Block Size	Blocks Needed	Total Space
5	745,151	10,086,381
6	475,263	9,998,531
7	283,559	9,920,446
8	164,660	10,101,104
9	94,105	10,537,195
10	53,399	11,179,987
11	30,496	11,974,072

Fig. 4. The number of extra blocks needed for 2^{20} vertices on a uniform distribution in 2D, and the total space required if we allocate 30% more blocks than are needed.

assumptions about the hash function, if the memory pool is at most 75% full, the probability that this technique will fail to find an $i \leq 127$ is at most $.75^{128} \simeq 10^{-16}$.

If the vertices are labeled sparsely (so that new labels can be generated dynamically), our implementation also makes use of a hash mapping between labels and vertex data blocks. One byte of memory is allocated per label; if the label is in use, this byte contains a hash pointer to the first data block for that vertex.

One bit is stored with each block to indicate whether the current block is the last in the sequence. For the first block this bit is stored with the degree of the vertex; for subsequent blocks it is stored as the eighth bit of the one-byte pointer to that block.

There is a tradeoff in the sizes of the blocks used. Large blocks are inefficient since they contain unused space; small blocks are inefficient since they require space for pointers to other blocks. In addition, there is a cost associated with computing hash pointers by searching for unused blocks in the memory pool. Figure 4 shows the tradeoff between these factors for our Delaunay triangulation algorithm run on 2^{20} uniformly distributed points in the unit square. We chose a block size of 7 since it gives the most efficient use of space.

To improve the efficiency of lookups our implementation uses a caching system. When a query or update is made, the blocks associated with the appropriate vertex are decoded. The information is represented in uncompressed form as a list with one vertex in the link per element of the list. The lists are kept in a FIFO cache with a maximum capacity of 2000 nodes. Update operations may affect the lists while they are in the cache. The lists are encoded back into blocks when they are flushed from the cache.

6.2. *3D Triangulation*

Our 3-dimensional structure is implemented as a slight generalization of our 2-dimensional structure. Recall that our 3D data structure keeps a map from each vertex v to all of its representative out-edges. This is stored as a difference coded