Theory and Applications of
Mathematics & Computer Science

(ISSN 2067-2764, EISSN 2247-6202)
http://www.uav.ro/applications/se/journal/index.php/tamcs

Theory and Applications of Mathematics & Computer Science 4 (2) (2014) 140–153

Counting Sets of Lattice Points in the Plane with a Given Diameter under the Manhattan and Chebyshev Distances

Mugurel Ionuţ Andreica[a,*]

[a]*Computer Science Department, Politehnica University of Bucharest, Splaiul Independenţei 313, RO-060042, sector 6, Bucharest, Romania.*

Abstract

In this paper we present new algorithms for counting the sets of lattice points in the plane whose diameter is a given value D, under the Manhattan (L_1) and Chebyshev (L_∞) distances. We consider two versions of the problem: counting all sets within a given lattice $U \times V$, and counting all sets that are not equivalent under translations.

Keywords: lattice points, Chebyshev distance, Manhattan distance, diameter.
2010 MSC: 52Cxx, 11P21.

1. Introduction

In this paper we present new algorithms for counting the sets of lattice points in the plane with a given diameter, under the Manhattan (L_1) and Chebyshev (L_∞) distances. We consider two versions of the problem. In the first version we assume that a fixed size 2D grid is given and the sets must be placed inside this grid. Two sets are different if they have a different number of points or the positions of their points inside the grid are not all identical. In the second version we assume that two sets are considered identical (and, thus, need to be counted only once) if one can be obtained from another by translation operations.

The rest of this paper is structured as follows. In Section 2 we present the problems in more details, together with some preliminaries required by the algorithms presented in the other sections. In Section 3 we present an algorithm with $O(D \cdot log(D))$ arithmetic operations for the Chebyshev (L_∞) distance which can solve both versions of the problem. In Section 4 we present a more efficient algorithm, with only $O(log(D))$ arithmetic operations, for the Chebyshev distance, but only for the second version of the problem. In Sections 5 and 6 we present algorithms with a similar number of arithmetic operations for the Manhattan distance and for the same versions of

[*]Corresponding author
Email address: `mugurel.andreica@cs.pub.ro` (Mugurel Ionuţ Andreica)

the problem. In Section 7 we present experimental results regarding the two algorithms for the Manhattan distance. In Section 8 we discuss related work. In Section 9 we conclude and discuss future work.

2. Problem Statement and Preliminaries

In this paper we consider sets of lattice points in the plane. A lattice point is a point with integer coordinates. The diameter of a set of points is the maximum distance between any two points in the set. In this paper we will consider two distances. The Manhattan distance (also called the L_1 distance) between two points (x_1, y_1) and (x_2, y_2) is defined as $|x_1 - x_2| + |y_1 - y_2|$. The L_∞ distance (also called the Chebyshev distance) between two points (x_1, y_1) and (x_2, y_2) is defined as $max\{|x_1 - x_2|, |y_1 - y_2|\}$. When the coordinates of the points are integer (i.e. when we consider only lattice points) both the L_1 and the L_∞ distances are integers.

We consider two versions of the problem for counting sets of lattice points having exactly a given diameter D (under the L_1 or L_∞ distances). The first version assumes that a 2D grid of fixed size $U \times V$ is given (U is the number of points along the OX axis and V is the number of points along the OY axis). We may assume that the points of the grid have coordinates (x, y) with $0 \leq x \leq U - 1$ and $0 \leq y \leq V - 1$. In this case two sets of points are considered different if they consist of a different number of points or if the positions of their points are not all identical. The dimensions of the grid (U and V) are part of the input of the algorithms presented for this version.

For the second version we assume that two sets A and B are identical if one can be obtained from another by translation operations. To be more precise, set A is considered identical to B if there exist the integer numbers TX and TY such that by adding TX to the x-coordinate of each point of A and TY to the y-coordinate of each point of A we obtain exactly the set B (note that this automatically implies that A and B have the same number of points). In this case the sets are not constrained to be located within a fixed size grid, so the parameters U and V from the first version of the problem do not exist here.

We are interested in computing the number of sets of points with a given value of the diameter D ($D \geq 1$) under both versions of the problem and considering either the L_1 or the L_∞ distance. Let's consider, for instance, the second version of the problem. For $D = 1$ there are two sets of lattice points for the Manhattan distance, each consisting of two adjacent lattice points. In the first set the two points are horizontally adjacent and in the second set the two points are vertically adjacent. On the other hand, there are 9 sets of lattice points for $D = 1$ and the Chebyshev distance.

In order for a set of points in the plane to have diameter D under the L_∞ distance all the points must be located inside a square of side length D and for at least one pair of opposite sides there must be at least one point from the set located on each of the two sides.

The diameter of a set of points A under the Manhattan distance is equivalent to the diameter under the L_∞ distance of a modified set of points B (Indyk, 2001). B is obtained by transforming each point (x, y) of A into the point $(x - y, x + y)$ in B. Thus, the two problems considered in this paper are strongly connected to each other. The transformed coordinates correspond to *diagonal coordinates*.

In a 2D plane we have two types of diagonals: *main diagonals* (running from north-east to south-west) and *secondary diagonals* (running from north-west to south-east). All the points (x, y)

on the same *main diagonal* have the same value of $x - y$ and all the points (x, y) on the same *secondary diagonal* have the same value of $x + y$. The *index of a main diagonal* is the difference $x - y$ of all the points (x, y) on it. Similarly, the *index of a secondary diagonal* is the sum $x + y$ of all the points (x, y) on it. The *parity* of a diagonal (main or secondary) is defined as the parity of its index. The distance between two diagonals of the same type (main or secondary) is defined as the absolute difference of their indices.

After the transformation to *diagonal coordinates* we can easily see that in order for a set of points to have exactly diameter D under the Manhattan distance one of the following conditions must hold:

1. It should have at least two points located on main diagonals at distance D apart, while the other pairs of main diagonals of all the points are located at distance at most D apart and the pairs of secondary diagonals of all the points are at distance strictly less than D or
2. It should have at least two points located on secondary diagonals at distance D apart while the other pairs of secondary diagonals of all the points are located at distance at most D apart and the pairs of main diagonals of all the points are at distance strictly less than D or
3. It should have at least two points located on main diagonals at distance D apart and at least two points located on secondary diagonals at distance D apart and all the other pairs of main and secondary diagonals of the points are at distance at most D apart.

The three cases correspond to different types of sets of points (i.e. each set of points having diameter D under the Manhattan distance belongs to exactly one of the three cases). Note that there is a bijection between the sets of points corresponding to cases 1 and 2. Each set of points corresponding to case 1 can be transformed into a set of points corresponding to case 2 (by switching the order of the diagonals). Similarly, each set of points corresponding to case 2 can be transformed into a set of points corresponding to case 1.

Thus, for the second version of the problem, it will be enough to count the number of sets of points corresponding to case 1 (C_1) and the number of sets of points corresponding to case 3 (C_3) in order to obtain the total number of sets of points having diameter D under the Manhattan distance. Then, the total number of sets of lattice points having diameter D (under the Manhattan distance) is equal to $2 \cdot C_1 + C_3$.

3. Algorithm 1 for Counting Sets of Lattice Points of Diameter D under the Chebyshev (L_∞) Distance

In this section we will present an algorithm which computes the number of sets of lattice points having diameter D under the Chebyshev distance for both versions of the problem. The algorithm will make use of a function denoted by $CNTSETS(LX, LY)$ which will compute the number of sets of lattice points contained in a rectangle having side length LX along the OX axis and side length LY along the OY axis and such that each counted set has at least one point on each of the 4 sides of the rectangle. Moreover, the corners of the rectangle are lattice points.

We will start with some simple cases. We have $CNTSETS(0, 0) = 1$ and $CNTSETS(P, 0) = CNTSETS(0, P) = 2^{P-1}$. When both LX and LY are greater than or equal to 1 we will use the following approach. We will first identify the 4 corners of the rectangle. We will consider each of the 2^4 binary configurations of 4 bits. Let BC denote the current binary configuration and $BC(i)$ will denote bit i in the configuration $(0 \leq i \leq 3)$. Each bit will correspond to one of the 4 corners. If $BC(i)$ is 1 we will assume that the corresponding corner is selected to be part of the set; otherwise we will assume that it is not selected. Lets consider now each of the horizontal sides of the rectangle. If at least one corner located on the considered side was selected, then there are 2^{LX-1} possibilities left for selecting the remaining points (non-corners) of the horizontal side (note that the side contains $LX + 1$ points overall, out of which 2 are corners). If none of the corners of the side are selected, then there are only $2^{LX-1} - 1$ possibilites left for selecting the remaining points of the horizontal side. The situation is similar for the vertical sides: if at least one corner is selected from a vertical side, there are 2^{LY-1} possibilities of selecting the remaining points of the vertical side; otherwise the number of possibilities is only $2^{LY-1} - 1$. After considering all the 4 sides of the rectangle we need to consider the points located strictly inside the rectangle. There are $NIN = (LX - 1) \cdot (LY - 1)$ points located strictly inside the rectangle. Each of these inner points may be selected or not, meaning that there are 2^{NIN} possibilities of selecting these points. For a given binary configuration BC the number of possibilities of selecting points according to it is equal to the product of five terms: four of which are the number of possibilities corresponding to each of the 4 sides of the rectangle and the 5th term corresponds to the number of possibilities of selecting the inner points of the rectangle. The value returned by $CNTSETS(LX, LY)$ is equal to the sum of the numbers of possibilities of selecting points corresponding to each of the 2^4 binary configurations.

We will use a variable C ranging from 0 to D. For each value of C we will first compute $CNTSETS(D, C)$. Note that this value corresponds to the number of sets of lattice points having diameter D and which are contained in a minimum bounding rectangle of side lengths D (along the OX axis) and C (along the OY axis). For the first version of the problem, each set counted by $CNTSETS(C, D)$ may appear multiple times inside the grid - in fact, it may appear $(U-D)\cdot(V-C)$ times (as that's the number of possibilities of placing a $D \cdot C$ rectangle inside the grid). Thus, we will add the term $CNTSETS(D, C) \cdot (U - D) \cdot (V - C)$ to the final answer for the first version of the problem (or 0, if $D > U$ or $C > V$). In the second version of the problem we simply need to add $CNTSETS(D, C)$ to the final answer for the second version of the problem. This is because all the sets counted by $CNTSETS(D, C)$ are different under translation operations.

If $C < D$ we will also compute $CNTSETS(C, D)$ (which is identical in value to $CNTSETS(D, C)$). For the first version of the problem we will add to the final answer the value $CNTSETS(C, D) \cdot (U - C) \cdot (V - D)$ (or 0, if $C > U$ or $D > V$). For the second version of the problem we will add to the final answer the value $CNTSETS(C, D)$.

The algorithm presented in this section uses $O(D \cdot log(D))$ arithmetic operations, because it considers $O(D)$ cases and for each case it needs to perform a constant number of exponentiations where the base 2 logarithm of the exponent is of the order $O(log(D))$. All the exponentiations raise 2 to a given exponent. If D is not very large we may consider precomputing all the powers of 2 from 0 to D (we may achieve this with only $O(D)$ multiplications because we can write $2^P = 2^{P-1} \cdot 2$ for $P \geq 1$ and we can consider the values of P in ascending order). However, NIN is of the order $O(D^2)$. If D is sufficiently small then we may precompute powers of 2 up to D^2 (using $O(D^2)$ multiplications). If, however, D^2 is too large, then we need to notice that, as C increases from 0 to D, NIN also increases. We will assume that our algorithm considers the values of C in ascending order (note that NIN is the same for both $CNTSETS(D, C)$ and $CNTSETS(C, D)$). Let's assume that $PREVNIN$ is equal to the value of NIN for the case $C - 1$ and $RESPREVNIN = 2^{PREVNIN}$. We will initially (for $C = 0$) have $PREVNIN = 0$ and $RESPREVNIN = 1$. When we need to compute 2^{NIN} for a case we will first compute the difference $DIFNIN = NIN - PREVNIN$. We will always have $DIFNIN = D - 1$. Thus, we can compute 2^{NIN} with only one multiplication, as $RESPREVNIN \cdot 2^{DIFNIN}$ (note that 2^{DIFNIN} is taken from the table of precomputed powers of two). After handling the current value of C we will update $PREVNIN = NIN$ and $RESPREVNIN = 2^{NIN}$ (where 2^{NIN} was just computed by the method we presented). Using this approach we only need $O(D)$ arithmetic operations instead of $O(D \cdot log(D))$.

So far we assumed that we want to compute the number of sets of lattice points exactly. In this case we will need to work with numbers which have $O(4 \cdot D)$ bits. However, there are many situations when the exact numbers are not required. For instance, if we are only interested in computing the number of sets modulo a given number M, then we only need numbers having $O(2 \cdot log(M))$ bits for storing intermediate and final results. If M is sufficiently small (e.g. a 32-bit number) then we can practically assume that on the current machine architectures the numbers we use have a constant number of bits. However, the exponents to which 2 is raised can still be pretty large numbers (having $O(log(D))$ bits). This may not necessarily be a problem, but we may inadvertently face some challenging algorithmic problems. For instance, when multiplying $(LX - 1)$ by $(LY - 1)$ in the $CNTSETS$ function we need to multiply together two numbers having $O(log(D))$ bits. The naive algorithm would use $O(log^2(D))$ time for computing the result. In order to speed up the multiplication we may need to use more complicated algorithms (Schonhage & Strassen, 1971), (Furer, 2009) which reduce the time complexity to $O(log(D) \cdot log(log(D)) \cdot log(log(log(D))))$ or slightly better. Nevertheless, there is a simple situation when all these complications are not needed: when M is an odd prime. In this case we know that $A^{M-1} = 1$ (modulo M) for any natural number $1 \leq A \leq M - 1$. Since we only need to raise 2 at some powers (modulo M), we notice that we only need the remainder of the exponent when divided by $M - 1$ in order to compute the required result. Thus, instead of using exact exponents we will only use the exponents modulo $M - 1$. This way we can avoid the complicated multiplication of $(LX - 1)$ by $(LY - 1)$ and replace it with the multiplication of *((LX-1) mod (M-1))* by *((LY-1) mod (M-1))*. This way we will need to spend $O(log(D))$ time in order to compute the remainders of numbers having $O(log(D))$ bits when divided by $M - 1$, but we do not need to multiply together two large numbers.

4. Algorithm 2 for Counting Sets of Lattice Points of Diameter D under the Chebyshev (L_∞) Distance

The algorithm presented in this section can only solve the second version of the problem (i.e. when two sets are identical if one can be obtained from another by using translation operations). We will first define the following function: $NSETS(LX, LY)$=the number of sets of lattice points contained in a rectangle of horizontal side length LX and vertical side length LY such that at least one point is located on each of the opposite vertical sides (for this function we will ignore the fact the two sets are identical if one can be obtained from another by translation operations). We assume $LX \geq 1$ and $LY \geq 0$, both numbers are integers and the corners of the rectangle are lattice points. Such a rectangle contains $(LX + 1) \cdot (LY + 1)$ lattice points inside of it or on its borders. It is easy to see that $NSETS(LX, LY) = (2^{LY+1} - 1)^2 \cdot 2^{(LX+1)\cdot(LY+1)-2\cdot(LY+1)}$. This formula corresponds to the following cases. On each of the two opposite vertical sides we must have one selected point. Thus, there are $2^{LY+1} - 1$ possibilities of choosing lattice points on each of these two sides. Each of the remaining $(LX + 1) \cdot (LY + 1) - 2 \cdot (LY + 1)$ lattice points can be selected or not to be part of the set. Thus, we have $2^{(LX+1)\cdot(LY+1)-2\cdot(LY+1)}$ possibilities for selecting these points. If $LY < 0$, by definition, we will have $NSETS(LX, LY) = 0$.

In order for a set of points in the plane to have diameter D under the Chebyshev distance all the points must be located inside a square of side length D, such that at least one pair of opposite sides has at least one point from the set on each side from the pair. We will consider three cases:

1. both of the vertical opposite sides of the square contain points from the set on them, but not both horizontal sides of the square contain points from the set: the number of sets corresponding to this case is $NSETS(D, D-1) - NSETS(D, D-2)$ (this forces every set to have a point selected on the bottom side of the square of side length D)
2. both of the horizontal opposite sides of the square contain points from the set on them, but not both vertical sides of the square contain points from the set: the number of sets corresponding to this case is also $NSETS(D, D - 1) - NSETS(D, D - 2)$.
3. both of the horizontal opposite sides and both of the vertical opposite sides of the square contain points from the set on them: the number of sets corresponding to this case is $NSETS(D, D) - 2 \cdot NSETS(D, D - 1) + NSETS(D, D - 2)$. We actually made use of the inclusion-exclusion principle here. From all the sets of lattice points with points on both opposite vertical sides ($NSETS(D, D)$) we subtracted the sets of lattice points which do not have points on the top or bottom horizontal side ($2 \cdot NSETS(D, D - 1)$). In doing this we over-subtracted the sets of lattice points which do not have points on any of the horizontal sides ($NSETS(D, D - 2)$) thus, we need to add this number back.

By adding together the numbers corresponding to the cases 1, 2 and 3, we obtain the total number of sets of lattice points having diameter D under the Chebyshev distance: $2 \cdot (NSETS(D, D - 1) - NSETS(D, D - 2)) + NSETS(D, D) - 2 \cdot NSETS(D, D - 1) + NSETS(D, D - 2)$, which simplifies to $NSETS(D, D) - NSETS(D, D - 2)$.

This method requires $O(log(D^2)) = O(log(D))$ arithmetic operations in order to compute the answer (this number corresponds to raising 2 to a power whose value is of the order $O(D^2)$). In case exact results are not needed, the same discussion from the previous section applies to this case, too, because in the $NSETS$ function we need to multiply two numbers of $O(log(D))$ bits each: $(LX + 1)$ and $(LY + 1)$.

5. Algorithm 1 for Counting Sets of Lattice Points of Diameter D under the Manhattan (L_1) Distance

In this section we present an algorithm similar in essence to the one from section 3. The algorithm can compute the number of sets of lattice points having diameter D under the Manhattan distance for both versions of the problem. The algorithm will make use of a function $CNTEQ(C, X)$ which computes the number of sets of lattice points such that:

- the main diagonals of at least two points are at distance exactly D apart

- all the other pairs of main diagonals of the points are at distance at most D apart
- the secondary diagonals of at least two points are at distance exactly C apart
- all the other pairs of secondary diagonals of the points are at distance at most C apart
- $X = 0$ means that the parity of the first secondary diagonal is equal to the parity of the first main diagonal, while $X = 1$ means that these parities differ (the first diagonal of each type is the one with the smallest index)

The algorithm will simply iterate through all the values of C (from 0 to D), and for each value of C, through all the values of X (from 0 to 1).

For the first version of the problem we will need to compute the minimum bounding rectangle for the sets counted by $CNTEQ(C, X)$ $(X = 0, 1)$. Let's assume that the minimum bounding rectangle has side length $MBRX$ along the OX axis and $MBRY$ along the OY axis. We will add to the final answer the value $CNTEQ(C, X) \cdot (U - MBRX) \cdot (V - MBRY)$ $(X = 0, 1)$, or 0 if $MBRX > U$ or $MBRY > V$. If $C < D$ then we have a set of symmetric sets of lattice points by switching the role of main and secondary diagonals. These sets have a minimum bounding rectangle with side length $MBRY$ along the OX axis and $MBRX$ along the OY axis. Thus, we will also add to the final answer the value $CNTEQ(C, X) \cdot (U - MBRY) \cdot (V - MBRX)$ $(X = 0, 1)$, or 0 if $MBRX > V$ or $MBRY > U$.

For the second version of the problem C_1 will be equal to the sum of the values $CNTEQ(C, X)$ $(0 \leq C \leq D - 1, 0 \leq X \leq 1)$ and C_3 will be equal to $CNTEQ(D, 0) + CNTEQ(D, 1)$.

When computing $CNTEQ(C, *)$, we need to consider a figure containing lattice points enclosed by a pair of main diagonals at distance D and a pair of secondary diagonals at distance C. We will denote the first main diagonal as the *left* diagonal, the second main diagonal as the *right* diagonal, the first secondary diagonal as the *bottom* diagonal and the second secondary diagonal as the *top* diagonal. We will need to compute the following numbers:

- NLEFT=the number of lattice points on the left diagonal of the figure
- NRIGHT=the number of lattice points on the right diagonal of the figure
- NUP=the number of lattice points on the top diagonal of the figure
- NDOWN=the number of lattice points on the bottom diagonal of the figure
- NTOTAL=the total number of lattice points inside the figure and on its borders

Then, we will need to identify the corners of the figure. A corner is a lattice point which belongs to two adjacent diagonals (a main diagonal and a secondary diagonal). Note that we may have 0, 2 or 4 corners. Let's assume that we have NC corners. We will make sure to decrease the corresponding numbers ($NLEFT$, $NRIGHT$, NUP, $NDOWN$) by the number of corners among the set of lattice points which were counted (e.g. if the left diagonal has Q corners on it, we will decrease $NLEFT$ by Q).

In Fig. 1, 2, 3, 4 we present all the cases which may occur during the computation of the $CNTEQ(C, X)$ function (the remaining cases are reducible to these 4 cases by symmetry). Lattice points on the main diagonals are drawn in green, lattice points on the secondary diagonals are drawn in red, corners are drawn in cyan and inner lattice points are drawn in yellow. In Fig. 1 we have $D = 10$, $C = 8$ and $X = 0$. Notice that we obtain $NC = 4$ corners. Note that two adjacent diagonals form a corner if they have the same parity. In Fig. 2 we have $D = 10$, $C = 7$ and $X = 0$. In this case we obtain only $NC = 2$ corners. This is because the top diagonal has a different parity from both the left and the right diagonals, thus forming no corners with them. In Fig. 3 we have $D = 9$, $C = 7$ and $X = 0$; we obtain $NC = 2$ corners. In Fig. 4 we have $D = 12$, $C = 8$ and $X = 1$; no corner is formed in this case.

The main algorithm for computing $CNTEQ(C, X)$ is as follows. We will consider each possible binary configuration of NC bits. If bit i ($0 \le i \le NC - 1$) is set to 1 we will assume that the corresponding corner (i) belongs to the set of lattice points; otherwise, it doesn't belong to the set. After deciding the states of the corners we will check which of the first and second main and secondary diagonals have no selected corners on them. For each such diagonal we will have $2^{NP} - 1$ possibilities of choosing lattice points on it (where NP is the number of lattice points on it, excluding the corners). This equation makes sure that at least one lattice point is selected on each such diagonal. For each of the other diagonals we will have 2^{NP} possibilities of choosing lattice points on them (for these diagonals it is possible to not select any of the lattice points on them, because they already have a selected corner). Then each of the interior lattice points of the figure can be selected as part of the set or not (there are $NIN = NTOTAL - (NUP + NDOWN + NLEFT + NRIGHT + NC)$ lattice points inside the figure and, thus, there are 2^{NIN} possibilities of choosing the inner points). The answer for each binary configuration of the corners is the product between the number of possibilities for each of the 4 diagonals and for the inner points of the figure. $CNTEQ(C, X)$ is the sum of all the answers for each binary configuration of corners. Note that this algorithm works even when $NC = 0$ (there is one binary configuration of 0 bits).

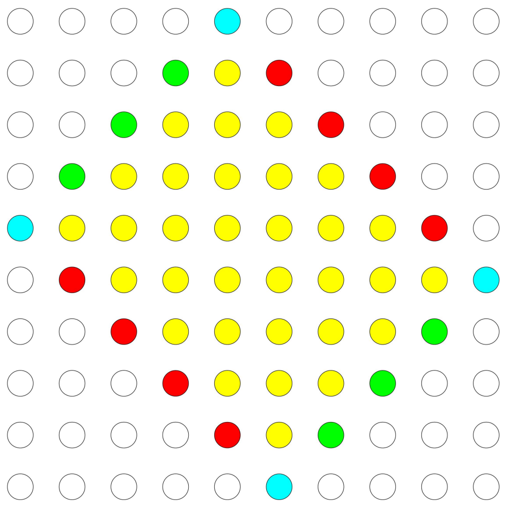
Figure 1. D=10, C=8, X=0, NC=4.

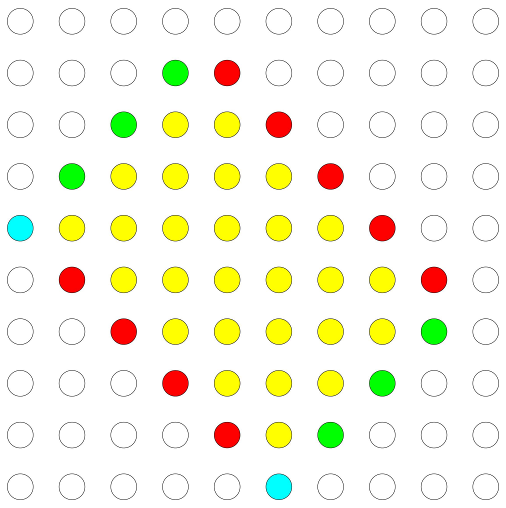
Figure 2. D=10, C=7, X=0, NC=2.

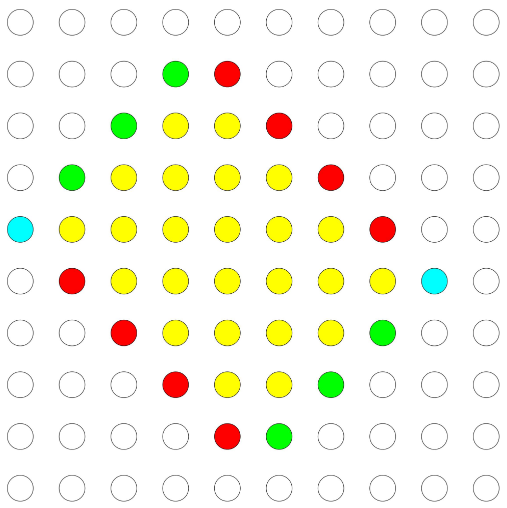
Figure 3. D=9, C=7, X=0, NC=2.

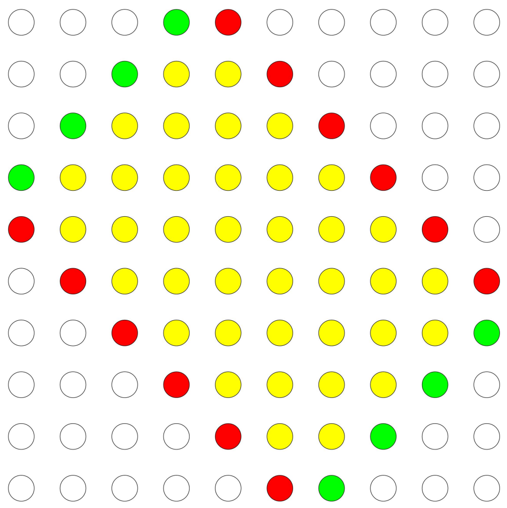
Figure 4. D=12, C=8, X=1, NC=0.

What is left is to identify the values NUP, $NDOWN$, $NLEFT$, $NRIGHT$, $NTOTAL$ and the corners depending on the values of C, X and the parity of D. We will also define the parameter Y, which is defined similarly as X, but for the second secondary diagonal (i.e. $Y = 0$ if the second secondary diagonal has the same parity as the first main diagonal, and $Y = 1$ otherwise). Note that $Y = X$ if C is even, and $Y = 1 - X$ if C is odd. From now on we will assume that the value of Y is computed when evaluating the function $CNTEQ(C, X)$.

We will first consider the case when D is even. If $C = 0$ and $X = 0$ then $CNTEQ(0, 0) = 2^{D/2-1}$. If $C = 0$ and $X = 1$ then $CNTEQ(0, 1) = 0$. Let's consider now that $C \geq 1$. If $X = 0$ then $NDOWN = (D/2) + 1$ and if $X = 1$ then $NDOWN = D/2$. Note that whenever we use the division operator "/" in this paper we refer to integer division. Similarly, if $Y = 0$ then $NUP = (D/2) + 1$, and if $Y = 1$ then $NUP = D/2$. If $X = 0$ then we have $NLEFT = NRIGHT = (C/2) + 1$; otherwise, if $X = 1$ then we have $NLEFT = NRIGHT = (C + 1)/2$. $NTOTAL$ is equal to $NLEFT \cdot ((D/2) + 1) + (C + 1 - NLEFT) \cdot (D/2)$.

If D is odd then we have the following values. $NUP = NDOWN = (D + 1)/2$. If $X = 0$ then $NLEFT = (C/2) + 1$ and $NRIGHT = (C + 1)/2$; otherwise, if $X = 1$ then $NLEFT = (C + 1)/2$ and $NRIGHT = (C/2) + 1$. $NTOTAL$ is equal to $(C + 1) \cdot ((D + 1)/2)$.

The exact formulas we presented for NUP, $NDOWN$, $NLEFT$, $NRIGHT$ and $NTOTAL$ can be easily derived by a careful analysis of all the relevant cases. Let's consider now the cases from Fig. 1, 2, 3, 4 and verify the formulas for those cases. In Fig. 1 we have $NLEFT = (4/2) + 1 = 5$, $NRIGHT = (4/2) + 1 = 5$, $NDOWN = (10/2) + 1 = 6$, $NUP = (10/2) + 1 = 6$ and $NTOTAL = 5 \cdot ((10/2) + 1) + (8 + 1 - 5) \cdot (10/2) = 50$. In Fig. 2 we have $NLEFT = (7/2) + 1 = 4$, $NRIGHT = (7/2) + 1 = 4$, $NDOWN = (10/2) + 1 = 6$, $NUP = 10/2 = 5$ and $NTOTAL = 4 \cdot ((10/2) + 1) + (7 + 1 - 4) \cdot (10/2) = 44$. In Fig. 3 we have $NLEFT = (7/2) + 1 = 4$, $NRIGHT = (7 + 1)/2 = 4$, $NDOWN = (9 + 1)/2 = 5$, $NUP = (9 + 1)/2 = 5$ and $NTOTAL = (7 + 1) \cdot ((9 + 1)/2) = 40$. In Fig. 4 we have $NLEFT = (8 + 1)/2 = 4$, $NRIGHT = (8 + 1)/2 = 4$, $NDOWN = 12/2 = 6$, $NUP = 12/2 = 6$ and $NTOTAL = 4 \cdot ((12/2) + 1) + (8 + 1 - 4) \cdot (12/2) = 58$.

We will show now how to compute the sizes $MBRX$ and $MBRY$ of the minimum bounding rectangle corresponding to the sets counted by $CNTEQ(C, X)$ $(X = 0, 1)$. $MBRX = NLEFT - 1 + NUP - 1 + Y$ and $MBRY = NLEFT - 1 + NDOWN - 1 + X$. Let's verify now these formulas for the cases presented in Fig. 1, 2, 3, 4. In Fig. 1 we have $MBRX = 5 - 1 + 6 - 1 + 0 = 9$ and $MBRY = 5 - 1 + 6 - 1 + 0 = 9$. In Fig. 2 we have $MBRX = 4 - 1 + 5 - 1 + 1 = 8$ and $MBRY = 4 - 1 + 6 - 1 + 0 = 8$. In Fig. 3 we have $MBRX = 4 - 1 + 5 - 1 + 1 = 8$ and $MBRY = 4 - 1 + 5 - 1 + 0 = 7$. In Fig. 4 we have $MBRX = 4 - 1 + 6 - 1 + 1 = 9$ and $MBRY = 4 - 1 + 6 - 1 + 1 = 9$.

After initializing the NUP, $NDOWN$, $NLEFT$, $NRIGHT$ and $NTOTAL$ values, we need to identify the corners. We will consider each pair of (main diagonal, secondary diagonal) and check if they have the same parity (note that the parity of each main and secondary diagonal can be uniquely determined relative to the parity of the first main diagonal from the values X, Y and D; for instance, if $X = 0$ the first main diagonal and the first secondary diagonal have the same parity, if $Y = 0$ the first main diagonal and the second secondary diagonal have the same parity, if $(X = 0)$ and (D is even) the second main diagonal and the first secondary diagonal have the same parity, if $(Y = 0)$ and (D is even) the second main diagonal and the second secondary diagonal have the same parity). Whenever a main diagonal and a secondary diagonal have the same parity, they form a corner. Whenever a corner is identified, the number of lattice points corresponding to the two diagonals is decremented by 1. For instance, if the first main diagonal and the first secondary diagonal form a corner then $NLEFT$ and $NDOWN$ are both decremented by 1. If the first main diagonal and the second secondary diagonal form a corner then both $NLEFT$ and NUP are decremented by 1. If the second main diagonal and the first secondary diagonal form a corner then $NRIGHT$ and $NDOWN$ are both decremented by 1. If the second main diagonal and the second secondary diagonal form a corner then both $NRIGHT$ and NUP are decremented by 1. NC is set to the number of identified corners and the corners are placed in an array on positions 0 to $NC - 1$, so that we know exactly to which diagonals each corner i $(0 \le i \le NC - 1)$ belongs to.

The algorithm presented in this section uses $O(D \cdot log(D))$ arithmetic operations, because it considers $O(D)$ cases and for each case it needs to perform a constant number of exponentiations where the base 2 logarithm of the exponent is of the order $O(log(D))$. In order to reduce the number of arithmetic operations to $O(D)$ we can use the same approach as in section 3. We will assume that our algorithm considers the values of C in ascending order and for each value of C it first computes $CNTEQ(C, 0)$ and then $CNTEQ(C, 1)$. Let's assume that $PREVNIN$ is equal to the value of NIN for the case $C - 1$ and $X = 1$ and $RESPREVNIN = 2^{PREVNIN}$. We will initially have $PREVNIN = 0$ and $RESPREVNIN = 1$. When we need to compute 2^{NIN} for a case we will first compute the difference $DIFNIN = NIN - PREVNIN$. We will always have $0 \le DIFNIN \le D$. Thus, we can compute 2^{NIN} with only one multiplication, as $RESPREVNIN \cdot 2^{DIFNIN}$ (note that 2^{DIFNIN} is taken from the table of precomputed powers of two). After handling the case $(C, X = 1)$ we will update $PREVNIN = NIN$ and $RESPREVNIN = 2^{NIN}$ (where 2^{NIN} was just computed by the method we presented). Using this approach we only need $O(D)$ arithmetic operations instead of $O(D \cdot log(D))$. The same discussion as in Section 3, regarding the computation of exact numbers or of numbers modulo a given number M, applies here, too. In this case $NTOTAL$ is the value of order $O(D^2)$ which is obtained by multiplying together two numbers which are of the order $O(D)$.

6. Algorithm 2 for Counting Sets of Lattice Points of Diameter D under the Manhattan (L_1) Distance

Our second algorithm for the Manhattan distance (and only for the second version of the problem) will use a function $CNTLEQ(C, X)$, where $C \leq D$ and $X = 0$ or 1. $CNTLEQ(C, X)$ computes the number of sets of lattice points such that:

- the main diagonals of at least two points are at distance exactly D apart
- all the other pairs of main diagonals of the points are at distance at most D apart
- all the pairs of secondary diagonals of the points are at distance at most C apart
- $X = 0$ means that the parity of the first secondary diagonal is equal to the parity of the first main diagonal, while $X = 1$ means that these parities differ (the first diagonal of each type is the one with the smallest index)

For $C < 0$ we have $CNTLEQ(C, 0) = CNTLEQ(C, 1) = 0$, by definition. We will present solutions for $C \geq 0$ depending on the parity of D.

We will first consider the case when D is even. In this case we have $CNTLEQ(0, 0) = 2^{D/2-1}$ and $CNTLEQ(0, 1) = 0$ (note that every time we use division we consider integer division). Let's consider now the case $C \geq 1$. All the points must be contained between two main diagonals located at distance D apart and between two secondary diagonals located at distance C apart.

For $X = 0$ this figure has $P = (C/2) + 1$ lattice points on each of the main diagonals and has $R = ((C/2) + 1) \cdot ((D/2) + 1) + (C - (C/2)) \cdot (D/2)$ lattice points in total inside of it and on its borders. $CNTLEQ(C, 0)$ is equal to $(2^P - 1)^2 \cdot 2^{R-2 \cdot P}$.

For $X = 1$ the figure has $P = (C + 1)/2$ lattice points on each of the main diagonals and has $R = ((C + 1)/2) \cdot ((D/2) + 1) + (C + 1 - (C + 1)/2) \cdot (D/2)$ lattice points in total inside of it and on its borders. $CNTLEQ(C, 1)$ is defined identically as $CNTLEQ(C, 0)$, except that we use these new values for P and R.

Let's consider now the case when D is odd. We have $CNTLEQ(C, 0) = CNTLEQ(C, 1)$. The figure defined by the main and secondary diagonals has $P = (C/2) + 1$ lattice points on the first main diagonal and $Q = (C + 1)/2$ lattice points on the second main diagonal. In total, the figure contains $R = (C + 1) \cdot ((D + 1)/2)$ lattice points inside of it and on its borders. We have $CNTLEQ(C, 0) = CNTLEQ(C, 1) = (2^P - 1) \cdot (2^Q - 1) \cdot 2^{R-P-Q}$.

Note that the $CNTLEQ$ function ignores the fact that two sets are identical if one can be obtained from another by translation operations. Instead, it considers two sets to be different if they correspond to different subsets of points belonging to the figure. However, this aspect will be considered when deriving the final formula for the number of sets of lattice points with a given diameter, by using the inclusion-exclusion principle.

The total number of sets of lattice points corresponding to case 1 is equal to $C_1 = CNTLEQ(D-1, 0) + CNTLEQ(D - 1, 1) - CNTLEQ(D - 2, 0) - CNTLEQ(D - 2, 1)$. The total number of sets of lattice points corresponding to case 3 is equal to $C_3 = (CNTLEQ(D, 0) - CNTLEQ(D - 1, 0) - CNTLEQ(D-1, 1)+CNTLEQ(D-2, 1))+(CNTLEQ(D, 1)-CNTLEQ(D-1, 0)-CNTLEQ(D-$

$1,1) + CNTLEQ(D-2,0))$. Again we made use of the inclusion-exclusion principle when computing C_1 and C_3.

It is easy to see that this algorithm uses $O(log(D))$ arithmetic operations (from a constant number of exponentiations where the base 2 logarithm of the exponent is of the order $O(log(D))$). The same discussion as in Section 3, regarding the computation of exact numbers or of numbers modulo a given number M, applies to this case, too. In this case R is the value which is obtained by multiplying together two numbers having $O(log(D))$ bits each.

7. Experimental Results

We implemented the two algorithms for the Manhattan distance and the second version of the studied problem, presented in Sections 5 and 6. For the algorithm from Section 5 we used its $O(D)$ optimized version. We computed the values modulo a prime number $M = 10^9 + 7$, in order to make use of all the computation optimizations possible. We also implemented a backtracking algorithm which generates every set independently (i.e. it enumerates all the valid sets of lattice points having diameter D). We used several values of D in order to compare the running times of the three algorithms. Note that for some values of D some of the algorithms were too slow and we stopped them after a running time of 5 minutes. The running times are presented in Table 1 (a ”-” is shown where the running time exceeded the 5 minutes threshold). All the three algorithms were implemented in C/C++ and the code was compiled using the G++ compiler version 3.3.1. The tests were run on a machine running Windows 7 with an Intel Atom N450 1.66 GHz CPU and 1 GB RAM.

As expected, the $O(log(D))$ algorithm is much faster than the other two algorithms. The $O(D)$ algorithm is faster for odd values of D than for even values. This is because, when D is odd, we can never obtain a figure with 4 corners (in order to have 4 corners both the main and secondary diagonals would need to have the same parity, but when D is odd the main diagonals have different parities).

8. Related Work

There is a large body of work in the scientific literature concerned with counting lattice points in various multidimensional structures. In (Loera, 2005) the general problem of counting lattice points in polytopes was considered. The general problem of counting lattice points in a bounded subset of the Euclidean space was considered in (Widmer, 2012). Harmonic analysis is applied in (Chamizo, 2008) for counting lattice points in large parts of space.

A problem concerned with counting configurations of lattice points obtained when translating a convex set in the plane was considered in (Huxley & Zunic, 2009), (Huxley & Zunic, 2013). Two configurations were considered identical under similar conditions as the ones used in this paper. Counting arrangements of connected polyominoes (equivalent under translation) and other figures was considered in (Rechnitzer, 2000). The problem of counting directed lattice walkers in horizontal strips of finite width was considered in (Chan & Guttman, 2003). Counting lattice triangulations was studied in (Keibel & Ziegler, 2003).

As far as we are aware, the problems we considered in this paper have not been considered before in any other publication.

Table 1. Running time (in sec) of the three algorithms for several values of D.

D	Backtracking Algorithm	O(D) Algorithm	O(log(D)) Algorithm
1	0.002	0.002	0.002
2	0.002	0.002	0.002
3	0.004	0.002	0.002
4	0.065	0.002	0.002
5	3.91	0.002	0.002
6	-	0.002	0.002
10	-	0.002	0.002
11	-	0.002	0.002
10^4	-	0.09	0.003
$10^4 + 1$	-	0.08	0.003
10^5	-	0.81	0.003
$10^5 + 1$	-	0.54	0.003
10^6	-	7.84	0.003
$10^6 + 1$	-	5.2	0.003
10^7	-	78.3	0.003
$10^7 + 1$	-	51.9	0.003
10^8	-	-	0.003
$10^8 + 1$	-	-	0.003
10^9	-	-	0.004
$10^9 + 1$	-	-	0.004

9. Conclusions

In this paper we presented novel, efficient algorithms for computing the number of sets of lattice points in the plane whose diameter is exactly equal to D, when considering the Manhattan (L_1) or the Chebyshev (L_∞) distance. We considered two versions for defining the equivalence of two such sets of lattice points. The first version forces the sets of points to be fully included inside a given 2D grid. The second version defines two sets of lattice points to be equivalent if one can be obtained from another by using translation operations. Our algorithms require $O(D \cdot log(D))$ or $O(D)$ arithmetic operations (additions, multiplications) for the first version of the problem and only $O(log(D))$ arithmetic operations for the second version of the problem for both distances. We also discussed the possibility of computing the results modulo a given number M, as a way of simplifying some parts of the algorithms (in particular, in order to use numbers with a number of bits independent of D).

As future work we intend to approach the same problems described in this paper but for a number of dimensions greater than 2. Note that in the 1D case the two problems are identical and very simple to solve (for instance, the answer is always 2^{D-1} for the second version of the problem, because we must have two points in the set at distance D and all the other $D - 1$ points between them may be selected or not to be part of the set).

References

Chamizo, F. (2008). Lattice point counting and harmonic analysis. In: *Bibl. Rev. Mat. Iberoamericana, Proceedings of the "Segundas Jornadas de Teoria de Numeros"*. pp. 83–99.

Chan, Y.-B. and A.J. Guttman (2003). Some results for directed lattice walkers in a strip. *Discrete Mathematics and Theoretical Computer Science* **AC**, 27–38.

Furer, M. (2009). Faster integer multiplication. *SIAM Journal on Computing* **39**(3), 979–1005.

Huxley, M.N. and J. Zunic (2009). The number of configurations in lattice point counting i. *Forum Mathematicum* **22**(1), 127–152.

Huxley, M.N. and J. Zunic (2013). The number of configurations in lattice point counting ii. *Proceedings of the London Mathematical Society*.

Indyk, P. (2001). Algorithmic applications of low-distortion geometric embeddings. In: *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*.

Keibel, V. and G.M. Ziegler (2003). Counting lattice triangulations. In: *Surveys in Combinatorics* (C.D. Wensley, Ed.).

Loera, J.A. De (2005). The many aspects of counting lattice points in polytopes. *Mathematische Semesterberichte* **52**(2), 175–195.

Rechnitzer, A.D. (2000). Some Problems in the Counting of Lattice Animals, Polyominoes, Polygons and Walks. PhD thesis. University of Melbourne, Department of Mathematics and Statistics.

Schonhage, A. and V. Strassen (1971). Schnelle multiplikation grosser zahlen. *Computing* **7**, 281–292.

Widmer, M. (2012). Lipschitz class, narrow class, and counting lattice points. *Proceedings of the American Mathematical Society* **140**(2), 677–689.