A Two-Pass Algorithm for Unordered Colored Bin Packing

Hamza Alsarhan, Davin Chia, Ananya Christman*, Shannia Fu, and Yanfeng Jin

Middlebury College, Middlebury VT 05753
{halsarhan, dchia, achristman, sfu, yjin}@middlebury.edu

Abstract. In the Colored Bin Packing problem a set of items with varying weights and colors must be packed into bins of uniform weight limit such that no two items of the same color may be packed adjacently within a bin. We consider a version of the problem where there is no ordering among the items. We present exact, linear-time algorithms for this problem for the cases where there are two or more colors when the items have zero weight and when the items have unit weight. Our algorithms require only two passes over the input items. We also find closed-form expressions for the optimal number of bins.

Keywords: algorithms, combinatorial optimization, bin packing, colors, class constraints

1 Introduction

The Bin Packing problem is a classical optimization problem with many practical applications related to minimizing time or space. The *Colored Bin Packing* problem is a variation proposed by Bohm et al. [2] and independently by Dosa et al. [4], where we are given a set of items each with a weight and a color. We have an unlimited number of bins of uniform weight limit and we assume that no item weighs more than the bin weight limit. The goal is to pack the items in as few bins as possible while maintaining that no two items of the same color are packed adjacently within a bin. Bohm et al. [2] studied this problem in the *restricted offline setting*, where the set of items is received as a sequence and must be packed in order. We consider a relaxed variation of the problem where the items have no particular ordering and therefore may be packed in any order.

Formally, the input is a set S of n items and a bin weight limit L. We first consider the case where the items have zero weight and present a linear time algorithm, Alternate-Zero, that solves this problem optimally. We then present another linear time algorithm, Alternate-Unit, that solves the problem optimally for unit-weight items. If the input is as described above, our algorithms require exactly two pass over each item to perform the packing. In the first pass, we simply separate the items into

* Corresponding author

In: A. Kononov et al. (eds.): DOOR 2016, Vladivostok, Russia, published at http://ceur-ws.org

subsets of items of the same color while maintaining a count of each subset and the most frequent color. The second pass performs the packing. We note that the first pass is a pre-processing step and does not involve any packing. Therefore if the input was instead given as (1) k subsets $S_1, S_2, \ldots, S_k$ where k is the number of colors, and S_1 contains the items of the most frequent color and S_i for $i > 1$ contains the items of color i, and (2) $|S_1|$ (i.e. the number of items of the most frequent color) and $|S_2| + |S_3| + \ldots + |S_k|$ (i.e. the number of items of other colors), our algorithm would require just one pass to pack the items. Another advantage of our algorithms is that, similar to many classical Bin Packing algorithms (Best-Fit, First-Fit, Next-Fit), they are simple; however their optimality is not obvious.

This *Colored Bin Packing* problem has many practical applications as described in [2]. For example, television and radio stations often schedule a set of programs on different channels. Each program may fall into a genre such as comedy, documentary, and sports, on TV, or various musical genres on radio. To maintain a diverse schedule of programs, the station would like to avoid broadcasting two programs of the same genre one after the other. This problem can be modeled as Colored Bin Packing where the items correspond to programs, colors to genres, and bins to channels.

Another application involves generating diverse content to be displayed on websites. Many websites prefer displays that alternate between different types of information and advertisements and the Colored Bin Packing problem can be used to generate such a display. The items correspond to the contents to display, the colors to the type of content, and the bins to the size of the display.

The remainder of this paper is organized as follows. In Section 1.1 we discuss several previous results related to the Colored Bin Packing problem. In Sections 2 and 3 we present our results for zero-weight and unit-weight items, respectively. In Section 4 we provide closed-form expressions for the optimal number of bins.

1.1 Related Work

Classic Bin Packing is a well known problem, with many applications throughout various fields. Bin Packing with color constraints is relatively recent, arising with the increasing complexity of contemporary industrial processes. The use of color gives classic Bin Packing added flexibility in modeling real-world problems. For example, Oh and Son used color, specifically, the requirement that two items with the same color cannot be packed in the same bin, to efficiently assign tasks in real time within a multiprocessor system [10]. Their result was a modified First Fit algorithm that is 1.7-competitive in the worst case, and 1.1 in the average case. Dawande et al. investigated a version of Colored Bin Packing where each bin has a maximum *color capacity* [3] i.e. a limit on the number of items of a particular color. Such a problem models the slab design problem in the production planning process of a steel plant. The authors present two 3-approximation algorithms, classical First-Fit-Decreasing, and a modified First Fit. Epstein et al. also studied Colored Bin with color capacities. They prove upper and lower bounds for several variants of this problem in both the offline and online settings. Xavier and Miyazawa use Bin Packing with Colors (they refer to the colors as *classes*) to model Video-on-Demand applications [12]. Specifically, there is a server of multiple disks that each have limited storage capacity and can hold video files of varying sizes and genres (i.e. classes). The application is given an expected number of requests for movies based on movie/genre popularity. The goal is to construct a server that maximizes the number of satisfied requests. A more generalized version of constrained bin packing where the constraints among items are defined in a conflict graph have been studied by Jansen [6] and Muritiba et al. [9].

Balogh et al. introduced the Black and White Bin Packing problem with *alternation* constraints, where two items with the same color cannot be packed adjacently to each other in a bin [1]. They studied both the offline and online versions of the problem. For the offline version the authors presented a 2.5-approximation algorithm. For the online version, they proved that classical algorithms (First, Best, Worst, Next, Harmonic) are not constant competitive, and hence do not perform well. They further proved a universal lower bound of approximately 1.7213. This exceeds classical Bin Packing's upper bound of 1.5889, proving that Black and White Bin Packing with alternation is harder [11]. The authors' main result was a 3-competitive online algorithm, *Pseudo*. The main concept of the algorithm is quite elegant. Using "pseudo", i.e. dummy, items of different color and zero weight, their algorithm creates an acceptable sequence of colors that are later substituted with actually released items. If any bin overflows, Pseudo redistributes the extra items using Any Fit into available bins, and creates new bins if necessary.

Bohm et al. explored the online Colored Bin Packing with alternation constraints where there two or more colors [2]. They study this problem for the *restricted offline* setting, i.e. where the items are presented in a particular order and must be packed in this order, which is a harder version of the problem we address. For zero-weight items, they present a recursive algorithm whose straightforward implementation runs in $O(n^2)$ time, where n is the number of items. Although our algorithms solve an easier problem, they do so with at most two passes over the input. Bohm et al. also show that the optimal number of bins equals the color *discrepancy* - the absolute difference between the number of items of the two colors. The paper's main result is an optimal online algorithm for the zero-weight case, named Balancing Any Fit, that operates on the simple principle that an item should be packed in an available bin containing the most oppositely-colored items. This is made easier since weight is not an issue.

Dosa and Epstein [4] extend the results from [1]. They prove that online Colored Bin Packing with alternation for 3 or more colors is harder than the 2-color version. Furthermore, the authors show that 2-color algorithms do not apply to 3 or more color problems. They also proved that no optimal algorithm exists for online Colored Bin Packing when there are more than two colors even when the items have zero-weight. Lastly, they present a 4-competitive algorithm for the problem. This is a modified version of the earlier Pseudo, appropriately named Balanced-Pseudo. Instead of randomly assigning items to available bins, Balanced-Pseudo assigns an item to the bin whose top color is the most frequent.

In this paper we consider the offline Colored Bin Packing problem where there is no ordering among the items. We present exact linear time algorithms that solve this problem for two or more colors when the items have zero-weight and when the items have unit-weight. This problem is related to the problem of finding alternating Euler tours in edge-colored graphs [7]. However, our work is different in that (1) it applies to bin packing rather than graphs, (2) the unit-weight case of our problem solves the bounded-length version of the alternating tour problem which has not yet been explicitly solved [8], and (3) our algorithms solve the problems using at most two passes over the input items.

2 Zero-Weight Colored Bin Packing

For the zero-weight case, the input is a set S of n items, where each item has a color from a set of k colors and weight zero. The Algorithm ALTERNATE-ZERO solves the zero-weight case. We let $MaxColor$ denote the most frequent color, i.e. the color with the most number of items, $OtherColors$ denote the set of all other colors and $MaxCount$ and $OtherCount$ denote the number of items of $MaxColor$ and $OtherColors$, respectively. In the zero-weight case, the bins do not have a weight limit so the only constraint is that no items of the same color may be packed adjacently in a bin. As in [2], we find that the number of bins depends on the *discrepancy* of the items, i.e. the difference between the number of $MaxColor$ and $OtherColors$ items.

Algorithm 1: ALTERNATE-ZERO(S: set of n items)

```
1: Separate $S$ into $S_1, S_2, \ldots, S_k$ where $S_1$ contains the items of the most frequent
   color, $MaxColor$, and $S_i$, for $i > 1$, contains the set of all items of color $i$.
   Also maintain:
   $MaxCount$: number of items of $MaxColor$,
   $OtherColors$: set of all non-$MaxColor$ colors,
   and $OtherCount$: number of items of $OtherColors$
2: $D = MaxCount - OtherCount$
3: if $D \leq 0$ then
4:    Alternate between $OtherColors$ items until there is one fewer $OtherColors$ item
      than $MaxColor$ item remaining.
5:    Start with a $MaxColor$ item and alternate between an $OtherColors$ item and a
      $MaxColor$ item until all items are packed.
6: else
7:    Alternate between a $MaxColor$ item and an $OtherColors$ item until all
      $OtherColors$ items are packed.
8:    There will be $MaxCount - OtherCount - 1$ remaining $MaxColor$ items so pack
      each in its own bin.
9: end if
```

Theorem 1. *The Algorithm* ALTERNATE-ZERO *is optimal for the Zero-Weight Colored Bin Packing Problem.*

Proof. We let $D = MaxCount - OtherCount$ denote the discrepancy and consider two cases based on D.

Case 1: $D \leq 0$. In this case there are at least as many $OtherColors$ items as there are $MaxColor$ items. Therefore, we need only one bin, which is the fewest number of bins any algorithm will use. If $D = 0$, we simply alternate between $MaxColor$ and $OtherColors$ items. If $D < 0$, we alternate between items of $OtherColors$ until there is one fewer $OtherColors$ item than $MaxColor$ item. At this point, we add one $MaxColor$ item to the bin and then alternate between an $OtherColors$ item and a $MaxColor$ item until all items are packed. Therefore, we use one bin. Note that to pack the items using exactly one pass, we must first alternate between $OtherColors$ items before packing any $MaxColor$ items. This ensures that we are not left with two or more $OtherColors$ items of the same color.

Also note that when $D < 0$ (i.e. there are more $MaxColor$ items than $OtherColors$ items) there will be enough $OtherColors$ items to alternate between until there is one fewer $OtherColors$ item than $MaxColor$ item. Suppose, by contradiction, that this is not the case, so when we can no longer alternate between $OtherColors$ items, the number of $OtherColors$ items is equal to or more than the number of $MaxColor$ items. If we can no longer alternate between $OtherColors$ items, it must mean that there is only one color, say C, remaining in the set of $OtherColors$ items left to pack. Let c denote the number of C-colored items left to pack, so $c \geq MaxCount$. Since we must have packed at least one C-colored item and no $MaxColor$ items, there must have been at least $c + 1 \geq MaxCount + 1$ C-colored items in the input, which contradicts that $MaxColor$ is the most frequent color.

As an example, if we are given 4 White (W), 3 Black (B), 3 Yellow (Y), and 1 Red (R) items, then $MaxColor$ is White, $MaxCount = 4$, $OtherCount = 7$, so $D = -3$. An optimal packing is: BYBYWBWYWRW. Note that if we started by packing a $MaxColor$ item (a tempting choice), we would need to use at least two passes to pack all the items.

Case 2: $D > 0$. In this case, there are enough $MaxColor$ items that each $OtherColors$ item is needed to be packed in between two $MaxColor$ items. Therefore, all $OtherColors$ items can be regarded as a single color not equal to $MaxColor$, making this the black/white bin packing problem [1]. The first bin will start with a $MaxColor$ item, alternate between $MaxColor$ and $OtherColors$ items, and end with a $MaxColor$ item. Thus, the first bin will contain all the $OtherColors$ items and $OtherCount + 1$ $MaxColor$ items. There will be $MaxCount - OtherCount - 1$ $MaxColor$ items remaining, each of which must be packed in a separate bin. This gives a total of $MaxCount - OtherCount$ bins, i.e. the discrepancy. Since the first bin contains as many $MaxColor$ items as possible and none of the additional bins can be combined, this is the optimal packing. □

As an example, if we are given 8 W, 2 B, and 2 Y items, then $MaxCount = 8$, $OtherCount = 4$, so $D = 4$. An optimal packing is: WBWBWYWYW / W / W / W (where / denotes a new bin).

3 Unit-Weight Colored Bin Packing

For the unit-weight case, the input is a set S of n items, where each item has a color from a set of k colors and unit weight, and a bin weight limit L. The general idea of Algorithm ALTERNATE-UNIT is as follows. The unit-weight case introduces weight

Algorithm 2: ALTERNATE-UNIT(S: set of n items, L: bin weight limit)

```
 1: Separate $S$ into $S_1, S_2, \ldots, S_k$ where $S_1$ contains the items of the most frequent
    color, $MaxColor$, and $S_i$, for $i > 1$, contains the set of all items of color $i$.
    Also maintain:
    $MaxCount$: number of items of $MaxColor$,
    $OtherColors$: set of all non-$MaxColor$ colors,
    and $OtherCount$: number of items of $OtherColors$
 2: $D = MaxCount - OtherCount$
 3: **if** $D \leq 0$ **then**
 4:   ALTERNATE-ZERO($S$). //apply a modified ALTERNATE-ZERO that packs at most $L$
      items per bin.
 5: **else**
 6:   **if** $L$ is even **then**
 7:     **if** $OtherCount < D(L/2) - 1$ **then**
 8:       **while** There are $OtherColors$ items remaining **do**
 9:         Pack each bin, alternatingly, with $L/2$ $MaxColor$ items and $L/2 - 1$
            $OtherColors$ items.
10:       **end while**
11:       There will be $MaxColor$ items remaining. Pack each in a separate bin.
12:     **else**
13:       **for** $i = 1$ to $D$ **do**
14:         Pack each bin, alternatingly, with $L/2$ $MaxColor$ items and $L/2 - 1$
            $OtherColors$ items.
15:       **end for**
16:       Let $R$ denote the set of remaining items.
17:       ALTERNATE-ZERO($R$) //apply a modified ALTERNATE-ZERO that packs
          at most $L$ items per bin.
18:     **end if**
19:   **else**
20:     **if** $OtherCount < D\lfloor L/2 \rfloor$ **then**
21:       **while** There are $OtherColors$ items remaining **do**
22:         Pack each bin, alternatingly, with $\lceil L/2 \rceil$ $MaxColor$ items and $\lfloor L/2 \rfloor$
            $OtherColors$ items.
23:       **end while**
24:       There will be $MaxColor$ items remaining. Pack each in a separate bin.
25:     **else**
26:       **for** $i = 1$ to $D$ **do**
27:         Pack each bin, alternatingly, with $\lceil L/2 \rceil$ $MaxColor$ items and $\lfloor L/2 \rfloor$
            $OtherColors$ items.
28:       **end for**
29:       Let $R$ denote the set of remaining items.
30:       ALTERNATE-ZERO($R$) //apply a modified ALTERNATE-ZERO that packs
          at most $L$ items per bin.
31:     **end if**
32:   **end if**
33: **end if**
```

constraints. However, color constraints, specifically the discrepancy, may force us to use more bins than the weight constraint alone. Specifically, if discrepancy does not pose a problem (i.e. is not more than zero), we simply pack the items using a modified ALTERNATE-ZERO that packs at most L items per bin. If discrepancy is more than zero, the packing depends on whether L is even or odd. In both cases, we start with a $MaxColor$ item and alternate between $MaxColor$ and $OtherColors$ items. If L is even, it may be optimal to not pack each bin full. Instead we should save $OtherColors$ items for packing in between excess $MaxColor$ items. If L is odd, every full bin will contain one more $MaxColor$ item than $OtherColors$ item, so our packing may eventually reduce discrepancy to zero. If so, we pack the remaining items as we did in the zero-weight case. Otherwise, we will have bins containing only a single $MaxColor$ item.

Lemma 1. *If $D > 0$ we can pack at most D bins that contain one more $MaxColor$ item than $OtherColors$ item before the discrepancy reduces to zero.*

Proof. Packing one such bin reduces the discrepancy to $D - 1$, packing two such bins reduces the discrepancy to $D - 2$, and so on. Therefore, packing D such bins reduces the discrepancy to $D - D = 0$. Therefore, if $OtherCount \geq D(L/2 - 1)$ for even L or $OtherCount \geq D\lfloor L/2 \rfloor$ for odd L, then we can pack D bins that contain one more $MaxColor$ item than $OtherColors$ item.

□

Theorem 2. *The Algorithm* ALTERNATE-UNIT *is optimal for the Unit-Weight Colored Bin Packing Problem.*

Proof. As in the zero-weight case, we consider two cases based on the discrepancy:

Case 1: $D \leq 0$. We know from the zero-weight case that ALTERNATE-ZERO satisfies the color constraint when discrepancy is less than or equal to 0. Therefore, we call a modified ALTERNATE-ZERO that packs at most L items per bin. Since ALTERNATE-ZERO is optimal, this packing will yield the optimal number of bins.

As an example, if we are given 4 W, 3 B, and 2 Y items, and $L = 3$, then $D = -1$ and an optimal packing is BYW / BWB / WYW.

Case 2: $D > 0$. This is the complex case as now the packing depends on both the weight and the color constraints. We pack bins by starting with a $MaxColor$ item and alternating between $OtherColors$ and $MaxColor$ items. Since the discrepancy is more than 0, there may be $MaxColor$ items remaining. The remainder of the packing depends on whether L is even or odd.

We first note that when the discrepancy of the remaining items is more than zero, packing as many $MaxColor$ items as possible into a bin can never yield a sub-optimal solution. This is because the only reason a $MaxColor$ item should be saved for another bin is if it is needed for packing between two $OtherColors$ items. However, this will not happen if there are more $MaxColor$ items than $OtherColors$ items. We use this fact to prove the optimality of our algorithms when discrepancy is more than zero.

- L is even.
  Since $D > 0$ we may end up with single-item bins containing a $MaxColor$ item; we refer to these as *M-bins*. Since $D > 0$, we pack each bin with as many $MaxColor$

items as possible (i.e. $L/2$). To reduce the number of M-bins, we save $OtherColors$ items from some bins. The number of $OtherColors$ items we save is essential - if we save too few, then we may end up with unnecessary M-bins; if we save too many, then we may end up with $OtherColors$ items of the same color that would need to be packed into separate bins. From Lemma 1, we know that if we save one $OtherColors$ item from each of D bins, then the discrepancy of remaining items will be at most zero. Therefore we pack at most D bins this way and refer to this as the *initial packing.* We note that there may not be enough $OtherColors$ items to pack D such bins and in this case, the packing will inevitably yield some M-bins. In particular, if $OtherCount < D(L/2 - 1)$, then we will pack fewer than D such bins, there will be $MaxColor$ items remaining, and each of these will be packed in an M-bin. If $OtherCount \geq D(L/2 - 1)$ then we can pack D such bins and the discrepancy reduces to zero. We then apply the modified ALTERNATE-ZERO to pack the remaining items. The bins packed during the initial packing contain as many $MaxColor$ items as possible, and we know from Lemma 1, they also contain the optimal number of $OtherColors$ items. From Theorem 1, we know the bins packed using ALTERNATE-ZERO are optimally packed. Therefore the entire packing is optimal.
As an example, if we are given 11 W, 3 B, and 3 Y items, and $L = 6$, then $D = 5$ and $OtherCount < D(L/2 - 1)$. An optimal packing is WBWBW / WBWYW / WYWYW / W / W.

If we are given 11 W, 6 B, and 3 Y items, and $L = 6$, then $D = 2$ and $OtherCount \geq D(L/2-1)$. An optimal packing is WBWBW / WBWBW / BWBWYW / YWYW.

- L is odd.
  We begin by packing bins full, starting with a $MaxColor$ item, alternating between $OtherColors$ and $MaxColor$ items, and topping with a $MaxColor$ item. We refer to this as the *initial packing.* Since L is odd each bin will contain one more $MaxColor$ item than $OtherColors$ item. From Lemma 1 we know that D reduces to zero if we are able to pack D bins this way. As in the even case, if $OtherCount < D\lfloor L/2 \rfloor$, then we will pack fewer than D such bins, there will be $MaxColor$ items remaining, and each of these will be packed in an M-bin. If $OtherCount \geq D\lfloor L/2 \rfloor$ then we can pack D such bins and the discrepancy reduces to zero. We then apply the modified ALTERNATE-ZERO to pack the remaining items. For the same reason as in the even case, this yields an optimal packing.
  As an example, if we are given 9 W, 3 B, and 3 Y items, and $L = 7$, then $D = 3$, so $OtherCount < D\lfloor L/2 \rfloor$. An optimal packing is: WBWBWBW / WBWBWBW / W.

  If we are given 7 W, 3 B, and 3 Y items, and $L = 7$, then $D = 1$, so $OtherCount \geq D\lfloor L/2 \rfloor$. An optimal packing is: WBWBWBW / YWYWYW.

□

3.1 Two Pass Algorithms

Our algorithms use exactly two passes over the set of input items. Specifically, in the first pass, the algorithms separate the items into subsets of items of the same color while maintaining $OtherColors$, $MaxCount$, $OtherCount$ and determining $MaxColor$. This process requires examining each item exactly once. It then performs the packing using exactly one pass over each input item.

4 Number of Bins

In this section, we provide closed-form expressions for the optimal number of bins. The zero-weight case is straightforward: if the discrepancy D is no more than 0, then one bin is needed; otherwise D bins are needed. For the unit-weight case we consider three sub-cases.

1. $D \leq 0$: In this case there are enough $OtherColors$ items to pack all the $MaxColor$ items such that there will be no M-bins. The number of bins depends on the bin capacity: specifically there will be $\lceil n/L \rceil$ bins.
2. $D > 0$ and L is even:
   We consider two sub-cases based on $OtherCount$:
   - $OtherCount < D(L/2 - 1)$:
     In this case, we will run out of $OtherColors$ items before packing D bins. In the initial packing we pack $\lceil OtherCount/(L/2-1) \rceil$ bins that contain $L/2$ $MaxColor$ and $L/2{-}1$ $OtherColors$ items, so we pack $\lceil OtherCount/(L/2-1) \rceil$ more $MaxColor$ items than $OtherColors$ items during the initial packing. Therefore the initial packing also yields $MaxCount{-}OtherCount{-}\lceil OtherCount/(L/2-1) \rceil$ $MaxColor$ items that are each packed in an M-bin. Therefore, the total number of bins is $\lceil OtherCount/(L/2-1) \rceil + MaxCount - OtherCount - \lceil OtherCount/(L/2-1) \rceil = D$.
   - $OtherCount \geq D(L/2 - 1)$:
     In this case, there are enough $OtherColors$ items such that packing D bins with $L/2$ $MaxColor$ items and $L/2 - 1$ $OtherColors$ items will eventually reduce D to zero, so there will be no M-bins. Each of these D bins will contain $L - 1$ items so there will be $n - D(L - 1)$ items remaining after D bins are packed and we will need $\left\lceil \frac{n-D(L-1)}{L} \right\rceil$ bins for these items, so the total number of bins is $D + \left\lceil \frac{n-D(L-1)}{L} \right\rceil$.
3. $D > 0$ and L is odd:
   Again we consider two sub-cases based on $OtherCount$:
   - $OtherCount < D(L/2 - 1)$:
     This case is identical to the even case, so the total number of bins is D.
   - $OtherCount \geq D(L/2 - 1)$:
     This case is similar to the even case - the only difference is that each of the D bins will contain L (instead of $L - 1$) items. So the total number of bins is $D + \lceil \frac{n-DL}{L} \rceil$.

References

1. J. Balogh, J. Bekesi, G. Dosa, H. Kellerer and Z. Tuza. Black and White Bin Packing. *Approximation and Online Algorithms, 10th International Workshop, WAOA, Revised Selected Papers*, pp. 131 - 144, 2012.
2. M. Bohm, J. Sgall and P. Vesely. Online Colored Bin Packing. *Approximation and Online Algorithms, 12th International Workshop, WAOA 2014, Revised Selected Papers*, pp. 35 - 45, 2014.
3. M. Dawande, J. Kalagnanam and J. Sethuraman. Variable Sized Bin Packing with Color Constraints. *IElectronic Notes in Discrete Mathematics, Brazilian Symposium on Graphs, Algorithms and Combinatorics*, vol. 7, pp. 154-157, 2001.
4. G. Dosa and L. Epstein. Colorful Bin Packing. *Algorithm Theory, SWAT*, pp. 170-181, 2014.
5. L. Epstein, C. Imreh, and A. Levin. Class Constrained Bin Packing Revisited. *Theoretical Computer Science*, vol. 411, no. 34, pp. 3073-3089, 2010.
6. K. Jansen. An Approximation Scheme for Bin Packing with Conflicts. *Journal of Combinatorial Optimization*, vol. 3, no. 4, pp. 363-377, 1999.
7. Y. Manoussakis. Alternating Paths in Edge-Colored Complete Graphs. *Discrete Applied Mathematics*, vol. 56, no. 2, pp. 297-309, 1995.
8. Y. Manoussakis. Personal communication, April 2016.
9. A. E. F. Muritiba, M. Iori, E. Malaguti, and P. Toth. Algorithms for the Bin Packing Problem with Conflicts. *INFORMS Journal on Computing*, vol. 22, no. 3, pp. 401-415, 2010.
10. Y. Oh and S. H. Son. On a Constrained Bin Packing Problem. *Technical report CS-95-14. Department of Computer Science, University of Virginia, VA*, 1995.
11. S. Seiden. On the Online Bin Packing Problem . *Journal of the ACM (JACM)* , pp. 640-671, 2002.
12. E. Xavier and F. K. Miyazawa. The Class Constrained Bin Packing Problem with Applications to Video-on-Demand. *Theoretical Computer Science*, vol. 393, no 1, pp. 240-259, 2008.