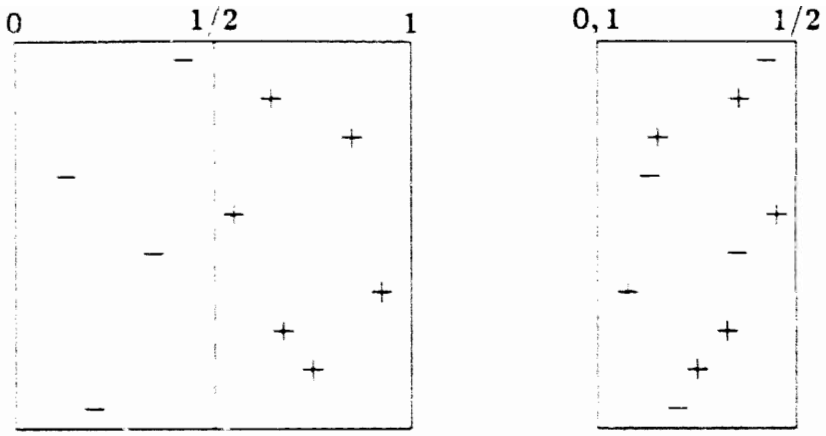

Figure 5.2.2: Representing items as points in the plane.

to the right of it, and furthermore that MBF finds a maximum such matching. Every + is above the − with which it is matched, since if two items are put in a bin by MBF, the first is larger than $\frac{1}{2}$ (i.e., is a + item) and the second is smaller than $\frac{1}{2}$ (i.e., a − item). Every + is to the right of the corresponding −, since if it was to the left, the two items would have sizes summing to more than 1 and so could not fit into a single bin. Furthermore, the algorithm MBF processes the −'s from top to bottom, matching each with the leftmost available +. (Here *available* means the + is above and to the right of the − and is still unmatched.) This is the algorithm shown in [KLM] to produce an optimal up-right matching. We will prove this in Lemma 5.2.5 below, using a different technique.

Since there is a bound of $O(n^{1/2}\log^{3/4} n)$ on the expected number of unmatched points in an optimal up-right matching, we obtain a bound of $n/2 + O(n^{1/2}\log^{3/4} n)$ on the number of bins used. ∎

We now show that the algorithm in MBF produces an optimal up-right matching. We use a different technique from the one in [KLM]. Our technique is slightly more complicated, but will generalize to a lemma that we need for the analysis of First Fit, which the easier technique cannot handle.

We first show that if there is a perfect matching, then the MBF algorithm

will find it.

Lemma 5.2.3: Suppose that there is a perfect up-right matching on some distribution of + and − points in a unit square. Then the MBF algorithm will find a perfect up-right matching on these points.

For the proof we will need a definition.

Definition 5.2.4: A pair of *crossed edges* in an up-right matching M is a pair of edges (a, b) and (c, d) where a and c are − points, b and d are + points, $y_a > y_c$ and $x_b > x_d$, and all possible $(-, +)$ edges, namely (a, b), (a, d), (c, b) and (c, d) are up-right. Here x_p and y_p are the x- and y-coordinates of point p.

A perfect matching found by the algorithm MBF will have no crossed edges. This is because the MBF algorithm processes the − point a before the − point c. The algorithm MBF will match a to the leftmost available +. Since neither a nor c is matched at this time, both b and d are available, and d is to the left of b. Thus, a pair of crossed edges will never be produced by the algorithm MBF.

Proof of Lemma 5.2.3 We now show that if there is a perfect up-right matching, then there is a perfect up-right matching with no crossed edges. To produce a perfect matching without crossed edges, given a perfect matching, all we do is repeatedly find pairs of crossed edges and uncross them. That is, if (a, b) and (c, d) are crossed, we replaced them by (a, d) and (c, b). If this process terminates, we are left with a perfect matching without crossed edges. To show it terminates, we show that the sum over all edges in our matching of the y-coordinate of the − point times the x-coordinate of the + point,

$$\sum_{e \in M} y_{e-} \cdot x_{e+}$$

always decreases when a pair of edges is uncrossed. This is because if (a, b) and (c, d) are crossed edges, then

$$y_a x_b + y_c x_d \geq y_a x_d + y_c x_b,$$

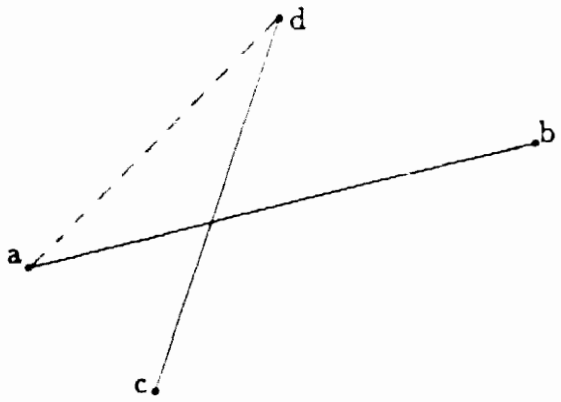

Figure 5.2.3: Crossed edges.

which follows from

$$(y_a - y_c)(x_b - x_d) \geq 0.$$

We now prove that if there exists a perfect matching with no crossed edges, then it is found by MBF. Suppose that there is a perfect up-right matching M with no crossed edges. Let M' be the up-right matching (not necessarily perfect) found by MBF. We wish to show that $M = M'$. Suppose they differ. Let a be the highest $-$ point such that its edge $(a, b) \in M$ is not also in M'. Since the point a is matched in M, and the $-$ points above a are matched to the same $+$ points in M and M', there was at least one available $+$ point (b) when point a was matched in M'. Therefore, a must be matched to some point, say d, in M'. The point d is in some edge in the perfect matching M. Let (c, d) be this edge. Then we claim (a, b) and (c, d) are crossed edges in M. (See Figure 5.2.3) We have $y_a > y_c$ since a is the highest $-$ point matched to different points in M and M'. We have $x_b > x_d$ since d was the leftmost available $+$ point when a was matched in M'. Clearly, the edges (a, b), (c, d) and (a, d) are up-right since they are edges in M and M'. The edge (c, b) is up-right since $x_b > x_d > x_c$ and $y_b > y_a > y_c$. We have found a pair of crossed edges in M, contradicting our assumption. Thus, a perfect matching with no crossed edges is found by the MBF algorithm.

Now, if there is a perfect up-right matching, there is also a perfect up-right matching with no crossed edges, and this matching is found by MBF. This proves the lemma. ∎

We now prove that MBF finds a maximum cardinality up-right matching on a set of + and − points.

Lemma 5.2.5: Given a placement of + and − points in the square, MBF finds a maximum cardinality up-right matching on these points.

Proof: By Lemma 5.2.1, if an item is added to list L' to obtain L, then $\#MBF(L) \leq \#MBF(L') + 1$. The number of 2-item bins in the packing, or edges in the matching $MBF(L)$, is $|L| - \#MBF(L)$. Since $|L| = |L'| + 1$, we get

$$|L| - \#MBF(L) \geq |L| - \#MBF(L') - 1 = |L'| - \#MBF(L').$$

Thus, if a point is added to a set of points, the cardinality of the matching found by the algorithm MBF does not decrease. Consider the subset $L'' \subseteq L$ of the points appearing in an optimal up-right matching M. By Lemma 5.2.3, MBF will find a perfect matching inL''. But since $L'' \subseteq L$, MBF matches at least as many points of L as of L'', so MBF finds an optimal matching of L. ∎

5.3. The Lower Bound

We now prove a lower bound on the wasted space for BF. We will do this by showing we can obtain an up-right matching from a Best Fit packing. Then a lower bound on the number of unmatched points in an up-right matching gives a lower bound on the number of bins used. We do this by showing that many bins are packed by putting a large item in the bin first and a small one second. This gives an up-right matching which has an expected value of $\Omega(\sqrt{n}\log^{3/4} n)$ unmatched points. These unmatched points give rise to wasted space.

In this section, we will say that the items in a bin are stacked in the same order as they were put in the bin. Thus, the first item to be put in a bin is on the *bottom* and the last item on the *top*. The *height* of a bin is the sum of the sizes of the items it contains.

For the proof, we will consider what happens to the items with size between $\frac{1}{3}$ and $\frac{2}{3}$. We will call items with size between $\frac{1}{3}$ and $\frac{1}{2}$ *s-items* (s for small) and items with size between $\frac{1}{2}$ and $\frac{2}{3}$ *b-items* (b for big). We now prove the following lemma.

Lemma 5.3.1: The probability of getting a bin with a b-item on top of an s-item is less than or equal to the probability of getting a bin with two s-items.

Proof: We can only get a bin with a b-item on top of an s-item if there is an s-item in a bin r which is less than half full. At any time, there can be only one such bin r. Suppose this bin is filled to height a. (the size of the s-item in the bin r can be smaller than a.) We claim that if the next item has size between a and $1-a$, it will go into r. Clearly, it will fit into the bin r. Furthermore, this bin is the only non-empty one with height less than $1-a$. This was certainly true when the first item was put into r, since otherwise it would not have been put there. It has remained true, since no items of size less than $1-a$ have started bins since. Thus, the probability of a b-item being put into the bin r is the probability of getting a b-item between $\frac{1}{2}$ and $1-a$. This is equal to the probability of getting an s-item between a and $\frac{1}{2}$, which is less than or equal to the probability of an s-item being put into r. ∎

We will call a bin with two s-items a $[^{s}_{s}]$ bin, and denote the number of these bins by $^{s}_{s}$. We will call a bin with a b-item on top of an s-item a $[^{b}_{s}]$ bin, and so on. If we restrict our attention to items in the range $(\frac{1}{3}, \frac{2}{3})$, there are five types of bins: $[^{s}_{s}]$, $[^{s}_{b}]$, $[^{b}_{s}]$, $[_{s}]$ and $[_{b}]$. There may also be non-empty bins with no items larger than $\frac{1}{3}$ in them. As we are proving a lower bound, we may ignore these

bins.

There is an equal probability of receiving an s-item and receiving a b-item. (The probability is 1/6.) By using bounds for the tail of the binomial distribution, we get that with probability at least $1 - 1/n$, the number of b-items and the number of s-items differ by $O(\sqrt{n \log n})$. It follows that

$$\left|\begin{matrix} \\ b \end{matrix}\right| = 2\left|\begin{matrix} s \\ s \end{matrix}\right| + \left|\begin{matrix} \\ s \end{matrix}\right| \pm O(\sqrt{n \log n}).$$

The previous lemma shows that at any time, the probability of a $\left[\begin{smallmatrix} b \\ s \end{smallmatrix}\right]$ bin is less than the probability of an $\left[\begin{smallmatrix} s \\ s \end{smallmatrix}\right]$ bin. Thus, with probability at least $1 - 1/n$,

$$\left|\begin{matrix} b \\ s \end{matrix}\right| \leq \left|\begin{matrix} s \\ s \end{matrix}\right| + O(\sqrt{n \log n}).$$

The probability of getting an s-item and the probability of getting a b-item are both $\frac{1}{6}$. Thus, with probability $1 - 1/n$, there are $n/3 \pm O(\sqrt{n \log n})$ b-items and s-items total (items with size between $\frac{1}{3}$ and $\frac{2}{3}$). The $\left[\begin{smallmatrix} s \\ b \end{smallmatrix}\right]$ bins form an up-right matching. By Theorem 3.1.1 on up-right matching, with high probability there must be $\Omega(\sqrt{n} \log 3/4n)$ unmatched points. Thus, we have

$$\left|\begin{matrix} s \\ b \end{matrix}\right| = \frac{n}{6} - \Omega(\sqrt{n} \log^{3/4} n).$$

We have that the total number of b-items is $n/6 \pm O(\sqrt{n \log n})$. This gives

$$\left|\begin{matrix} s \\ b \end{matrix}\right| + \left|\begin{matrix} \\ b \end{matrix}\right| + \left|\begin{matrix} b \\ s \end{matrix}\right| = \frac{n}{6} \pm O(\sqrt{n \log n}).$$

Combining the two above equations, we obtain

$$\left|\begin{matrix} \\ b \end{matrix}\right| + \left|\begin{matrix} b \\ s \end{matrix}\right| = \Omega(\sqrt{n} \log^{3/4} n).$$

Using the inequalities on $\left|\begin{smallmatrix} \\ b \end{smallmatrix}\right|$ and $\left|\begin{smallmatrix} b \\ s \end{smallmatrix}\right|$ obtained above, we get

$$3\left|\begin{matrix} s \\ s \end{matrix}\right| + \left|\begin{matrix} \\ s \end{matrix}\right| = \Omega(\sqrt{n} \log^{3/4} n).$$

Thus, there are $\Omega(\sqrt{n} \log^{3/4} n)$ bins containing no items larger than $\frac{1}{2}$. Since the number of items larger than $\frac{1}{2}$ is with high probability $n/2 \pm O(\sqrt{n \log n})$, this shows that the number of bins used is with high probability $n/2 \pm O(\sqrt{n} \log^{3/4} n)$.

∎

Chapter 6. First Fit

6.1. Introduction

In the algorithm First Fit (FF), the bins are kept in order. As each item arrives, it is placed into the first bin in which it fits. If it does not fit in any bin, it is placed in an empty bin at the end of list of bins. We also define 2-First Fit (2FF) and Matching First Fit (MFF) as we did for Best Fit. In 2FF, only two items can go into a bin, and items are packed into the first bin they fit in satisfying this constraint. In MFF, no item can be put into a bin that has an item smaller than $\frac{1}{2}$ in it.

In this chapter, we prove the following:

Theorem 6.1.1: The expected wasted space produced by the algorithm First Fit when packing n items that are uniformly distributed on $[0,1]$ is $\Omega(n^{2/3})$ and $O(n^{2/3}\log^{1/2} n)$.

The upper bound is obtained in a manner similar to that of Best Fit, and the lower bound uses some new techniques. We will discuss the upper bound first.

6.2. The Upper Bound

The proof of the upper bound for First Fit is quite similar to the proof for Best Fit. As before, we convert the problem to a matching problem. We

then prove an upper bound on the matching problem by producing a suitable matching.

We use MFF to obtain the upper bound. Lemma 5.2.1 holds for 2FF and MFF as well as for 2BF and MBF. The proof is identical.

Lemma 6.2.1: If L' is a list obtained by removing one item from L, then

$$\#2FF(L) \geq \#2FF(L') \geq \#2FF(L) - 1$$

and

$$\#MFF(L) \geq \#MFF(L') \geq \#MFF(L) - 1.$$

The proof that $\#MFF(L) \geq \#FF(L)$ proceeds exactly as did the proof for Best Fit; $\#2FF(L) \geq \#FF(L)$ is shown in [BJLMM]. As before, we represent items as points in the plane with x-coordinate size and y-coordinate time. We then mark items greater than $\frac{1}{2}$ with $+$ and items less than $\frac{1}{2}$ with $-$ and fold the plane around the line $x = \frac{1}{2}$, as in Figure 5.2.2. We derive a matching by connecting a $+$ point and a $-$ point corresponding to two items in the same bin in MFF. This will be an up-right matching for the same reasons that MBF was up-right. However, MFF differs from MBF in that it used a different algorithm to find the matching. The MFF matching is derived by processing the $-$'s from top to bottom, and matching each $-$ with the *highest* available $+$. This algorithm no longer produces a maximum cardinality up-right matching. We will show that it produces a maximum cardinality up-right matching satisfying the following condition:

Condition 1: If $e_1 = (a, b)$ and $e_2 = (c, d)$ are $(-, +)$ edges and their y-coordinates satisfy $y_a < y_c < y_d < y_b$, then their x-coordinates satisfy $x_b < x_c$. (See Figure 6.2.1.)

The reason that any MFF matching satisfies this condition is that if c were to the left of b, then c would have been matched with b rather than with d. The

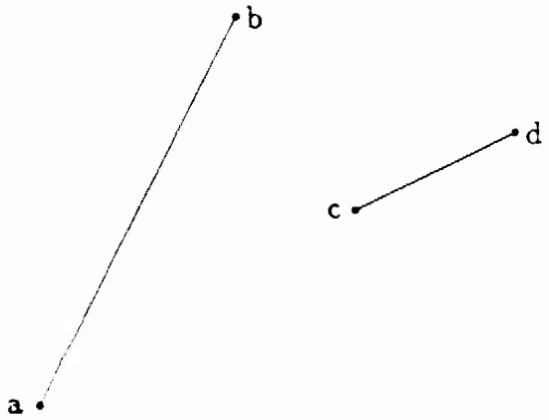

Figure 6.2.1: Condition 1.

proof that MFF gives a maximum cardinality matching satisfying this condition follows.

Lemma 6.2.2: The algorithm MFF gives an up-right matching satisfying Condition 1 of maximum cardinality.

We will first show that any *perfect* matching satisfying Condition 1 is a MFF matching. Then we will use this to prove the lemma.

Claim 6.2.3: If a perfect up-right matching M satisfying Condition 1 exists, it will be found by the algorithm MFF.

Proof of Claim: Suppose we have a perfect matching M satisfying Condition 1. We show that if a $-$ point is not matched with the highest available $+$ point, then there is a pair of edges violating Condition 1. Consider the highest $-$ point which was not matched according to the rules of MFF. Call this $-$ point c. Since M is a perfect matching, this $-$ must be matched to some $+$ point by M, say point d. Let b be the $+$ point that point c would have been matched to using the MFF rule. Since b was the highest available $+$ point when we processed c, point b must be higher than point d. Since it was unmatched when we processed c, point b must be matched in M to a $-$ point below c, say a. Then (a, b) and (c, d) form a pair of edges in M violating Condition 1. This contradicts our

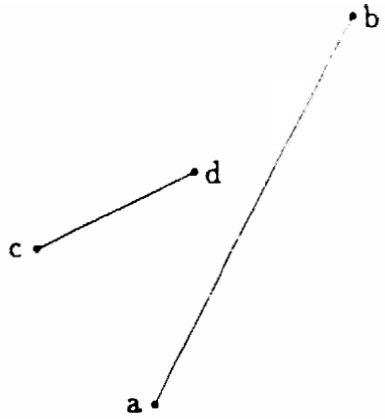

Figure 6.2.2: Condition 2.

assumption and thus proves the claim. ∎

Proof of Lemma: Let L be the list of items to be packed. The proof of Lemma 6.2.1 applies to MFF as well as to 2FF. Thus, if m items are added to a list L', then at most m bins are added to the packing given by MFF. Let M be a maximum cardinality up-right matching satisfying Condition 1. Let L' be those items of L matched by M. By the above Claim, MFF must match every item in L'. By Lemma 6.2.1, MFF must match at least as many items of L as it does of L'. Thus, MFF produces a maximum cardinality matching. ∎

Next, we prove that if we have a up-right matching satisfying Condition 2 given below, we can obtain an equal cardinality matching satisfying Condition 1. We do this by successive switches of pairs of edges that satisfy Condition 2 but not Condition 1.

Condition 2: If $e_1 = (a, b)$ and $e_2 = (c, d)$ are $(-, +)$ edges, and their y-coordinates satisfy $y_a < y_c < y_d < y_b$, then their x-coordinates satisfy $x_a < x_d$. (See Figure 6.2.2.)

Any two edges satisfying Condition 1 must also satisfy Condition 2.

We now prove the following lemma:

Lemma 6.2.4: Given a set of + and − points in a unit square, suppose that

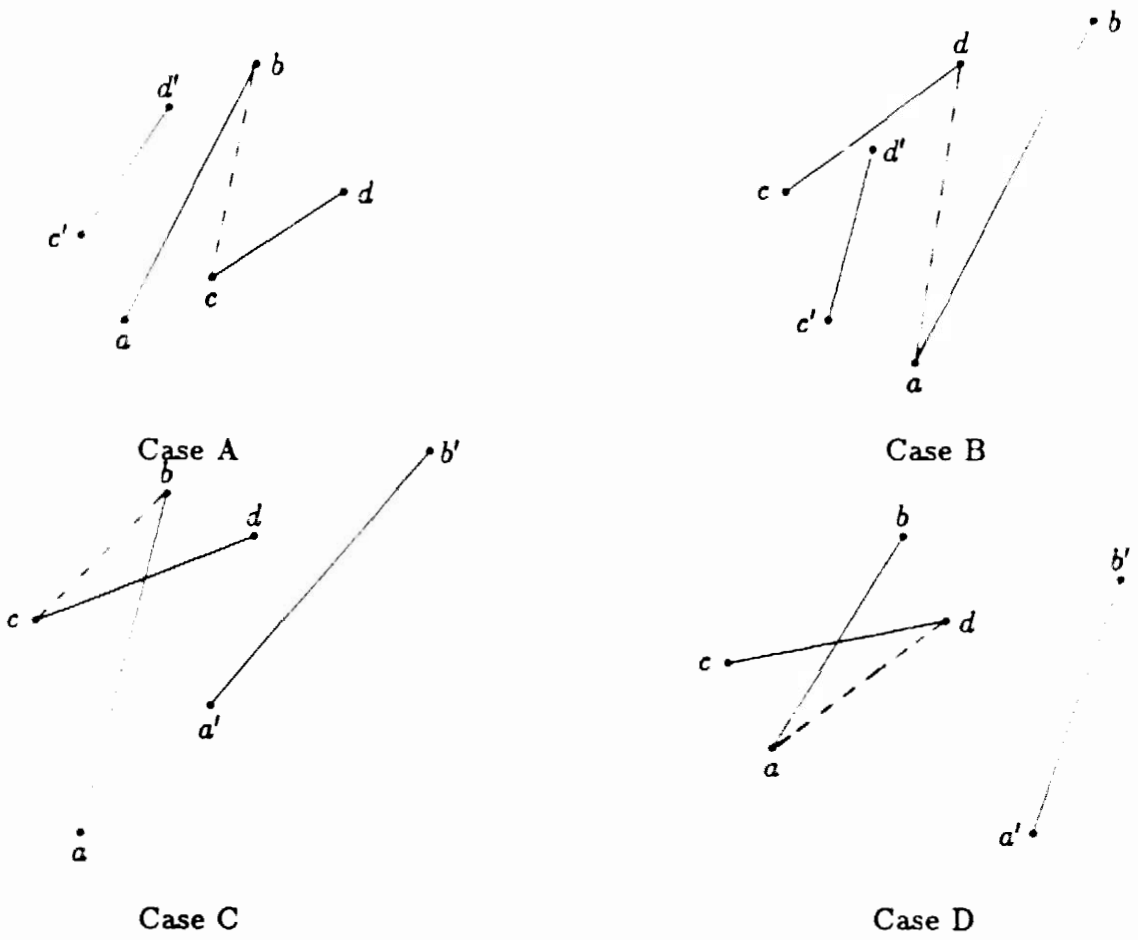

Figure 6.2.3:

there is an up-right matching satisfying Condition 2. Then there is an up-right matching of equal cardinality satisfying Condition 1.

Proof: Suppose that we have an up-right matching satisfying Condition 2 but not Condition 1. Consider all pairs of edges e_1 and e_2 which violate Condition 1. Of these, choose for e_1 an edge (a, b) such that b is the rightmost point of all such edges. Next, choose for e_2 an edge (c, d) such that (a, b) and (c, d) violate Condition 1, and c is the leftmost point in all such (c, d) edges. (We require a, b, c, d to be the points in the order stated in Condition 1.) Replace (a, b) and (c, d) with (a, d) and (c, b). We claim that this new matching satisfies Condition 2. Furthermore, by repeating this step, we will eventually obtain a matching that satisfies Condition 1.

We must first show that the new matching still satisfies Condition 2. This is done by case analysis. There are four cases, all of which are relatively easy. We must show, for example, that there is no edge (c', d') such that (c, b) and (c', d') violate condition 2.

Case A: Suppose (c, b) and (c', d') do not satisfy Condition 2, with (c', d') the edge e_2 as stated in the condition. Then we must have c' to the left of c. But (a, b) and (c', d') now violate Condition 1. This contradicts our choice of c as the leftmost point violating Condition 1 with (a, b). (See Figure 6.2.3A.)

Case B: Suppose (a, d) and (c', d') do not satisfy Condition 2, with $e_1 = (a, d)$ and $e_2 = (c', d')$. Then (a, b) and (c', d') violate Condition 2, contradicting our assumption that we had a matching satisfying Condition 2. (See Figure 6.2.3B.)

Case C: Suppose (a', b') and (c, b) do not satisfy Condition 2, with $e_1 = (a', b')$ and $e_2 = (c, b)$. Then a' is to the right of b, implying that b' is to the right of b. Now (a', b') and (c, d) violate Condition 1, contradicting our choice of b as the rightmost point in an edge $e_1 = (a, b)$ violating Condition 1. (See Figure 6.2.3C.)

Case D: Suppose $e_1 = (a', b')$ and $e_2 = (a, d)$ do not satisfy Condition 2. Then we have a' to the right of d. Thus, (a', b') and (c, d) violate Condition 2, a contradiction. (See Figure 6.2.3D.)

We can continue the above process of switching pairs of edges that do not satisfy Condition 1 as long as our matching does not satisfy Condition 1. Thus, if the process terminates, the resulting matching must satisfy Condition 1. It is a simple calculation to see that the replacement always reduces the sum $\sum_e (y_{e-} - y_{e-})^2$ over all edges $e = (e_+, e_-)$ in the matching. Since there are a finite number of matchings, this sum cannot decrease forever. Thus, the process terminates in

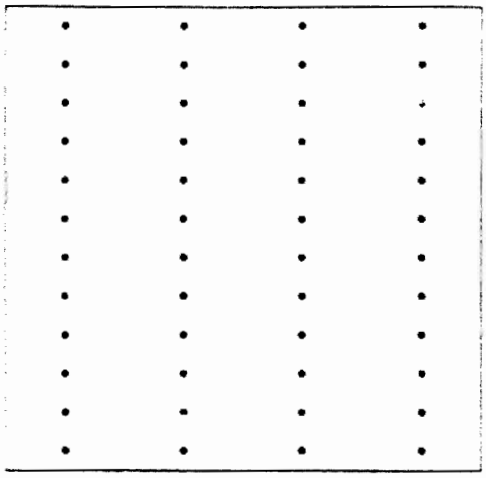

Figure 6.2.4: A $u \times v$ grid in the square.

a matching satisfying Condition 1. ∎

We will now produce a matching satisfying Condition 2 and using all but $O(n^{2/3}\log^{1/2} n)$ points. We will need the following lemma on grid matching to prove this. This lemma is proved in Part I of this thesis.

Lemma 6.2.5: Suppose n points are randomly distributed in a unit square, and there is a $\sqrt{n} \times \sqrt{n}$ grid imposed on this square. Then with probability at least $1 - 1/n$ we can match each point to a grid point with edges of length $O(\log^{3/4} n/\sqrt{n})$.

We now produce the matching satisfying Condition 2. Assume that we have $2n$ points, approximately half +'s and half −'s, in the unit square. We will first lay out a grid of $n^{1/3}\log^{1/4} n \times n^{2/3}\log^{-1/4} n$ points, with $u = n^{1/3}\log^{1/4} n$ points in each row and $v = n^{2/3}\log^{-1/4} n$ points in each column. (See Figure 6.2.4) From the lemma above, we show that by using edges that pass at most $k\log^{3/4} n$ grid points, we can match all but $n^{2/3}\log^{1/4} n$ of the + points and of the − points to grid points. We do this by breaking the grid into $v/u = n^{1/3}\log^{-1/2} n$ square grids of size $u \times u$ and matching points within each of these using the above lemma. (See Figure 6.2.5.) With probability $1 - 1/n^2$, in each of these

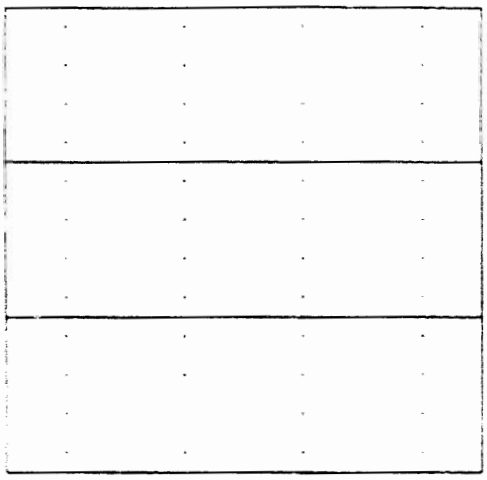

Figure 6.2.5: Breaking the grid into square grids.

square grids the number of grid points, + points, and − points will differ by $O(u\sqrt{\log u}) = O(n^{1/3}\log^{3/4} n)$. Using the Lemma above, we can match + points to grid points in each of the square grids so that each edge passes $O(\log^{3/4} n)$ grid points and so that only the $O(n^{1/3}\log^{3/4} n)$ extra points in each grid are unmatched. We do the same for the − points. There are $n^{1/3}\log^{-1/2} n$ square grids, so the total number of unmatched points is $O(n^{2/3}\log^{1/4} n)$.

We now have nearly all the + points and nearly all the − points matched to grid points with edges that move only $k\log^{3/4} n$ grid points horizontally and vertically, where k is a constant. Let $l = k\log^{3/4} n$. We match the + points to the − points by matching the + point matched to one grid point to the − point matched to a nearby grid point. In order to ensure an upward right matching, we must match a − point to a + point l grid points up and to the right. We can ensure a matching satisfying Condition 2 by making the edges towards the left side of the square longer vertically than the ones towards the right. If we do this properly, we match all but $O(n^{2/3}\log^{1/2} n)$ of the points.

We match the − point matched to the (i, j) grid point to the + point matched to the $(i + 7l, j + u - i)$ grid point, where $u = n^{1/3}\log^{1/4} n$ is the number of