Approximating Bin Packing with Conflict Graphs via Maximization Techniques

Ilan Doron-Arad and Hadas Shachnai(✉)

Computer Science Department, Technion, Haifa 3200003, Israel
{idoron-arad,hadas}@cs.technion.ac.il

Abstract. We give a comprehensive study of *bin packing with conflicts* (BPC). The input is a set I of items, sizes $s : I \to [0, 1]$, and a conflict graph $G = (I, E)$. The goal is to find a partition of I into a minimum number of independent sets, each of total size at most 1. Being a generalization of the notoriously hard graph coloring problem, BPC has been studied mostly on polynomially colorable conflict graphs. An intriguing open question is whether BPC on such graphs admits the same best known approximation guarantees as classic bin packing.

We answer this question negatively, by showing that (in contrast to bin packing) there is no asymptotic polynomial-time approximation scheme (APTAS) for BPC already on seemingly easy graph classes, such as *bipartite* and *split* graphs. We complement this result with improved approximation guarantees for BPC on several prominent graph classes. Most notably, we derive an asymptotic 1.391-approximation for bipartite graphs, a 2.445-approximation for perfect graphs, and a $\left(1 + \frac{2}{e}\right)$-approximation for split graphs. To this end, we introduce a generic framework relying on a novel interpretation of BPC allowing us to solve the problem via *maximization* techniques. Our framework may find use in tackling BPC on other graph classes arising in applications.

1 Introduction

We study the *bin packing with conflicts (BPC)* problem. We are given a set I of n items, sizes $s : I \to [0, 1]$, and a conflict graph $G = (I, E)$ on the items. A *packing* is a partition $(A_1, \ldots, A_t)$ of I into independent sets called *bins*, such that for all $b \in \{1, \ldots, t\}$ it holds that $s(A_b) = \sum_{\ell \in A_b} s(\ell) \leq 1$. The goal is to find a packing in a minimum number of bins. Let $\mathcal{I} = (I, s, E)$ denote a BPC instance. We note that BPC is a generalization of *bin packing (BP)* (where $E = \emptyset$) as well as the graph coloring problem (where $s(\ell) = 0 \ \forall \ell \in I$).[1] BPC captures many real-world scenarios such as resource clustering in parallel computing [2], examination scheduling [21], database storage [16], and product delivery [3]. As the special case of graph coloring cannot be approximated within a ratio better than $n^{1-\varepsilon}$ [28], most of the research work on BPC has focused on families of

[1] See the formal definitions of *graph coloring* and *independent sets* in Sect. 2.

A full version of the paper is available in [6].

D. Paulusma and B. Ries (Eds.): WG 2023, LNCS 14093, pp. 261–275, 2023.
https://doi.org/10.1007/978-3-031-43380-1_19

conflict graphs which can be optimally colored in polynomial time [4,5,8,15–17,22,23].

Let $\mathrm{OPT} = \mathrm{OPT}(\mathcal{I})$ be the value of an optimal solution for an instance $\mathcal{I}$ of a minimization problem $\mathcal{P}$. As in the bin packing problem, we distinguish between *absolute* and *asymptotic* approximation. For $\alpha \geq 1$, we say that $\mathcal{A}$ is an absolute α-approximation algorithm for $\mathcal{P}$ if for any instance $\mathcal{I}$ of $\mathcal{P}$ we have $\mathcal{A}(\mathcal{I})/\mathrm{OPT}(\mathcal{I}) \leq \alpha$, where $\mathcal{A}(\mathcal{I})$ is the value of the solution returned by $\mathcal{A}$. Algorithm $\mathcal{A}$ is an *asymptotic* α-approximation algorithm for $\mathcal{P}$ if for any instance $\mathcal{I}$ it holds that $\mathcal{A}(\mathcal{I}) \leq \alpha\mathrm{OPT}(\mathcal{I}) + o(\mathrm{OPT}(\mathcal{I}))$. An APTAS is a family of algorithms $\{\mathcal{A}_\varepsilon\}$ such that, for every $\varepsilon > 0$, $\mathcal{A}_\varepsilon$ is a polynomial time asymptotic $(1+\varepsilon)$-approximation algorithm for $\mathcal{P}$. An *asymptotic fully polynomial-time approximation scheme (AFPTAS)* is an APTAS $\{\mathcal{A}_\varepsilon\}$ such that $\mathcal{A}_\varepsilon(\mathcal{I})$ runs in time $\mathrm{poly}(|\mathcal{I}|, \frac{1}{\varepsilon})$, where $|\mathcal{I}|$ is the encoding length of the instance $\mathcal{I}$.

It is well known that, unless P=NP, BP cannot be approximated within ratio better than $\frac{3}{2}$ [10]. This ratio is achieved by First-Fit Decreasing (FFD) [26].[2] Also, BP admits an AFPTAS [19], and an additive approximation algorithm which packs any instance $\mathcal{I}$ in at most $\mathrm{OPT}(\mathcal{I}) + O(\log(\mathrm{OPT}(\mathcal{I})))$ bins [14]. Despite the wide interest in BPC on polynomially colorable graphs, the intriguing question whether BPC on such graphs admits the same best known approximation guarantees as classic bin packing remained open.

Table 1. Known results for Bin Packing with Conflict Graphs

	Absolute		**Asymptotic**	
	Lower Bound	Upper Bound	Lower Bound	Upper Bound
General graphs	$n^{1-\varepsilon}$ [28]	$O\left(\frac{n(\log\log n)^2}{(\log n)^3}\right)$ [13]	$n^{1-\varepsilon}$ [28]	$O\left(\frac{n(\log\log n)^2}{(\log n)^3}\right)$ [13]
Perfect graphs	·	**2.445** (2.5 [8])	$\mathbf{c > 1}$	**2.445** (2.5 [8])
Chordal graphs	·	$\frac{7}{3}$ [8]	$\mathbf{c > 1}$	$\frac{7}{3}$ [8]
Cluster graphs	·	2 [1]		1 [5]
Cluster complement	·	**3/2**	**3/2**	**3/2**
Split graphs	·	$\mathbf{1 + 2/e}$ (2 [15])	$\mathbf{c > 1}$	$\mathbf{1 + 2/e}$ (2 [15])
Bipartite graphs	·	$\frac{5}{3}$ [15]	$\mathbf{c > 1}$	**1.391** ($\frac{5}{3}$ [15])
Partial k-trees	·	$2 + \varepsilon$ [17]		1 [16]
Trees	·	$\frac{5}{3}$ [15]		·
No conflicts	$\frac{3}{2}$ [10]	$\frac{3}{2}$ [26]		1 [25]

We answer this question negatively, by showing that (in contrast to bin packing) there is no APTAS for BPC even on seemingly easy graph classes, such as *bipartite* and *split* graphs. We complement this result with improved approximation guarantees for BPC on several prominent graph classes. For BPC on bipartite graphs, we obtain an asymptotic 1.391-approximation. We further derive improved bounds of 2.445 for perfect graphs, $\left(1 + \frac{2}{e}\right)$ for split graphs, and $\frac{5}{3}$ for

[2] We give a detailed description of Algorithm FFD in [6].

bipartite graphs.[3] Finally, we obtain a tight $\frac{3}{2}$-asymptotic lower bound and an absolute $\frac{3}{2}$-upper bound for graphs that are the complements of cluster graphs (we call these graphs below *complete multi-partite*).

Table 1 summarizes the known results for BPC on various classes of graphs. New bounds given in this paper are shown in boldface. Entries that are marked with $\cdot$ follow by inference, either by using containment of graph classes (trees are partial k-trees), or since the hardness of BPC on all considered graph classes follows from the hardness of classic BP. Empty entries for lower bounds follow from tight upper bounds.

1.1 Related Work

The BPC problem was introduced by Jansen and Öhring [17]. They presented a general algorithm that initially finds a coloring of the conflict graph, and then packs each color class separately using the First-Fit Decreasing algorithm. This approach yields a 2.7-approximation for BPC on perfect graph. The paper [17] includes also a 2.5-approximation for subclasses of perfect graphs on which the corresponding *precoloring extension problem* can be solved in polynomial time (e.g., interval and chordal graphs). The authors present also a $(2+\varepsilon)$-approximation algorithm for BPC on cographs and partial k-trees.

Epstein and Levin [8] present better algorithms for BPC on perfect graphs (2.5-approximation), graphs on which the precoloring extension problem can be solved in polynomial time ($\frac{7}{3}$-approximation), and bipartite graphs ($\frac{7}{4}$-approximation). Their techniques include matching between *large* items and a sophisticated use of new item *weights*. Recently, Huang et al. [15] provided fresh insights to previous algorithms, leading to $\frac{5}{3}$-approximation for BPC on bipartite graphs and a 2-approximation on split graphs.

Jansen [16] presented an AFPTAS for BPC on *d-inductive* conflict graphs, where $d \geq 1$ is some constant. This graph family includes trees, grid graphs, planar graphs, and graphs with constant treewidth. For a survey of *exact* algorithms for BPC see, e.g., [15].

1.2 Techniques

There are several known approaches for tackling BPC instances. One celebrated technique introduced by Jansen and Öhring [17] relies on finding initially a minimum coloring of the given conflict graph, and then packing each color class using a bin packing heuristic, such as First-Fit Decreasing. A notable generalization of this approach is the sophisticated integration of *precoloring extension* [8,17], which completes an initial partial coloring of the conflict graph, with no increase to the number of color classes. Another elegant technique is a matching-based algorithm, applied by Epstein and Levin [8] and by Huang et al. [15].

[3] Recently, Huang et al. [15] obtained a $\frac{5}{3}$-approximation for bipartite graphs, simultaneously and independently of our work. We note that the techniques of [15] are different than ours, and their algorithm is more efficient in terms of running time.

The best known algorithms (prior to this work), e.g., for perfect graphs [8] and split graphs [15] are based on the above techniques. While the analyses of these algorithms are tight, the approximation guarantees do not match the existing lower bounds for BPC on these graph classes; thus, obtaining improved approximations requires new techniques.

In this paper we present a novel point of view of BPC involving the solution of a maximization problem as a subroutine. We first find an *initial packing* of a subset $S \subseteq I$ of items, which serves as a baseline packing with *high potential* for adding items (from $I \setminus S$) without increasing the number of bins used. The remaining items are then assigned to extra bins using a simple heuristic. Thus, given a BPC instance, our framework consists of the following main steps.

1. Find an initial packing $\mathcal{A} = (A_1, \ldots, A_m)$ of high potential for $S \subseteq I$.
2. Maximize the total size of items in $\mathcal{A}$ by adding items in $I \setminus S$.
3. Assign the remaining (unpacked) items to extra bins using a greedy approach respecting the conflict graph constraints.

The above generic framework reduces BPC to cleverly finding an initial packing of high potential, and then efficiently approximating the corresponding maximization problem, while exploiting structural properties of the given conflict graph. One may view classic approaches for solving BP (e.g., [20]), as an application of this technique: find an initial packing of high potential containing the *large* items; then add the *small* items using First-Fit. In this setting, the tricky part is to find an initial high potential packing, while adding the small items is trivial. However, in the presence of a conflict graph, solving the induced maximization problem is much more challenging.

Interestingly, we are able to obtain initial packings of high potential for BPC on several conflict graph classes. To solve the maximization problem, we first derive efficient approximation for maximizing the total size of items within a *single* bin. Our algorithm is based on finding a maximum weight independent set of *bounded* total size in the graph, combined with enumeration over items of large sizes. Using the single bin algorithm, the maximization problem is solved via application of the *separable assignment problem (SAP)* [9] framework, adapted to our setting. Combined with a hybrid of several techniques (to efficiently handle different types of instances) this leads to improved bounds for BPC on perfect, split, and bipartite graphs (see Sects. 3, 4, and the full version of the paper [6]). Our framework may find use in tackling BPC on other graph classes arising in applications.

1.3 Organization

In Sect. 2 we give some definitions and preliminary results. Section 3 presents an approximation algorithm for BPC on perfect graphs and an asymptotic approximation on bipartite graphs. In Sect. 4 we give an algorithm for split graphs. We present our hardness results in Sect. 5 and conclude in Sect. 6. Due to space constraints, some of our results and proofs are given in the full version of the paper [6].

2 Preliminaries

For any $k \in \mathbb{R}$, let $[k] = \{1, 2, \ldots, \lfloor k \rfloor\}$. Also, for a function $f : A \to \mathbb{R}_{\geq 0}$ and a subset of elements $C \subseteq A$, we define $f(C) = \sum_{e \in C} f(e)$.

2.1 Coloring and Independent Sets

Given a graph $G = (V, E)$, an *independent set* in G is a subset of vertices $S \subseteq V$ such that for all $u, v \in S$ it holds that $(u, v) \notin E$. Let $\mathsf{IS}(G)$ be the collection of all independent sets in G. Given weight function $w : V \to \mathbb{R}_{\geq 0}$, a *maximum independent set w.r.t. w* is an independent set $S \in \mathsf{IS}(G)$ such that $w(S)$ is maximized. A *coloring* of G is a partition $(V_1, \ldots, V_t)$ of V such that $\forall i \in [t] : V_i \in \mathsf{IS}(G)$; we call each subset of vertices V_i *color class i*. Let $\chi(G)$ be the minimum number of colors required for a coloring of G. A graph G is *perfect* if for every induced subgraph G' of G the cardinality of the maximum clique of G' is equal to $\chi(G')$; note that G' is also a perfect graph. The following well known result is due to [12].

Lemma 2.1. *Given a perfect graph $G = (V, E)$, a minimum coloring of G and a maximum weight independent set of G can be computed in polynomial time.*

2.2 Bin Packing with Conflicts

Given a BPC instance $\mathcal{I}$, let $G_{\mathcal{I}} = (I, E)$ denote the conflict graph of $\mathcal{I}$. A *packing* of a subset of items $S \subseteq I$ is a partition $\mathcal{B} = (B_1, \ldots, B_t)$ of S such that, for all $i \in [t]$, B_i is an independent set in $G_{\mathcal{I}}$, and $s(B_i) \leq 1$. Let $\#\mathcal{B}$ be the number of bins (i.e., entries) in $\mathcal{B}$.

In this paper we consider BPC on several well studied classes of perfect graphs and the acronym BPC refers from now on to perfect conflict graphs. For *bin packing with bipartite conflicts (BPB)*, where the conflict graph is bipartite, we assume a bipartition of V is known and given by X_V and Y_V. Recall that $G = (V, E)$ is a split graph if there is a partition K, S of V into a clique and an independent set, respectively. We call this variant of BPC *bin packing with split graph conflicts (BPS)*.

The following notation will be useful while enhancing a partial packing by new items. For two packings $\mathcal{B} = (B_1, \ldots, B_t)$ and $\mathcal{C} = (C_1, \ldots, C_r)$, let $\mathcal{B} \oplus \mathcal{C} = (B_1, \ldots, B_t, C_1, \ldots, C_r)$ be the *concatenation* of $\mathcal{B}$ and $\mathcal{C}$; also, for $t = r$ let $\mathcal{B} + \mathcal{C} = (B_1 \cup C_1, \ldots, B_t \cup C_t)$ be the *union* of the two packings; note that the latter is not necessarily a packing. We denote by $\mathsf{items}(\mathcal{B}) = \bigcup_{i \in [t]} B_i$ the set of items in the packing $\mathcal{B}$. Finally, let $\mathcal{I} = (I, s, E)$ be a BPC instance and $T \subseteq I$ a subset of items. Define the BPC instances $\mathcal{I} \cap T = (T, s, E_T)$ and $\mathcal{I} \setminus T = (I \setminus T, s, E_{I \setminus T})$ where for all $X \in \{T, I \setminus T\}$ $E_X = \{(u, v) \in E \mid u, v \in X\}$.

2.3 Bin Packing Algorithms

We use $\mathcal{I} = (I, s)$ to denote a BP instance, where I is a set of n items for some $n \geq 1$, and $s : I \to [0, 1]$ is the size function. Let $L_{\mathcal{I}} = \{\ell \in I \mid s(\ell) > \frac{1}{2}\}$ be the

set of *large* items, $M_{\mathcal{I}} = \{\ell \in I \mid \frac{1}{3} < s(\ell) \leq \frac{1}{2}\}$ the set of *medium* items, and $S_{\mathcal{I}} = \{\ell \in I \mid s(\ell) \leq \frac{1}{3}\}$ the set of *small* items. Our algorithms use as building blocks also algorithms for BP. The results in the next two lemmas are tailored for our purposes. We give the detailed proofs in [6].[4]

Lemma 2.2. *Given a* BP *instance* $\mathcal{I} = (I, s)$*, there is a polynomial-time algorithm* First-Fit Decreasing (FFD) *which returns a packing* $\mathcal{B} = (B_1, \ldots, B_t)$ *of* $\mathcal{I}$ *where* $\#\mathcal{B} \leq (1 + 2 \cdot \max_{\ell \in I} s(\ell)) \cdot s(I) + 1$*. Moreover, it also holds that* $\#\mathcal{B} \leq |L_{\mathcal{I}}| + \frac{3}{2} \cdot s(M_{\mathcal{I}}) + \frac{4}{3} \cdot s(S_{\mathcal{I}}) + 1$.

Lemma 2.3. *Given a* BP *instance* $\mathcal{I} = (I, s)$*, there is a polynomial-time algorithm* AsymptoticBP *which returns a packing* $\mathcal{B} = (B_1, \ldots, B_t)$ *of* $\mathcal{I}$ *such that* $t = \mathrm{OPT}(\mathcal{I}) + o(\mathrm{OPT}(\mathcal{I}))$*. Moreover, if* $\mathrm{OPT}(\mathcal{I}) \geq 100$ *then* $t \leq 1.02 \cdot \mathrm{OPT}(\mathcal{I})$.

3 Approximations for Perfect and Bipartite Graphs

In this section we consider the bin packing problem with a perfect or bipartite conflict graph. Previous works (e.g., [8,17]) showed the usefulness of the approach based on finding first a minimum coloring of the given conflict graph, and then packing each color class as a separate bin packing instance (using, e.g., algorithm FFD). Indeed, this approach yields efficient approximations for BPC; however, it does reach a certain limit. To enhance the performance of this *coloring based* approach, we design several subroutines. Combined, they cover the problematic cases and lead to improved approximation guarantees (see Table 1).

Our first subroutine is the coloring based approach, with a simple modification to improve the asymptotic performance. For each color class $C_i, i = 1, \ldots, k$ in a minimum coloring of the given conflict graph, we find a packing of C_i using FFD, and another packing using AsymptoticBP (see Lemma 2.3). We choose the packing which has smaller number of bins. Finally, the returned packing is the concatenation of the packings of all color classes. The pseudocode of Algorithm Color_Sets is given in Algorithm 1.

Algorithm 1. Color_Sets($\mathcal{I} = (I, s, E)$)

1: Compute a minimum coloring $\mathcal{C} = (C_1, \ldots, C_k)$ of $G_{\mathcal{I}}$.
2: Initialize an empty packing $\mathcal{B} \leftarrow ()$.
3: **for** $i \in [k]$ **do**
4: Compute $\mathcal{A}_1 \leftarrow \mathsf{FFD}((C_i, s))$ and $\mathcal{A}_2 \leftarrow \mathsf{AsymptoticBP}((C_i, s))$.
5: $\mathcal{B} \leftarrow \mathcal{B} \oplus \arg\min_{\mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2\}} \#\mathcal{A}$.
6: **end for**
7: Return $\mathcal{B}$.

For the remainder of this section, fix a BPC instance $\mathcal{I} = (I, s, E)$. The performance guarantees of Algorithm Color_Sets are stated in the next lemma.

[4] For more details on algorithms FFD and AsymptoticBP see, e.g., [27].

Lemma 3.1. *Given a* BPC *instance* $\mathcal{I} = (I, s, E)$*, Algorithm* Color_Sets *returns in polynomial time in* $|\mathcal{I}|$ *a packing* $\mathcal{B}$ *of* $\mathcal{I}$ *such that* $\#\mathcal{B} \leq \chi(G_\mathcal{I}) + |L_\mathcal{I}| + \frac{3}{2} \cdot s(M_\mathcal{I}) + \frac{4}{3} \cdot s(S_\mathcal{I})$*. Moreover, if* $\mathcal{I}$ *is a* BPB *instance then* $\#\mathcal{B} \leq \frac{3}{2} \cdot |L_\mathcal{I}| + \frac{4}{3} \cdot (\mathrm{OPT}(\mathcal{I}) - |L_\mathcal{I}|) + o(\mathrm{OPT}(\mathcal{I}))$.

Note that the bounds may not be tight for instances with many large items. Specifically, if $|L_\mathcal{I}| \approx \mathrm{OPT}(\mathcal{I})$ then a variant of Algorithm Color_Sets was shown to yield a packing of at least $2.5 \cdot \mathrm{OPT}(\mathcal{I})$ bins [8]. To overcome this, we use an approach based on the simple yet crucial observation that there can be at most one large item in a bin. Therefore, we view the large items as *bins* and assign items to these bins to maximize the total size packed in bins including large items. We formalize the problem initially on a single bin.

Definition 3.2. *In the* bounded independent set problem (BIS) *we are given a graph* $G = (V, E)$*, a weight function* $w : V \to \mathbb{R}_{\geq 0}$*, and a budget* $\beta \in \mathbb{R}_{\geq 0}$*. The goal is to find an independent set* $S \subseteq V$ *in* G *such that* $w(S)$ *is maximized and* $w(S) \leq \beta$*. Let* $\mathcal{I} = (V, E, w, \beta)$ *be a BIS instance.*

Towards solving BIS, we need the following definitions. For $\alpha \in (0, 1]$, $\mathcal{A}$ is an α-approximation algorithm for a maximization problem $\mathcal{P}$ if, for any instance $\mathcal{I}$ of $\mathcal{P}$, $\mathcal{A}$ outputs a solution of value at least $\alpha \cdot OPT(\mathcal{I})$. A *polynomial-time approximation scheme (PTAS)* for $\mathcal{P}$ is a family of algorithms $\{A_\varepsilon\}$ such that, for any $\varepsilon > 0$, A_ε is a polynomial-time $(1 - \varepsilon)$-approximation algorithm for $\mathcal{P}$. A *fully PTAS (FPTAS)* is a PTAS $\{A_\varepsilon\}$ where, for all $\varepsilon > 0$, A_ε is polynomial also in $\frac{1}{\varepsilon}$. We now describe a PTAS for BIS. Fix a BIS instance $\mathcal{I} = (V, E, w, \beta)$ and $\varepsilon > 0$. As there can be at most ε^{-1} items with weight at least $\varepsilon \cdot \beta$ in some optimal solution OPT for $\mathcal{I}$, we can *guess* this set F of items via enumeration. Then, to add smaller items to F, we define a residual graph G_F of items with weights at most $\varepsilon \cdot \beta$ which are not adjacent to any item in F. Formally, define $G_F = (V_F, E_F)$, where

$$V_F = \{v \in V \backslash F \mid w(v) \leq \varepsilon \cdot \beta, \forall u \in F : (v, u) \notin E\},\ E_F = \{(u, v) \in E \mid u, v \in V_F\}$$

Now, we find a maximum weight independent set S in G_F. Note that this can be done in polynomial time for perfect and bipartite graphs. If $w(F \cup S) \leq \beta$ then we have an optimal solution; otherwise, we discard iteratively items from S until the remaining items form a feasible solution for $\mathcal{I}$. Since we discard only items with relatively small weights, we lose only an ε-fraction of the weight relative to the optimum. The pseudocode for the scheme is given in Algorithm 2.

Lemma 3.3. *Algorithm 2 is a* PTAS *for* BIS.

We note that by a result of [7], unless P=NP, BIS does not admit an *efficient* PTAS, even on bipartite graphs.[5] Thus, our PTAS for this problem is of an independent interest.

[5] An *efficient PTAS* is a PTAS $\{A_\varepsilon\}$ where, for all $\varepsilon > 0$, the running time of A_ε is given by $f(1/\varepsilon)$ times a polynomial of the input size.

Algorithm 2. PTAS$((V, E, w, \beta), \varepsilon)$

```
Initialize A ← ∅.
for all independent sets F ⊆ V in (V, E) s.t. |F| ≤ ε^{-1}, w(F) ≤ β do
    Define the residual graph G_F = (V_F, E_F).
    Find a maximum independent set S of G_F w.r.t. w.
    while w(F ∪ S) > β do
        Choose arbitrary z ∈ S.
        Update S ← S \ {z}.
    end while
    if w(A) < w(F ∪ S) then
        Update A ← F ∪ S.
    end if
end for
Return A.
```

We now define our maximization problem for multiple bins. We solve a slightly generalized problem in which we have an initial partial packing in t bins. Our goal is to add to these bins (from unpacked items) a subset of items of maximum total size. Formally,

Definition 3.4. *Given a* BPC *instance* $\mathcal{I} = (I, s, E)$, $S \subseteq I$, *and a packing* $\mathcal{B} = (B_1, \ldots, B_t)$ *of* S, *define the* maximization problem *of* $\mathcal{I}$ *and* $\mathcal{B}$ *as the problem of finding a packing* $\mathcal{B}+\mathcal{C}$ *of* $S \cup T$, *where* $T \subseteq I \setminus S$ *and* $\mathcal{C} = (C_1, \ldots, C_t)$ *is a packing of* T, *such that* $s(T)$ *is maximized.*

Our solution for BIS is used to obtain a $(1-\frac{1}{e}-\varepsilon)$-approximation for the maximization problem described in Definition 3.4. This is done using the approach of [9] for the more general *separable assignment problem (SAP)*.

Lemma 3.5. *Given a* BPC *instance* $\mathcal{I} = (I, s, E)$, $S \subseteq I$, *a packing* $\mathcal{B} = (B_1, \ldots, B_t)$ *of* S, *and a constant* $\varepsilon > 0$, *there is an algorithm* MaxSize *which returns in time polynomial in* $|\mathcal{I}|$ *a* $(1-\frac{1}{e}-\varepsilon)$*-approximation for the maximization problem of* $\mathcal{I}$ *and* $\mathcal{B}$. *Moreover, given an* FPTAS *for* BIS *on the graph* (I, E), *the weight function* s, *and the budget* $\beta = 1$, MaxSize *is a* $(1-\frac{1}{e})$*-approximation algorithm for the maximization problem of* $\mathcal{I}$ *and* $\mathcal{B}$.

We use the above to obtain a feasible solution for the instance. This is done via a reduction to the maximization problem of the instance with a singleton packing of the large items and packing the remaining items in extra bins. Specifically, in the subroutine MaxSolve, we initially put each item in $L_{\mathcal{I}}$ in a separate bin. Then, additional items from $S_{\mathcal{I}}$ and $M_{\mathcal{I}}$ are added to the bins using Algorithm MaxSize (defined in Lemma 3.5). The remaining items are packed using Algorithm Color_Sets. The pseudocode of the subroutine MaxSolve is given in Algorithm 3.

The proof of Lemma 3.6 uses Lemmas 3.1, 3.3, and 3.5.

Lemma 3.6. *Given a* BPC *instance* $\mathcal{I} = (I, s, E)$ *and an* $\varepsilon > 0$, *Algorithm* MaxSolve *returns in polynomial time in* $|\mathcal{I}|$ *a packing* $\mathcal{C}$ *of* $\mathcal{I}$ *such that there are* $0 \leq x \leq s(M_{\mathcal{I}})$ *and* $0 \leq y \leq s(S_{\mathcal{I}})$ *such that the following holds.*

Algorithm 3. MaxSolve($\mathcal{I} = (I, s, E), \varepsilon$)

1: Define $T \leftarrow (\{\ell\} \mid \ell \in L_{\mathcal{I}})$.
2: $\mathcal{A} \leftarrow$ MaxSize($\mathcal{I}, L_{\mathcal{I}}, T, \varepsilon$).
3: $\mathcal{B} \leftarrow$ Color_Sets($\mathcal{I} \setminus$ items($\mathcal{A}$)).
4: Return $\mathcal{A} \oplus \mathcal{B}$.

1. $x + y \leq \mathrm{OPT}(\mathcal{I}) - |L_{\mathcal{I}}| + \left(\frac{1}{e} + \varepsilon\right) \cdot \frac{|L_{\mathcal{I}}|}{2}$.
2. $\#\mathcal{C} \leq \chi(G_{\mathcal{I}}) + |L_{\mathcal{I}}| + \frac{3}{2} \cdot x + \frac{4}{3} \cdot y$.

Lemma 3.6 improves significantly the performance of Algorithm Color_Sets for instances with many large items. However, Algorithm MaxSize may prefer small over medium items; the latter items will be packed by Algorithm Color_Sets (see Algorithm 3). The packing of these medium items may harm the approximation guarantee. Thus, to tackle instances with many medium items, we use a reduction to a maximum matching problem for packing the large and medium items in at most $\mathrm{OPT}(\mathcal{I})$ bins.[6] Then, the remaining items can be packed using Algorithm Color_Sets. The graph used for the following subroutine Matching contains all large and medium items; there is an edge between any two items which can be assigned to the same bin in a packing of the instance $\mathcal{I}$. Formally,

Definition 3.7. *Given a* BPC *instance* $\mathcal{I} = (I, s, E)$*, the* auxiliary graph of $\mathcal{I}$ *is* $H_{\mathcal{I}} = (L_{\mathcal{I}} \cup M_{\mathcal{I}}, E_H)$*, where* $E_H = \{(u, v) \mid u, v \in L_{\mathcal{I}} \cup M_{\mathcal{I}}, s(\{u, v\}) \leq 1, (u, v) \notin E\}$.

Algorithm Matching finds a maximum matching in $H_{\mathcal{I}}$ and outputs a packing of the large and medium items where pairs of items taken to the matching are packed together, and the remaining items are packed in extra bins using Algorithm Color_Sets. The pseudocode of the subroutine Matching is given in Algorithm 4.

Algorithm 4. Matching($\mathcal{I} = (I, s, E)$)

1: Find a maximum matching $\mathcal{M}$ in $H_{\mathcal{I}}$.
2: $\mathcal{B} \leftarrow (\{u, v\} \mid (u, v) \in \mathcal{M}) \oplus (\{v\} \mid v \in M_{\mathcal{I}} \cup L_{\mathcal{I}}, \forall u \in M_{\mathcal{I}} \cup L_{\mathcal{I}} : (u, v) \notin \mathcal{M})$.
3: Return $\mathcal{B} \oplus$ Color_Sets($\mathcal{I} \setminus (M_{\mathcal{I}} \cup L_{\mathcal{I}})$).

The proof of Lemma 3.8 follows by noting that the cardinality of a maximum matching in $H_{\mathcal{I}}$ in addition to the number of unmatched vertices in $L_{\mathcal{I}} \cup M_{\mathcal{I}}$ is at most $\mathrm{OPT}(\mathcal{I})$.

Lemma 3.8. *Given a* BPC *instance* $\mathcal{I} = (I, s, E)$*, Algorithm* Matching *returns in polynomial time in* $|\mathcal{I}|$ *a packing* $\mathcal{A}$ *of* $\mathcal{I}$ *such that* $\#\mathcal{A} \leq \mathrm{OPT}(\mathcal{I}) + \chi(G_{\mathcal{I}}) + \frac{4}{3} \cdot s(S_{\mathcal{I}})$.

[6] We note that a maximum matching based technique for BPC is used also in [8,15].

We now have the required components for the approximation algorithm for BPC and the asymptotic approximation for BPB. Our algorithm, ApproxBPC, applies all of the above subroutines and returns the packing which uses the smallest number of bins. We use $\varepsilon = 0.0001$ for the error parameter in MaxSolve. The pseudocode of ApproxBPC is given in Algorithm 5.

Algorithm 5. ApproxBPC($\mathcal{I}$)

1: Let $\varepsilon = 0.0001$.
2: Compute $\mathcal{A}_1 \leftarrow$ Color_Sets($\mathcal{I}$), $\mathcal{A}_2 \leftarrow$ MaxSolve($\mathcal{I}, \varepsilon$), $\mathcal{A}_3 \leftarrow$ Matching($\mathcal{I}$).
3: Return $\arg\min_{\mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3\}} \#\mathcal{A}$.

We give below the main result of this section. The proof follows by the argument that the subroutines Color_Sets, MaxSolve, and Matching handle together most of the difficult cases. Specifically, if the instance contains many large items, then MaxSolve produces the best approximation. If there are many large and medium items, then Matching improves the approximation guarantee. Finally, for any other case, our analysis of the Color_Sets algorithm gives us the desired ratio. We summarize with the next result.

Theorem 3.9. *Algorithm 5 is a* 2.445*-approximation for* BPC *and an asymptotic* 1.391*-approximation for* BPB.

4 Split Graphs

In this section we enhance the use of maximization techniques for BPC to obtain an absolute approximation algorithm for BPS. In particular, we improve upon the recent result of Huang et al. [15]. We use as a subroutine the maximization technique as outlined in Lemma 3.5. Specifically, we start by obtaining an FPTAS for the BIS problem on split graphs. For the following, fix a BPS instance $\mathcal{I} = (I, s, E)$. It is well known (see, e.g., [11]) that a partition of the vertices of a split graph into a clique and an independent set can be found in polynomial time. Thus, for simplicity we assume that such a partition of the split graph G is known and given by K_G, S_G. We note that an FPTAS for the BIS problem on split graphs follows from a result of Pferschy and Schauer [24] for *knapsack with conflicts*, since split graphs are a subclass of chordal graphs. We give a simpler FPTAS for our problem in [6].

Lemma 4.1. *There is an algorithm* FPTAS-BIS *that is an* FPTAS *for the* BIS *problem on split graphs.*

Our next goal is to find a suitable initial packing $\mathcal{B}$ to which we apply MaxSize. Clearly, the vertices $K_{G_\mathcal{I}}$ must be assigned to different bins. Therefore, our initial packing contains the vertices of $K_{G_\mathcal{I}}$ distributed to $|K_{G_\mathcal{I}}|$ bins as $\{\{v\} \mid v \in K_{G_\mathcal{I}}\}$. In addition, let $\alpha \in \{0, 1, \ldots, \lceil 2 \cdot s(I) \rceil + 1\}$ be a *guess* of $\mathrm{OPT}(\mathcal{I}) - |K_{G_\mathcal{I}}|$;

then, $(\emptyset)_{i\in[\alpha]}$ is a packing of α bins that do not contain items. Together, the two above packings form the initial packing $\mathcal{B}_\alpha$. Our algorithm uses MaxSize to add items to the existing bins of $\mathcal{B}_\alpha$ and packs the remaining items using FFD. Note that we do not need an error parameter ε, since we use MaxSize with an FPTAS (see Lemma 3.5). For simplicity, we assume that $\mathrm{OPT}(\mathcal{I}) \geq 2$ (else we can trivially pack the instance in a single bin) and omit the case where $\mathrm{OPT}(\mathcal{I}) = 1$ from the pseudocode. We give the pseudocode of our algorithm for BPS in Algorithm 6.

Algorithm 6. Split-Approx($\mathcal{I} = (I, s, E)$)

1: **for** $\alpha \in \{0, 1, \ldots, \lceil 2 \cdot s(I) \rceil + 1\}$ **do**
2: Define $\mathcal{B}_\alpha = \{\{v\} \mid v \in K_{G_\mathcal{I}}\} \oplus (\emptyset)_{i\in[\alpha]}$
3: $\mathcal{A}_\alpha \leftarrow \mathsf{MaxSize}(\mathcal{I}, K_{G_\mathcal{I}}, \mathcal{B}_\alpha)$.
4: $\mathcal{A}^*_\alpha \leftarrow \mathcal{A}_\alpha \oplus \mathsf{FFD}(\mathcal{I} \setminus \mathsf{items}(\mathcal{A}_\alpha))$.
5: **end for**
6: Return $\arg\min_{\alpha\in\{0,1,\ldots,\lceil 2\cdot s(I)\rceil+1\}} \#\mathcal{A}^*_\alpha$.

By Lemmas 4.1 and 3.5 we have a $\left(1 - \frac{1}{e}\right)$-approximation for the maximization problem of the BPS instance $\mathcal{I}$ and an initial partial packing $\mathcal{B}$. Hence, for a correct guess $\alpha = \mathrm{OPT}(\mathcal{I}) - |K_{G_\mathcal{I}}|$, the remaining items to be packed by FFD are of total size at most $\frac{s(I)}{e}$ and can be packed in $\frac{2\cdot\mathrm{OPT}(\mathcal{I})}{e}$ bins. Thus, we have

Theorem 4.2. *Algorithm 6 is a* $\left(1 + \frac{2}{e}\right)$*-approximation for* BPS.

5 Asymptotic Hardness for Bipartite and Split Graphs

In this section we show that there is no APTAS for BPB and BPS, unless $P = NP$. We use a reduction from the *Bounded 3-dimensional matching (B3DM)* problem, that is known to be MAX SNP-complete [18].

For the remainder of this section, let $c > 2$ be some constant. A B3DM instance is a four-tuple $\mathcal{J} = (X, Y, Z, T)$, where X, Y, Z are three disjoint finite sets and $T \subseteq X \times Y \times Z$; also, for each $u \in X \cup Y \cup Z$ there are at most c triples in T to which u belongs. A *solution* for $\mathcal{J}$ is $M \subseteq T$ such that for all $u \in X \cup Y \cup Z$ it holds that u appears in at most one triple of M. The objective is to find a solution M of maximum cardinality. Let $\mathrm{OPT}(\mathcal{J})$ be the value of an optimal solution for $\mathcal{J}$. We use in our reduction a *restricted* instance of B3DM defined as follows.

Definition 5.1. *For* $k \in \mathbb{N}$*, a* B3DM *instance* $\mathcal{J}$ *is* k-restricted *if* $\mathrm{OPT}(\mathcal{J}) \geq k$.

In the next lemma we show the hardness of k-restricted B3DM. Intuitively, since B3DM instances $\mathcal{J}$ with $\mathrm{OPT}(\mathcal{J}) \leq k$ are polynomially solvable for a fixed k (e.g., by exhaustive enumeration), it follows that restricted-B3DM must be hard to approximate, by the hardness result of Kann [18].

Lemma 5.2. *There is a constant $\alpha < 1$ such that for any $k \in \mathbb{N}$ there is no α-approximation for the k-restricted* B3DM *problem unless* P=NP.

We give below the main idea of our reduction, showing the asymptotic hardness of BPB and BPS. A more formal description and the proof of Lemma 5.2 are given in [6]. For a sufficiently large $n \in \mathbb{N}$, let $\mathcal{J} = (X, Y, Z, T)$ be an n-restricted instance of B3DM, and let the components of $\mathcal{J}$, together with appropriate indexing, be $U = X \cup Y \cup Z$ and T, where

$$X = \{x_1, \ldots, x_{\tilde{x}}\}, Y = \{y_1, \ldots, y_{\tilde{y}}\}, Z = \{z_1, \ldots, z_{\tilde{z}}\}, T = \{t_1, \ldots, t_{\tilde{t}}\}.$$

We outline our reduction for BPB and later show how it can be modified to yield the hardness result for BPS. Given an n-restricted B3DM instance, we construct a sequence of BPB instances. Each BPB instance contains an item for each element $u \in U$, and an item for each triple $t \in T$. There is an edge (u, t) if $u \in U$ and $t \in T$, and u does not appear in t, i.e., we forbid packing an element u in the same bin with a triple not containing u, for any $u \in U$. Since we do not know the exact value of $\mathrm{OPT}(\mathcal{J})$, we define a family of instances with different number of *filler items*; these items are packed in the optimum of our constructed BPB instance together with elements not taken to the solution for $\mathcal{J}$.

Specifically, for a *guess* $i \in \{n, n+1, \ldots, |T|\}$ of $\mathrm{OPT}(\mathcal{J})$, we define a BPB instance $\mathcal{I}_i = (I_i, s, E)$. The set of items in $\mathcal{I}_i$ is $I_i = U \cup P_i \cup T \cup Q_i$, where P_i, Q_i are a set of $\tilde{t} - i$ (filler) items and a set of $\tilde{x} + \tilde{y} + \tilde{z} - 3 \cdot i$ (filler) items, respectively, such that $P_i \cap U = \emptyset$ and $Q_i \cap U = \emptyset$. The bipartite (conflict) graph of $\mathcal{I}_i$ is $G_i = (I_i, E)$, where $E = E_X \cup E_Y \cup E_Z$ is defined as follows.

$$E_X = \{(x, t) \mid x \in X, t = (x', y, z) \in T, x \neq x'\}$$
$$E_Y = \{(y, t) \mid y \in Y, t = (x, y', z) \in T, y \neq y'\}$$
$$E_Z = \{(z, t) \mid z \in Z, t = (x, y, z') \in T, z \neq z'\}$$

Finally, define the sizes of items in $\mathcal{I}_i$ to be

$$\forall u \in U, p \in P_i, q \in Q_i, t \in T: \ s(u) = 0.15, s(p) = 0.45, s(q) = 0.85, s(t) = 0.55.$$

By the above, the only way to pack three items from $x, y, z \in U$ with a triple $t \in T$ is if $(x, y, z) = t$; also, $s(\{x, y, z, t\}) = 1$. For an illustration of the reduction see Fig. 1.

Given a packing $(A_1, \ldots, A_q)$ for the BPB instance $\mathcal{I}_i$, we consider all *useful bins* A_b in the packing, i.e., $A_b = \{x, y, z, t\}$, where $x \in X, y \in Y, z \in Z$ and $t = (x, y, z)$. The triple t from bin A_b is taken to our solution for the original n-restricted B3DM instance $\mathcal{J}$. Note that taking all triples as described above forms a feasible solution for $\mathcal{J}$, since each element is packed only once. Thus, our goal becomes to find a packing for the reduced BPB instance with a maximum number of useful bins. Indeed, since $s(A_b) = 1$ for any useful bin A_b, finding a packing with many useful bins coincides with an efficient approximation for BPB.

For the optimal guess $i^* = \mathrm{OPT}(\mathcal{J})$, it is not hard to see that the optimum for the BPB instance $\mathcal{I}_{i^*}$ satisfies $s(I_{i^*}) = \mathrm{OPT}(\mathcal{I}_{i^*})$; that is, all bins in the optimum

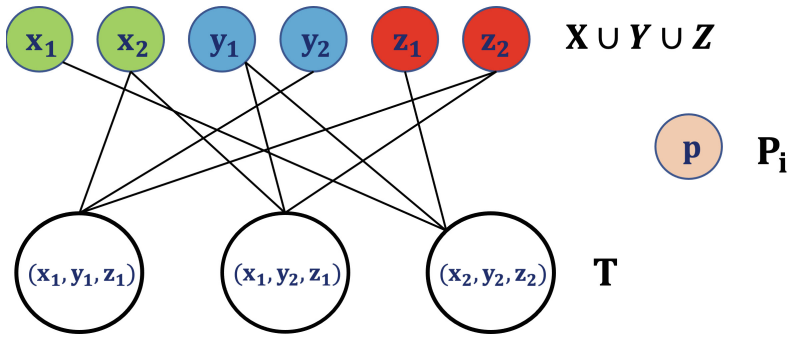

Fig. 1. An illustration of the BPB instance $\mathcal{I}_i = (I_i, s, E)$, where $i = \mathrm{OPT}(\mathcal{J}) = 2$. The optimal solution for $\mathcal{I}_i$ contains the bins $\{x_1, y_1, z_1, (x_1, y_1, z_1)\}, \{x_2, y_2, z_2, (x_2, y_2, z_2)\}$, and $\{p, (x_1, y_2, z_1)\}$; this corresponds to an optimal solution $(x_1, y_1, z_1), (x_2, y_2, z_2)$ for the original B3DM instance. Note that in this example $Q_i = \emptyset$.

are *fully* packed. For a sufficiently large n, and assuming there is an APTAS for BPB, we can find a packing of $\mathcal{I}_{i^*}$ with a large number of bins that are fully packed. A majority of these bins are useful, giving an efficient approximation for the original B3DM instance. A similar reduction to BPS is obtained by adding to the bipartite conflict graph of the BPB instance an edge between any pair of vertices in T; thus, we have a *split* conflict graph. We summarize the above discussion in the next result (the proof is given in [6]).

Theorem 5.3. *There is no* APTAS *for* BPB *and* BPS, *unless* P=NP.

6 Discussion

In this work we presented the first theoretical evidence that BPC on polynomially colorable graphs is harder than classic bin packing, even in the special cases of bipartite and split graphs. Furthermore, we introduced a new generic framework for tackling BPC instances, based on a reduction to a maximization problem. Using this framework, we improve the state-of-the-art approximations for BPC on several well studied graph classes.

We note that better bounds for the maximization problems solved within our framework will imply improved approximation guarantees for BPC on perfect, bipartite, and split graphs. It would be interesting to apply our techniques to improve the known results for other graph classes, such as chordal graphs or partial k-trees.

References

1. Adany, R., et al.: All-or-nothing generalized assignment with application to scheduling advertising campaigns. ACM Trans. Algorithms **12**(3), 38:1–38:25 (2016)

2. Beaumont, O., Bonichon, N., Duchon, P., Larchevêque, H.: Distributed approximation algorithm for resource clustering. In: Shvartsman, A.A., Felber, P. (eds.) SIROCCO 2008. LNCS, vol. 5058, pp. 61–73. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69355-0_7
3. Christofides, N.: The vehicle routing problem. Combinatorial optimization (1979)
4. Doron-Arad, I., Kulik, A., Shachnai, H.: An APTAS for bin packing with clique-graph conflicts. In: Lubiw, A., Salavatipour, M. (eds.) WADS 2021. LNCS, vol. 12808, pp. 286–299. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-83508-8_21
5. Doron-Arad, I., Kulik, A., Shachnai, H.: An AFPTAS for bin packing with partition matroid via a new method for LP rounding. In: Proceedings of APPROX (2023)
6. Doron-Arad, I., Shachnai, H.: Approximating bin packing with conflict graphs via maximization techniques. arXiv preprint arXiv:2302.10613 (2023)
7. Doron-Arad, I., Shachnai, H.: Tight bounds for budgeted maximum weight independent set in bipartite and perfect graphs. arXiv preprint arXiv:2307.08592 (2023)
8. Epstein, L., Levin, A.: On bin packing with conflicts. SIAM J. Optim. **19**(3), 1270–1298 (2008)
9. Fleischer, L., Goemans, M.X., Mirrokni, V.S., Sviridenko, M.: Tight approximation algorithms for maximum separable assignment problems. Math. Oper. Res. **36**(3), 416–431 (2011)
10. Garey, M.R., Johnson, D.S.: Computers and intractability. A Guide to the (1979)
11. Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs. Elsevier, Amsterdam (2004)
12. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization, vol. 2. Springer, Berlin (2012)
13. Halldórsson, M.M.: A still better performance guarantee for approximate graph coloring. Inf. Process. Lett. **45**(1), 19–23 (1993)
14. Hoberg, R., Rothvoß, T.: A logarithmic additive integrality gap for bin packing. In: Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 2616–2625. SIAM (2017)
15. Huang, Z., Zhang, A., Dósa, G., Chen, Y., Xiong, C.: Improved approximation algorithms for bin packing with conflicts. Int. J. Found. Comput. Sci. 1–16 (2023)
16. Jansen, K.: An approximation scheme for bin packing with conflicts. J. Comb. Optim. **3**(4), 363–377 (1999)
17. Jansen, K., Öhring, S.R.: Approximation algorithms for time constrained scheduling. Inf. Comput. **132**(2), 85–108 (1997)
18. Kann, V.: Maximum bounded 3-dimensional matching is MAX SNP-complete. Inf. Process. Lett. **37**(1), 27–35 (1991)
19. Karmarkar, N., Karp, R.M.: An efficient approximation scheme for the one-dimensional bin-packing problem. In: 23rd Annual Symposium on Foundations of Computer Science, pp. 312–320. IEEE (1982)
20. de La Vega, W.F., Lueker, G.S.: Bin packing can be solved within $1+\varepsilon$ in linear time. Combinatorica **1**(4), 349–355 (1981)
21. Laporte, G., Desroches, S.: Examination timetabling by computer. Comput. Oper. Res. **11**(4), 351–360 (1984)
22. McCloskey, B., Shankar, A.: Approaches to bin packing with clique-graph conflicts. University of California, Computer Science Division (2005)
23. Oh, Y., Son, S.: On a constrained bin-packing problem. Technical Report CS-95-14 (1995)
24. Pferschy, U., Schauer, J.: The knapsack problem with conflict graphs. J. Graph Algorithms Appl. **13**(2), 233–249 (2009)

25. Rothvoß, T.: Approximating bin packing within O(log OPT * log log OPT) bins. In: 54th Annual IEEE Symposium on Foundations of Computer Science, pp. 20–29. IEEE Computer Society (2013)
26. Simchi-Levi, D.: New worst-case results for the bin-packing problem. Naval Res. Logist. (NRL) **41**(4), 579–585 (1994)
27. Vazirani, V.V.: Approximation Algorithms. Springer, Berlin, Heidelberg (2001)
28. Zuckerman, D.: Linear degree extractors and the inapproximability of max clique and chromatic number. In: Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing, pp. 681–690 (2006)