On the Monotonicity of Certain Bin Packing Algorithms

Frank D. Murgolo

Technical Report No. 244

December 1984

Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92717

Supported by National Science Foundation Grant MCS 79-04997

On The Monotonicity of Certain Bin Packing Algorithms

Frank D. Murgolo

Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92717

ABSTRACT

This paper examines the monotonicity of the approximation bin packing algorithms Worst-Fit (WF), Worst-Fit Decreasing (WFD), Best-Fit (BF), Best-Fit Decreasing (BFD), and Next-Fit-k (NF-k). Let X and Y be two sets of items such that the set X can be derived from the set Y by possibly deleting some members of Y or by reducing the size of some members of Y. If an algorithm never uses more bins to pack X than it uses to pack Y we say that algorithm is monotonic. It is shown that NF and NF-2 are monotonic. It was already known that First-Fit and First-Fit Decreasing were non-monotonic and we give examples which show BF, BFD, WF, and WFD also suffer from this anomaly. One may consider First-Fit as the limiting case of NF-k. We notice that NF-1 is monotonic while First-Fit is not, suggesting there exists some critical k for which NF-k' is monotonic, for $k' \leq k$, while NF-k' is not monotonic for any $k' > k$. We establish that this is indeed the case and determine that critical k. An upper bound on the non-monotonicity of selected algorithms is also provided.

December 7, 1984

Supported by National Science Foundation Grant MCS 79-04997

On The Monotonicity of Certain Bin Packing Algorithms

Frank D. Murgolo

Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92717

1. Introduction

It has become common when working with approximation algorithms for combinatorial optimization problems to analyze the performance ratio of the algorithm under study. If we let $AL(I)$ be the measure of the performance of a given algorithm for an instance I of a particular combinatorial optimization problem and $OPT(I)$ be the measure of an optimal algorithm for the same instance then the ratio of $AL(I)$ to $OPT(I)$ provides us with an indicator of the quality of the given algorithm.

It has been shown that certain algorithms, although having comparatively good performance ratios, possess the undesirable property of performing "worse" when their inputs and/or constraints are made "better". For example, in the common bin-packing problem we are given a list of items L and asked to pack them into as few bins as possible. We form a new list L' by deleting some elements of L and/or reducing the size of some elements of L. If an algorithm never uses more bins to pack L' than it uses to pack L we say that algorithm is monotonic, otherwise we say the algorithm is non-monotonic. It could be important to know that an algorithm has the characteristic of not performing better when seemingly favorable changes are made to its inputs.

In [Gr66] it is shown that certain multiprocessor scheduling algorithms suffer from this anomaly. There it is shown that by changing the scheduling constraints for a list of tasks in an apparently favorable fashion, i.e. decreasing the time required for each task to complete or increasing the number of processors, we may actually increase the time it takes for the task list to be completed. In [Gr76] an example is given in which the bin packing algorithm First-Fit Decreasing uses fewer bins to pack a set L of items than it uses to pack a subset of L.

In this paper we classify other bin packing algorithms showing Worst-Fit, Worst-Fit Decreasing, Best Fit, and Best-Fit Decreasing are not monotonic. We show that a simple packing scheme NF-k is either monotonic or non-monotonic,

Supported by National Science Foundation Grant MCS 79-04997

depending on k. We provide upper bounds on the anomalous behavior of First-Fit Decreasing and Best-Fit Decreasing. We also provide an upper bound for a particular class of bin packing algorithms. This class includes all of the commonly studied approximation algorithms.

2. Definitions

1. Let $Y = \{y_1, y_2, \ldots, y_n\}$, $X = \{x_1, x_2, \ldots, x_m\}$, with $s(x_i) > 0$, $s(y_i) > 0$, $s(y_i) \geq s(y_{i+1})$, $s(x_i) \geq s(x_{i+1})$. The set Y ***dominates*** the set X if $n \geq m$ and $s(y_i) \geq s(x_i)$, $1 \leq i \leq m$.
2. x_i, y_i are the i^{th} items of X and Y, respectively.
3. $s(x_i)$ is the size of x_i. We assume $s(x_i) \in (0,1]$.
4. $B_{X,i}$ is the i^{th} bin allocated by the packing algorithm in packing X. Allocated bins are thought of as appearing left to right in order of allocation. This may be abbreviated as B_i where the set being packed is clear from context.
5. $x_i \rightarrow j$ means item x_i is packed into $B_{X,j}$.
6. *Time t of a packing* is that time in the packing when items y_{t-1} and x_{t-1} have been packed but y_t and x_t have not. Time 1 is prior to any items being packed and time $n+1$ is after all items have been packed.
7. The space remaining in $B_{X,i}$ at time t is:

$$REM^{t}(B_{X,i}) = 1 - \sum_{k: x_k \rightarrow i \text{ and } k<t} s(x_k)$$

   The superscript will be omitted if the above property is to hold for all times t of a packing.
8. $AL(X)$ is the number of bins used by bin packing algorithm AL to pack the set X.
9. A packing algorithm is ***reasonable*** if it never packs an item into an empty bin if the item can fit into a non-empty bin.

All definitions pertaining to X and x_i also apply to Y and y_i.

The bin packing problem we consider can be stated as follows: given a list L of piece sizes in (0,1] place the pieces into a minimum number of bins so that in no bin does the sum of the piece sizes exceed 1.

The First-Fit Decreasing Algorithm may be described as follows: let B_1, B_2, ..., be the bins available for packing a set of pieces X. Arrange the pieces X into non-increasing order of piece size and then pack the pieces in order according to the following rule. Pack x_i into the least j such that $REM^i(B_j) \geq s(x_i)$.

Best-Fit packs piece x_i in the following way: find that j such that $REM^i(B_j) \geq s(x_i)$ and for all bins k with $REM^i(B_k) \geq s(x_i)$ we have $REM^i(B_k) \geq REM^i(B_j)$. If j is not unique choose the least j. Pack x_i into B_j.

Best-Fit Decreasing uses the same rules as Best-Fit with the additional step of first placing its input into non-increasing order.

Worst-Fit Decreasing orders its input in the same way as FFD but it chooses that j for which $REM^i(B_j) \geq s(x_i)$ and for all bins k with $REM^i(B_k) \geq s(x_i)$ we have $REM^i(B_k) \leq REM^i(B_j)$. If no such j exists a new bin is allocated to pack x_i.

Next-Fit can be described in the following way: let X be a list of items to be packed. Pack x_1 into a new bin. If x_{i+1} can fit into the same bin as x_i pack it there, otherwise pack x_{i+1} into a new bin.

3. The Monotonicity of Some Algorithms

We exhibit a function which will aid us in examining the monotonicity of several of the algorithms examined. Let AL be a reasonable bin packing algorithm. We compare how AL performs on two sets of items X and Y with Y dominating X by imagining AL packing items of Y and X alternately and then examining the space remaining in the non-empty bins.

Let $S = \{1,2,...,AL(Y)\}$ and define the function $g:S \rightarrow S$ initially as $g(i)=i$ for all $i \in S$. We will say that AL ***supports*** g ***in monotonicity,*** or simply AL ***supports*** g, if at all times t in the packings g can be maintained so the following conditions are true:

(i) $g:S \rightarrow S$ and,

(ii) For all j, $\sum_{i:g(i)=j} REM(B_{Y,i}) \leq REM(B_{X,j})$

Lemma 1: Let X and Y be two sets of items with Y dominating X which are packed by the reasonable algorithm AL. If AL supports g then $AL(X) \leq AL(Y)$.

proof

Because AL supports g we know that for all times t of the packings if y_t is packed into $B_{Y,i}$ then $REM^t(B_{X,g(i)}) \geq REM^t(B_{Y,i}) \geq s(y_t)$. Because Y dominates X we further know $REM^t(B_{X,g(i)}) \geq s(x_t)$. Therefore since AL is reasonable x_t will be packed into a bin numbered no higher than $AL(Y)$.

3.1. Next-Fit

Theorem 1: Let X and Y be two sets of items that satisfy the constraint Y dominates X. Then $NF(X) \leq NF(Y)$.

proof

We start by proving the following

Claim: If NF packs item y_t into $B_{Y,m}$ then NF will pack item x_t into $B_{X,m'}$ with $m' \leq m$.

proof of claim (We use an induction on the items)
Basis: i=1 When x_1 and y_1 are packed they are each placed into new bins so the claim is clearly true.

Inductive step: Assuming the claim is true for all items $\{x_1,x_2,...,x_{t-1}\}$, $\{y_1,y_2,...,y_{t-1}\}$ it is also true for y_t and x_t.

case.i) $y_{t-1} \epsilon B_{Y,j}$, $x_{t-1} \epsilon B_{X,k}$ with $k < j$.

Here even if $x_t \rightarrow B_{X,k+1}$ we still have $k+1 \leq j$ so the claim is true.

case.ii) $y_{t-1} \epsilon B_{Y,j}$, $x_{t-1} \epsilon B_{X,j}$.

Let the items in $B_{X,j}$ be, in order from bottom to top, $(x_{t-k},x_{t-k+1},...,x_{t-1})$. How many items can be in $B_{Y,j}$?

ii.a) $B_{Y,j}$ contains k or more items.
By our assumption that Y dominates X and the way that NF works we know $\sum_{y \epsilon B_{Y,j}} s(y) \geq \sum_{x \epsilon B_{X,j}} s(x)$. Because $s(y_t) \geq s(x_t)$, if y_t is packed into $B_{Y,j}$ then x_t can fit into $B_{X,j}$.

ii.b) $B_{Y,j}$ contains fewer than k items.
This tells us that at time $t-k$ item y_{t-k} was packed into $B_{Y,j'}$, $j' < j$, but item x_{t-k} was packed into $B_{X,j}$; this is a contradiction of the inductive hypothesis. Therefore, $B_{Y,j}$ can not contain fewer than k items and the claim is proved.

The theorem follows immediately.

3.2. Next-Fit k

We have shown that NF is monotonic. One can generalize NF to NF-k where the algorithm is allowed to pack an item into any of the last k allocated bins, examining the bins in increasing order of allocation. NF then becomes NF-1. The limiting case of NF-k is the algorithm FF which was shown to be non-

monotonic indicating that there exists a ***critical k*** such that *NF*-k is monotonic and *NF*-k+1 is not. We determine that critical k.

Theorem 2: *NF*-2 is monotonic.

proof

Claim: *NF*-2 supports g.

proof of claim

To show this we show that we can maintain the function g on the last 2 allocated bins in a *NF*-2 packing. Since these are the only bins whose configurations can influence whether or not an item causes a new bin to be allocated, this restricted g mapping is sufficient for proving the result. We assume at least two bins are necessary to pack Y and initially define $g(1)=1$, $g(2)=2$. (If the Y items can be packed into only one bin by *NF*-2 then clearly so can the X items and the proof is complete.)

BASIS: Pieces y_1 and x_1 are packed.

Since $y_1 \rightarrow B_{Y,1}$ and $x_1 \rightarrow B_{X,1}$ we leave g defined as is.

Induction step: Assume g is maintained on bins $k-1$ and k after items y_{t-1} and x_{t-1} have been packed. We show that g continues to be maintained after y_t and x_t have been packed. (For our analysis we allocate a new empty bin for X each time a new bin is allocated by *NF*-2 for Y and maintain g between the last two allocated bins of X and Y even though the last two artificially allocated bins of X may not be used by *NF*-2 in packing X. If *NF*-2 packs an X item into an X bin of index less than $k-1$ we can ignore it in our analysis because g can be maintained by being left alone. As a notational convenience we set $i=k-1$, $j=k$ and let $r_{\alpha,\lambda}$ denote $REM^{\,t}(bin_{\alpha,\lambda})$. Under the initial configuration for g shown in Figure 1 we have the 4 following possible cases:

1) $y_t \rightarrow B_{Y,i}$
2) $y_t \rightarrow B_{Y,j}$, $x_t \rightarrow B_{X,g(j)}$
3) $y_t \rightarrow B_{Y,j}$, $x_t \rightarrow B_{X,g(i)}$ **and** $r_{Y,i} < r_{Y,j}$
4) $y_t \rightarrow B_{Y,j}$, $x_t \rightarrow B_{X,g(i)}$ **and** $r_{Y,i} \geq r_{Y,j}$

The analysis of these cases shows they may give rise to only one new g configuration, (see Figure 1.) Under this new g configuration we have two possible occurrences:

5) $x_t \rightarrow B_{X,i}$
6) $x_t \rightarrow B_{X,j}$

recalling $i=k-1$, $j=k$.

Finally, under either g configuration we have the case of $NF{-}2$ allocating a new bin for y_t:

7) $y_t \rightarrow B_{Y,j+1}$

We now describe the maintenance of g in each of the 7 cases.

Case 1: $y_t \rightarrow B_{Y,i}$

We know $r_{X,g(i)} \geq r_{Y,i}$, $s(y_t) \geq s(x_t)$ and $r_{Y,i} \geq s(y_t)$. Together this implies $r_{X,g(i)} \geq s(x_t)$, so NF-2 will pack x_t into $B_{X,g(i)}$. Because $s(y_t) \geq s(x_t)$ we know $r_{Y,i} - s(y_t) \leq r_{X,g(i)} - s(x_t)$, so no modification of g is necessary.

Case 2: $y_t \rightarrow B_{Y,j}$, $x_t \rightarrow B_{X,g(j)}$

This case is very similar to case 1.

Case 3: $y_t \rightarrow B_{Y,j}$, $x_t \rightarrow B_{X,g(i)}$ **and** $r_{Y,i} < r_{Y,j}$

Because y_t was packed into $B_{Y,j}$ we know $(*)$ $s(y_t) > r_{Y,i}$. We also have $r_{X,g(j)} \geq r_{Y,j}$ which implies $r_{X,g(j)} - s(y_t) \geq r_{Y,j} - s(y_t)$. Therefore, we can take away an amount of space equal to $s(y_t)$ from $r_{X,g(j)}$ and still have g properly defined for bins $B_{Y,j}$ and $B_{X,g(j)}$. In addition, by $(*)$ we know that we can set $g(i) \leftarrow g(j)$ and have g properly defined for bins $B_{Y,i}$ and $B_{X,g(i)}$. Thus we modify g by setting $g(i) \leftarrow g(j)$ and leaving $g(j)$ unaffected.

Case 4: $y_t \rightarrow B_{Y,j}$, $x_t \rightarrow B_{X,g(i)}$ **and** $r_{Y,i} \geq r_{Y,j}$

The NF-2 rule would not allow $B_{Y,i}$ to be passed over to pack $y_t \rightarrow B_{Y,j}$, therefore this case is not possible.

Cases 5 and 6 consider the g configuration created by step 3.

Case 5: $x_t \rightarrow B_{X,i}$

Clearly, since neither $g(i)$ nor $g(j)$ is affected by this packing we need not modify g in order to maintain it.

Case 6: $x_t \rightarrow B_{X,j}$

Here we see that x_t is **always** packed into the X bin that functionally corresponds to the Y bin that y_t was packed in. So again no modification is necessary.

Case 7: $y_t \rightarrow B_{Y,j+1}$

We know at that time $g(k) = k$ and either $g(k{-}1) = k{-}1$ or $g(k{-}1) = k$ which means we do not need the remaining space of $B_{X,k-1}$ to maintain g when bins k and $k{+}1$ are the last two bins allocated. That is important because when

bins k and $k+1$ are the last two bins allocated the space remaining in $B_{X,k-1}$ is unavailable for the maintenance of g. To maintain g we do the following: if $x_t \rightarrow B_{X,k+1}$ set $g(k+1) \leftarrow k+1$; if $x_t \rightarrow B_{X,k}$ we set $g(k+1) \leftarrow k+1$ and change $g(k)$ to $k+1$. This works for reasons similar to those described in Case 3. Since this does not introduce any new g configurations the proof of the lemma is complete. The theorem is derived immediately from Lemma 1.

To show that *NF*-k for $k>2$ is non-monotonic we start with the example on p. 67 of [Jo73]. This example shows the First-Fit heuristic can use three bins to pack a set of items Y and four bins to pack a set X even though Y dominates X. Because *NF*-3 only allows packing an item into the last three allocated bins of a packing using a First-Fit strategy the example shows the non-monotonicity of *NF*-3. We can extend that example to lists of arbitrary length and arbitrary $k>3$ by prefixing any number of items of size equal to the bin size to the items of the original example.

3.3. Best-Fit and Best-Fit Decreasing

When proving *NF*-2 monotonic we were able to maintain g as a well-defined function satisfying the requirements necessary to support that algorithm in monotonicity. We now show how one can use the function g to construct an example which shows an algorithm is not monotonic.

Examining the conditions that must be maintained if an algorithm is to support g in monotonicity, we make the following observation. Assume item y_t is packed into $B_{Y,i}$ and item x_t is packed into $B_{X,g(j)}$ with $r_{Y,i}>r_{Y,j}$ and $r_{X,g(i)}>r_{X,g(j)}$. It might be that $B_{X,g(j)}$ is completely filled after x_t is packed thereby requiring us to use the space in $B_{X,g(i)}$ to accommodate both $B_{Y,i}$ and $B_{Y,j}$ in maintaining g. We can do that because we know by the way *BFD* works, that $s(y_t) \geq r_{Y,j}$. This enables us to use the space in $B_{X,g(i)}$ to cover the space in both $B_{Y,j}$ and the space left in $B_{Y,i}$ after y_t has been packed. (See Figure 2.)

This leaves us with a properly maintained g which is not 1-1. In analyzing the bin configurations which may arise under this scenario we find cases where g can not be properly maintained. In particular this occurs in the following example. (See Figure 3.)

Item y_t is packed into $B_{Y,j}$, x_t is packed into $B_{X,g(i)}$ with $r_{Y,i}>r_{Y,j}$ and $r_{X,g(i)}<r_{X,g(j)}$. If $B_{X,g(j)}$ were the image of only $B_{Y,j}$, which would be true if g were 1-1, then one could interchange $g(i)$ and $g(j)$ and continue to have g support *BFD*. However, because g is not 1-1, $r_{X,g(j)}$ could be due to several Y bins mapping to $B_{X,g(j)}$ with the amount of space in $B_{X,g(j)}$ which is assigned to the space in $B_{Y,j}$ being of size less than $r_{Y,i}$. We do know that an amount of space equal to $s(y_t)$ in $B_{X,g(j)}$ is available for re-mapping and that this space plus the space left

in $B_{X,g(i)}$ is sufficient to cover the space in $B_{Y,i}$. Intuitively we know this because the packing of x_t and y_t removes an amount of space equal to $s(x_t)$ from $g(B_{Y,i})$ while simultaneously adding an amount of space equal to $s(y_t)$. However, the bin it adds to is different from the bin from which it subtracted.

We also know the segment of space in $B_{X,g(j)}$ assigned to $B_{Y,i}$ after this remapping is large enough to accommodate the next item to be packed, x_{t+1}, because the input is in non-increasing order of size. These observations suggest that one might change the conditions for g to support *BFD* in monotonicity by incorporating slightly weaker conditions which make use of the non-increasing order of the input.

This would be of little use however, since the conditions which result from the aforementioned example can be exploited to achieve the following anomaly:

$$Y=\{.7,\ .68,\ .5399,\ .3201,\ .15,\ .14,\ .08(x5),\ .07\}$$

$$X=\{.7,\ .68,\ .5399,\ .32,\ .15,\ .14,\ .08(x5),\ .07\}.$$

The notation $.08(x5)$ means 5 items of size .08. Here we see *BFD* uses 3 bins to pack Y and 4 bins to pack X even though Y dominates X. (See Figure 4.)

In addition to the above example we provide a simple example showing *BF*'s non-monotonicity.

$$Y=\{8/12,\ 6/12,\ 5/12,\ 8/12,\ 1/12,\ 4/12,\ 4/12\}$$

$$X=\{6/12,\ 6/12,\ 5/12,\ 8/12,\ 1/12,\ 4/12,\ 4/12\}.$$

BF uses three bins to pack Y and four bins to pack X. (See Figure 5.)

3.4. Worst-Fit Decreasing and Worst-Fit

When creating item sets to cause an algorithm to be non-monotonic, it is useful to choose item sizes that cause g to be not 1-1. Although this is not always sufficient, as we saw in *NF*-2, we find it useful in causing *WFD* to be non-monotonic. Consider the following item sets:

$$Y=\{.6,\ .55,\ .451,\ .35,\ .25,\ .2,\ .2,\ .2,\ .199\}$$

$$X=\{.58,\ .55,\ .4,\ .35,\ .25,\ .2,\ .2,\ .2,\ .199\}$$

After items x_3 and y_3 have been packed g is not 1-1, i.e. $g(3)=3$ and $g(2)=3$. Once g is no longer 1-1 it is possible to cause a packing where g is no longer maintainable. As shown in Figure 6, *WFD* uses three bins to pack Y while using four bins to pack X. This example also shows that *WF* is non-monotonic.

NF-2 is monotonic even though g is not maintained in a 1-1 fashion. This suggests that if we restrict a reasonable algorithm to considering only the last two allocated bins we can cause the algorithm to be monotonic. This hypothesis is not true. If we replace the *FF* packing heuristic of *NF*-2 with either the *BF* or *WF* heuristic we obtain a non-monotonic algorithm. It is the packing rule in

conjunction with the special set of bins considered which causes *NF*-2 to be monotonic.

4. Bounds for FFD, BFD and Any Reasonable Algorithm

Theorem 3.1: If Y dominates X then $FFD(X) \leq \frac{11}{9} FFD(Y)+4$.

proof

Clearly $OPT(X) \leq OPT(Y) \leq FFD(Y)$, from which we derive

$$\frac{11}{9} OPT(X)+4 \leq \frac{11}{9} OPT(Y)+4 \leq \frac{11}{9} FFD(Y)+4 \qquad (1)$$

By results in [Jo73] we have $FFD(X) \leq \frac{11}{9} OPT(X)+4$ which together with (1) proves the theorem.

Theorem 3.2: If Y dominates X then $BFD(X) \leq \frac{11}{9} BFD(Y)+4$.

proof

Same as Theorem 3.1 except replace *FFD* with *BFD*.

Theorem 4: If Y dominates X and RA is any reasonable algorithm then $RA(X) \leq \frac{3}{2} RA(Y)$.

proof

Let $k = RA(Y)$ and $k' = RA(X)$ with $k' > k$. Define X_{ADD} as the set of X items that are packed into $B_{X,k+1}, B_{X,k+2}, \cdots, B_{X,k'}$; $r_{\max} = \max \{REM(B_{X,i})\}$, $1 \leq i \leq k$; $REM(X) = \sum_{i=1}^{k} REM(B_{X,i})$, and $REM(Y) = \sum_{i=1}^{k} REM(B_{Y,i})$.

Because Y dominates X we know that for all $x \epsilon X_{ADD}$, $s(x_i) \leq \frac{1}{2}$. This gives an immediate bound of

$$RA(X) \leq RA(Y) + \frac{|X_{ADD}|}{2} \qquad (*)$$

Claim: $|X_{ADD}| \leq RA(Y)$

proof of claim

Assume not. Because each $x_i \epsilon X_{ADD}$ has $s(x_i) > r_{\max}$, we know:

$$\sum_{x_i \epsilon X_{ADD}} s(x_i) > \sum_{x_i \epsilon X_{ADD}} r_{max}. \qquad (1)$$

By assumption: $|X_{ADD}| > k$ which gives us

$$\sum_{x_i \epsilon X_{ADD}} r_{\max} > \sum_{i=1}^{k} r_{\max} \geq REM(X). \tag{2}$$

Together, (1) & (2) imply:

$$\sum_{x_i \epsilon X_{ADD}} s(x_i) > REM(X). \tag{3}$$

Now observe:

$$\sum_{x_i \epsilon X} s(x_i) = k - REM(X) + \sum_{x_i \epsilon X_{ADD}} s(x_i) \tag{4}$$

Together, (3) & (4) imply:

$$\sum_{x_i \epsilon X} s(x_i) > k - REM(X) + REM(X) = k \geq \sum_{y_j \epsilon Y} s(y_j).$$

$$or \quad \sum_{x_i \epsilon X} s(x_i) > \sum_{y_j \epsilon Y} s(y_j).$$

This contradicts the fact that Y dominates X and completes the proof of the claim.

Substituting into equation (*) gives us the theorem.

Theorem 5: Define $\max\{Y\} = \max\{s(y) : y \epsilon Y\}$, $\max\{X\} = \max\{s(x) : x \epsilon X\}$. Let X, Y, RA, $RA(X)$, and $RA(Y)$ be as in theorem 4. If $\max\{X\} \leq r$, $r \leq 1$, then $RA(X) \leq \frac{1}{1-r} RA(Y) + 1$.

proof

We first observe:

$$\sum_{x_i \epsilon X} x_i > (1-r)(RA(X)-1). \tag{1}$$

Certainly $RA(Y) \geq \sum_{y_i \epsilon Y} y_i$, which taken with equation (1) and the fact Y dominates X implies:

$$RA(Y) \geq (1-r)(RA(X)-1).$$

Or,

$$RA(X) \leq \frac{RA(Y)}{(1-r)} + 1.$$

This bound is only interesting for small values of r.

Theorem 6: (A bound on performance ratios): $RA(X) \leq 2OPT(X) + 1$.

proof

In a packing of X by RA an item $x_i \epsilon X$ will not be placed into a new bin, B_j, if it can fit into B_{j-1}. Therefore, if x_i is placed into a new bin we know that $\sum_{x' \epsilon B_{j-1}} s(x') + s(x_i) > 1$. This inequality is true for all pairs of successive bins; taking into account the possibility of $RA(X)$ odd we have:

$$\sum_{x \epsilon X} s(x) \geq \frac{RA(X)-1}{2} \tag{1}$$

Clearly, $OPT(X) \geq \sum_{x \epsilon X} s(x)$ which together with (1) yields the result.

5. Further Work

The author intends to strengthen the bounds for *FFD* and study the monotonicity of other approximation algorithms for combinatorial optimization problems to see how they perform on sets of inputs that are related by the dominance relation. Besides answering the question for the algorithms studied he hopes to find basic reasons why some algorithms possess their monotonicity and others do not. These reasons may then shed some light on those characteristics of combinatorial optimization problems that enable such anomalies to occur.

6. Acknowledgements

I would like to thank George Lueker for suggesting Theorem 3, and for many other helpful comments and suggestions.

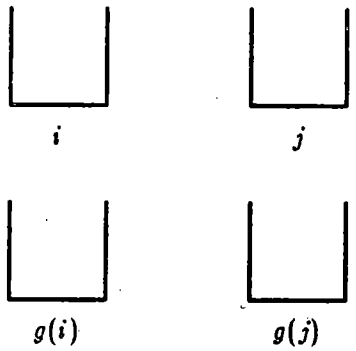
References

[Gr66] R.L. Graham, *Bounds for certain multiprocessing anomalies,* Bell System Technical Journal, 45, no.9 (1966) pp. 1563-1581.

[Gr76] R.L. Graham, *Bounds on Performance of Scheduling Algorithms,* in Computer and Job Shop Scheduling, E.G. Coffman, Jr. (ed.), John Wiley & Sons, Inc. New York, 1976.

[Jo73] D.S. Johnson, Near-optimal bin packing algorithms, Doctoral Thesis, Mass. Inst. of Tech., Cambridge, Mass., 1973

Original *NF*- 2 g Configuration



A g Configuration Created by *NF*- 2

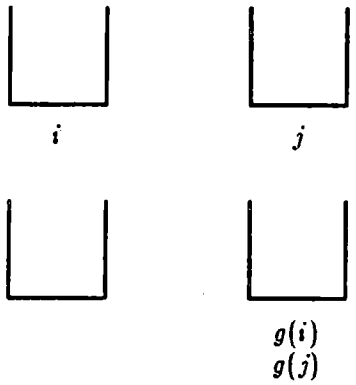
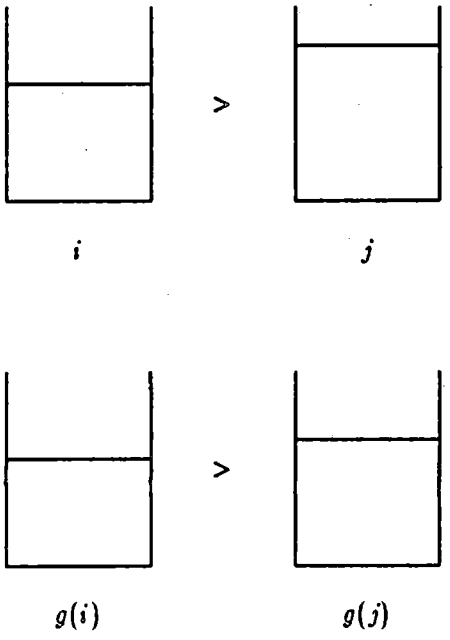


Figure 1.

Causing g to Become not 1– 1

Before y_t and x_t are packed:



After y_t and x_t have been packed:

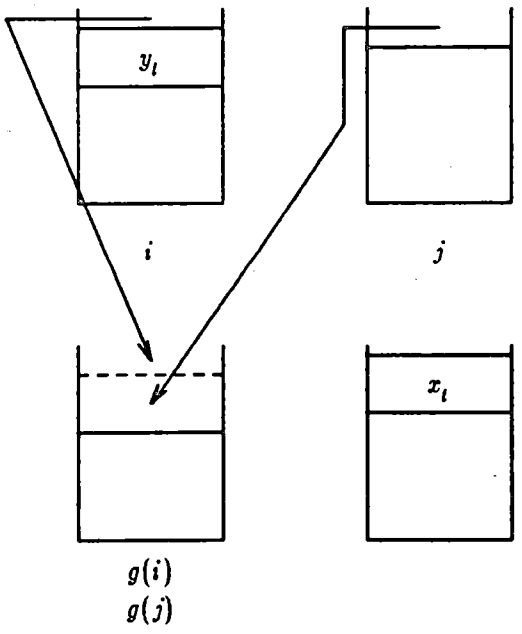


Figure 2.

Causing g to Become not Well-defined

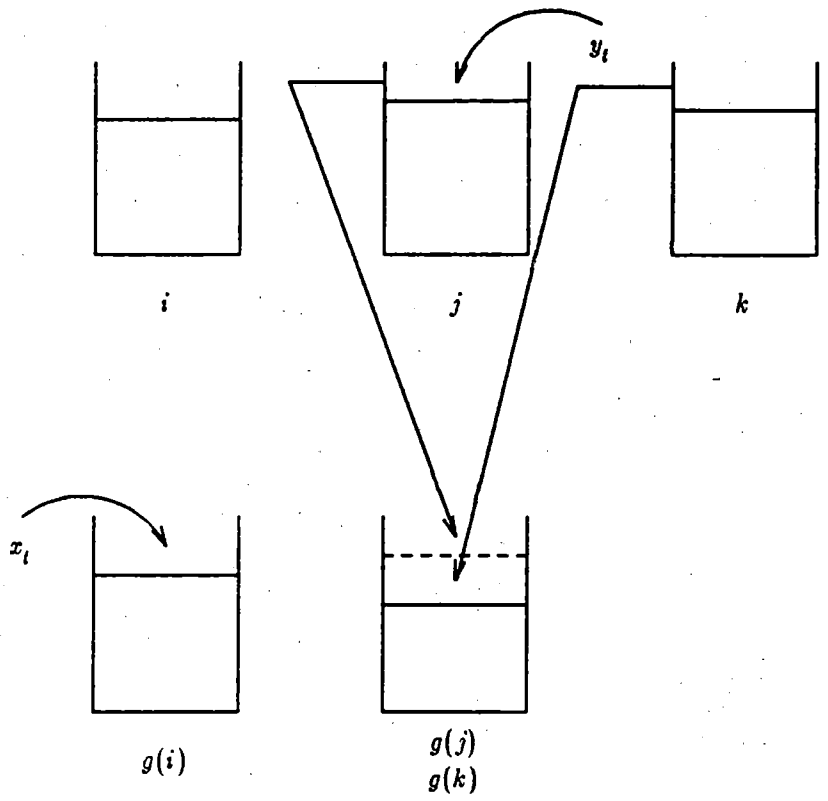
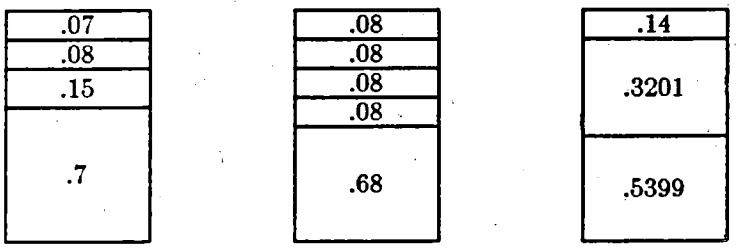


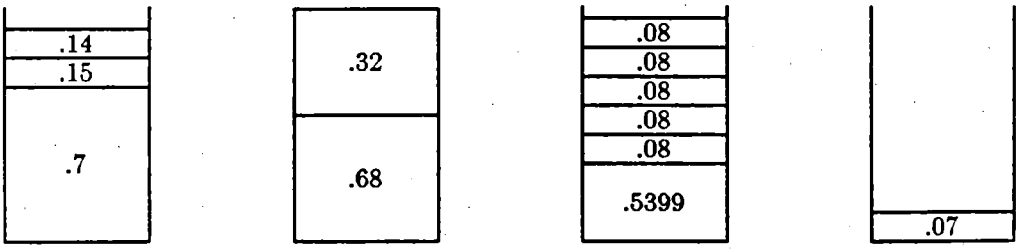
Figure 3.

Non-Monotonicity of BFD

The packing of Y by *BFD*



The packing of X by *BFD*

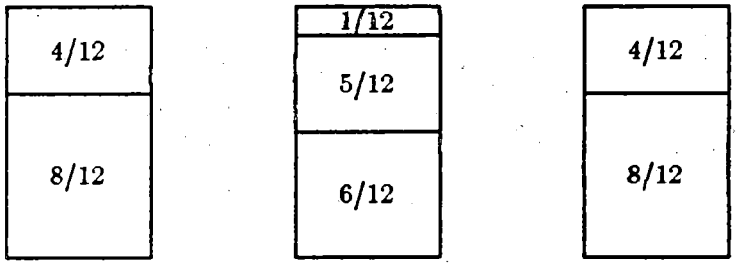


$Y=\{.7, .68, .5399, .3201, .15, .14, .08(x5), .07\}$

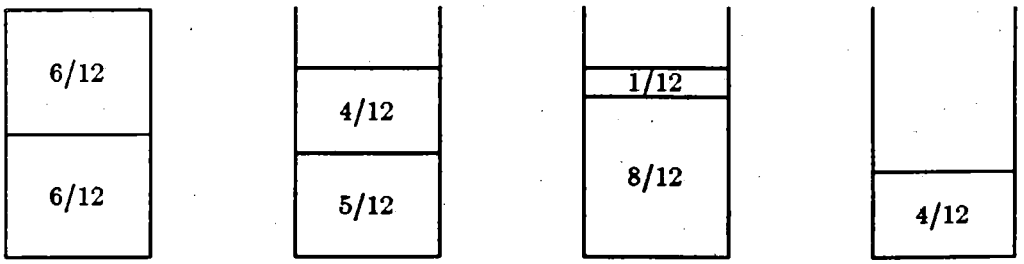
$X=\{.7, .68, .5399, .32, .15, .14, .08(x5), .07\}$

Figure 4.

The packing of Y by BF



The packing of X by BF



Y={8/12, 6/12, 5/12, 8/12, 1/12, 4/12, 4/12}

X={6/12, 6/12, 5/12, 8/12, 1/12, 4/12, 4/12}

Figure 5.