SIAM J. COMPUT.
Vol. 28, No. 2, pp. 574–611

FULLY DYNAMIC ALGORITHMS FOR BIN PACKING: BEING (MOSTLY) MYOPIC HELPS[*]

ZORAN IVKOVIĆ[†] AND ERROL L. LLOYD[‡]

Abstract. The problem of maintaining an approximate solution for *one-dimensional bin packing* when items may arrive and depart dynamically is studied. In accordance with various work on fully dynamic algorithms, and in contrast to prior work on bin packing, it is assumed that the packing may be arbitrarily rearranged to accommodate arriving and departing items. In this context our main result is a *fully dynamic* approximation algorithm for bin packing *MMP* that is $\frac{5}{4}$-competitive and requires $\Theta(\log n)$ time per operation (i.e., for an *Insert* or a *Delete* of an item). This competitive ratio of $\frac{5}{4}$ is nearly as good as that of the best practical off-line algorithms. Our algorithm utilizes the technique (introduced here) whereby the packing of an item is done with a total disregard for already packed items of a smaller size. This *myopic* packing of an item may then cause several smaller items to be repacked (in a similar fashion). With a bit of additional sophistication to avoid certain "bad" cases, the number of items (either individual items or "bundles" of very small items treated as a whole) that needs to be repacked is bounded by a constant.

Key words. bin packing, fully dynamic algorithm

AMS subject classifications. 68P05, 68Q25, 68R05

PII. S0097539794276749

1. Introduction.

In the (one-dimensional) bin packing problem, a list $L = (a_1, a_2, ..., a_n)$ of items of size $size(a_i)$ in the interval (0,1] is given. The goal is to find the minimum k such that all of the items a_i can be packed into k unit-size bins. Bin packing was shown to be *NP*-complete in [15].

For the past quarter century, bin packing has been a central area of research activity in the algorithms and operations research communities (see [3, 7]). Despite its advanced age, bin packing has retained its appeal by being a fertile ground for the study of approximation algorithms (more than a decade ago, bin packing was labeled "the problem that wouldn't go away" [3]). In this paper, we consider *fully dynamic* bin packing, where

- items may arrive and depart from the packing dynamically, and
- items may be moved from bin to bin as the packing is adjusted to accommodate arriving and departing items.

In general, *fully dynamic* algorithms are aimed at situations where the problem instance is changing over time. Fully dynamic algorithms incorporate these incremental changes without any knowledge of the existence and nature of future changes.

Each of the earlier works on *on-line* and *dynamic* bin packing differ from this notion of fully dynamic bin packing in either of two ways: either they do not allow an item to be moved from a bin (of course, this has a predictably bad effect on the achievable quality of the packing), or they restrict themselves to dynamic arrivals of items—there are no departures.

[*]Received by the editors November 7, 1994; accepted for publication (in revised form) September 10, 1996; published electronically July 28, 1998. This research was partially supported by National Science Foundation grant CCR-9120731.

http://www.siam.org/journals/sicomp/28-2/27674.html

[†]School of Management, Yale University, 135 Prospect Street, New Haven, CT 06520 (ivkovich@isis.som.yale.edu).

[‡]Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716 (elloyd@cis.udel.edu).

Although most of the existing work on fully dynamic algorithms has been directed toward problems known to be in P, some recent attention has been paid to fully dynamic *approximation* algorithms for problems that are NP-complete [9, 16].

In this paper we develop a fully dynamic approximation algorithm for bin packing that is "competitive" with existing off-line algorithms. In this case, being competitive with off-line algorithms means that the quality of the approximation produced by the fully dynamic approximation algorithm should be as good as that produced by the off-line algorithms. Further, the running time per operation (i.e., change) of the fully dynamic algorithm should be as small as possible.

1.1. Bin packing—Existing results. The usual measure of the quality of a solution produced by a bin packing algorithm A is its *competitive ratio* $R(A)$ defined as

$$R(A) = \lim_{n\to\infty} \sup_{OPT(L)=n} \frac{A(L)}{OPT(L)},$$

where $A(L)$ and $OPT(L)$ denote, respectively, the number of bins used for packing of the list L by A and some optimal packing of L. Here, we say that A is $R(A)$-*competitive*.

In the domain of off-line algorithms, the value of R has been successively improved [3, 19, 5, 13, 4]. Indeed, it has been shown that for any value of $R \geq 1$, there is an $\mathcal{O}(n \log n)$-time algorithm with that competitive ratio [14]. Unfortunately, the running times for these algorithms involve exceedingly large constants (actually, these "constants" depend on how close R is to 1). Among algorithms of practical importance, the best result is an $\mathcal{O}(n \log n)$ algorithm for which R is $\frac{71}{60}$ [13].

With respect to on-line bin packing, the problem has been defined strictly in terms of arrivals (*Inserts*)—items never depart from the packing (i.e., there are no *Deletes*). Further, most on-line algorithms have operated under the restriction that each item must be packed into some bin, and it should remain in that bin permanently. In this context, it is shown that for every on-line linear time algorithm A, $R(A) \geq 1.536...$ [3]. Further, the upper bound has been improved over the years to roughly 1.6 [10, 11, 12, 17, 18].

The work reported in [6] focused on a variant of on-line bin packing, again supporting *Inserts* only, in which each item may be moved a constant number of times (from one bin to another). Two algorithms were provided: one with a linear running time (linear in n, the number of *Inserts*, which is also the number of items) and a competitive ratio of 1.5, and one with an $\mathcal{O}(n \log n)$ running time and a competitive ratio of $\frac{4}{3}$.

Another notion that is related to, but distinct from, fully dynamic bin packing is dynamic bin packing of [2], where each item is associated with not only its size, but also with an arrival time and a departure time (interpreted in the natural way). Here, again (and unlike [6]), items cannot be moved once they are assigned to some bin, unless they depart from the system permanently (at their departure time). This variant differs from fully dynamic bin packing in that items are not allowed to be moved once they are assigned to a bin and through the departure time information. It was shown in [2] that for any such algorithm A, $R(A) \geq 2.5$, and that for their dynamic first fit (FF), $2.770 \leq R(FF) \leq 2.898$.

1.2. Competitive ratio and running time for fully dynamic approximation algorithms. In this section we discuss the notions of competitiveness and running time in the context of developing fully dynamic approximation algorithms.

We begin by noting that with respect to the definition of *competitive ratio* there is no need to make a distinction between fully dynamic and off-line algorithms. In each case, these measures reflect the size of the packing produced by the algorithm relative to the size of optimal packing.

With respect to running times, we say that a fully dynamic approximation algorithm B for bin packing has *running time* $\mathcal{O}(f(n))$ if the time taken by B to process a change (an *Insert* or *Delete*) to an instance of n items is $\mathcal{O}(f(n))$. If $\mathcal{O}(f(n))$ is a *worst-case* time bound, then B is *uniform.* If $\mathcal{O}(f(n))$ is an *amortized* time bound, then B is *amortized.*

Our general goal in developing fully dynamic approximation algorithms for bin packing is to design algorithms with competitive ratios close to those of the best off-line algorithms such that the changes are processed quickly. Of particular interest are algorithms that are, in a sense, the best possible relative to the existing off-line methods. For bin packing the best known off-line algorithms require time $\Theta(n \log n)$. Thus, a fully dynamic algorithm that runs in time $\Theta(\log n)$ per operation is, in that sense, the best possible. Indeed, the fully dynamic algorithm *MMP* that we give in this paper runs in precisely this time per operation.

The algorithms that we present process a sequence of *Inserts* (arrivals) and *Deletes* (departures) of items. Further, our algorithm is designed to handle "lookup" queries of the following form:

- *size*—returns in $\mathcal{O}(1)$ time the number of bins in the current packing;
- *packing*—returns a description of the packing in the form of a list of pairs $(x, \text{Bin}(x))$, where $\text{Bin}(x)$ denotes the bin into which an item x is packed, in time linear in the number of items in the current instance.

These queries may be interspersed in the *Insert/Delete* sequence as follows.

1.3. What's to come. The main result of this paper is a fully dynamic algorithm *MMP* that is $\frac{5}{4}$-competitive and requires $\Theta(\log n)$ time per operation. Relative to the best off-line algorithms, *MMP* has a running time that is best possible, and it has a competitive ratio that is nearly the equal of the best practical off-line algorithms. This is a surprising result even in terms of off-line bin packing, since it is the first practical bin packing algorithm that has a competitive ratio of less than $\frac{4}{3}$ that does not rely on packing the items in sorted order (as discussed in section 2, dynamically maintaining a packing based on a sorted list is problematic). That the algorithm is fully dynamic is all the more remarkable.

With the preliminaries concluded, the remainder of the paper is organized as follows. In the next section we review the basic definitions and define two key properties of *MMP* packing: *LLS-maximality* and *M-thoroughness.* We further provide a sketch of *MMP*'s *Insert* and *Delete* operations, and we focus on the techniques *MMP* utilizes to maintain the above properties: *myopic packing*, *bundles*, and *LLS-coalitions.*

In section 3 we describe the (rather complex) data structure, and the details of *MMP.* In section 4 we prove the competitive ratio of $\frac{5}{4}$ and the uniform logarithmic running time per *Insert/Delete* operation. Finally, in section 5 we furnish some concluding remarks.

2. Toward full dynamization of bin packing. Motivated by the notions of competitiveness introduced in the preceding section, a natural approach to the development of fully dynamic bin packing algorithms is to adapt existing methods to work in the fully dynamic situation. Unfortunately, this is easier said than done. The difficulty is that most of the off-line algorithms perform bin packing in two distinct stages. First, there is a *preprocessing stage* in which the items are organized in some fashion (this reorganization should have a positive effect on the resulting packing). This is followed by a *packing stage* where the actual packing is accomplished. In the off-line situation, this two stage approach is quite natural since the entire list of items is available at the outset. However, in a dynamic environment a two-stage process becomes awkward. Consider, for example, the algorithm first fit decreasing (*FFD*), which is $\frac{11}{9}$-competitive. This algorithm first sorts the items and then packs them in order of decreasing size using the *FF* packing rule.[1] What about a fully dynamic version of *FFD*? There is, of course, no difficulty in maintaining a sorted list of the elements. *But* there is great difficulty in maintaining the packing based on that sorted list, since the insertion (or deletion) of a single item can result in a large number of changes to that packing. It would seem that the packing induced by the sorted list of items is "too specific" to be maintained dynamically, and that perhaps a less specific rule might be of use. Indeed, in this paper we utilize a weaker notion: Johnson's *grouping* [10, 11].

2.1. Some definitions. Before proceeding, we require a few definitions that will be used throughout the remainder of the paper. In particular, for a bin B, $level(B)$ is the sum of the sizes of the items packed in B; $gap(B)$ is $1 - level(B)$, i.e., the amount of empty space in B; and $content(B)$ is the set of items packed in B.

We can assume that the bins are numbered in such a way that every bin has a unique number with the property that, for any two bins, the bin with the lower number is placed "to the left" of the bin with the higher number. In other words, we assume that, for conceptual purposes, the bins are numbered in increasing order from left to right.

Following Johnson's *grouping* [10, 11], we partition the items according to their respective sizes. In particular, an item a is a B-item (big) if $size(a) \in (\frac{1}{2}, 1]$, an L-item (large) if $size(a) \in (\frac{1}{3}, \frac{1}{2}]$, an S-item (small) if $size(a) \in (\frac{1}{4}, \frac{1}{3}]$, a T-item (tiny) if $size(a) \in (\frac{1}{5}, \frac{1}{4}]$, and an M-item (miniscule) if $size(a) \in (0, \frac{1}{5}]$.

Let $\mathcal{B}$, $\mathcal{L}$, $\mathcal{S}$, $\mathcal{T}$, and $\mathcal{M}$ denote the number of B-items, L-items, S-items, T-items, and M-items in L, respectively.

When the meaning is otherwise clear, the fact that a is a B-item (L-item, S-item, T-item, M-item) will be abbreviated as $a \in$ B (L,S,T,M). A bin is a B-bin (L-bin, S-bin, T-bin, M-bin) if its largest item is a B-item (L-item, S-item, T-item, M-item). There are several types of B-bins: bins containing one B-item and one L-item, and no more B-items, L-items, S-items, or T-items will be called bins of type BL; bins of type BST, BS, BTT, BT, and B are defined analogously. Likewise, there are several types of L-bins, several types of S-bins, and several types of T-bins. The possible types of B-bins, L-bins, S-bins, and T-bins are illustrated in Figure 1. Note that we did not take into consideration the M-items: while it is certainly the case that bins may contain M-items, accounting for them will have no substantive effect on the competitive ratio of *MMP*.

By way of preliminaries, we introduce a binary relation of *superiority* over types of bins. First, all of the types of B-bins, L-bins, S-bins, and T-bins are superior to M-bins. Second, we consider non-M-bins. Here the following ordering of relevant types of items is assumed: B $\prec$ L $\prec$ S $\prec$ T $\prec$ Z, where Z denotes a fictitious item of size 0. We imagine that each bin contains, on top of its B-items, L-items, S-items, and T-items (M-items may be present but are being ignored), a fictitious item of type

[1]Informally, bins are ordered from left to right, and an item is packed into the leftmost bin into which it will fit.

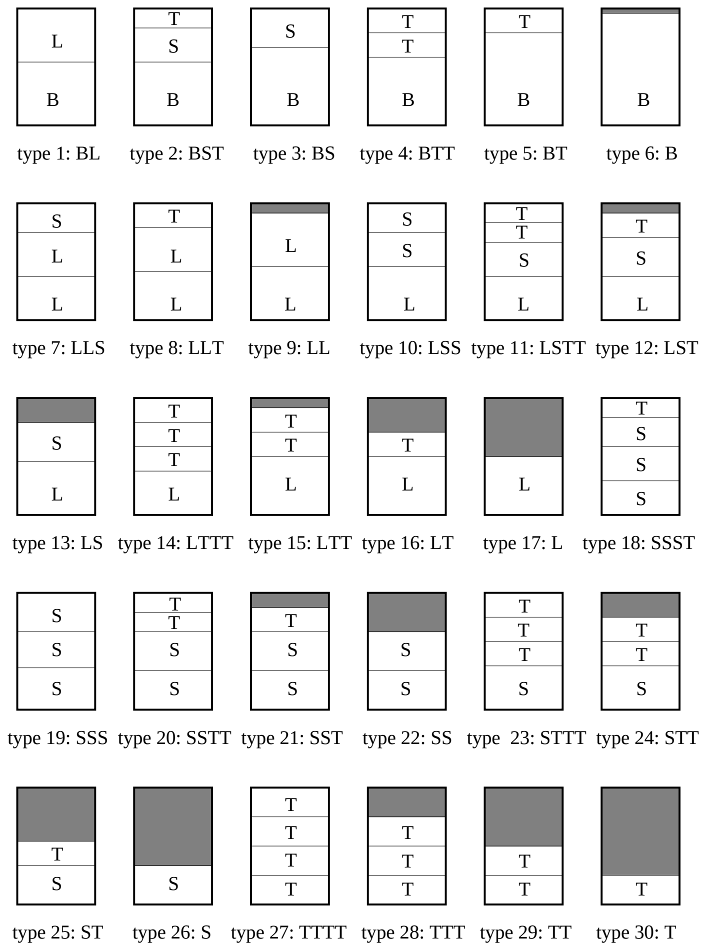

FIG. 1. *Possible types of bins in* MMP.

Z (zero), of size 0. Zero items are introduced solely for technical convenience, as their presence will enable us to impose the desired ordering on the types of bins. Thus, in view of the introduction of Z-items, the types of bins are BLZ, BSTZ, BSZ, ..., TTZ, and TZ. For these types of bins, the relation of superiority is defined as the lexicographical ordering over the types of bins. For example, a bin of type BLZ is superior to a bin of type BSTZ. In the remainder of this paper, we omit Z from the notation describing the types of bins.[2] Finally, we will sometimes find it convenient to refer to these types of bins according to their canonical index in this lexicographical ordering, as depicted in Figure 1: a bin of type 1 is a bin of type BL, a bin of type 2 is a bin of type BST, ..., a bin of type 30 is a bin of type T. We assert naturally that if B_j is superior to B_i, then B_i is *inferior* to B_j.

The *allowed* types of bins in the packings produced by *MMP* are BL, BST, BS, BTT, BT, B, LLS, LLT, LL, SSST, SSS, and TTTT, and, of course, M-bins. This restriction may result in at most six unpacked items: one L-item, two S-items, and

[2]Although Z is omitted, it is needed to ensure that, e.g., BLZ is superior to BZ. The reader should keep in mind that the relation of superiority relies on the presence of Z in each non-M-bin.

three T-items. Clearly, these items could be packed into at most two additional bins (a bin of type LTT and a bin of type SST). *MMP* will utilize the *regular packing*, consisting at all times only of bins of the allowed types, and the *auxiliary storage*, containing the items that are not (currently) packed into a bin from the regular packing.

2.2. LLS-maximality and M-thoroughness. We next define the properties of packings that play a key role for the competitive ratio of *MMP*. We first define the *thoroughness property*. Next, we define the *LLS-maximality property*, a property that is similar to, and (much) stronger than, the thoroughness property. Finally, we define the *M-thoroughness property* aimed at the M-items and their role in the packing. Intuitively, maintaining the LLS-maximality property leads to the competitive ratio of $\frac{5}{4}$ for packings of lists of non-M-items; maintaining LLS-maximality *and* the M-thoroughness property leads to the competitive ratio of $\frac{5}{4}$ for packings of arbitrary lists. We begin with two definitions.

DEFINITION 1. *Let $\mathcal{P}_{B,L,S,T}$ be a set of packings of B-items, L-items, S-items, and T-items such that each packing $P \in \mathcal{P}_{B,L,S,T}$ consists only of the allowed types of bins (BL, BST, BS, BTT, BT, B, LLS, LLT, LL, SSST, SSS, and TTTT), where all of the bins of type BL are to the left of all the non-BL-bins, all of the bins of type BST are to the left of all the non-BL-bins and non-BST-bins, etc.*

DEFINITION 2. *Let a packing $P \in \mathcal{P}_{B,L,S,T}$. Then*

1. *Bins of type BL are thorough in P iff there does not exist a B-item b and an L-item l such that size(b) + size$(l) \leq 1$, and item b is either in a bin of type inferior to BL in the packing P or in the auxiliary storage, and item l is either in a bin of type inferior to BL in the packing P or in the auxiliary storage, i.e., iff it is not possible to pack a B-item from a bin of type inferior to BL or from the auxiliary storage, and an L-item from a bin of type inferior to BL or from the auxiliary storage into a bin.*
2. *Bins of type BST are thorough in P iff there does not exist a bin B of type BS in P, where b and s are the B-item and the S-item packed into B, and a T-item t such that size(b) + size(s) + size$(t) \leq 1$, and the item t is in a bin of type inferior to BST in the packing P or the auxiliary storage.*
3. *Bins of type BS are thorough in P iff there does not exist a B-item b and an S-item s such that size(b) + size$(s) \leq 1$, and the item b is either in a bin of type inferior to BS in the packing P or in the auxiliary storage, and the item s is in a bin of type inferior to BS in the packing P or in the auxiliary storage.*
4. *Bins of type BTT are thorough in P iff there does not exist a bin B of type BT in P, where b and t_1 are the B-item and the T-item packed into B, and a T-item t_2 such that size(b) + size(t_1) + size$(t_2) \leq 1$, and the item t_2 is in a bin of type inferior to BTT in the packing P or in the auxiliary storage.*
5. *Bins of type BT are thorough in P iff there does not exist a B-item b and a T-item t such that size(b) + size$(t) \leq 1$, and the item b is either in a bin of type inferior to BS in the packing P or in the auxiliary storage, and the item t is in a bin of type inferior to BT in the packing P or in the auxiliary storage.*
6. *Bins of type LLS are thorough in P iff there does not exist a bin B of type LLT or LL in P, where l_1 and l_2 are the L-items packed into B, and an S-item s such that size(l_1) + size(l_2) + size$(s) \leq 1$, and the item s is either in a bin of type inferior to LLS in the packing P or in the auxiliary storage.*

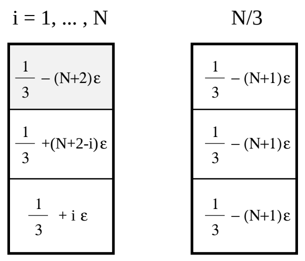

Fig. 2. *An example of a thorough but not LLS-maximal packing. In the packing above, N is an arbitrary integer. The bottom L-item a_1 from the first bin ($i = 1$), size$(a_1) = \frac{1}{3} + \epsilon$, and the top L-item a_2 from the second bin ($i = 2$), size$(a_2) = \frac{1}{3} + N\epsilon$ can fit together with an S-item (all S-items have the size of $\frac{1}{3} - (N+1)\epsilon$). The same is true of the bottom L-item from the second bin ($i = 2$) and the top L-item from the third bin ($i = 3$), ..., the bottom L-item from the $(N-1)$st bin and the top L-item from the Nth bin. Thus, although the packing above is thorough, it is far from LLS-maximal, since many bins of type LLS could be packed from the items in the packing, and all of the items are packed into bins of type inferior to LLS.*

7. *Bins of type LLT are thorough in P iff there does not exist a bin B of type LL in P, where l_1 and l_2 are the L-items packed into B, and a T-item t such that size(l_1) + size(l_2) + size$(t) \leq 1$, and the item t is either in a bin of type inferior to LLT in the packing P or in the auxiliary storage.*
8. *Bins of type SSST are thorough in P iff there does not exist a bin B of type SSS in P, where s_1, s_2, and s_3 are the S-items packed into B, and a T-item t such that size(s_1) + size(s_2) + size(s_3) + size$(t) \leq 1$, and the item t is either in a bin of type inferior to SSST in the packing P or in the auxiliary storage.*

Finally, a packing $P \in \mathcal{P}_{B,L,S,T}$ is thorough iff all of the above types of bins are thorough in P.

LLS-maximality. *MMP* will take some pains to be guaranteed of packing a certain portion of certain L-items and S-items into bins of type LLS (we will call this endeavor "seeking *LLS-coalitions*"). Leading toward this guarantee, we define *LLS-maximality*. LLS-maximality strengthens thoroughness: maintenance of thoroughness does not require LLS-coalitions, and the absence of coalitions leads to a competitive ratio of at least $\frac{4}{3}$ (see the lower bound example for *FFG* in [10]).

Definition 3. *Let a packing $P \in \mathcal{P}_{B,L,S,T}$. Then P is LLS-maximal iff P is thorough and bins of type LLS are LLS-maximal in P; i.e., there does not exist an L-item l_1, another L-item l_2, and an S-item s such that size(l_1)+size(l_2)+size$(s) \leq 1$, and the item l_1 is either in a bin of type inferior to LLS in the packing P or in the auxiliary storage, and the item l_2 is in a bin of type inferior to LLS in the packing P or in the auxiliary storage, and s is in a bin of type inferior to LLS in the packing P or in the auxiliary storage.*

Note that LLS-maximality is a (much) stronger property than thoroughness: there might be packings that are thorough but not maximal. In Figure 2 we give an example, a variant of the lower bound example for *FFG* from [10], of a packing that is thorough but not maximal.

The key factor that distinguishes between thoroughness and LLS-maximality is that when considering whether or not it is possible to pack two L-items and an S-item from bins of type inferior to LLS or the auxiliary storage into a bin, LLS-maximality, unlike thoroughness, does *not* insist that the two L-items must come from the same bin. We note that it can be shown that the maintenance of thoroughness, but not

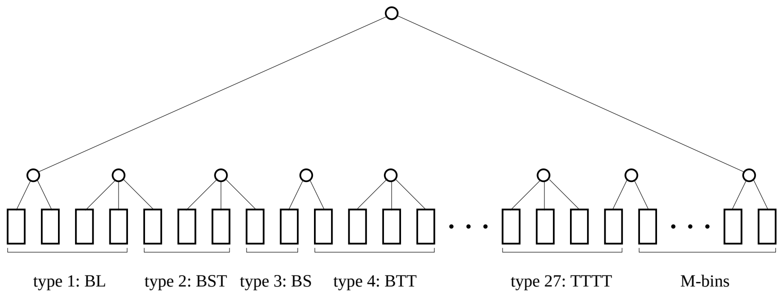

FIG. 3. *A sketch of the* 2-3 *tree of bins in* MMP. *Note that it contains only the allowed types of bins.*

LLS-maximality, leads to a simpler algorithm that also runs in uniform logarithmic time per *Insert/Delete* operation, and is $\frac{4}{3}$–competitive (see [8]).

M-thoroughness. M-thoroughness is the third property we require. It pertains to the role of M-items in the *MMP* packings. Ideally, we would like to be able to develop a method that would enable *MMP* to pack as many M-items into non-M-bins as possible. However, this is not necessary, as it turns out that maintaining M-thoroughness (a much weaker goal), coupled with LLS-maximality, of course, is quite sufficient to guarantee a competitive ratio of $\frac{5}{4}$. Later in this section we show that *MMP* maintains the M-thoroughness property.

DEFINITION 4. *A packing P is M-thorough iff precisely one of the following two conditions is satisfied:*

1. *there are no M-bins in P, or*
2. *there is at least one M-bin in P, and all of the non-M-bins have a level exceeding $\frac{4}{5}$ (i.e., a gap less than $\frac{1}{5}$), and all of the M-bins, except for possibly the rightmost bin in the packing, also have a level exceeding $\frac{4}{5}$.*

2.3. A sketch of *Insert* and *Delete*.

In the next section we describe the data structure of *MMP* in detail. Here we briefly note that all of the bins in the packing will be stored at the leaves of the 2-3 tree of bins, with the bins of type BL placed in the leftmost leaves of the 2-3 tree of bins, with the bins of type BST placed in the leftmost remaining leaves (those not holding bins of type BL) of the 2-3 tree of bins, etc. for all of the other allowed types of bins, and, finally, with the M-bins placed in the rightmost leaves of the 2-3 tree of bins,[3] as depicted in Figure 3.

We now consider, somewhat informally, how to *Insert/Delete* an item. This is done using three major ideas: *myopic packing*, *bundles*, and *LLS-coalitions*.

***Insert* and *Delete* of non-M-items.** We first consider how *MMP Insert*s an item $a \in \mathrm{B} \cup \mathrm{L} \cup \mathrm{S}$. We begin by explaining how *myopic packing* is used to maintain thoroughness. Based on Johnson's grouping [10, 11], when an item a is being packed, a should be more insensitive to previously packed items of "smaller" types than the type of a. Thus, what would a "see" in the bins? Only the items of its own type or of "larger" type. In this sense, a K-item (K is B, L, S, or T) is *myopic* in that it can "see" relatively large items (K-items or larger), and it cannot "see" relatively small ones (smaller than K-items). Based on that view of the packing, a is packed in an *FF*

[3]Recall that the bins are numbered in increasing order from left to right. A numbering of bins that ranges $1, \ldots, MMP(L)$ corresponds to the graph theoretic notion of a preorder numbering of leaves of the 2-3 tree of bins.

fashion (in a's "K or larger" world) using the information stored in the internal nodes and leaves of the 2-3 tree of bins.[4] Let B be the bin into which a was packed. This packing of a results in a forceful eviction of items of smaller types from B, if there are such items at all. The evicted items will be temporarily "set aside" into the auxiliary storage and will eventually be reinserted. Next, an attempt is made to restore the thoroughness of the packing by trying to pack additional items into B, starting from the available items of the largest type that are smaller than K, i.e., the items of the largest type that are smaller than K from the auxiliary storage and from the bins that are inferior to the type of bin the algorithm is trying to reconstruct for B. This effort continues until there are no more available items of that type that can fit with the current bin content. Next, *MMP* continues with the available items (auxiliary storage or inferior bins) of the next largest type, until there are no more available items of that type that would fit into the bin, etc. Here, if an item is taken from some bin, that bin is deleted from the packing, and its contents, except for the item that was taken, are temporarily moved into the auxiliary storage. Upon completing the filling of B and inserting B into the packing, *MMP* reinserts the items from the auxiliary storage into the packing. Their reinsertion may, of course, disturb some other bins and move their contents to the auxiliary storage for later reinsertion. Eventually, all of the items from the auxiliary storage (except perhaps at most one L-item, two S-items, and/or three T-items, of course) are reinserted into the packing, and that packing is thorough.

In addition to thoroughness, *MMP* maintains LLS-maximality. This is done by using LLS-coalitions. To avoid the situation of a list that is thorough but far from LLS-maximal (see Figure 2), an amendment is made to the myopic discipline outlined above. Namely, the insertion of an L-item a is carried out as follows: first, packing a into a B-bin is attempted in a standard myopic fashion. If this attempt fails, a is authorized to try to form an *LLS-coalition* with another L-item and an S-item. The latter two can each be sought in any, and not necessarily the same, bin whose type is inferior to LLS or in the auxiliary storage. If such a coalition is possible, a and the two items are packed into a bin of type LLS, and that bin is inserted into the regular packing. The bins that yielded some or all of these two items need to be deleted, and their remaining content will eventually be reinserted. If the coalition is not possible, the packing of a is completed by resuming the standard myopic steps. Similarly, insertion of an S-item a would involve first packing a into a B-bin in a standard myopic fashion. If this fails, a will seek two L-items coming from any, and not necessarily the same, bins whose type is inferior to LLS or from the auxiliary storage. If two such items are found, an LLS-coalition is formed, and the bin of type LLS is inserted into the regular packing. If not, the packing of a is completed by resuming the standard myopic steps. A careful implementation can guarantee that the added complexity of this *mostly myopic* discipline does not asymptotically add to the running time.

Deletes are implemented as follows: the bin in which a (the item that needs to be deleted) resides is emptied and is deleted. Upon discarding a, the remaining contents of the deleted bin are temporarily moved to the auxiliary storage, from which they are reinserted into the packing as a part of this *Delete* operation.

We show later that *Inserts* and *Deletes* can be carried out in $\Theta(\log n)$ uniform running time, since the number of bins inserted and deleted by an *Insert/Delete*

[4]Note that we do not provide for all the details here. A more detailed description of the data structure and the algorithm *MMP* will be furnished in the coming sections.

operation is bounded by a fixed constant. Intuitively, the discipline of "touching" only the inferior types of bins provides for the desired running time.

Handling M-items. We now consider how *MMP* packs the lists that contain M-items. The goal here is to utilize *bundles* to manipulate many M-items at once within logarithmic uniform running time. At the same time, a proper manipulation of M-items will be important for the M-thoroughness property.

In general, the simplest approach would be to pack the M-items independently of the B-items, L-items, S-items, and T-items by packing them into totally separate bins. This would, however, lead to a competitive ratio greater than $\frac{5}{4}$. Rather, the M-items need to be packed, whenever possible, into non-M-bins. Thus, *MMP* inserts M-items just like any other items, according to their myopic view of the packing (of course, they actually "see" the entire packing). However, the presence of M-items in the packing gives rise to several important considerations.

First, upon insertion of an M-item a into a bin no items will be evicted—M-items are the smallest items! This makes the insertion of an M-item very efficient.

Second, the insertion of B-items, L-items, S-items, and T-items in situations where the input lists contain M-items needs to be examined very carefully. In particular, if the algorithm were to follow only the simple logic of myopic packing, its striving to maintain a thorough packing might require relocation of as many as $\mathcal{O}(n)$ M-items, leading to $\mathcal{O}(n \log n)$ time per *Insert/Delete* operation. This would happen during both insertions and deletions that require relocation of items from the bins of type inferior to that of the bin that is currently being filled. Furthermore, the number of bins that could be inserted and deleted per operation would be huge: it would be possible, for example, to delete as many as $\mathcal{O}(n)$ bins of type BST for the sake of taking a few M-items from each of them and packing those M-items into a single bin of type BL. The disaster does not stop here: each of the items from those many bins of type BST needs to be reinserted, and each reinsertion may again cause an avalanche of deleted bins.

Third, in case the simple myopic discipline is followed, the deletion of an M-item would cause the temporary relocation of B-items, L-items, S-items, T-items, and potentially many M-items into the auxiliary storage. Packing all of these items back into the bins might be very costly: following the same argument as above, $\mathcal{O}(n \log n)$ time might be required to reinsert a single non-M-item, with many inserted and deleted bins.

Thus handling M-items in the same manner as the other items will not do. We solve this apparent difficulty by introducing the technique of *bundling* (see [1]). The idea is that the M-items in each bin (in the auxiliary storage) are collected into *bundles* g_i. All of the bundles in a bin (in the auxiliary storage) have the cumulative size of $\frac{1}{10} < size(g_i) \leq \frac{1}{5}$, except for at most one bundle whose cumulative size is $\leq \frac{1}{10}$. The former kind of bundles is called *closed*, while the latter kind is called *open*.

The purpose of bundles is to allow efficient manipulation of large numbers of M-items at one time: in response to the need to move M-items from a bin to the auxiliary storage, or from the auxiliary storage to a bin, the algorithm will only move entire bundles. During this process, when a bundle is inserted into a bin (or temporarily stored into auxiliary storage), it is first checked to see whether it could be merged with the open bundle, if any, from that bin (or from the auxiliary storage), and, if so, the merging is carried out. While this does not asymptotically increase the running time required for the insertion of an M-item, it drastically decreases the running time of other operations involving M-bundles and makes *MMP* fast ($\Theta(\log n)$ running time per *Insert/Delete* operation).

The algorithm will treat bundles of M-items like any other item (except for the occasional merging of bundles to maintain the property that each bin (auxiliary storage) can have at most one open bundle). Note that a bin can contain at most 10 bundles; hence, we say that no bin can contain more than 10 items. Bundling is one of the tools used to accomplish *M-thoroughness.* It is natural to ask whether or not the technique of bundling is essential for *MMP*; the answer is in the affirmative, since it can be shown that moving only a constant number of very small items per *Insert/Delete* operation disallows competitive ratios below $\frac{4}{3}$, regardless of the running time (see [8]).

3. The data structure and the details of *MMP*.

3.1. The data structure of *MMP*. In this section we describe the data structure utilized by *MMP.* The data structure is rather complex, and it consists of several components.

1. *The regular packing.* The regular packing consists of the bins of the allowed types and some of the information required for the maintenance of an *MMP* packing (LLS-maximal and M-thorough). As mentioned before, the regular packing is maintained via a 2-3 tree. The leaves represent the bins, while the internal nodes store some of the information required for the proper maintenance of the packing (LLS-maximal and M-thorough).

Each leaf contains a record with the following information that provides for a full description of a bin B:

- *content*(B)—Five doubly linked circular lists are utilized to record the set of items currently packed in B, one list for the B-items, one for the L-items, one for the S-items, one for the T-items, and one for the M-bundles packed in B. The entries of these lists are the individual records associated with the B-items, L-items, S-items, T-items, and M-bundles that are currently packed in B. Recall that each bin can contain at most one B-item, two L-items, three S-items, four T-items, and ten M-bundles (at most nine closed bundles and at most one open bundle). Thus these lists are very short.
- Each leaf contains a pointer to the open bundle in B, if any.
- Five numbers that record the myopic levels of B include the following: $\mathit{level}_{\mathrm{B}}(B) = \sum_{a\in content(B)\wedge a\in \mathrm{B}} \mathit{size(a)}$, $\mathit{level}_{\mathrm{L}}(B) = \sum_{a\in content(B)\wedge a\in \mathrm{B}\cup\mathrm{L}} \mathit{size(a)}$, $\mathit{level}_{\mathrm{S}}(B) = \sum_{a\in content(B)\wedge a\in \mathrm{B}\cup\mathrm{L}\cup\mathrm{S}} \mathit{size(a)}$, $\mathit{level}_{\mathrm{T}}(B) = \sum_{a\in content(B)\wedge a\in \mathrm{B}\cup\mathrm{L}\cup\mathrm{S}\cup\mathrm{T}} \mathit{size(a)}$, and $\mathit{level}_{\mathrm{M}}(B) = \mathit{level}(B)$.
- Each leaf contains the type of B.

Each internal node contains the following information about its left, middle, and right (if present) subtree: (1) the largest gaps in the subtree ($\mathit{gap}_{\mathrm{B}} = 1 - \mathit{level}_{\mathrm{B}}$, $\mathit{gap}_{\mathrm{L}} = 1 - \mathit{level}_{\mathrm{L}}$, $\mathit{gap}_{\mathrm{S}} = 1 - \mathit{level}_{\mathrm{S}}$, $\mathit{gap}_{\mathrm{T}} = 1 - \mathit{level}_{\mathrm{T}}$, and $\mathit{gap}_{\mathrm{M}} = 1 - \mathit{level}_{\mathrm{M}}$) and (2) the most inferior type of bin in the subtree.

2. *Inferior trees.* One of the consequences of executing an *Insert* or *Delete* operation is that, in order to maintain a packing that is LLS-maximal, the packing of a bin B may require that some items be removed from bins inferior to B and packed into B. In order to efficiently search for such items in bins inferior to B, *MMP* maintains for each allowed bin type i a set of items of each type of item that appears in a bin of type i. These sets are each represented by a heap (implemented as a 2-3 tree), and we call them the *inferior trees.* To accomplish this efficient search, *MMP* utilizes 15 min-heaps implemented as 2-3 trees: (1) L-items in bins of type BL, (2) L-items in bins of type LLS, (3) L-items in bins of type LLT, (4) L-items in bins of type LL,

(5) S-items in bins of type BST, (6) S-items in bins of type BS, (7) S-items in bins of type LLS, (8) S-items in bins of type SSST, (9) S-items in bins of type SSS, (10) T-items in bins of type BST, (11) T-items in bins of type BTT, (12) T-items in bins of type BT, (13) T-items in bins of type LLT, (14) T-items in bins of type SSST, and (15) T-items in bins of type TTTT.

For example, if the algorithm attempts to pack a B-item into a bin of type BST, it needs to search for an S-item and a T-item from bins whose type is inferior to BST. Thus, an S-item will be searched for in *Aux* (see below) and in the following inferior trees: the tree of S-items in bins of type BS, the tree of S-items in bins of type LLS, the tree of S-items in bins of type SSST, and the tree of S-items in bins of type SSS. Similarly, a T-item will be searched for in *Aux* and in the following inferior trees: the tree of T-items in bins of type BTT, the tree of T-items in bins of type BT, the tree of T-items in bins of type LLT, the tree of T-items in bins of type SSST, and the tree of T-items in bins of type TTTT.

In an inferior tree, each leaf contains an item, or more precisely the record associated with that item, and the internal nodes contain, for left, middle, and (if present) right subtree, the size of the smallest item in the subtree. This enables an easy search for the smallest item in that inferior tree.

3. *Aux.* Admitting only allowed types of bins to the regular packing possibly results in a few excess items that cannot be packed into the regular packing: at most one L-item, two S-items, and three T-items. These items are stored in *Aux.* In addition, in the course of maintaining an *MMP* packing, all of the bins that lose an item(s) a (so that a could be packed into a superior bin), or simply had an item deleted, need to be deleted from the regular packing, and their content, except for the lost/deleted item(s), temporarily stored into *Aux.* The items that are temporarily stored into *Aux* within an operation must all, except for at most one L-item, two S-items, and three T-items, be reinserted into the packing as an integral part of an *Insert/Delete* operation. Later in this section we show that the number of items stored in *Aux* at any time during the execution of *MMP* is bounded by a rather small constant.

Aux is implemented using five min-heaps (again, each heap is implemented as a 2-3 tree), one for each type of item: Aux_{B} for B-items, Aux_{L} for L-items, Aux_{S} for S-items, Aux_{T} for T-items, and Aux_{M} for M-bundles. Each leaf contains an item (bundle), or more precisely the record associated with that item (bundle), and the internal nodes contain, for left, middle, and (if present) right subtree, the size of the smallest item in the subtree. This enables an easy search for the smallest B-item, L-item, S-item, T-item, or M-bundle in *Aux.*

As it turns out, only closed M-bundles will be stored in the 2-3 tree of M-bundles. *Aux* will contain at any time at most one open M-bundle that will be stored separately from the 2-3 tree of (closed) M-bundles.

4. *M-items/bundles.* As mentioned earlier, M-items will be collected in bundles. These bundles may undergo a number of changes in the course of execution of *MMP*: M-items may be inserted to/deleted from bundles; two bundles could be merged into one bundle of M-items; further, there is a need to allow at most one open bundle per bin (at most one open bundle in *Aux*) and to ensure that all of the other bundles in a bin (in *Aux*) are closed. Each M-bundle b will be represented by a 2-3 tree of M-items that are collected into b. With each M-bundle b there will be an associated record that stores detailed information about b.

2-3 trees are particularly suitable for the maintenance and manipulation of bundles. Insertions and deletions of individual M-items into/from a bundle correspond to the operations *Insert* and *Delete*, which are supported by 2-3 trees in logarithmic time. Furthermore, if there is a need to pack a bundle b into a bin B (or store it into *Aux*), b can be easily added onto the list of M-bundles in B (inserted into the 2-3 tree of M-bundles in *Aux* by means of an *Insert* operation on that tree in case b is closed or designated as open and stored into *Aux* in case b is open). If there is a need to merge two bundles, this can be easily accomplished by executing the operation *Union*, also supported by 2-3 trees in logarithmic time, on the two 2-3 trees representing the two bundles.

5. *The list of items in L.* Each item a will have, for the duration of its presence in L (i.e., from operation *Insert(a)* to operation *Delete*(a)), an associated record that maintains detailed information about a ($size(a)$ and a few pointers for manipulation of a in the data structures utilized by *MMP*). In addition, *MMP* will maintain L by storing its items at the leaves of a 2-3 tree of the current items of L. The leaves of that tree are each associated with an item currently in L. Each leaf stores, next to the pointers required for the manipulation of the 2-3 tree of the current items, a unique identifier associated with the item and the pointer to the corresponding record associated with that item.

When an operation *Insert(a)* is initiated, the new item a is inserted into the 2-3 tree of items, and its associated record is created. This is followed by other actions described in subsequent sections. Conversely, the operation *Delete(a)* involves certain actions (described in subsequent sections) and, finally, removes a from the 2-3 tree of the current items and destroys a's associated record. The processing of an *Insert/Delete* operation will typically cause changes to the packing and will affect several items in the packing. None of those changes will, however, have any impact on the 2-3 tree of the current items.

Finally, we note that, in the course of executing an *Insert/Delete* of an item a, the 2-3 tree of current items is used to locate the bin in which a is packed (or to realize that a is in *Aux*) and, furthermore, if a is an M-item, to locate the bundle b that a belongs to. Given an identifier associated with a, $\Theta(\log n)$ running time is required to locate the leaf in the 2-3 tree of current items that is associated with a. Then the pointer to the corresponding record associated with a is followed to gain access to that record within $\mathcal{O}(\log n)$ uniform running time.

3.2. Details of *MMP*.

In this section we furnish the details of *MMP*. In the following subsections we will first provide a top level description of *MMP* and then furnish the details of *clear_Aux*, a key function from that description. In the following, b denotes a B-item, l denotes an L-item, s denotes an S-item, t denotes a T-item, and m denotes a M-bundle.

3.2.1. Top level description of *MMP*.

We now describe *MMP*. Simply put, both *Insert*(x) and *Delete*(x) rely heavily on the function *clear_Aux*. The idea behind *Insert* is to insert the item x into *Aux* and then let *clear_Aux* complete the insertion of x in a manner that will maintain LLS-maximality and M-thoroughness. The idea behind *Delete* is similar, except that before invoking *clear_Aux*, the *Delete* operation needs to remove x from all of the data structures utilized by *MMP*.

In the description below, we utilize the following functions:

- *store_Aux*(x)—store the item x into *Aux*.
- *store_Aux*(X)—store the items and M-bundles from the set X into *Aux*. Then, for each L-item, S-item, and T-item from X, delete that item from the inferior tree

to which it belonged (if any). The set X will be either all of the content of some bin B or a part of it. In the latter case, adjust the myopic levels of B.

• *locate_structure*(x)—return a pointer to the structure to which x belongs (either a bin B or *Aux*).

• *terminate*(x,p)—delete x from all of the data structures. A pointer p points to the structure containing x (either a bin B or *Aux*). If x is an M-item, then delete x from the M-bundle m to which x belongs; m is contained in the structure pointed to by p (either a bin B or *Aux*). If x is a non-M-item, then (1) delete x from the structure to which p points (either a bin B or *Aux*), and then (2) if p points to a bin B, then delete x from the inferior tree to which x belongs (if any). At the conclusion of *terminate*, adjust the myopic levels of B, delete x from L, and delete the record associated with x.

• *remove*(B)—delete a bin B from the 2-3 tree representing the regular packing. Then destroy the record associated with B.

• *add*(B)—insert a bin B into the 2-3 tree representing the regular packing. Let TYPE be the type of B. *add* will insert B immediately after the highest numbered bin of type superior or equal to TYPE. This is accomplished via the information stored at the internal nodes (the most inferior type of bin in the left, middle, and right (if present) subtree). *add* starts at the root and walks down the tree, always to the subtree that stores the largest type that is still inferior to TYPE, until a leaf q is reached, or no subtree with a type that is inferior to TYPE could be found. In the latter case, B is the most inferior bin in the packing, and it will be inserted as the rightmost leaf of the 2-3 tree of bins. In the former case, B is inserted immediately to the left of q.

Both *remove* and *add* will update the information in the interior nodes on the path from B to the root of the 2-3 tree representing the regular packing.

• *open_new_bin*(B)—create a record associated with a new, unpacked bin; call it B. Initialize *content*(B) as empty, and B's myopic levels to 0. Leave the type of B unspecified.

• *pack*(B, $x_1, \ldots, x_k$, TYPE)—pack items/bundles $x_1, \ldots, x_k$ into B. Set the type of B to TYPE, if the value of TYPE is one of the allowed bin types (including M-bins). If TYPE is 0, then *pack* does not set the type of B. The latter option will be convenient when packing M-bundles into non-M, nonempty bins. Note that B need not be empty prior to the execution of the *pack* operation.

• *search_myopic*_K(x) (K $\in$ {L, S, T, M}, x is a K-item)—perform a search for the leftmost bin into which x can fit, with a myopic "K or larger" view of the packing. The search utilizes the information on the largest gap_{K} in the left, middle, and right (if present) subtree stored at the internal nodes of the 2-3 tree representing the regular packing and searches that tree in an *FF* fashion. *search_myopic*_K(x) returns a pointer p to the bin that can accommodate x (in the myopic sense). In this case we let B_p refer to that bin. If the value of the pointer p is *null*, then x could not fit into any of the bins in the (current) regular packing.

• *unload_Aux*$_{\mathrm{M}}$(B)—pack the following M-bundles into a bin B for as long as they fit or until Aux_{M} is empty: (1) the open M-bundle from Aux_{M}, (2) one by one, the smallest remaining closed M-bundle from Aux_{M}. *unload_Aux*$_{\mathrm{M}}$ terminates when either an M-bundle from *Aux* that cannot fit into B is found or Aux_{M} is empty.

• *top_with*_M(B)—fill up a bin B with M-bundles from Aux_{M}. If Aux_{M} is emptied in the process, the rightmost M-bin B_r of the regular packing (if any) is removed from the regular packing, and all of the M-bundles from B_r are inserted into Aux_{M}, at which point filling B with M-bundles from Aux_{M} is resumed.

```
Insert(x):
    store_Aux(x);
    clear_Aux;

Delete(x):
    p =locate_structure(x);
    terminate(x,p);
    if ($x \in$ M) then
        begin
            top_with_M($B_p$); /* $B_p$ is a bin pointed to by p */
            reload_M;
        end;
    else /* x is a non-M-item */
        if p points to a bin $B_p$ then
            begin
                store_Aux(content($B_p$)); /* x is already deleted from $B_p$ */
                remove($B_p$);
                clear_Aux;
            end;
```

FIG. 4. *A top level description of MMP's Insert and Delete.*

```
top_with_M(B):
    unload_$Aux_M$(B);
    while $Aux_M = \emptyset$ and $B_r$ is a M-bin do /* $B_r$—rightmost bin in packing */
        begin
            store_Aux(content($B_r$));
            remove($B_r$);
            unload_$Aux_M$(B);
        end;
```

• *discharge*($x_1, \dots, x_k$,TYPE)—creates a new bin B, packs the items/bundles $x_1, \dots, x_k$ into B, sets the type of B to TYPE, and then inserts B into the regular packing:

```
discharge($x_1, \dots, x_k$, TYPE):
    open_new_bin(B);
    pack(B,$x_1, \dots, x_k$,TYPE);
    add(B);
```

• *reload*_M—empty Aux_M by packing each of its bundles into the regular packing via *FF*:

```
reload_M:
    while $Aux_M \neq \emptyset$ do
        begin
            b=delete_min($Aux_M$); /* Extract the smallest M-bundle m from $Aux_M$ */
            p =search_myopic_M;
            if p = null then /* pack m into a new bin */
                discharge(m,M-bin);
            else pack($B_p$,0); /* pack m into $B_p$, the leftmost possible bin */
        end;
```

We complete this subsection with an observation that it is easy to see that all of the functions outlined above run in logarithmic uniform running time (using the data structures outlined in section 3.1).

We now summarize the top level description of *Insert* and *Delete* in Figure 4.

```
   clear_Aux:
17     phase 1: clear all of the B-items from $Aux_B$;
18     phase 2: pack L-items, S-items, and T-items from Aux into B-bins of the regular
           packing;
19     phase 3: form LLS-coalitions;
20     phase 4: pack the remaining L-items, S-items, and T-items from Aux into non-
           B-bins;
21     phase 5: reload_M;
```

Fig. 5. *The five phases of clear_Aux.*

3.2.2. Details of ***clear_Aux*****.** In this subsection we provide a description of *clear_Aux*, the most involved function invoked by *Insert* and *Delete* operations. *clear_Aux* proceeds in five phases. The first four phases (Figures 6 through 9) are aimed at the maintenance of LLS-maximality, while the last phase is a simple invocation of *reload*_M aimed at the maintenance of M-thoroughness. The five phases are listed in Figure 5.

We proceed with a detailed description of the first four phases. We utilize the functions defined in the previous subsection, as well as the function *seek* defined below.

• *seek*$(x_1, \ldots, x_k, y_1[s_1^1, \ldots, s_{y1}^1], \ldots, y_j[s_1^j, \ldots, s_{yj}^j])$—given items $x_1, \ldots, x_k$, *seek* searches for an item y_1 in the sources from the list $[s_1^1, \ldots, s_{y1}^1]$, ... , and for an item y_j in the sources from the list $[s_1^j, \ldots, s_{yj}^j]$ such that all of the x's and y's fit together; i.e., their cumulative size does not exceed 1. The first source on each list is always Aux_{K_y} (the type of K_y is the same as the type of the corresponding y item), while the remaining sources are the appropriate inferior trees (see code). The length of each list is always bounded by a small constant. This search gives preference to the "recruitment" of as many y's from *Aux* as possible. When an item currently packed into a bin must be utilized, it is preferred to search for items from the second source, and if that fails from the third source, etc. The search in any source is nothing more than checking the minimum size item in that source, which is easily accomplished in logarithmic time, since each source is maintained as a min-heap.

If the search is successful and the desired y's are found, *seek* proceeds by deleting all of the entries associated with the y's that are currently packed into bins from their respective inferior trees. Further, *seek* deletes such y's from their bins, stores the remaining content of these bins into *Aux*, and finally deletes these bins from the regular packing and destroys the records associated with these bins.

Note that *seek* runs in logarithmic uniform running time, since the number of searches of various balanced trees in the data structure from section 3.1 is bounded by a small constant (depending on the lengths of the lists of sources).

3.2.3. Implementation of queries. In this subsection we briefly comment on the implementation of queries *size* and *packing*. The query *size* asks for the number of bins in the (current) *MMP* packing. It is easy to implement it in $\mathcal{O}(1)$ uniform running time by maintaining a global integer variable *number* that reflects the number of bins in the regular packing. When a query *size* is processed, the algorithm will first look into *Aux*, which will at that time contain at most one L-item, two S-items, and three T-items. *MMP* will then compute (in constant time) z, the number of bins needed to pack these items from *Aux* ($z \leq 2$), and return *number* $+ z$ as the response to query *size*.

Recall that the query *packing* requests a description of the packing in the form of

```
   phase 1:
22   while $Aux_B \neq \emptyset$ do
23     begin
24       $b$ = delete_min($Aux_B$);
25       if seek($b, l[Aux_L$,4,3,2]) then
26         discharge($b, l$,BL);
27       else if seek($b, s[Aux_S$,9,8,7,6]) then
28         if seek($b, s, t[Aux_T$,15,14,13,12,11]) then
29           discharge($b, s, t$,BST);
30         else discharge($b, s$,BS);
31       else if seek($b, t[Aux_T$,15,14,13,12]) then
32         if seek($b, t, t'[Aux_T$,15,14,13,12]) then
33           discharge($b, t, t'$,BTT);
34         else discharge($b, t$,BT);
35       else discharge($b$,B);
36     end;
```

FIG. 6. *Phase* 1 *of clear_Aux: clear all of the B-items from* Aux_B.

a list of pairs $(x, \mathrm{Bin}(x))$, where $\mathrm{Bin}(x)$ denotes the bin into which an item x is packed, in time linear in the number of items in the current instance. Such a description of the regular packing can be obtained by a preorder traversal of the 2-3 tree representing the regular packing, and a computation of the packing of the items from *Aux* (without actually packing the items from *Aux* into bins, thereby removing these items from *Aux*). It is easy to see that this processing requires a uniform running time that is linear in n, the size of the (current) list L.

4. Competitive ratio and running time of *MMP*.

4.1. *MMP* is $\frac{5}{4}$-competitive. In this section the proof of the upper bound on the competitive ratio of *MMP* is presented. Lower bound examples with a $\frac{5}{4}$ ratio can be given; hence, this bound is tight. See Figure 10 for details.

4.1.1. Overview of the proof of the upper bound on the competitive ratio of *MMP*. The proof of the upper bound on the competitive ratio of *MMP* consists of several parts. First, we establish that *MMP* maintains regular packings that are LLS-maximal. Second, we show that *MMP* maintains M-thoroughness. Once these two important facts about *MMP* are proved (see subsections 4.1.2 and 4.1.3 below), we consider only lists of non-M-items and prove that the upper bound on the competitive ratio of *MMP* for such lists is $\frac{5}{4}$. This is the most difficult part of the proof. We then prove that *MMP* is $\frac{5}{4}$-competitive for arbitrary lists by an easy application of M-thoroughness.

4.1.2. *MMP* maintains LLS-maximality. In this subsection we prove the LLS-maximality of packings produced by *MMP*. In the proof, we will appeal to the code presented in the previous section.

LEMMA 1. *MMP maintains LLS-maximality of packings of lists of non-M-items.*

Proof. The proof proceeds by induction on the number of *Insert* and *Delete* operations processed by *MMP*. Suppose that the packing P produced by *MMP* is LLS-maximal immediately before an *Insert* or *Delete* operation is requested from *MMP*. We proceed to show that *MMP* will process that operation in a manner that maintains LLS-maximality.

```
phase 2:
    do
        $l = \min(Aux_{\mathrm{L}})$;
        $p = search\_myopic\_\mathrm{L}(l)$;
        if $p$ points to a B-bin, call it $B_p$ then
            begin
                success = true;
                $l = delete\_\min(Aux_{\mathrm{L}})$;
                $store\_Aux(\{y \in content(B_p) \mid y$ of type $\prec \mathrm{L}\})$;
                $pack(B_p, l, \mathrm{BL})$;
                $top\_with\_\mathrm{M}(B_p)$;
            end
    while $Aux_{\mathrm{L}} \neq \emptyset$ and success;
    do
        $s = \min(Aux_{\mathrm{S}})$;
        $p = search\_myopic\_\mathrm{S}(s)$;
        if $p$ points to a B-bin, call it $B_p$ then
            begin
                success = true;
                $s = delete\_\min(Aux_{\mathrm{S}})$;
                $store\_Aux(\{y \in content(B_p) \mid y$ of type $\prec \mathrm{S}\})$;
                if $seek(b, s, t[Aux_{\mathrm{T}}, 15, 14, 13, 12, 11])$ then /* $b \in content(B_p)$ */
                    $pack(B_p, s, t, \mathrm{BST})$;
                else $pack(B_p, s, \mathrm{BS})$;
                $top\_with\_\mathrm{M}(B_p)$;
            end
    while $Aux_{\mathrm{S}} \neq \emptyset$ and success;
    do
        $t = \min(Aux_{\mathrm{T}})$;
        $p = search\_myopic\_\mathrm{T}(t)$;
        if $p$ points to a B-bin, call it $B_p$ then
            begin
                success = true;
                $t = delete\_\min(Aux_{\mathrm{T}})$;
                $store\_Aux(\{y \in content(B_p) \mid y$ of type $\prec \mathrm{T}\})$;
                if $seek(b, t, t'[Aux_{\mathrm{T}}, 15, 14, 13, 12])$ then /* $b \in content(B_p)$ */
                    $pack(B_p, t, t', \mathrm{BTT})$;
                else $pack(B_p, t, \mathrm{BT})$;
                $top\_with\_\mathrm{M}(B_p)$;
            end
    while $Aux_{\mathrm{T}} \neq \emptyset$ and success;
```

FIG. 7. *Phase 2 of clear_Aux: pack L-items, S-items, and T-items from Aux into B-bins of the regular packing.*

LLS-maximality concerns only non-M-items. Thus, operations of interest here are *Inserts* and *Deletes* of non-M-items. To complete the proof, it suffices to examine the first four phases of *clear_Aux* and to verify that their execution maintains LLS-maximality. This is in turn easy to verify by inspection of the definitions of thoroughness and LLS-maximality (Definitions 2 and 3, respectively) and the code of *clear_Aux*. □

```
phase 3:
    do
        $l = \min(Aux_{\mathrm{L}})$;
        success = $seek(l, l'[Aux_{\mathrm{L}},4,3], s[Aux_{\mathrm{S}},9,8])$;
        if success then $discharge(l, l', s, \mathrm{LLS})$;
    while $Aux_{\mathrm{L}} \neq \emptyset$ and success;
    do
        $s = \min(Aux_{\mathrm{S}})$;
        success = $seek(s, l_1[Aux_{\mathrm{L}},4,3], l_2[Aux_{\mathrm{L}},4,3])$;
        if success then $discharge(l_1, l_2, s, \mathrm{LLS})$;
    while $Aux_{\mathrm{S}} \neq \emptyset$ and success;
```

FIG. 8. *Phase* 3 *of clear_Aux: form LLS–coalitions.*

```
phase 4:
    while $|Aux_{\mathrm{L}}| \geq 2$ do
        begin
            $l_1 = delete\_\min(Aux_{\mathrm{L}})$;
            $l_2 = delete\_\min(Aux_{\mathrm{L}})$;
            if $seek(l_1, l_2, t[Aux_{\mathrm{T}},15,14])$ then $discharge(l_1, l_2, t, \mathrm{LLT})$;
            else $discharge(l_1, l_2, \mathrm{LL})$;
        end;
    while $|Aux_{\mathrm{S}}| \geq 3$ do
        begin
            $s_1 = delete\_\min(Aux_{\mathrm{S}})$;
            $s_2 = delete\_\min(Aux_{\mathrm{S}})$;
            $s_3 = delete\_\min(Aux_{\mathrm{S}})$;
            if $seek(s_1, s_2, s_3, t[Aux_{\mathrm{T}},15])$ then $discharge(s_1, s_2, s_3, t, \mathrm{SSST})$;
            else $discharge(s_1, s_2, s_3, \mathrm{SSS})$;
        end;
    while $|Aux_{\mathrm{T}}| \geq 4$ do
        begin
            $t_1 = delete\_\min(Aux_{\mathrm{T}})$;
            $t_2 = delete\_\min(Aux_{\mathrm{T}})$;
            $t_3 = delete\_\min(Aux_{\mathrm{T}})$;
            $t_4 = delete\_\min(Aux_{\mathrm{T}})$;
            $discharge(t_1, t_2, t_3, t_4, \mathrm{TTTT})$;
        end;
```

FIG. 9. *Phase* 4 *of clear_Aux: pack the remaining L-items, S-items, and T-items from Aux into non-B-bins.*

4.1.3. *MMP* maintains M-thoroughness. In this subsection we prove that *MMP* maintains M-thorough packings. This proof also proceeds by induction on the number of *Insert* and *Delete* operations processed by *MMP*. Suppose that the packing P produced by *MMP* is M-thorough immediately before an *Insert* or *Delete* operation is requested from *MMP*. We proceed to show that *MMP* will process that operation in a manner that maintains M-thoroughness.

We first consider the operation *Insert*(x): the item x is simply stored into Aux, and then *clear_Aux* is invoked. Within *clear_Aux*, each bin B whose content is changed (B can be a new bin or a bin whose content partially changes in the course of execution

$$L = \left(\frac{1}{2} + \epsilon, \frac{1}{2} + 2\epsilon, \ldots, \frac{1}{2} + \frac{\mathrm{N}}{2}\epsilon, \ldots, \frac{1}{2} + \mathrm{N}\epsilon, \right.$$
$$\left. \frac{1}{2} - \mathrm{N}\epsilon, \frac{1}{2} - (\mathrm{N}-1)\epsilon, \ldots, \frac{1}{2} - \frac{\mathrm{N}}{2}\epsilon, \ldots, \frac{1}{2} - \epsilon \right)$$

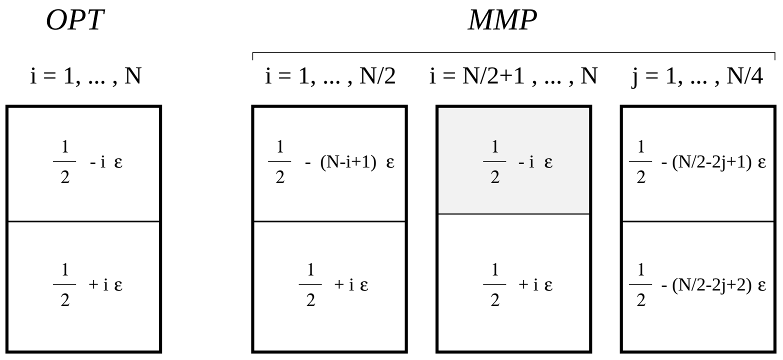

FIG. 10. *A lower bound example for MMP. Here the items are inserted precisely in the order in which they are listed in L above. This example proves that there are arbitrarily large lists L (N is an arbitrarily large integer divisible by 4) for which OPT(L) = N, and MMP(L) = $\frac{5}{4}$N, thus establishing a lower bound of $\frac{5}{4}$ on the competitive ratio of MMP, i.e., R(MMP) $\geq \frac{5}{4}$.*

of *clear_Aux*) is filled with M-bundles via the *top_with_*M function. This function continues filling B until an M-bundle that cannot fit into B is found, in which case $level(B) > \frac{4}{5}$ or until there are no more M-bins and Aux_{M} is empty. Finally, phase 5 of *clear_Aux* is an execution of the function *reload_*M. This function will, in case Aux_{M} is *not* empty at the conclusion of maintaining LLS-maximality (phases 1 through 4), accomplish M-thoroughness by packing the M-bundles from Aux_{M} into the regular packing in an *FF* fashion. This will guarantee that (1) there could only be M-bins if each non-M-bin has a level $> \frac{4}{5}$ and (2) any M-bin, except, perhaps, for the rightmost bin in the packing, has a level $> \frac{4}{5}$.

Next, we consider the operation *Delete*(x), where x is a non-M-item. If x is packed into a bin B_p, x is deleted from B_p, the remaining content of B_p is inserted into *Aux*, B_p is deleted from the regular packing, and *clear_Aux* is invoked. (As argued above *clear_Aux* maintains the M-thoroughness of the packing.) Note that deletion of non-M-items from *Aux* has no effect on M-thoroughness of *MMP*. Finally, deletion of M-items (see code: lines 3–9 in *Delete*) also preserves M-thoroughness via functions *top_with_*M and *reload_*M.

Thus we proved the following lemma.

LEMMA 2. *MMP maintains M-thoroughness of packings of arbitrary lists.*

4.1.4. Consideration of lists with no M-items. We fix an arbitrary list L that contains no M-items. Recall that L may be obtained by an arbitrary sequence of *Inserts* and *Deletes* of items. This arbitrariness may lead to various *MMP* packings of L. We thus fix an arbitrary *MMP* packing P_L of L. P_L consists of the regular packing P_0 and the auxiliary storage *Aux*.

We then fix an arbitrary optimal packing OPT_0 of L and derive from it another optimal packing OPT of L. OPT is a reordering of the bins from OPT_0 such that the bins of type BL are the leftmost bins of OPT, the bins of type BST are the next leftmost bins of OPT, ... , and the bins of type T are the rightmost bins of OPT.

We note that it will be convenient to fix P, a reordering of the bins from P_0. P is defined as follows: the kth leftmost B-bin in OPT and the kth leftmost B-bin in P must contain the same B-item; the order of non-B-bins of P_0 and P is identical.

For both OPT and P, let the *index* of a B-bin B in the packing $OPT(P)$ be the number assigned to B in the "left to right" numbering of the bins from $OPT(P)$. Clearly, for an arbitrary B-item b from L, the indices of B-bins into which b is packed in OPT and P are equal.

Intuitively, the above construction will enable us to view OPT and P in a special way: we may imagine that the B-items of L are "static"—they "remain in the same bin" in an imaginary transformation between OPT and P, and the L-items, S-item, and T-items "migrate," since they may be packed into the k_1th leftmost bin in OPT and into the k_2th leftmost bin in P, where $k_1 \neq k_2$.

The underlying idea. The underlying idea of the proof is to develop an elegant way of capturing the following imaginary series of events. At the outset, someone "glued" the B-items from L into $\mathcal{B}$ bins, one B-item per bin. That person is then required to take the non-B-items from L and pack them into those $\mathcal{B}$ bins and as many additional bins as necessary (these will be non-B-bins), so as to create OPT. Next, that person is required to remove all of the non-B-items from bins and retain the bins containing the glued B-items in the "left to right" order produced by OPT. Finally, that person is again required to take the non-B-items from L and pack them into those $\mathcal{B}$ bins and as many additional bins as necessary (again, these will be non-B-bins), so as to create P. As indicated in the course of defining P, we are interested in "migrations" of non-B-items *from* bins of all types in OPT *to* B-bins in P. The notion of glued B-items motivated the definitions of OPT and P: B-items may be viewed as "static"; they do not move from bin to bin when the packing is changed from OPT to P. L-items, S-items, and T-items, on the other hand, may "migrate"; i.e., the indices of the bins they are packed into in OPT and P may differ. The remainder of the proof explores this "itemographic process."

We perform an extensive analysis of the structure of P in terms of different types of bins in P and OPT and their respective multiplicity f_i, $1 \leq i \leq 30$, where f_i denotes the number of bins of type i in OPT (i is the canonical index of bin types with respect to the superiority relation. See Figure 1.) The proof will *not at all* depend on the particular sequence of *Inserts* and *Deletes* that led to L and its *MMP* packing P_L. The proof will depend *only* on the fact, proved in subsection 4.1.2, that *MMP* produces packings that are LLS-maximal. Based on that property, we derive lower bounds on the quantities L_B, S_B, and T_B, and N_{LLS} defined thus.

DEFINITION 5. *Let L_B (S_B, T_B) denote the number of L-items (S-items, T-items) in B-bins in P. Let N_{LLS} denote the number of bins of type LLS in P.*

A preliminary result. We begin with a preliminary lemma. We establish an upper bound on the number of bins *MMP* would require to pack a given number of L-items, S-items, and T-items.

LEMMA 3. *Suppose L contains only L-items, S-items, and T-items.* MMP *will pack L into at most $\frac{\mathcal{L}}{2} + \frac{\mathcal{S}}{3} + \frac{\mathcal{T}}{4} + 2$ bins.*

Proof. Recall that *MMP* maintains LLS-maximality (see Lemma 1). P_0 will require precisely $\lfloor \frac{\mathcal{L}}{2} \rfloor$ bins with two L-items (these bins may each contain an additional S-item or an additional T-item), followed by at most $\lfloor \frac{\mathcal{S}}{3} \rfloor$ bins with three S-items (analogously, these bins may each contain an additional T-item), followed by at most

$\lfloor \frac{\mathcal{T}}{4} \rfloor$ bins with four T-items. In addition, *Aux* might contain at most one L-item, two S-items, and three T-items. All of these items can fit into at most two bins (one bin of type LTT and one bin of type SST). Hence the number of bins required by *MMP* to pack all of the items of L is at most

$$\left\lfloor \frac{\mathcal{L}}{2} \right\rfloor + \left\lfloor \frac{\mathcal{S}}{3} \right\rfloor + \left\lfloor \frac{\mathcal{T}}{4} \right\rfloor + 2 \le \frac{\mathcal{L}}{2} + \frac{\mathcal{S}}{3} + \frac{\mathcal{T}}{4} + 2. \qquad \square$$

Several technical results. We now proceed with several technical results that are important in estimating the quantities L_B, S_B, T_B, and N_{LLS}. A precise estimate of these quantities is the key ingredient in the proof. Each of the following lemmas contains three statements that are quite analogous and are moreover proved analogously. We adopt this particular style of presentation to avoid unnecessary repetition.

LEMMA 4. *Consider an optimal packing P_{opt} consisting only of k bins of type BL (BS; LLS). Consider a list L whose optimal packing could be P_{opt} and is constructed in stages:*[5]

```
stage 0:          X = the set of items in P_opt;
                  L = ∅;
                  go to stage 1;
stage i:          Y = (the smallest B-item in X (B-item; two L-items),
(1 ≤ i ≤ k)          the smallest L-item in X (S-item; S-item));
                  L = L · Y;
                  X = X − set of items in Y;
                  go to stage i + 1;
stage k + 1:      return L;
```

Then the B-item and the L-item (the B-item and the S-item; the two L-items and the S-item) assigned to Y in stages 1 through $\alpha = \lceil \frac{k}{2} \rceil$ ($\lceil \frac{k}{2} \rceil$; $\lceil \frac{k}{3} \rceil$), respectively, must fit together into a bin of type BL (BS, LLS).

Proof. Suppose by way of contradiction that it is *not* the case that the B-item and the L-item (the B-item and the S-item; the two L-items and the S-item) assigned to Y in all of the stages 1 through $\alpha = \lceil \frac{k}{2} \rceil$ ($\lceil \frac{k}{2} \rceil$; $\lceil \frac{k}{3} \rceil$), respectively, fit together into a bin of type BL (BS, LLS).

Then there exists a positive integer $1 \le \beta < \alpha$ such that only the B-items and the L-items (the B-items and the S-items; the L-items and the S-items) assigned to Y in stages 1 through β, respectively, fit into a bin of type BL (BS; LLS).

Consider L arranged in a table with 2 (2;3) rows and k columns as follows: ith column contains exactly the items added to L in the ith stage of the construction of L, where within a column the items are arranged in successive rows according to the order in which they were added to L. Refer to Figure 11 for details.

Note that only the items from the first β columns of this table can, according to the supposition, fit together into bins: one bin per column of type BL (BS; LLS).

Let $\mathcal{C}_1$ be the set of all of the items that lie in the first row (first row; first two rows) and are also in the first β columns. Let $n_1 = card(\mathcal{C}_1)$. Clearly $n_1 = \beta$ (β; 2β). Let $\mathcal{C}_2$ be the set of all of the items in the last row that belong to the columns $\beta + 1, \beta + 2, ..., k$. Let $n_2 = card(\mathcal{C}_2)$. Clearly $n_2 = k - \beta$.

Any item $a \in \mathcal{C}_2$ could only be packed into a bin of type BL (BS; LLS) if b, the B-item (the B-item; at least one of the two L-items) with which a would be packed,

[5]In the construction below, the operator $\cdot$ denotes concatenation; i.e., $\cdot$ is a binary operator that appends its second operand to its first operand.

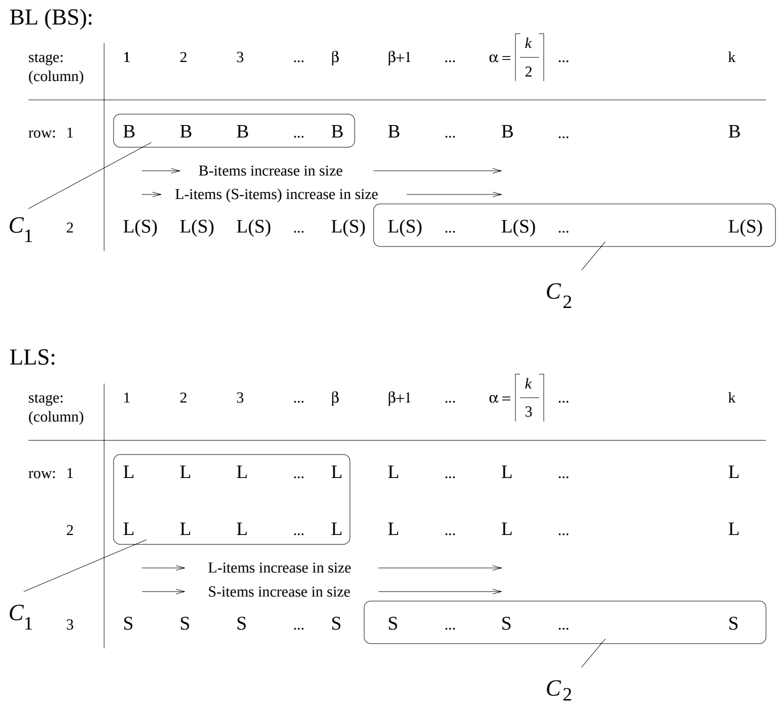
BL (BS):
stage: (column)
1 2 3 ... β β+1 ... α = ⌈k/2⌉ ... k
row: 1
B B B ... B B ... B ... B
B-items increase in size
L-items (S-items) increase in size
2
L(S) L(S) L(S) ... L(S) L(S) ... L(S) ... L(S)
C1
C2
LLS:
stage: (column)
1 2 3 ... β β+1 ... α = ⌈k/3⌉ ... k
row: 1
L L L ... L L ... L ... L
2
L L L ... L L ... L ... L
L-items increase in size
S-items increase in size
3
S S S ... S S ... S ... S
C1
C2

FIG. 11. *A tabular arrangement of L. Letters B, L, and S denote B-items, L-items, and S-items of L, respectively.*

would be from $\mathcal{C}_1$; otherwise the level of such a bin would exceed 1. Thus, since the optimal packing of L contains only bins of type BL (BS; LLS), there should be sufficiently many items in $\mathcal{C}_1$ to ensure the packing of each $a \in \mathcal{C}_2$; i.e., it should be the case that $n_1 \geq n_2$, i.e., $n_1 - n_2 \geq 0$. However,

$$n_1 - n_2 = (c-1)\beta - (k-\beta) \; = \; c\beta - k \; \leq c\left(\left\lceil \frac{k}{c} \right\rceil - 1\right) - k$$
$$\leq \left(\frac{k+c-1}{c}\right) - c - k \; = \; -1,$$

where c=2 (2;3) bins of type BL (BS; LLS). This is a contradiction. ∎

Clearly, if *MMP* packs this particular list L in a series of *Inserts* of items in the order in which the items appear in L, *MMP* will pack at least α bins of type BL (BS; LLS).

COROLLARY 1. *The MMP packing of the list L defined in the statement of Lemma 4, carried out as a series of Inserts of items in the order in which the items appear in L, must contain at least $\alpha = \lceil \frac{k}{2} \rceil$ ($\lceil \frac{k}{2} \rceil$; $\lceil \frac{k}{3} \rceil$) bins of type BL (BS, LLS).*

Thus we have shown that a very particular sequence of *Inserts* will pack an L with a very specific optimal packing in the manner that will, informally, "salvage" about one half (one half; one third) of the bins of type BL (BS; LLS). The following lemma shows that this particular order of packing the items of L, inserted one by one in the order in which they appear in L, is not at all essential. Furthermore, it is not essential that an optimal packing of L consists only of bins of type BL (BS; LLS). In other words, to determine the lower bound on the number of bins of type BL (BS; LLS) in an *MMP* packing of a list L containing only B-items, L-items, S-items, and T-items (B-items, S-items, and T-items [note a careful avoidance of L-items); L-items, S-items, and T-items (note a careful avoidance of B-items)), the only relevant information is the maximum number of bins of type BL (BS; LLS) that can be constructed from items in L. *MMP* will, informally, "salvage" about one half (one half; one third) of that maximum number of bins of type BL (BS; LLS). We again choose this particular style of presentation to avoid the unnecessary repetition.

LEMMA 5. *Let L' be a list containing B-items, L-items, S-items, and T-items (B-items, S-items, and T-items; L-items, S-items, and T-items). Let k be the maximum number of bins of type BL (BS; LLS) that could be packed from the items of L'. Then any MMP packing of L', obtained by an arbitrary sequence of Inserts and Deletes leading to L', contains at least $\alpha = \lceil \frac{k}{2} \rceil$ ($\lceil \frac{k}{2} \rceil$; $\lceil \frac{k}{3} \rceil$) bins of type BL (BS; LLS).*

Proof. Suppose by way of contradiction that there is an *MMP* packing P_{bad} of L' that contains only $\gamma < \alpha$ bins of type BL (BS; LLS). Let $P_{desired}$ be a maximum cardinality packing of B-items and L-items (B-items and S-items; L-items and S-items) from L' into bins of type BL (BS; LLS) (note the correspondence of $P_{desired}$ with P_{opt} from Lemma 11). By a hypothesis, the maximum number of bins of type BL (BS; LLS) in the $P_{desired}$ packing is precisely k. Let L be the list of items derived from $P_{desired}$ in the manner of Lemma 11. Let P' be the *MMP* packing of L obtained by a sequence of *Inserts* of items from L in the order in which the items appear in L (note the correspondence with the situation from Corollary 1). Since there are only $\gamma < \alpha$ bins of type BL (BS; LLS) in P_{bad}, there must be at least one B-item b (one B-item b; two L-items l_1, l_2) and one L-item l (one S-item s; one S-item s) that are packed in one of the first α bins of type BL (BS; LLS) in P', and are not packed in a bin of type BL (BS; LLS) in P_{bad}. Note that the size of each of b and l (b and s; l_1, l_2, and s) is no greater than the size of the B-item and the L-item (the B-item and the S-item; the two L-items and the S-item) from the αth bin in P_{bad}. Thus b and l (b and s; l_1, l_2, and s) can fit into a bin. This is a contradiction, since *MMP* produces packings that are thorough (thorough; LLS-maximal) (Lemma 1) for bins of type BL (BS; LLS), and would therefore have packed at least one more bin of type BL (BS; LLS) in P_{bad}. $\square$

Toward a lower bound on N_{LLS}. We proceed by stating several properties that will be used to obtain a lower bound on the value of N_{LLS}. We begin by considering bins of type LSTT. We show that two L-items and one S-item, each from any, and not necessarily the same, bin of type LSTT must fit into a bin.

LEMMA 6. *Let l_1, l_2, and s be two L-items and an S-item, each from any, and not necessarily the same, bin of type LSTT. Then l_1, l_2, and s can fit into a bin.*

Proof. It suffices to show that $2L_{max} + S_{max} \leq 1$, where L_{max} (S_{max}) denotes the maximum size of L-items (S-items) in any bin of type LSTT;

$$L_{max} = 1 - S_{min} - 2T_{min} < 1 - \frac{1}{4} - 2 \cdot \frac{1}{5} = \frac{7}{20},$$
$$S_{max} = 1 - L_{min} - 2T_{min} < 1 - \frac{1}{3} - 2 \cdot \frac{1}{5} = \frac{4}{15},$$
$$2L_{max} + S_{max} < 2 \cdot \frac{7}{20} + \frac{4}{15} = \frac{29}{30} < 1,$$

where S_{min} and T_{min} denote the minimum size of an S-item and an T-item, respectively. $\square$

Establishing a good lower bound on the number of bins of type LLS that would be produced by *MMP* is very important later in the proof, since that bound is a measure of the success of the technique of LLS-coalitions utilized by *MMP*. To that end, we state and prove the following lemma.

LEMMA 7. *Suppose that L is a list of non-B-items. Let P_{opt} be an arbitrary optimal packing of L satisfying the following three conditions:*

1. *P_{opt} contains at least n_1 bins of type LLS;*
2. *at least n_2 L-items from L are not packed into bins of type LLS in P_{opt}, and each of these L-items can fit into a bin of type LSTT;*
3. *at least n_2 S-items from L are not packed into bins of type LLS in P_{opt}, and each of these S-items can fit into a bin of type LSTT.*

Then the number of bins of type LLS in any MMP *packing of L is at least* $\lceil \frac{n_1+n_2}{4} \rceil - 1$.

Proof. First, the following fact follows immediately from Lemma 6.

FACT 1. *The maximum number of bins of type LLS that can be packed from the items of L is at least $n_0 = n_1 + \lfloor \frac{n_2}{2} \rfloor$.*

Now to prove this lemma, we consider two cases.

Case 1. $n_1 \geq n_2$.

By Lemma 5, *MMP* will pack at least $\lceil \frac{n_0}{3} \rceil$ bins of type LLS:

$$\left\lceil \frac{n_0}{3} \right\rceil = \left\lceil \frac{n_1 + \lfloor \frac{n_2}{2} \rfloor}{3} \right\rceil \geq \left\lceil \frac{n_1 + \frac{n_2-1}{2}}{3} \right\rceil = \left\lceil \frac{\frac{3}{4}n_1 + \frac{1}{4}n_1 + \frac{1}{2}n_2 - \frac{1}{2}}{3} \right\rceil$$

$$= \left\lceil \frac{n_1+n_2}{4} - \frac{1}{6} \right\rceil \geq \left\lceil \frac{n_1+n_2}{4} \right\rceil - 1.$$

Case 2. $n_1 < n_2$.

By Lemma 6 and the fact that *MMP* maintains LLS-maximality of bins of type LLS (Lemma 1), *MMP* will pack at least $\lfloor \frac{n_2}{2} \rfloor$ bins of type LLS. Further, note that $n_1 + n_2 < 2n_2$:

$$\left\lfloor \frac{n_2}{2} \right\rfloor \geq \left\lfloor \frac{\frac{n_1+n_2}{2}}{2} \right\rfloor = \left\lfloor \frac{n_1+n_2}{4} \right\rfloor \geq \left\lceil \frac{n_1+n_2}{4} \right\rceil - 1. \qquad \square$$

Toward lower bounds on L_B, S_B, and T_B. The next several lemmas establish important properties that will be used later to bound L_B, S_B, and T_B.

LEMMA 8. *A B-item from a bin of type BL can be packed together with any S-item.*

Proof. It suffices to show that $B_{max} + S_{max} < 1$, where B_{max} denotes the maximum size of B-items in any bin of type BL, and S_{max} denotes the maximum size of S-items:

$$B_{max} = 1 - L_{min} < 1 - \frac{1}{3} = \frac{2}{3}, \;\; S_{max} = \frac{1}{3}, \;\; B_{max} + S_{max} < \frac{2}{3} + \frac{1}{3} = 1,$$

where L_{min} denotes the minimum size of L-items. $\square$

The next nine lemmas are proved quite analogously.

LEMMA 9. *A B-item from a bin of type BL can be packed together with any T-item.*

LEMMA 10. *A B-item from a bin of type BST can be packed together with an L-item from a bin of type LLS.*

LEMMA 11. *A B-item from a bin of type BST can be packed together with an L-item from a bin of type LSTT.*

LEMMA 12. *A B-item from a bin of type BST can be packed together with any S-item.*

LEMMA 13. *A B-item from a bin of type BST can be packed together with any T-item.*

LEMMA 14. *A B-item from a bin of type BS can be packed together with any T-item.*

LEMMA 15. *A B-item from a bin of type BTT can be packed together with an L-item from a bin of type LSTT.*

LEMMA 16. *A B-item from a bin of type BTT can be packed together with any S-item.*

LEMMA 17. *A B-item from a bin of type BTT can be packed together with any T-item.*

***MMP* is $\frac{5}{4}$-competitive for lists with no M-items.** Recall that there are 30 possible types of bins involving B-items, L-items, S-items, and T-items: BL, BST, ... , T (see Figure 1). We will find it convenient to refer to these types of bins according to their canonical index in the lexicographical ordering based on the relation of superiority. For example, a bin of type 1 is a bin of type BL, a bin of type 2 is a bin of type BST, ... , a bin of type 30 is a bin of type T, as depicted in Figure 1.

Next, recall that only a subset of the above types of bins is *allowed* in *MMP* packings. The allowed bin types are BL, BST, BS, BTT, BT, B, LLS, LLT, LL, SSST, SSS, and TTTT. Alternatively, using the canonical indices, the allowed bin types are 1, 2, 3, 4, 5, 6, 7, 8, 9, 18, 19, and 27.

Intuitively, certain types of bins will be more important for the proof of the upper bound on the competitive ratio than the others. We therefore classify the bin types into the *front* (more important) types of bins and the *back* (less important) types of bins.

DEFINITION 6. *The* front *types of bins are BL, BST, BS, BTT, BT, B, LLS, and LSTT (types $1, 2, \dots, 7, 11$). All of the other types are* back *types of bins. Let $\mathcal{L}_{back}$ ($\mathcal{S}_{back}$, $\mathcal{T}_{back}$) denote the number of L-items (S-items, T-items) in the bins of back types in OPT.*

Recall that f_i, $1 \le i \le 30$, denotes the number of bins of type i in *OPT*. The following equations hold (see Figure 1):

$$\mathcal{L}_{back} = 2(f_8 + f_9) + f_{10} + \sum_{i=12}^{17} f_i,$$

$$\mathcal{S}_{back} = 3(f_{18} + f_{19}) + 2(f_{10} + f_{20} + f_{21} + f_{22}) + f_{12} + f_{13} + \sum_{i=23}^{26} f_i,$$

$$\begin{aligned}\mathcal{T}_{back} &= 4f_{27} + 3(f_{14} + f_{23} + f_{28}) + 2(f_{15} + f_{20} + f_{24} + f_{29}) \\ &\quad + f_8 + f_{12} + f_{16} + f_{18} + f_{21} + f_{25} + f_{30}, \\ \mathcal{L} &= f_1 + 2f_7 + f_{11} + \mathcal{L}_{back}, \\ \mathcal{S} &= f_2 + f_3 + f_7 + f_{11} + \mathcal{S}_{back}, \\ \mathcal{T} &= f_2 + 2f_4 + f_5 + 2f_{11} + \mathcal{T}_{back}.\end{aligned}$$

We next furnish a technical definition of the *list of indices*.

DEFINITION 7. *A list of indices $l = i_1, i_2, \dots, i_k$ is a list of positive integers. $*$ will be used as an abbreviation for the list* 8, 9, 10, 12, 13, ..., 30.[6]

[6]Note that this list corresponds to the back types of bins.

We would like to use this notation to count the number of bins in OPT whose type is an element of the list of indices l. To that end, we furnish the following definition.

DEFINITION 8. *Let l denote a list of indices. Let f_l denote the number of bins in OPT whose type is an element of l. Then $f_l = \sum_{i \in l} f_i$.*

For example, the number of bins in OPT that contains a B-item and an S-item (bins of type BST (2) or BS (3)) is $f_{2,3} = f_2 + f_3$.

Let us define another technical definition of *ranges of bins*. Each range of bins r_i captures precisely the indices of B-bins of type i in OPT.

DEFINITION 9. *Ranges of bins r_i, $1 \leq i \leq 6$, are segments of positive integers defined as follows: $r_1 = [1, \dots, f_1]$, $r_2 = [f_1 + 1, \dots, f_{1,2}]$, $r_3 = [f_{1,2} + 1, \dots, f_{1,2,3}]$, $r_4 = [f_{1,2,3} + 1, \dots, f_{1,2,3,4}]$, $r_5 = [f_{1,\dots,4} + 1, \dots, f_{1,\dots,5}]$, and $r_6 = [f_{1,\dots,5} + 1, \dots, f_{1,\dots,6}]$.*

We would like to count the number of items of a certain type (L-items, S-items, or T-items) that are packed into a bin of type i in OPT and into a B-bin with an index from the range r_j in P. Intuitively, we would like to count how many items of a certain type "migrated" from bins of type i in OPT (intuitively, the source bins) to B-bins in P that "used to be" bins of type j in OPT; i.e., the index of such bins in both OPT and P is from the range r_j (intuitively, the destination bins).

We would also like to count the number of items of a certain type (L-items, S-items, or T-items) that are packed into a bin in OPT whose type is an element of the list of indices l_s (intuitively, the types of source bins), and into a B-bin in P whose index is from a range of bins indexed by an element of the list of indices l_d (intuitively, the ranges of destination bins).

DEFINITION 10. *Let L_d^s (S_d^s, T_d^s), $s \in \{1, 2, \dots, 5, 7, 8, \dots, 30\}$ and $d \in \{1, 2, \dots, 6\}$, denote the number of L-items (S-items, T-items) that are packed into bins of type s (source) in OPT, and into B-bins with an index from r_d (destination) in P.*

Let l_s and l_d denote lists of indices. Let $L_{l_d}^{l_s}$ ($S_{l_d}^{l_s}$, $T_{l_d}^{l_s}$) denote the number of L-items (S-items, T-items) that are packed into a bin in OPT whose type is an element of the list of indices l_s, and into a B-bin in P whose index is from a range of bins indexed by an element of the list of indices l_d. Then

$$L_{l_d}^{l_s} = \sum_{i \in l_s} \sum_{j \in l_d} L_j^i \quad \left(S_{l_d}^{l_s} = \sum_{i \in l_s} \sum_{j \in l_d} S_j^i \ , \ \ T_{l_d}^{l_s} = \sum_{i \in l_s} \sum_{j \in l_d} T_j^i \right).$$

We illustrate this notation with two examples.

First, L_1^7 denotes the number of L-items that are packed into bins of type 7 (LLS) in OPT, and into bins with an index from r_1 in P, that is, into one of the f_1 leftmost bins, bins that contain a B-item that is packed into a bin of type 1 (BL) in OPT.

Second, the number of T-items that are packed into a bin of one of the types BST (2), BT (4), or LLT (8) in OPT, and into a B-bin whose index is from one of the ranges r_2 or r_3 (that is, into one of the bins that contains a B-item that is packed into a bin of type 2 (BST) or a bin of type 3 (BS) in OPT) is written as follows:

$$T_{2,3}^{2,4,8} = \sum_{i \in 2,4,8} \sum_{j \in 2,3} T_j^i = T_2^2 + T_2^4 + T_2^8 + T_3^2 + T_3^4 + T_3^8.$$

We now state and prove the main result of this subsection.

LEMMA 18. *$MMP(L) \leq \frac{5}{4} OPT(L) + 3$.*

Proof. We begin with a series of four claims that establish lower bounds on the values of L_B, S_B, T_B, and N_{LLS}, respectively.

CLAIM 1. $L_B \geq L^{1,7,11,*}_{1,3,\ldots,6} + L^{1,*}_2 + \min(2f_7 + f_{11} - L^{7,11}_{1,3,\ldots,6}, f_2 - L^{1,*}_2)$.

Proof. The key to the proof of this lemma is an observation that any B-item from a bin of type 2 (BST) can be packed together with any L-item from a bin of type 7 or 11 (LLS or LSTT) (Lemmas 10 and 11).

The right-hand side of the above inequality consists of the follwing three additive terms.

1. The first term accounts for all of the L-items that are packed into one of the bins of type 1, 7, 11, or into one of the bins of back type in *OPT*, and into one of the B-bins in P that contains a B-item that is packed into a bin of type 1, 3, 4, 5, or 6 in *OPT* (note a careful avoidance of bins of type 2 (BST)).

2. The second term accounts for all of the L-items that are packed into one of the bins of type 1, or into one of the bins of back type in *OPT*, and into one of the B-bins in P that contains a B-item that is packed into a bin of type 2 (again, note a careful avoidance of bins of type 2 (BST)).

3. The third term, the min term, is somewhat more involved.

The first operand of the min term is $2f_7 + f_{11} - L^{7,11}_{1,3,\ldots,6}$. This number denotes the number of L-items from bins of type 7 (LLS) and 11 (LSTT) in *OPT* that are *not* packed into a bin in P together with a B-item that is packed into a bin of type 1, 3, 4, 5, or 6 in *OPT* (note a careful avoidance of B-items from bins of type 2 (BST) in *OPT*).

The second operand of the min term is $f_2 - L^{1,*}_2$. This number denotes the number of B-items that are (1) packed into a bin of type 2 (BST) in *OPT* and (2) are *not* packed into a bin in P together with an L-item that is packed into a bin of type 1 (BL) or one of the back-type bins in *OPT* (note a careful avoidance of L-items from bins of type 7 (LLS) and 11 (LSTT) in *OPT*).

Thus, by Lemmas 10 and 11, any B-item accounted for by the second operand can be packed together with any L-item accounted for by the first operand of the min expression. By the LLS-maximality of *MMP* (see Lemma 1), P will feature $\min(2f_7 + f_{11} - L^{7,11}_{1,3,\ldots,6}, f_2 - L^{1,*}_2)$ bins of type 1 (BL) containing the above B-items and L-items. $\square$

The following two lemmas are proved quite analogously.

CLAIM 2. $S_B \geq S^{2,3,7,11,*}_{3,5,6} + \min(f_2 + f_3 + f_7 + f_{11} - S^{2,3,7,11}_{3,5,6}, f_1 + f_2 + f_4 - (L_B - L^{1,7,11,*}_{3,5,6}))$.

CLAIM 3. $T_B \geq T^{2,4,5,11,*}_{5,6} + \min(f_2 + 2f_4 + f_5 + 2f_{11} - T^{2,4,5,11}_{5,6},\ f_{1,\ldots,4} - (L_B - L^{1,7,11,*}_{5,6}) - (S_B - S^{2,3,7,11,*}_{5,6}))$.

CLAIM 4. $N_{LLS} \geq \frac{1}{4}(\max(0, f_7 - (L^7_{1,\ldots,6} + S^7_{1,\ldots,6})) + f_{11} - \max(L^{11}_{1,\ldots,6}, S^{11}_{1,\ldots,6})) - 1$.

Proof. The key to the proof of this lemma is the statement of Lemma 7. Let the set of items packed into non-B-bins in *OPT* play the role of L in Lemma 7. Let the non-B-bins of *OPT* play the role of P_{opt} in Lemma 7. Further, note the following conditions.

1. $\max(0, f_7 - (L^7_{1,\ldots,6} + S^7_{1,\ldots,6}))$ is a (conservative) lower bound on the number of bins of type LLS in *OPT* that do *not* contain an item that is packed into a B-bin in P. These $\max(0, f_7 - (L^7_{1,\ldots,6} + S^7_{1,\ldots,6}))$ bins of type LLS play the role of n_1 bins of type LLS in Lemma 7.

2. $f_{11} - \max(L^{11}_{1,\ldots,6}, S^{11}_{1,\ldots,6})$ is a (conservative) lower bound on the number of L-items and the number of S-items from bins of type LSTT in *OPT* that are *not* packed into B-bins in P. These $f_{11} - \max(L^{11}_{1,\ldots,6}, S^{11}_{1,\ldots,6})$ L-items and $f_{11} - \max(L^{11}_{1,\ldots,6}, S^{11}_{1,\ldots,6})$ S-items play the role of n_2 L-items and S-items in Lemma 7.

This claim, then, follows directly from Lemma 7. Note that, unlike in Lemma 7, we did not choose to apply the ceiling operator, as this will have no substantive effect on the analysis of the competitive ratio of *MMP*. ◻

Next, we apply Lemma 5 to establish lower bounds on the values of L_1^1 and S_3^3. Note that we do not provide a lower bound for T_5^5. Again, while a lower bound on T_5^5 could be established in an analogous manner, accounting for T_5^5 will have no substantive effect on the analysis of the competitive ratio of *MMP*.

Claim 5. $L_1^1 \geq \frac{1}{2}(f_1 - \min(f_1, L_1^{7,11,*} + L_{2,\ldots,6}^1))$.

Proof. The proof is immediate from Lemma 5. Let Γ be the set of B-items from L that are packed into bins of type BL in OPT and at the same time satisfy the following two properties.

1. They are packed into a bin in OPT together with L-items which satisfy the property that they are *not* packed into a bin of type BL in P together with a B-bin with an index from $r_2 \cup r_3 \ldots \cup r_6$. Informally, the B-items from Γ cannot be packed in OPT together with L-items that "migrate" from bins with indices from r_1 in OPT to other B-bins in P.

2. They are *not* packed into a bin in P together with L-items that are packed into non-B-bins in OPT.

A conservative estimate of the lower bound on the cardinality of Γ is $f_1 - \min(f_1, L_1^{7,11,*} + L_{2,\ldots,6}^1)$. We say that the estimate is conservative, since there may be B-items that violate both properties, whereas this bound assumes that each B-item violates at most one property; i.e., it is assumed that there are no "overlaps."

The list consisting of B-items of Γ and L-items that are packed together with a B-item from Γ in OPT now plays the role of L' from Lemma 5. Further, the above conservative estimate of the cardinality of Γ plays the role of k from Lemma 5.

The claim follows directly from Lemma 5. Note that, unlike in Lemma 5, we did not apply the ceiling operator, as this will have no substantive effect on the analysis of the competitive ratio of *MMP*. ◻

The following claim is proved quite analogously.

Claim 6. $S_3^3 \geq \frac{1}{2}(f_3 - \min(f_3, L_3^{1,7,11,*} + S_3^{2,7,11,*} + S_{1,2,4,5,6}^3))$.

Next, we note that the number of bins in OPT is precisely $OPT(L) = \sum_{i=1}^{30} f_i$. Let us now bound the value of *MMP*(L) in terms of f_i's, L_B, S_B, T_B, and N_{LLS}. The number of B-bins, $\mathcal{B} = \sum_{i=1}^{6} f_i = f_{1,2,3,4,5,6}$, is identical for OPT and P. Further, *MMP* packs L_B (S_B, T_B) L-items (S-items, T-items) into B-bins, and N_{LLS} bins of type LLS. Finally, we assume that the L-items, S-items, and T-items that are not explicitly accounted for via L_B, S_B, T_B, and N_{LLS} are packed in the "worst possible way." Hence, we bound the value of *MMP*(L) as follows:

$$\begin{aligned}
MMP(L) &\leq f_{1,\ldots,6} + N_{LLS} + \frac{1}{2}(\mathcal{L} - L_B - 2N_{LLS}) \\
&\quad + \frac{1}{3}(\mathcal{S} - S_B - N_{LLS}) + \frac{1}{4}(\mathcal{T} - T_B) + 2 \\
&= f_{1,\ldots,6} + \frac{1}{2}(f_1 + 2f_7 + f_{11} + \mathcal{L}_{back} - L_B) \\
&\quad + \frac{1}{3}(f_2 + f_3 + f_7 + f_{11} + \mathcal{S}_{back} - S_B - N_{LLS}) \\
&\quad + \frac{1}{4}(f_2 + 2f_4 + f_5 + 2f_{11} + \mathcal{T}_{back} - T_B) + 2.
\end{aligned}$$

Note that

$$\frac{\mathcal{L}_{back}}{2} + \frac{\mathcal{S}_{back}}{3} + \frac{\mathcal{T}_{back}}{4} \leq \frac{5}{4} f_*,$$

which can be verified by a straightforward substitution. Hence, to complete the proof it suffices to show that

$$\mathcal{K} \leq \frac{5}{4} f_{1,\ldots,7,11} + 1,$$

where $\mathcal{K}$ is defined as follows:

$$\begin{aligned}\mathcal{K} = f_{1,\ldots,6} &+ \frac{1}{2}(f_1 + 2f_7 + f_{11} - L_B)\\ &+ \frac{1}{3}(f_2 + f_3 + f_7 + f_{11} - S_B - N_{LLS})\\ &+ \frac{1}{4}(f_2 + 2f_4 + f_5 + 2f_{11} - T_B).\end{aligned}$$

We outline four conditions, (I),...,(IV), by observing Claims 1 through 4. Our case analysis will be guided by these conditions:

$$\begin{aligned}\text{(I) holds} &\Longleftrightarrow 2f_7 + f_{11} - L^{7,11}_{1,3,\ldots,6} \leq f_2 - L^{1,*}_2,\\ \text{(II) holds} &\Longleftrightarrow f_2 + f_3 + f_7 + f_{11} - S^{2,3,7,11}_{3,5,6} \leq f_1 + f_2 + f_4 - (L_B - L^{1,7,11,*}_{3,5,6}),\\ \text{(III) holds} &\Longleftrightarrow f_2 + 2f_4 + f_5 + 2f_{11} - T^{2,4,5,11}_{5,6}\\ &\qquad \leq f_{1,\ldots,4} - (L_B - L^{1,7,11,*}_{5,6}) - (S_B - S^{2,3,7,11,*}_{5,6}),\\ \text{(IV) holds} &\Longleftrightarrow L^7_{1,\ldots,6} + S^7_{1,\ldots,6} \leq f_7.\end{aligned}$$

Before proceeding with the analysis, we note several useful facts:

1. $L_B \geq \frac{1}{2} f_1$ (immediate from Lemma 5).
2. If (I) holds,

$$\begin{aligned}L_B &\geq L^{1,7,11,*}_{1,3,\ldots,6} + L^{1,*}_2 + 2f_7 + f_{11} - L^{7,11}_{1,3,\ldots,6}\\ &= L^{1,*}_{1,3,\ldots,6} + L^{1,*}_2 + 2f_7 + f_{11}\\ &\geq 2f_7 + f_{11}.\end{aligned}$$

3. If (I) does not hold ($\neg$(I) holds),

$$\begin{aligned}L_B &\geq L^{1,7,11,*}_{1,3,\ldots,6} + L^{1,*}_2 + f_2 - L^{1,*}_2\\ &= f_2 + L^{1,7,11,*}_{1,3,\ldots,6}\\ &= f_2 + L^1_1 + L^{1,7,11,*}_{3,\ldots,6} + L^{7,11,*}_1\\ &\geq f_2 + \frac{1}{2}(f_1 - \min(f_1, L^{7,11,*}_1 + L^1_{2,\ldots,6})) + L^{1,7,11,*}_{3,\ldots,6} + L^{7,11,*}_1\\ &= f_2 + \frac{1}{2}(f_1 - \min(f_1, L^{7,11,*}_1 + L^1_{2,\ldots,6})) + \left(\frac{1}{2} + \frac{1}{2}\right) L^1_{3,\ldots,6}\\ &\quad + L^{7,11,*}_{3,\ldots,6} + \left(\frac{1}{2} + \frac{1}{2}\right) L^{7,11,*}_1 + \left(\frac{1}{2} - \frac{1}{2}\right) L^1_2\\ &= f_2 + \frac{1}{2} f_1 + \frac{1}{2}(L^{7,11,*}_1 + L^1_{2,\ldots,6}) - \frac{1}{2}\min(f_1, L^{7,11,*}_1 + L^1_{2,\ldots,6})\end{aligned}$$

$$
\begin{aligned}
&+\frac{1}{2}L^{1}_{3,\ldots,6} + L^{7,11,*}_{3,\ldots,6} + \frac{1}{2}L^{7,11,*}_{1} - \frac{1}{2}L^{1}_{2}\\
&\geq \frac{1}{2}f_1 + f_2 - \frac{1}{2}L^{1}_{2} + \frac{1}{2}L^{1}_{3,\ldots,6} + L^{7,11,*}_{3,\ldots,6} + \frac{1}{2}L^{7,11,*}_{1}\\
&\geq \frac{1}{2}f_1 + f_2 - \frac{1}{2}L^{1}_{2} + \frac{1}{2}L^{7,11}_{1,3,\ldots,6}.
\end{aligned}
$$

4. If (II) holds,

$$
\begin{aligned}
S_B &\geq S^{2,3,7,11,*}_{3,5,6} + f_2 + f_3 + f_7 + f_{11} - S^{2,3,7,11,*}_{3,5,6}\\
&= f_2 + f_3 + f_7 + f_{11}.
\end{aligned}
$$

5. If $\neg$(II) holds,

$$
\begin{aligned}
S_B &\geq S^{2,3,7,11,*}_{3,5,6} + f_1 + f_2 + f_4 - L_B + L^{1,7,11,*}_{3,5,6}\\
&= f_1 + f_2 + f_4 - L_B + S^{2,7,11,*}_{5,6} + L^{1,7,11,*}_{5,6} + S^{3}_{3} + S^{2,7,11,*}_{3} + S^{3}_{5,6} + L^{1,7,11,*}_{3}\\
&\geq f_1 + f_2 + f_4 - L_B + S^{2,7,11,*}_{5,6} + L^{1,7,11,*}_{5,6} + \frac{1}{2}f_3\\
&\quad -\frac{1}{2}\min(f_3, L^{1,7,11,*}_{3} + S^{2,7,11,*}_{3} + S^{3}_{1,2,4,5,6}) + S^{2,7,11,*}_{3} + S^{3}_{5,6} + L^{1,7,11,*}_{3}\\
&= f_1 + f_2 + f_4 - L_B + S^{2,7,11,*}_{5,6} + L^{1,7,11,*}_{5,6} + \frac{1}{2}f_3\\
&\quad -\frac{1}{2}\min(f_3, L^{1,7,11,*}_{3} + S^{2,7,11,*}_{3} + S^{3}_{1,2,4,5,6}) + \left(\frac{1}{2}+\frac{1}{2}\right)S^{2,7,11,*}_{3}\\
&\quad + \left(\frac{1}{2}+\frac{1}{2}\right)S^{3}_{5,6} + \left(\frac{1}{2}+\frac{1}{2}\right)L^{1,7,11,*}_{3} + \left(\frac{1}{2}-\frac{1}{2}\right)S^{3}_{1,2,4}\\
&= f_1 + f_2 + f_4 - L_B + S^{2,7,11,*}_{5,6} + L^{1,7,11,*}_{5,6} + \frac{1}{2}f_3\\
&\quad +\frac{1}{2}(L^{1,7,11,*}_{3} + S^{2,7,11,*}_{3} + S^{3}_{1,2,4,5,6})\\
&\quad -\frac{1}{2}\min(f_3, L^{1,7,11,*}_{3} + S^{2,7,11,*}_{3} + S^{3}_{1,2,4,5,6})\\
&\quad +\frac{1}{2}S^{2,7,11,*}_{3} + \frac{1}{2}S^{3}_{5,6} + \frac{1}{2}L^{1,7,11,*}_{3} - \frac{1}{2}S^{3}_{1,2,4}\\
&\geq f_1 + f_2 + \frac{1}{2}f_3 + f_4 - L_B + \frac{1}{2}L^{1,7,11,*}_{3} + L^{1,7,11,*}_{5,6} + \frac{1}{2}S^{2,7,11,*}_{3}\\
&\quad +S^{2,7,11,*}_{5,6} + \frac{1}{2}S^{3}_{5,6} - \frac{1}{2}S^{3}_{1,2,4}\\
&\geq f_1 + f_2 + \frac{1}{2}f_3 + f_4 - L_B - \frac{1}{2}S^{3}_{1,2,4}.
\end{aligned}
$$

6. If (III) holds,

$$
\begin{aligned}
T_B &\geq T^{2,4,5,11,*}_{5,6} + f_2 + 2f_4 + f_5 + 2f_{11} - T^{2,4,5,11}_{5,6}\\
&= f_2 + 2f_4 + f_5 + 2f_{11}.
\end{aligned}
$$

7. If $\neg$ (III) holds,

$$
\begin{aligned}
T_B &\geq T^{2,4,5,11,*}_{5,6} + f_{1,\ldots,4} - (L_B - L^{1,7,11,*}_{5,6}) - (S_B - S^{2,3,7,11,*}_{5,6})\\
&\geq f_{1,\ldots,4} - (L_B - L^{1,7,11,*}_{5,6}) - (S_B - S^{2,3,7,11,*}_{5,6})\\
&\geq f_{1,\ldots,4} - L_B - S_B.
\end{aligned}
$$

Based on the bounds on L_B, S_B, T_B, and N_{LLS}, the proof of Lemma 18 is completed by an case analysis.

Case 1. (II) holds; (III) holds.

$$\begin{aligned}
\mathcal{K} &= f_{1,\ldots,6} + \frac{1}{2}(f_1 + 2f_7 + f_{11} - L_B) \\
&\quad + \frac{1}{3}(f_2 + f_3 + f_7 + f_{11} - S_B - N_{LLS}) \\
&\quad + \frac{1}{4}(f_2 + 2f_4 + f_5 + 2f_{11} - T_B) \\
&\leq f_{1,\ldots,6} + \frac{1}{2}(f_1 + 2f_7 + f_{11} - L_B) \\
&\quad + \frac{1}{3}(f_2 + f_3 + f_7 + f_{11} - (f_2 + f_3 + f_7 + f_{11}) - N_{LLS}) \\
&\quad + \frac{1}{4}(f_2 + 2f_4 + f_5 + 2f_{11} - (f_2 + 2f_4 + f_5 + 2f_{11})) \\
&= f_{1,\ldots,6} + \frac{1}{2}(f_1 + 2f_7 + f_{11} - L_B) - \frac{1}{3}N_{LLS} \\
&\leq f_{1,\ldots,6} + \frac{1}{2}\left(f_1 + 2f_7 + f_{11} - \frac{1}{2}f_1\right) \\
&= \frac{5}{4}f_1 + f_{2,\ldots,7} + \frac{1}{2}f_{11} \\
&\leq \frac{5}{4}f_{1,\ldots,7,11} + 1.
\end{aligned}$$

Case 2. (II) holds; $\neg$(III) holds.

$$\begin{aligned}
\mathcal{K} &= f_{1,\ldots,6} + \frac{1}{2}(f_1 + 2f_7 + f_{11} - L_B) \\
&\quad + \frac{1}{3}(f_2 + f_3 + f_7 + f_{11} - S_B - N_{LLS}) \\
&\quad + \frac{1}{4}(f_2 + 2f_4 + f_5 + 2f_{11} - T_B) \\
&\leq f_{1,\ldots,6} + \frac{1}{2}(f_1 + 2f_7 + f_{11} - L_B) \\
&\quad + \frac{1}{3}(f_2 + f_3 + f_7 + f_{11} - S_B - N_{LLS}) \\
&\quad + \frac{1}{4}(f_2 + 2f_4 + f_5 + 2f_{11} - (f_1 + f_2 + f_3 + f_4 - L_B - S_B)) \\
&= \frac{5}{4}f_1 + \frac{4}{3}f_2 + \frac{13}{12}f_3 + \frac{5}{4}f_4 + \frac{5}{4}f_5 + f_6 + \frac{4}{3}f_7 + \frac{4}{3}f_{11} \\
&\quad - \frac{1}{4}L_B - \frac{1}{12}S_B - \frac{1}{3}N_{LLS} \\
&\leq \frac{5}{4}f_1 + \frac{4}{3}f_2 + \frac{13}{12}f_3 + \frac{5}{4}f_4 + \frac{5}{4}f_5 + f_6 + \frac{4}{3}f_7 + \frac{4}{3}f_{11} \\
&\quad - \frac{1}{4}L_B - \frac{1}{12}(f_2 + f_3 + f_7 + f_{11}) - \frac{1}{3}N_{LLS} \\
&\leq \frac{5}{4}f_{1,2} + f_3 + \frac{5}{4}f_{4,5} + f_6 + \frac{5}{4}f_{7,11} \\
&\leq \frac{5}{4}f_{1,\ldots,7,11} + 1.
\end{aligned}$$

Case 3. $\neg$(II) holds; $\neg$(III) holds.

$$\begin{aligned}
\mathcal{K} &= f_{1,\ldots,6} + \frac{1}{2}(f_1 + 2f_7 + f_{11} - L_B)\\
&\quad + \frac{1}{3}(f_2 + f_3 + f_7 + f_{11} - S_B - N_{LLS})\\
&\quad + \frac{1}{4}(f_2 + 2f_4 + f_5 + 2f_{11} - T_B)\\
&\le f_{1,\ldots,6} + \frac{1}{2}(f_1 + 2f_7 + f_{11} - L_B)\\
&\quad + \frac{1}{3}(f_2 + f_3 + f_7 + f_{11} - S_B - N_{LLS})\\
&\quad + \frac{1}{4}(f_2 + 2f_4 + f_5 + 2f_{11} - (f_1 + f_2 + f_3 + f_4 + (L_{5,6}^{1,7,11,*} - L_B) + (S_{5,6}^{2,3,7,11,*} - S_B)))\\
&= \frac{5}{4}f_1 + \frac{4}{3}f_2 + \frac{13}{12}f_3 + \frac{5}{4}f_{4,5} + f_6 + \frac{4}{3}f_{7,11} - \frac{1}{4}L_B - \frac{1}{12}S_B - \frac{1}{3}N_{LLS}\\
&\quad - \frac{1}{4}L_{5,6}^{1,7,11,*} - \frac{1}{4}S_{5,6}^{2,3,7,11,*}\\
&\le \frac{5}{4}f_1 + \frac{4}{3}f_2 + \frac{13}{12}f_3 + \frac{5}{4}f_{4,5} + f_6 + \frac{4}{3}f_{7,11} - \frac{1}{4}L_B\\
&\quad - \frac{1}{12}\left(f_1 + f_2 + \frac{1}{2}f_3 + f_4 - L_B - \frac{1}{2}S_{1,2,4}^{3}\right) - \frac{1}{3}N_{LLS} - \frac{1}{4}L_{5,6}^{1,7,11,*} - \frac{1}{4}S_{5,6}^{2,3,7,11,*}\\
&\le \frac{7}{6}f_1 + \frac{5}{4}f_2 + \frac{25}{24}f_3 + \frac{7}{6}f_4 + \frac{5}{4}f_5 + f_6 + \frac{4}{3}f_7 + \frac{4}{3}f_{11} - \frac{1}{6}L_B - \frac{1}{3}N_{LLS}\\
&\quad - \frac{1}{4}L_{5,6}^{1,7,11,*} - \frac{1}{4}S_{5,6}^{2,3,7,11,*}\\
&\le \frac{7}{6}f_1 + \frac{5}{4}f_2 + \frac{25}{24}f_3 + \frac{7}{6}f_4 + \frac{5}{4}f_5 + f_6 + \frac{4}{3}f_7 + \frac{4}{3}f_{11} - \frac{1}{6}L_B - \frac{1}{12}S_{5,6}^{7,11} - \frac{1}{3}N_{LLS}.
\end{aligned}$$

Case 3.1. (I) holds.

$$\begin{aligned}
\mathcal{K} &\le \frac{7}{6}f_1 + \frac{5}{4}f_2 + \frac{25}{24}f_3 + \frac{7}{6}f_4 + \frac{5}{4}f_5 + f_6 + \frac{4}{3}f_7 + \frac{4}{3}f_{11} - \frac{1}{6}(2f_7 + f_{11})\\
&\le \frac{5}{4}f_{1,\ldots,7,11} + 1.
\end{aligned}$$

Case 3.2. $\neg$(I) holds.

$$\begin{aligned}
\mathcal{K} &\le \frac{7}{6}f_1 + \frac{5}{4}f_2 + \frac{25}{24}f_3 + \frac{7}{6}f_4 + \frac{5}{4}f_5 + f_6 + \frac{4}{3}f_7 + \frac{4}{3}f_{11}\\
&\quad - \frac{1}{6}\left(\frac{1}{2}f_1 + f_2 - \frac{1}{2}L_2^1 + \frac{1}{2}L_{1,3,\ldots,6}^{7,11}\right) - \frac{1}{12}S_{5,6}^{7,11} - \frac{1}{3}N_{LLS}\\
&= \frac{13}{12}f_1 + \frac{13}{12}f_2 + \frac{25}{24}f_3 + \frac{7}{6}f_4 + \frac{5}{4}f_5 + f_6 + \frac{4}{3}f_7 + \frac{4}{3}f_{11} + \frac{1}{12}L_2^1 - \frac{1}{12}L_{1,3,\ldots,6}^{7,11}\\
&\quad - \frac{1}{12}S_{5,6}^{7,11} - \frac{1}{3}N_{LLS}\\
&\le \frac{7}{6}f_1 - \frac{1}{12}S_1^{7,11} + \frac{1}{12}L_2^1 + \frac{7}{6}f_2 - \frac{1}{12}L_2^{7,11} - \frac{1}{12}S_2^{7,11} + \frac{9}{8}f_3 - \frac{1}{12}S_3^{7,11}\\
&\quad + \frac{5}{4}f_4 - \frac{1}{12}S_4^{7,11} + \frac{5}{4}f_5 + f_6 + \frac{4}{3}f_7 + \frac{4}{3}f_{11} - \frac{1}{12}L_{1,3,\ldots,6}^{7,11} - \frac{1}{12}S_{5,6}^{7,11} - \frac{1}{3}N_{LLS}\\
&\le \frac{5}{4}f_1 + \frac{7}{6}f_2 + \frac{9}{8}f_3 + \frac{5}{4}f_4 + \frac{5}{4}f_5 + f_6 + \frac{4}{3}f_7 + \frac{4}{3}f_{11} - \frac{1}{12}L_{1,\ldots,6}^{7,11} - \frac{1}{12}S_{1,\ldots,6}^{7,11} - \frac{1}{3}N_{LLS}
\end{aligned}$$

$$\begin{aligned}
&\le \frac{5}{4}f_1 + \frac{7}{6}f_2 + \frac{9}{8}f_3 + \frac{5}{4}f_4 + \frac{5}{4}f_5 + f_6 + \frac{4}{3}f_7 + \frac{4}{3}f_{11} - \frac{1}{12}L^{7,11}_{1,\dots,6} - \frac{1}{12}S^{7,11}_{1,\dots,6}\\
&\quad -\frac{1}{3}\left(\frac{1}{4}(\max(0, f_7 - (L^7_{1,\dots,6} + S^7_{1,\dots,6})) + f_{11} - \max(L^{11}_{1,\dots,6}, S^{11}_{1,\dots,6})) - 1\right)\\
&= \frac{5}{4}f_1 + \frac{7}{6}f_2 + \frac{9}{8}f_3 + \frac{5}{4}f_4 + \frac{5}{4}f_5 + f_6 + \frac{4}{3}f_7 + \frac{5}{4}f_{11} - \frac{1}{12}L^{7,11}_{1,\dots,6} - \frac{1}{12}S^{7,11}_{1,\dots,6}\\
&\quad -\frac{1}{12}\left(\max(0, f_7 - (L^7_{1,\dots,6} + S^7_{1,\dots,6})) + \frac{1}{12}\max(L^{11}_{1,\dots,6}, S^{11}_{1,\dots,6})\right) + \frac{1}{3}\\
&\le \frac{5}{4}f_{1,\dots,7,11} + 1 + \frac{1}{12}f_7 - \frac{1}{12}\left(\max(0, f_7 - (L^7_{1,\dots,6} + S^7_{1,\dots,6})) - \frac{1}{12}L^{7,11}_{1,\dots,6} - \frac{1}{12}S^{7,11}_{1,\dots,6}\right.\\
&\quad \left. + \frac{1}{12}\max(L^{11}_{1,\dots,6}, S^{11}_{1,\dots,6})\right)\\
&\le \frac{5}{4}f_{1,\dots,7,11} + 1 + \frac{1}{12}\mathcal{K}',
\end{aligned}$$

where $\mathcal{K}' = f_7 - (\max(0, f_7 - L^7_{1,\dots,6} - S^7_{1,\dots,6}) + L^7_{1,\dots,6} + S^7_{1,\dots,6})$.

Case 3.2.1. (IV) holds. $\mathcal{K}' = f_7 - (f_7 - L^7_{1,\dots,6} - S^7_{1,\dots,6} + L^7_{1,\dots,6} + S^7_{1,\dots,6}) = 0$.

Case 3.2.2. $\neg$(IV) holds. $\mathcal{K}' = f_7 - L^7_{1,\dots,6} - S^7_{1,\dots,6} < 0$.

Case 4. $\neg$(II) holds; (III) holds.

$$\begin{aligned}
\mathcal{K} &= f_{1,\dots,6} + \frac{1}{2}(f_1 + 2f_7 + f_{11} - L_B)\\
&\quad + \frac{1}{3}(f_2 + f_3 + f_7 + f_{11} - S_B - N_{LLS})\\
&\quad + \frac{1}{4}(f_2 + 2f_4 + f_5 + 2f_{11} - T_B)\\
&\le f_{1,\dots,6} + \frac{1}{2}(f_1 + 2f_7 + f_{11} - L_B)\\
&\quad + \frac{1}{3}\left(f_2 + f_3 + f_7 + f_{11} - \left(f_1 + f_2 + \frac{1}{2}f_3 + f_4 - L_B\right.\right.\\
&\quad \left.\left. + \frac{1}{2}L^{1,7,11,*}_3 + L^{1,7,11,*}_{5,6} + \frac{1}{2}S^{2,7,11,*}_3 + S^{2,7,11,*}_{5,6} + \frac{1}{2}S^3_{5,6} - \frac{1}{2}S^3_{1,2,4}\right) - N_{LLS}\right)\\
&\quad + \frac{1}{4}(f_2 + 2f_4 + f_5 + 2f_{11} - (f_2 + 2f_4 + f_5 + 2f_{11}))\\
&= \frac{7}{6}f_1 + f_2 + \frac{7}{6}f_3 + \frac{2}{3}f_4 + f_{5,6} + \frac{4}{3}f_7 + \frac{5}{6}f_{11} - \frac{1}{6}L_B - \frac{1}{3}N_{LLS} + \frac{1}{6}S^3_{1,2,4}\\
&\quad -\frac{1}{3}\left(\frac{1}{2}L^{1,7,11,*}_3 + L^{1,7,11,*}_{5,6} + \frac{1}{2}S^{2,7,11,*}_3 + S^{2,7,11,*}_{5,6} + \frac{1}{2}S^3_{5,6}\right)\\
&\le \frac{7}{6}f_1 + f_2 + \frac{7}{6}f_3 + \frac{2}{3}f_4 + f_{5,6} + \frac{4}{3}f_7 + \frac{5}{6}f_{11} - \frac{1}{6}L_B + \frac{1}{6}S^3_{1,2,4} - \frac{1}{12}S^7_{3,5,6} - \frac{1}{3}N_{LLS}.
\end{aligned}$$

Case 4.1. (I) holds.

$$\begin{aligned}
\mathcal{K} &\le \frac{7}{6}f_1 + f_2 + \frac{7}{6}f_3 + \frac{2}{3}f_4 + f_{5,6} + \frac{4}{3}f_7 + \frac{5}{6}f_{11} - \frac{1}{6}(2f_7 + f_{11}) + \frac{1}{6}S^3_{1,2,4}\\
&\le \frac{5}{4}f_1 - \frac{1}{12}S^3_1 + \frac{13}{12}f_2 - \frac{1}{12}S^3_2 + \frac{5}{4}f_3 - \frac{1}{12}S^3_{1,2,4} + \frac{3}{4}f_4 - \frac{1}{12}S^3_4\\
&\quad + f_{5,6,7} + \frac{2}{3}f_{11} + \frac{1}{6}S^3_{1,2,4}
\end{aligned}$$

$$\begin{aligned}
&= \frac{5}{4}f_1 + \frac{13}{12}f_2 + \frac{5}{4}f_3 + \frac{3}{4}f_4 + f_{5,6,7} + \frac{2}{3}f_{11}\\
&\le \frac{5}{4}f_{1,\dots,7,11} + 1.
\end{aligned}$$

Case 4.2. ¬ (I) holds.

$$\begin{aligned}
\mathcal{K} &\le \frac{7}{6}f_1 + f_2 + \frac{7}{6}f_3 + \frac{2}{3}f_4 + f_{5,6} + \frac{4}{3}f_7 + \frac{5}{6}f_{11} - \frac{1}{6}\left(\frac{1}{2}f_1 + f_2 - \frac{1}{2}L_2^1 + \frac{1}{2}L_{1,3,\dots,6}^7\right)\\
&\quad + \frac{1}{6}S_{1,2,4}^3 - \frac{1}{12}S_{3,5,6}^7 - \frac{1}{3}N_{LLS}\\
&\le \frac{5}{4}f_1 - \frac{1}{12}S_1^3 - \frac{1}{12}S_1^7 + \frac{7}{6}f_2 - \frac{1}{12}L_2^1 - \frac{1}{12}S_2^3 - \frac{1}{12}L_2^7 - \frac{1}{12}S_2^7\\
&\quad + \frac{5}{4}f_3 - \frac{1}{12}S_{1,2,4}^3 + \frac{3}{4}f_4 - \frac{1}{12}S_4^3 - \frac{1}{12}S_4^7 + f_{5,6} + \frac{4}{3}f_7 + \frac{5}{6}f_{11}\\
&\quad + \frac{1}{12}L_2^1 - \frac{1}{12}L_{1,3,\dots,6}^7 + \frac{1}{6}S_{1,2,4}^3 - \frac{1}{12}S_{3,5,6}^7 - \frac{1}{3}N_{LLS}\\
&= \frac{5}{4}f_1 + \frac{7}{6}f_2 + \frac{5}{4}f_3 + \frac{3}{4}f_4 + f_{5,6} + \frac{4}{3}f_7 + \frac{5}{6}f_{11} - \frac{1}{12}L_{1,\dots,6}^7 - \frac{1}{12}S_{1,\dots,6}^7 - \frac{1}{3}N_{LLS}\\
&\le \frac{5}{4}f_1 + \frac{7}{6}f_2 + \frac{5}{4}f_3 + \frac{3}{4}f_4 + f_{5,6} + \frac{4}{3}f_7 + \frac{5}{6}f_{11} - \frac{1}{12}L_{1,\dots,6}^7 - \frac{1}{12}S_{1,\dots,6}^7\\
&\quad - \frac{1}{3}\left(\frac{1}{4}(\max(0, f_7 - (L_{1,\dots,6}^7 + S_{1,\dots,6}^7)) + f_{11} - \max(L_{1,\dots,6}^{11}, S_{1,\dots,6}^{11})) - 1\right)\\
&\le \frac{5}{4}f_1 + \frac{7}{6}f_2 + \frac{5}{4}f_3 + \frac{3}{4}f_4 + f_{5,6} + \frac{4}{3}f_7 + \frac{11}{12}f_{11} - \frac{1}{12}L_{1,\dots,6}^{7,11} - \frac{1}{12}S_{1,\dots,6}^{7,11}\\
&\quad - \frac{1}{12}\left(\max(0, f_7 - (L_{1,\dots,6}^7 + S_{1,\dots,6}^7)) + \frac{1}{12}\max(L_{1,\dots,6}^{11}, S_{1,\dots,6}^{11})\right) + \frac{1}{3}\\
&\le \frac{5}{4}f_{1,\dots,7,11} + 1 + \frac{1}{12}f_7 - \frac{1}{12}\left(\max(0, f_7 - (L_{1,\dots,6}^7 + S_{1,\dots,6}^7)) - \frac{1}{12}L_{1,\dots,6}^{7,11} - \frac{1}{12}S_{1,\dots,6}^{7,11}\right.\\
&\quad \left. + \frac{1}{12}\max(L_{1,\dots,6}^{11}, S_{1,\dots,6}^{11})\right)\\
&\le \frac{5}{4}f_{1,\dots,7,11} + 1 + \frac{1}{12}\mathcal{K}',
\end{aligned}$$

where $\mathcal{K}' = f_7 - (\max(0, f_7 - L_{1,\dots,6}^7 - S_{1,\dots,6}^7) + L_{1,\dots,6}^7 + S_{1,\dots,6}^7)$.

Case 4.2.1. (IV) holds. $\mathcal{K}' = f_7 - (f_7 - L_{1,\dots,6}^7 - S_{1,\dots,6}^7 + L_{1,\dots,6}^7 + S_{1,\dots,6}^7) = 0$.

Case 4.2.2. ¬(IV) holds. $\mathcal{K}' = f_7 - L_{1,\dots,6}^7 - S_{1,\dots,6}^7 < 0$. □

4.1.5. Consideration of arbitrary lists. We established that the competitive ratio of *MMP* packings of lists of non-M-items is $\frac{5}{4}$. We now use the M-thoroughness of *MMP*, established in Lemma 2, to establish $\frac{5}{4}$ as the competitive ratio of *MMP* for arbitrary lists.

THEOREM 1. *Let* L *be a list of items of arbitrary size from* $(0, 1]$. *Then*

$$MMP(L) \le \frac{5}{4}OPT(L) + 3.$$

Proof. We consider the following two cases.

Case 1. There are no M-bins in any *MMP* packing of L.

We may disregard all of the M-items from L and directly apply Lemma 18.

Case 2. There exists at least one *MMP* packing of L that contains an M-bin.

Fix an arbitrary *MMP* packing P_L of L that contains an M-bin. Let P_0 and *Aux* be the regular packing and the auxiliary storage of P_L, respectively. By M-thoroughness of *MMP* (Lemma 2), all of the bins of the regular packing have a level of at least $\frac{4}{5}$, except for, possibly, the last bin. This immediately implies that the regular packing P_0 requires at most $\frac{5}{4}OPT(L) + 1$ bins to pack all of the items of L, *except* for the excess items from *Aux*: at most one L-item, two S-items, and three T-items, all of which can be packed into two bins. Thus the overall number of bins required by P_L is at most $\frac{5}{4}OPT(L) + 3$. □

4.2. The running time of *MMP* is $\Theta(\log n)$.

THEOREM 2. MMP *can be implemented to run in* $\Theta(\log n)$ *uniform running time per* Insert/Delete *operation.*

Proof. Recall from sections 3.2.1 and 3.2.2 that each of the functions utilized by *MMP* runs in logarithmic time. It is easy to see from the code that there is a small bounded number of calls to these functions preceding each packing of an item and in particular each insertion and deletion of a whole bin. Note that this in itself is not sufficient to prove the logarithmic running time of *MMP*. However, it does suffice to show that the number of bins that are inserted and deleted as a consequence of a single *Insert* or *Delete* operation is bounded by a constant.

The proof proceeds by a case analysis. The nontrivial cases are the following: *Insert* of a B-item, *Insert* of an L-item, *Insert* of an S-item, and *Insert* of a T-item; *Delete* of a B-item, *Delete* of an L-item, *Delete* of an S-item, and, finally, *Delete* of a T-item.

We only show the case of an *Insert* of an S-item and note that the proofs are quite analogous for each of the other cases.

To facilitate a simple exposition of the proof, we denote items by upper case letters corresponding to their type (e.g., a T-item will be denoted as T in this proof). Items that could form an LLS-coalition will be decorated with an apostrophe.

What is the longest possible sequence of bin insertions/deletions an *Insert* of an S-item S' could generate?

We consider two subcases. First, S' is inserted so as to form an LLS' bin. Second, S' is inserted into a B-bin. It can be shown that the more difficult case is the second case and with that case the subcase leading to a BS'T bin is more difficult than the subcase leading to a BS' bin.

We proceed with an analysis of the case of a BS'T bin. Before the *Insert* of S', the content of *Aux* was in the worst case: {L, 2S, 3T}.

To construct BS'T, S' may have in the worst case evicted two T-items from a BTT bin and "recruited" a T-item from another BTT bin. At this moment, one bin (BS'T) is inserted, one bin (BTT) is deleted, and the content of *Aux* is in the worst case: {B, L, 2S, 6T}.

The B-item from *Aux* could now, in the worst case, form a BTT bin with two T-items, each from a BT bin. At this moment, one bin (BTT) is inserted and two bins (BT, BT) are deleted. *Aux* is now {2B, L, 2S, 8T}.

The two B-items from *Aux* could now, in the worst case, each form a BT bin with a T-item from an LLT bin. At this moment, two bins (BT, BT) are inserted, and two bins (LLT, LLT) are deleted. *Aux* is now {5L, 2S, 8T}.

By LLS-maximality of *MMP*, none of the L-items from *Aux* can form LLS-coalitions. Thus, in the worst case, each of the five L-items from *Aux* can form an LLT bin with a T-item from an SSST bin. At this moment, five LLT bins are inserted, and five SSST bins are deleted. *Aux* is now {L, 17S, 8T}.

Again, by LLS-maximality of *MMP*, none of the S-items from *Aux* can form LLS-coalitions. Thus, in the worst case, five SSST bins are formed out of 15 S-items from *Aux*, coupled with five T-items, each from a TTTT bin. At this moment, five SSST bins are inserted, and five TTTT bins are deleted. *Aux* is now {L, 2S, 23T}.

Finally, five TTTT bins are inserted, each containing four T-items from *Aux*. At the conclusion, the content of *Aux* is {L, 2S, 3T}.

Thus, the overall number of inserted bins is 18, and the overall number of deleted bins is 15. Note that we ignored the possibility that M-bins may be deleted to execute *top_with*_M, and at the end of the *Insert* of S' *reload*_M. Clearly, consideration of M-bins could contribute only very few inserted or deleted M-bins. For example, a (generous) upper bound on the number of deleted M-bins is 18. $\square$

5. Conclusion.

We have studied the problem of maintaining an approximate solution for *one-dimensional bin packing* when items may arrive and depart dynamically and when the packing may be rearranged to accommodate arriving and departing items. Our main result is a fully dynamic bin packing algorithm *MMP* that is $\frac{5}{4}$-competitive and requires $\Theta(\log n)$ uniform running time per *Insert/Delete* operation. Relative to the best practical off-line algorithms, our algorithm is the best possible with respect to its running time and is nearly approximation-competitive with those algorithms (losing but a factor of $\frac{1}{15}$ to the best of those [13]).

The major unresolved issue is whether there exist fully dynamic bin packing algorithms (accommodating both *Inserts* and *Deletes*) that attain better competitive ratios, i.e., are there algorithms that are α-competitive for some $\alpha < \frac{5}{4}$, and require $o(n)$ time per operation. Here, both uniform and amortized algorithms are of interest.

Other unresolved issues are (1) what is the nature of the trade-off between running times and competitive ratios of fully dynamic bin packing algorithms for bin packing (both uniform and amortized), and (2) is there a competitive ratio for which there are no fully dynamic approximation algorithms for bin packing featuring sublinear running times (uniform or amortized)?

REFERENCES

[1] R. J. Anderson, E. W. Mayr, and M. K. Warmuth (1983), *Parallel approximation algorithms for bin packing*, Inform. and Comput., 82, pp. 262–277.

[2] E. G. Coffman, M. R. Garey, and D. S. Johnson (1983), *Dynamic bin packing*, SIAM J. Comput., 12, pp. 227–258.

[3] E. G. Coffman, M. R. Garey, and D. S. Johnson (1984), *Approximation algorithms for bin packing: An updated survey*, in Algorithm Design for Computer System Design, G. Ausiello, M. Lucertini, and P. Serafini, eds., Springer-Verlag, New York, pp. 49–106.

[4] W. Fernandez de la Vega and G. S. Lueker (1981), *Bin packing can be solved within* $1+\epsilon$ *in linear time*, Combinatorica, 1, pp. 349–355.

[5] D. K. Friesen and M. A. Langston (1991), *Analysis of a compound bin packing algorithm*, SIAM J. Discrete Math., 4, pp. 61–79.

[6] G. Gambosi, A. Postiglione, and M. Talamo (1990), *New algorithms for on–line bin packing*, in Algorithms and Complexity, Proceedings of the First Italian Conference, G. Aussiello, D. P. Bovet, and R. Petreschi, eds., World Scientific, Singapore, pp. 44–59.

[7] M. R. Garey and D. S. Johnson (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco.

[8] Z. Ivković (1995), *Fully Dynamic Approximation Algorithms*, Ph.D. thesis, University of Delaware, Newark, DE.

[9] Z. Ivković and E. L. Lloyd (1993), *Fully dynamic maintenance of vertex cover*, in Proc. 19th International Workshop on Graph–Theoretic Concepts in Computer Science, Lecture Notes in Computer Science, Springer-Verlag, New York, pp. 99–111.

[10] D. S. Johnson (1973), *Near–Optimal Bin Packing Algorithms*, Ph.D. thesis, MIT, Cambridge, MA.

[11] D. S. Johnson (1974), *Fast algorithms for bin packing*, J. Comput. System Sci., 8, pp. 272–314.

[12] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham (1974), *Worst–case performance bounds for simple one-dimensional packing algorithms*, SIAM J. Comput., 3, pp. 299–325.

[13] D. S. Johnson and M. R. Garey (1985), *A* 71/60 *theorem for bin packing*, J. Complexity, 1, pp. 65–106.

[14] N. Karmarkar and R. M. Karp (1982), *An efficient approximation scheme for the one-dimensional bin-packing problem*, in Proc. 23rd IEEE Symposium on Foundations of Computer Science, pp. 312–320.

[15] R. M. Karp (1972), *Reducibility among combinatorial problems*, in Complexity of Computations, R. E. Miller and J. W. Thatcher, eds., Plenum, New York, pp. 85–103.

[16] P. N. Klein and S. Sairam (1993), *Fully Dynamic Approximation Schemes for Shortest Path Problems in Planar Graphs*, manuscript.

[17] C. C. Lee and D. T. Lee (1985), *A simple on-line bin-packing algorithm*, J. Assoc. Comput. Mach., 3, pp. 562–572.

[18] P. Ramanan, D. J. Brown, C. C. Lee, and D. T. Lee (1989), *On-line bin-packing in linear time*, J. Algorithms, 3, pp. 305–326.

[19] A. C.–C. Yao (1980). *New algorithms for bin packing*, J. Assoc. Comput. Mach., 27, pp. 207–277.