SOME RESULTS OF THE SPACE REQUIREMENTS OF DYNAMIC MEMORY ALLOCATION ALGORITHMS+

Dennis W. Ting

TR 75-229

February 1975

Department of Computer Science
Cornell University
Ithaca, New York 14853

+This research was supported in part by NSF grants GJ-28176 and GJ-41115.

SOME RESULTS ON THE SPACE REQUIREMENTS OF DYNAMIC MEMORY ALLOCATION ALGORITHMS+

Dennis W. Ting

The problem of dynamic storage allocation has been studied by Robson [1,2,3] and Krogdahl [4]. We extend some of Robson's results in this paper.

To characterize the problem, we make the following assumptions:

(1) Total number of busy words in memory at any time never exceeds M.

(2) The size of any single request is chosen from the set $\{n_1, n_2, \ldots, n_\ell\}$ where $n_i < n_{i+1}$, $n_\ell < m$.

(3) Once a block has been allocated, it cannot be relocated.

(4) Any request - release sequence obeying (1) and (2) is possible.

We denote the minimum memory size necessary to satisfy all request-release sequences without possibility of overflow by $N^*(m, \{n_1 \ldots n_\ell\})$, and the limit $\lim_{m\to\infty} \frac{N^*(m,\{n_1\ldots n_\ell\})}{m}$ by $N(\{n_1 \ldots n_\ell\})$. Where there is no ambiguity, we denote $N^*(m,\{1\ldots n\})$ by $N^*(m,n)$, and $N(\{1\ldots n\})$ by $N(n)$.

Robson proved in [2] and [3] the following theorems:

Theorem 1: $N^*(m,n) < m(c + 0.86 \log n) + 4n^2 + n$ where c is a constant.

Theorem 2: $N(\{3,4\}) \geq \frac{19}{15}$

In view of Krogdahl's result [4]: $N^*(m,n) \geq \frac{1}{2} m \log n$, Theorem 1 seems quite strong. However, there is an implicit assumption that

+This research was supported in part by NSF grants GJ-28176 & GJ-41115.

$n \ll m$. For otherwise the $4n^2$ term becomes dominant and the bound trivial. We therefore propose:

Theorem 1-a: $N^*(m,n) \leq m\lceil \log n \rceil - \frac{m}{2} \quad \forall n,\ 0 < n \leq m$

We also propose an improvement to Robson's Theorem 2.

Theorem 2-a: $N(\{3,4\}) \geq \frac{23}{18}$

Proof of Theorem 1-a: We give an allocation strategy here which will satisfy all request-release sequences using only $m\lceil \log n \rceil - \frac{m}{2}$ words of storage. (We assume m to be a power of 2, if it is not, we simply use $2^{\lceil \log m \rceil}$ in place of m).

First we divide memory into $\lceil \log n \rceil$ regions R_0 through $R_{\lceil \log n \rceil - 1}$. If we denote the size of R_i by $|R_i|$, then $|R_0| = |R_2| = \ldots = |R_{\lceil \log n \rceil - 1}| = m$, $|R_1| = \frac{m}{2}$, and then

$$\sum_{0}^{\lceil \log n \rceil - 1} |R_i| = m\lceil \log n \rceil - \frac{m}{2}.$$

When a block of x words, $2^i < x \leq 2^{i+1} \leq \frac{m}{2}$, is requested, we search regions $R_0 \ldots R_{i+1}$ for a free area of 2^{i+1} words on a 2^{i+1} integral boundary and allocate it to this request. Notice that since m is assumed to be a power of 2, such a free area, if found, cannot cross region boundaries. Furthermore, in region R_i, each busy block must be larger than 2^{i-1} except of course R_0 and R_1. If the request is for a block of size greater than $\frac{m}{2}$, we search regions $R_{\lceil \log n \rceil - 2}$ and $R_{\lceil \log n \rceil - 1}$ for a sufficiently large free area and allocate it to the request. To see that this strategy cannot fail, we assume a block of size x: $2^i < x \leq 2^{i+1}$ causes overflow.

<u>Case 1</u>: $2^i < x \le 2^{i+1} \le \frac{m}{2}$, and there are no busy blocks in memory larger than $\frac{m}{2}$. It must be the case that in region R_j there is at least one busy block of size at least $2^{j-1}+1$ in each area of 2^{i+1} words in this region, for $j = 0,1,\ldots,i+1$. Hence there are more than

$$\frac{m}{2^{i+1}} + \frac{m}{2^{i+1}} + \sum_{2}^{i+1} \frac{m}{2^{i+1}} \; 2^{j-1} = \frac{m}{2^{i+1}} \; (2+2+4+\ldots+2^i) = m$$

busy words in store, hence the request was illegal.

<u>Case 2</u>: $2^i < x \le 2^{i+1} \le \frac{m}{2}$ and there is already a busy block of size larger than $\frac{m}{2}$ in store.

If $2^{i+1} \le \frac{m}{8}$, this case is identical to case 1. So we assume $2^{i+1} > \frac{m}{8}$. Since there are already a busy block of size $\frac{m}{2} + \ell, \ell > 0$ in store, 2^{i+1} can therefore be only $\frac{m}{2}$ or $\frac{m}{4}$ or $\frac{m}{8}$. In any case, there are more than $\frac{2m}{2^{i+1}} + \frac{m}{2^{i+1}} \sum_{2}^{i-1} 2^{j-1} = \frac{m}{4}$ busy words in regions $R_0 \ldots R_{i-1}$, and we know there is already a block of size $\frac{m}{2} + \ell$ somewhere in regions $R_{\lceil \log n \rceil -2}$ and $R_{\lceil \log n \rceil -1}$. Now if $2^{i+1} = \frac{m}{2}$, the size of the current request is $> \frac{m}{4}$. But there are already $\frac{m}{4} + \frac{m}{2} + \ell$ busy words in memory, this current request is therefore illegal. If $2^{i+1} = \frac{m}{4}$, there must be another $\frac{m}{2^{i+1}} = \frac{m}{4}$ busy words in region R_i, making the total number of busy words in memory more than m already.

<u>Case 3</u>: $x > \frac{m}{2}$.

If no free area is large enough in $R_{\lceil \log m \rceil -1}$ then there must be a block of size less than $\frac{m}{2}$ sitting on the area starting with word # $\frac{m}{2}$. (This is due to the fact that only blocks of size between $\frac{m}{4} + 1$ to $\frac{m}{2}$ can be in $R_{\lceil \log m \rceil -1}$, and that such block must start on square 0 or $\frac{m}{2}$ of $R_{\lceil \log m \rceil -1}$). Furthermore, the entire 1^{st} $\frac{m}{2}$ words of $R_{\lceil \log m \rceil -1}$ must be free. If there is no busy words in the last $\frac{m}{4}$ words of $R_{\lceil \log m \rceil -2}$ then there certainly is a free area large enough to accommodate the current request. So we assume there is a busy block (whose size must be more than $\frac{m}{8}$) in the last $\frac{m}{4}$ of $R_{\lceil \log m \rceil -2}$. But this block (of size $x > \frac{m}{8}$) must be on squares $\frac{3}{4}m, \frac{3}{4}m+1, \ldots, \frac{3}{4}m + x-1$ of $R_{\lceil \log m \rceil -2}$. But this leaves a free area of $\frac{m}{2} + \frac{m}{4} - x = \frac{3}{4}m-x$ words in between the above mentioned busy blocks. Surely, if there are already more than $\frac{m}{4} + x$ busy words in these two busy blocks, the current request cannot be larger than

$$m - \frac{m}{4} - x = \frac{3}{4}m - x \quad \text{which will fit in the free area.} \qquad \text{Q.E.D.}$$

<u>Theorem 2-a</u>: $N(\{3,4\}) \geq \frac{23}{18}$

The attacker's strategy involves first requesting as many 3-word blocks, using only m words, as possible. Depending on how the defender (the allocator) places these blocks, the attacker will release some of the blocks and then request blocks of 4. The proof lies in showing that after the defender is done placing the 4-blocks, memory can be partitioned into areas such that each of these areas will have a density of no more than $\frac{18}{23}$, thus proving that the defender must have used at least $\frac{23}{18}m$ words of memory.

In the following figures, an X represents a single busy block of 3 words, possibly preceeded by a free area whose size is an exact multiple of 4, a dash, -, represents a single free word, and finally a Y represents a single busy block of 4, again possibly preceeded by a free area of size which is an exact multiple of 4. We shall call a 4-word free area a "usable" area. Free areas of sizes up to 3 embedded in busy blocks obviously cannot be used to accommodate requests of 4, and thus are called "unusable" areas, or in the case of single words, "unusable" words. For example, X_ _ X would be $4n_1$ free words followed by a busy block of 3, then followed by $4n_2+2$ free words then another block of 3. There are n_1+n_2 usable areas and 2 unusable words.

Now we consider the memory configuration just after the defender has placed all blocks of size 3. We divide memory into zones of the following types using the state-diagram in figure 1. We can think of memory as a string of input, with input symbols _ for a free square, and X for a busy block of 3 words. A section of memory from word i to word j is of type k if the input string from word i to word j takes us from the start state to a terminal state whose label is k. We resume the process after reaching a terminal state by taking the next input symbol and start from the start state again. The reader should have no difficulty convincing himself that, except possibly at the right hand end, memory can be divided into the following types, with each section matching 1 and only 1 type.

(1) _ _ _ X

(2) _ _ X

(3) _ X

(4) X _ _ _ X

(5) X _ _ X

(6) XX

(7) X _ X _ ... X _ X _ _ _ X (i.e. n copies of X _ followed by X _ _ _ X, $6 > n \geq 1$)

(8) X _ X _ ... X _ X _ _ X (n copies of X _ followed by X _ _ X, $6 > n \geq 1$)

(9) X _ XX

(10) X _ X _ XX

(11) X _ X _ X _ XX _ _ _ X

(12) X _ X _ X _ XX _ _ X

(13) X _ X _ X _ XXX

(14) X _ X _ X _ X followed by (7)

(15) X _ X _ X _ X followed by (8)

(16) X _ X _ X _ XX _ XX

(17) X _ X _ X _ XX _ X _ X _ X

(18) X _ X _ X _ XX _ X _ XX

(19) X _ X _ X _ X _ XX

(20) X _ X _ X _ X _ X _X

Any leading subsequence of any type is possible at the right hand end. Although such a leading subsequence can in fact be of arbitrary length, the number of busy blocks in it must be bounded (by the # of busy blocks of type (20)). In fact, the only possible ending sequences are

(1) X

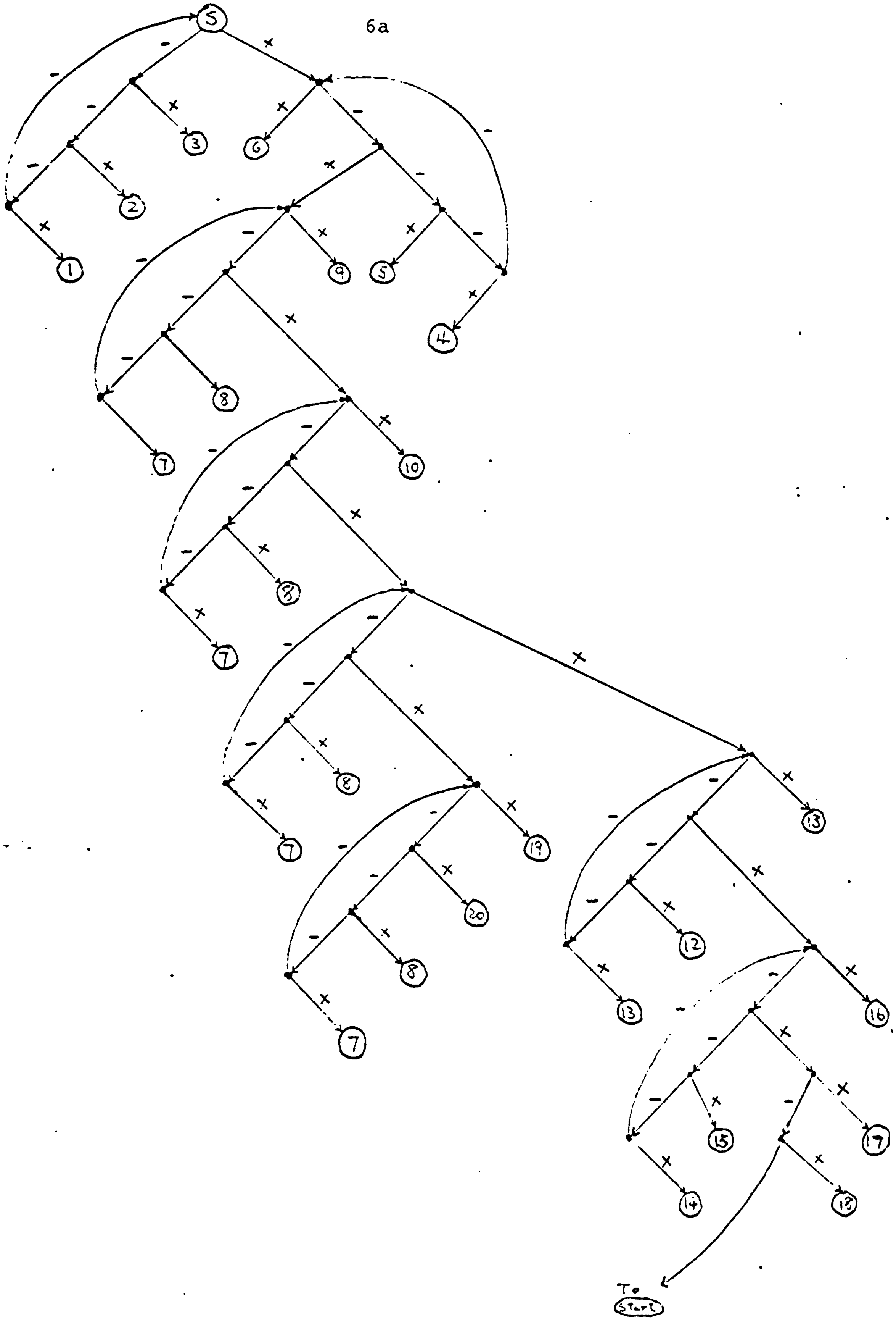
(2) X _ X

(3) X _ X _ X

(4) X _ X _ X _ XX

(5) _ _ ...

6a



Case (5) does not cause us any trouble. Cases (1) through (3) require 1 additional space elsewhere to reduce the density down to $\frac{3}{4}$. Case (4) requires 2 additional spaces to reduce the density to $\frac{3}{4}$. In any case, the 2 spaces become neglible as m becomes large.

We now specify actions associated with sections of each type.

For types (1), (2), (3), (4), (5), (7), (8), (11), (12), (14), (15), and (20) the densities are, respectively, $\frac{1}{2}$, $\frac{3}{5}$, $\frac{3}{4}$, $\frac{6}{9}$, $\frac{6}{8}$, $\frac{3n+6}{4n+9}$, $\frac{3n+6}{4n+8}$, $\frac{18}{24}$, $\frac{18}{23}$, $\frac{3n+18}{4n+24}$, $\frac{3n+18}{4n+23}$, and $\frac{18}{23}$. Each of these densities is $\leq \frac{18}{23}$, so no action is necessary. Note that when computing density, we are not counting the embedded usable area

For type (6) we release the first block, thus creating 3 unusable words. The 3 words released can be used to form part of a block of 4 which will eventually occupy another usable area. We count 3 of the 4 spaces of this other usable area along with the original 6 words used by the 2 blocks. Hence the resulting density is $\frac{6}{9} < \frac{18}{23}$ i.e. the original 6 busy words eventually occupy

(1) The 3 spaces still occupied by the 2nd busy block of size 3.

(2) The 3 spaces vacated but left unusable by the 1st busy block of size 3.

(3) 3 of the 4 additional spaces occupied by the block of 4 which was formed in part with the 3 words released from the 1st block.

For zones of type (9), we release the 1st 2 blocks, releasing 6 words and creating one usable area as well as 3 unusable words. Four of the 6 released words must be used to "kill" the created usable area. That still leaves us 2 words with which to form a new request of 4 which will again take up a new usable area. We must again count 2 of the 4 spaces of this usable area along with the original 10 spaces used. The resulting density is therefore $\frac{9}{12} < \frac{18}{23}$.

For zones of type (13), we simply release the 5th block, then the argument for type (6) applies and will give us a density of $\frac{18}{24} < \frac{18}{23}$.

Assume there are x_{10} zones of type (10), x_{16} zones of type (16), x_{17} zones of type (17), x_{18} zones of type (18) and x_{19} zones of type (19). We also assume the defender left a total of k words in usable areas (we call these "existing" usable areas as opposed to "created" usable areas created by the removal of blocks of 3). If $k \geq 2x_{10} + 2x_{16} + 2x_{17} + x_{18} + x_{19}$ then the overall density (density of all zones of types (10), (16), (17), (18), and (19)) would be

$$\frac{12x_{10} + 21x_{16} + 24x_{17} + 24x_{18} + 18x_{19}}{14x_{10} + 25x_{16} + 29x_{17} + 30x_{18} + 22x_{19} + k}$$

$$= \frac{12x_{10} + 21x_{16} + 24x_{17} + 24x_{18} + 18x_{19}}{16x_{10} + 27x_{16} + 31x_{17} + 31x_{18} + 23x_{19}} \leq \frac{18}{23}$$

(Since each term is $\leq \frac{18}{23}$)

So we may assume $k < 2x_{10} + 2x_{16} + 2x_{17} + x_{18} + x_{19}$.

Now for zones of type (10) we release the 2nd blocks; for zones of type (16) we release the 3rd blocks; for zones of type (17) we release the 3rd blocks; for zones of types (18) and (19) we release both the 2nd and 4th blocks. Thus we have released a total of $3x_{10} + 3x_{16} + 3x_{17} + 6x_{18} + 6x_{19}$ words. Less than $k = 2x_{10} + 2x_{16} + 2x_{17} + x_{18} + x_{19}$ of these will be allocated in "existing" usable areas. That leaves us more than $x_{10} + x_{16} + x_{17} + 5x_{18} + 5x_{19}$ words to form blocks of 4 which <u>must</u> be allocated in "created" usable areas. For zones of type (10), when the created area is used if it is (if its not, density is $< \frac{18}{23}$), one of the following configurations results:

(a) XY _ XX or

(b) X _ YXX

Note that when the number of free spaces in between the 1st and 2nd busy blocks may very well be $4n+1$ where $n > 2$. In this case the defender has a considerable amount of freedom to place the block of 4 (the Y). However the most judicious choices would be the cases considered. In any other choice, the defender would in fact leave more than 1 unusable word embedded between the busy blocks mentioned above.

In case (a) we release the 1st block, in case (b) we release the 3rd block. In either case for an "investment" of 4 words, we get a return of 3 words, thus a net "investment" of 1 word per zone will "kill" the created usable area. So to kill all created usable areas in zones of this type, we need at most x_{10} words of investment.

For zones of type (16), after the created usable area is "killed", one of the following configurations results:

(a) X _ XY _ XX _ XX or

(b) X _ X _ YXX _ XX .

In case (a) we release the 1^{st}, 2^{nd}, 5^{th} and 6^{th} blocks, freeing 12 words. However, 8 of these must be used to consume the newly created usable area, thus giving us 4 words back, i.e. for an investment of 4 words, we get 4 words back, for a net investment of 0 word per zone. In case (b) we simply release the 4^{th} block. Thus an investment of 4 words we get back 3 words for a net investment of 1 word per zone. Thus to kill all the created areas of this type we need no more than x_{16} words.

For zones of type (17), after the created usable area is used, one of the following configurations results:

(a) X _ XY _ XX _ X _ XX or

(b) X _ X _ YXX _ X _ XX

In case (a) we release the 6^{th} block, and asking for a 4-block, resulting in either,

(a)-1 X _ XY _ XXY _ XX or

(a)-2 X _ XY _ XX _ YXX

In case (a)-1 we release the 1^{st} 2 blocks as well as the 5^{th} one, giving us 9 words back, but 4 of the 9 words must be used to kill the created usable area. Thus in the end we invested a total of 12 words to get 12 words back, for a net investment of 0 words to kill the created usable area. The same argument carries through for case (a)-2 except we release the 7^{th} block instead of the 5^{th}. In case (b) we simply release the 4^{th} block, thus giving us 3 words back for the 4 invested. It requires therefore 1 word in net investment to kill the created usable area. In conclusion, for zones of type (17) we require no more than x_{17} words to kill all the created usable areas.

For zones of type (18), after the created usable area is used, one of the following results:

(a) XY _ XYX _ X _ X _ X

(b) X _ YXYX _ X _ X _ X

In case (a) we release the 1^{st} block, giving us 3 words back for the 8 invested. So we need a net of 5 words in investments. The same argument again go through with the 3^{rd} instead of the 1^{st} released. So for all zones of this type we need no more than $5x_{18}$ words to kill all created usable areas.

For zones of type (19) one of the following results:

(a) XY _ XY _ XX

(b) X _ YXY _ XX

(c) XY _ X _ YXX

(d) X _ YX _ YXX

after the created usable area is used. In case (a) we release the 1^{st} blcok to get 3 words back for the 8 invested. The net investment is again 5 words. Case (b) is the same except the 3^{rd} block, instead of the 1^{st} is released. Cases (c) and (d) would be exactly the same except the 5^{th} block is released. In all, we need only $5x_{19}$ words in investment to kill all the created usable areas in zones of this type.

All together, for zones of types (10), (16), (17), (18), and (19) we therefore need at most $x_{10} + x_{16} + x_{17} + 5x_{18} + 5x_{19}$ words in investment to have all the created usable areas killed.

Now the overall density of all of these zones combined is:

$$\frac{12x_{10}+21x_{16}+24x_{17}+24x_{18}+18x_{19}}{14x_{10}+25x_{16}+29x_{17}+30x_{18}+22x_{19}+k+(3x_{10}+3x_{16}+3x_{17}+6x_{18}+6x_{19}-k)-(x_{10}+x_{16}+x_{17}+5x_{18}+5x_{19})}$$

$$= \frac{12x_{10}+21x_{16}+24x_{17}+24x_{18}+18x_{19}}{16x_{10}+27x_{16}+31x_{17}+31x_{18}+23x_{19}} \leq \frac{18}{23}$$

i.e. the $12x_{10}+21x_{16}+24x_{17}+24x_{18}+18x_{19}$ busy words took up

(1) the original $14x_{10}+25x_{16}+29x_{17}+30x_{18}+22x_{19}$ spaces

(2) plus the k spaces in existing usable areas

(3) plus the $3x_{10}+3x_{16}+3x_{17}+6x_{18}+6x_{19}$ words in released 3-blocks but k of them went into existing usable areas.

(4) minus the $x_{10}+x_{16}+x_{17}+5x_{18}+5x_{19}$ words needed to kill the created usable areas.

Since each zone is of density $\leq \frac{18}{23}$, the defender must use at least $\frac{23}{18}$ in spaces to avoid overflow. Q.E.D.

Acknowledgement: The author would like to thank Professor Kip Moore for his critical comments and ideas.

References

[1] J. M. Robson, "An Estimate of the Store Size Necessary for Dynamic Storage Allocation", J. ACM 3 (1971), pp. 416-423.

[2] J. M. Robson, "Bounds for Some Functions Concerning Dynamic Storage Allocation", J. ACM 3 (1974), pp. 491-499.

[3] J. M. Robson, personal correspondence.

[4] S. Krogdahl, "A Dynamic Storage Allocation Problem", Information Processing Letters 4 (1973), pp. 96-99.