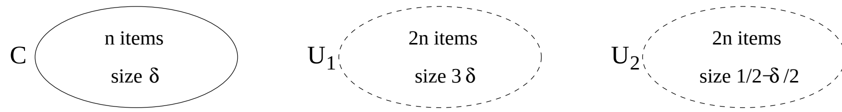
3.1 Standard Algorithms

In this section we analyze modifications of several standard bin packing algorithms that require the conflict graph to be precolored. These techniques were introduced by Jansen and Öhring [3]. Of course, all instances of BPCC can be colored optimally in linear time, since each k-clique requires k colors. Once the graph has been colored, standard techniques are employed to pack each color separately. Jansen's results carry over to our work with a few modifications.

Next Fit Once the graph is colored, we run ordinary next fit, introduced in Section 2.1, on each color. The following theorem is proved by Jansen and Öhring [3] for the BPAC problem.

Theorem 4 (Jansen and Öhring) *Let ω^{GNF} be the number of bins used by the next fit algorithm with graph coloring on BPAC. It satisfies $\omega^{GNF} \leq 3 \cdot \omega^*$.*

We show that this bound is asymptotically tight for BPCC. Consider the following conflict graph. The dashed circles represent collections of isolated nodes, each node of which is a trivial clique of size 1.



Let n be large. Let $\delta = \frac{1}{6n}$. The optimal packing places one element from C with two elements from U_2, filling n bins. Then it uses a final bin for all of the elements of U_1:

$$[\{c_1, u_{2,1}, u_{2,2}\}, \{c_2, u_{2,3}, u_{2,4}\}, \ldots, \{c_n, u_{2,2n-1}, u_{2,2n}\}, \{u_{1,1}, u_{1,2}, \ldots, u_{1,2n}\}].$$

A minimal coloring for this graph, using n colors, assigns each element of C its own color and all the elements of U_1 and U_2 the same color as, say, c_n. Given this coloring, and the ordering for the large color set $u_{2,1}, u_{1,1}, u_{2,2}, u_{1,2}, \ldots, u_{2,2n}, u_{1,2n}, c_n$, next fit uses $3n-1$ bins:

$$[\{c_1\}, \{c_2\}, \ldots, \{c_{n-1}\}, \{u_{2,1}, u_{1,1}\}, \{u_{2,2}, u_{1,2}\}, \ldots, \{u_{2,2n}, u_{1,2n}, c_n\}].$$

So in this instance next fit uses $\omega^{GNF} = 3n-1$ bins, and the optimal packing uses $\omega^* = n+1$ bins. In the limit,

$$\lim_{n\to\infty} \frac{\omega^{GNF}}{\omega^*} = \lim_{n\to\infty} \frac{3n-1}{n+1} = 3.$$

First Fit First fit was introduced in Section 2.2. The same algorithm can be applied to the set of items of a particular color.

Theorem 5 (Jansen and Öhring) *Let ω^{GFF} be the number of bins used by the first fit algorithm with graph coloring on BPAC. It satisfies $\omega^{GFF} \le 2.7 \cdot \omega^*$.*

Jansen [3] gives a set of instances that asymptotically achieve the 2.7 bound and use a clique-graph. Therefore, this algorithm is unable to take advantage of the specialized BPCC graph, and the BPAC bound holds.

First Fit Decreasing In this algorithm, the items of a given color are sorted in non-increasing size. Then they are placed using first fit. Clearly first fit decreasing for conflict graphs (GFFD) can do no worse than GFF. However, Jansen provides a set of instances that asymptotically achieve a ratio of $\sim$ 2.691 using a clique-graph. It is possible, though not proven, that the 2.7 bound is tight for first fit descending as well.

3.2 Smarter Coloring

As indicated at the beginning of this section, there are good and bad minimal colorings with respect to optimal packings. One observation is that it is useless to assign the same color to two items of size greater than 1/2, since they cannot possibly be placed in the same bin. A strategy that takes advantage of this, GFF-$\frac{1}{2}$, simply assigns distinct colors to the large items first, then colors the rest of the items as usual.

Jansen and Öhring prove that this strategy succeeds for BPAC.

Theorem 6

$$\omega^{GFF-\frac{1}{2}} \le 2.5 \cdot \omega^*$$

As before, the instances used by Jansen and Öhring to prove that the 2.5 bound is tight also have clique-graphs, so this bound is tight for BPCC as well.

3.3 Equivalence of Online and Offline Algorithms

In the general case, NFF behaves asymptotically badly [3]. However, when the conflict graph is restricted to cliques, NFF performs as well as GFF. In fact, the two algorithms are equivalent, even though GFF is an offline algorithm.

Theorem 7 *Let $\omega^{NFF}(\pi)$ be the number of bins used by the naive first fit algorithm for a particular ordering of the input items, π. Similarly, let $\omega^{GFF}(\pi, f)$ be the number of bins used by first fit with graph coloring, with ordering π and coloring f. Then*

$$\forall \pi. \ \exists f. \ \omega^{GFF}(\pi, f) = \omega^{NFF}(\pi)$$

Proof. We show that NFF's online placement of items into bins is equivalent to first minimally coloring them and then running FF on each color in the same order that they were given to NFF.

We infer a coloring from NFF's item placement as follows. In placing a given item x, NFF does one of three things:

1. It puts x into an existing bin B
2. Although there exists a bin B with which x does not conflict, it must create a new bin, because x does not fit in B
3. x conflicts with all existing bins, so NFF creates a new bin for it

For cases 1 and 2, color x the same color as the items in B. For case 3, color x with a new color. Notice that a new color is introduced only if an item could not be given any existing color. Such a graph coloring algorithm is minimal for clique graphs, since the new item must belong to a clique of size greater than the current number of colors. It is also clear that in cases 1 and 2, NFF packs the items in the same order as FF does on each color. Therefore NFF and GFF are equivalent. □

A similar result follows for GFFD.

Theorem 8 $\forall \pi. \ \exists f. \ \omega^{GFFD}(\pi, f) = \omega^{NFFD}(\pi)$

Proof. Since NFFD processes the elements in sorted order, all the items of a given induced color are placed in the bins in sorted order, just as GFFD would do. The rest of the proof is the same as above. □

4 Restricted Problems

4.1 Constant Item Sizes

BPCC with constant item size δ can be solved optimally in polynomial time as follows. Each bin holds $\lfloor \frac{1}{\delta} \rfloor$ items. To fill a bin, the algorithm picks off an independent set of size at most $\lfloor \frac{1}{\delta} \rfloor$ from the conflict graph by selecting one item from each of the $\lfloor \frac{1}{\delta} \rfloor$ largest cliques. If there are fewer than $\lfloor \frac{1}{\delta} \rfloor$ cliques, the algorithm takes from only as many as are available. One independent set is removed in each stage of the algorithm.

Claim 1 *If two cliques ever differ in size by at most 1 at any stage of the algorithm, then their size differs by at most 1 at any later stage.*

Proof. At any later stage, if the two cliques do differ in size by 1, then the algorithm will not select an item from the smaller clique without selecting an item from the larger clique, because it preferentially chooses larger cliques. □

Claim 2 *For any three cliques A, B, C, if at any stage of the algorithm A has the same size as B, and at any (potentially different) stage B has the same size as C, then after the later of these two stages A and C will never differ in size by more than 1.*

Proof. Consider the later of these two stages, say without loss of generality when B is the same size as C. At this point, A differs from B by at most 1 due to Claim 1. Therefore A differs from C by at most 1 at that point as well, and the previous argument holds for A and C. □

Theorem 9 *The constant item size algorithm produces an optimal packing in polynomial time.*

Proof. Choose a clique C of maximal size. It has k items. If the algorithm selects an item from C at every stage, then there are k stages, so k bins are produced. This is clearly optimal.

If, however, there is some stage at which the algorithm does not select an item from C, then there must be $\lfloor \frac{1}{\delta} \rfloor$ other cliques of the same or greater size at that stage; call them T. Since C was initially largest, each of the cliques in T must have been the same size as C at some stage. If the number of cliques ever drops below $\lfloor \frac{1}{\delta} \rfloor$, it must happen after this stage, since T has at least $\lfloor \frac{1}{\delta} \rfloor$ of them.

If the number of cliques never drops below $\lfloor \frac{1}{\delta} \rfloor$, then the algorithm fills all bins completely, so it must be optimal. Otherwise, at that point, one of the cliques $D \in T \cup \{C\}$ must have been eliminated. If $D = C$, then by Claim 1 all remaining cliques must be of size 1. Since there are fewer than $\lfloor \frac{1}{\delta} \rfloor$ of them, the algorithm will optimally place them all into the same bin. Otherwise, $D \in T$ and so we know that D must have had been the same size as C at some point, meaning that C now has size at most 1; furthermore, since C was the largest clique and now has size 1, it must have been the same size as every other remaining clique at some (other) point. By Claim 2, then, since D is of size 0, every other clique must be at most size 1, and the algorithm will place them all into the same final bin. Either way, the algorithm will produce a packing with only one bin of fewer than $\lfloor \frac{1}{\delta} \rfloor$ items, the final one. This packing is clearly optimal.

The algorithm needs to maintain a sorted list of cliques, which can be done with a heap. A constant number of items are removed at each stage. Removing a clique and reinserting it into the heap with a smaller size takes $O(\log n)$ time, so the total running time is $O(n \log n)$. □

4.2 Bounded Items Sizes

We considered the problem where item sizes vary within a given range $[\delta - \gamma, \delta]$. The constant size algorithm can be used to approximate a solution.

Theorem 10 *The constant size algorithm, run on items of size* $\in [\delta - \gamma, \delta]$, *uses* $\omega^{CONST} \leq \frac{\omega^*}{1-\delta-\gamma/\delta}$ *bins, assuming* $\gamma \leq \delta(1 - \delta)$.

Proof. If $\omega^* = k$, then, as above, the algorithm produces an optimal packing. Otherwise, when the item sizes are constant, each bin except the last contains $\lfloor \frac{1}{\delta} \rfloor$ items and wastes at most δ space. Reducing the item sizes by up to γ additionally wastes at most γ space per item. Thus the total waste per bin is at most $\delta + \gamma \lfloor \frac{1}{\delta} \rfloor$.

Let L be total size of the items. Each bin must be filled to height at least $1 - (\delta + \gamma \lfloor \frac{1}{\delta} \rfloor) \geq 1 - \delta - \frac{\gamma}{\delta}$. Therefore, by summing the total sizes in the bins,

$$\begin{aligned} \omega^{CONST} \cdot (1 - \delta - \frac{\gamma}{\delta}) &\leq L \\ \omega^{CONST} \cdot (1 - \delta - \frac{\gamma}{\delta}) &\leq \omega^* && \text{since } L \leq \omega^* \\ \omega^{CONST} &\leq \frac{\omega^*}{1 - \delta - \gamma/\delta} \end{aligned}$$

In this proof, we require that $\gamma \leq \delta(1 - \delta)$ so that the last step does not reverse the inequality. □

As an example, suppose we would like to get a 2-approximation for some δ.

$$\begin{aligned} \frac{1}{1 - \delta - \gamma/\delta} &\leq 2 \\ \gamma &\leq \delta(1/2 - \delta) \\ \delta - \gamma &\geq \delta(1/2 + \delta) \end{aligned}$$

In the digital music problem, a likely value of δ is 1/10 (meaning that no song is longer than 1/10 of an album). The minimum length that any song can be is $\delta - \gamma = \frac{6}{10}\delta = \frac{3}{50}$. So, for a set of fifty minute albums, if all songs are between three and five minutes, a 2-approximation of the optimal packing can be achieved in polynomial time.

5 Conclusion

We have presented a number of approaches to approximating BPCC. The next fit algorithm is subsumed by first fit. We have proved that the various on- and off-line incarnations of the first fit algorithm are equivalent, leaving us with four primary options; see Figure 1.

Some of our contributions have built upon those of others. We have proved that the naive next fit algorithm has asymptotically bad running time for BPCC. We have discovered a bound of $2 \cdot \omega^* + k$ for

Algorithm	Requirements	Bound
First Fit (NFF)	*none*	$\min(2\omega^* + k, 1.7\omega^* + 2.19k, 2.7\omega^*)$
Smart Coloring (GFF-$\frac{1}{2}$)	offline	$2.5\omega^*$
Constant Size (CONST)	offline; items of equal size	ω^*
Bounded Size (CONST)	offline; item size between $[\delta-\gamma, \delta]$; $\gamma \leq \delta(1-\delta)$	$\frac{\omega^*}{1-\delta-\gamma/\delta}$

Figure 1: A comparison of BPCC approximation algorithms.

the naive first fit algorithm, and we have proved that naive first fit is equivalent to first fit with graph coloring. We also have shown that the $3 \cdot \omega^*$ bound for the next fit algorithm with graph coloring is tight even for BPCC.

Finally, we have given an optimal algorithm for the case of constant item sizes, and have shown that this algorithm performs well when the item sizes have bounded variability.

In the future, we would like to investigate a generalization of this problem in which a constant number of conflicts per bin are tolerated. Additionally, Jansen [2] gives an APTAS for the specific case where the conflict graph is *d-inductive* for constant d. BPCC conflict graphs are $(k-1)$-inductive, where k is the size of the largest clique. We would like to find an APTAS that does not depend on k.

References

[1] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. *Approximation Algorithms for Bin Packing: A Survey*, pages 46–93. PWS Publishing, Boston, 1997.

[2] Klaus Jansen. An approximation scheme for bin packing with conflicts. In *Scandinavian Workshop on Algorithm Theory*, pages 35–46, 1998.

[3] Klaus Jansen and Sabine R. Öhring. Approximation algorithms for time constrained scheduling. In *Workshop on Parallel Algorithms for Irregularly Structured Problems*, pages 143–157, 1995.

[4] Yingfeng Oh and Sang H. Son. On a constrained bin-packing problem. Technical Report CS-95-14, University of Virginia, 3, 1995.