Practical Discrete Unit Disk Cover Using an Exact Line-Separable Algorithm*

Francisco Claude[1], Reza Dorrigiv[1], Stephane Durocher[2], Robert Fraser[1], Alejandro López-Ortiz[1,**], and Alejandro Salinger[1]

[1] Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada
{fclaude,rdorrigiv,r3fraser,alopez-o,ajsalinger}@cs.uwaterloo.ca
[2] Department of Computer Science, University of Manitoba, Winnipeg, Canada
durocher@cs.umanitoba.ca

Abstract. Given m unit disks and n points in the plane, the discrete unit disk cover problem is to select a minimum subset of the disks to cover the points. This problem is NP-hard [11] and the best previous practical solution is a 38-approximation algorithm by Carmi et al. [4]. We first consider the line-separable discrete unit disk cover problem (the set of disk centres can be separated from the set of points by a line) for which we present an $O(m^2n)$-time algorithm that finds an exact solution. Combining our line-separable algorithm with techniques from the algorithm of Carmi et al. [4] results in an $O(m^2n^4)$ time 22-approximate solution to the discrete unit disk cover problem.

1 Introduction

Recent interest in specific geometric set cover problems is partly motivated by applications in wireless networking. In particular, when wireless clients and servers are modelled as points in the plane and the range of wireless transmission is assumed to be constant (say one unit), the resulting region of wireless communication is a disk of unit radius centred on the point representing the corresponding wireless transmitting device. Under this model, sender a successfully transmits a wireless message to receiver b if and only if point b is covered by the unit disk centred at point a. This model applies more generally to a variety of facility location problems for which the Euclidean distance between clients and facilities cannot exceed a given radius, and clients and candidate facility locations are represented by discrete sets of points. Examples include (1) selecting locations for wireless servers (e.g., gateways) from a set of candidate locations to cover a set of wireless clients, (2) positioning a fleet of water bombers at airports such that every active forest fire is within a given maximum distance of a water bomber, (3) selecting a set of weather radar antennae to cover a set of cities, and (4)

* Funding for this project was provided by the NSERC Strategic Grant on Optimal Data Structures for Organization and Retrieval of Spatial Data.

** Part of this work took place while the fifth author was on sabbatical at the Max-Planck-Institut für Informatik in Saarbrücken, Germany.

Y. Dong, D.-Z. Du, and O. Ibarra (Eds.): ISAAC 2009, LNCS 5878, pp. 45–54, 2009.

selecting locations for anti-ballistic defenses from a set of candidate locations to cover strategic sites. These problems can be modelled by the *discrete unit disk cover problem* (DUDC), whose definition is: Given sets P of m points and Q of n points in the plane (candidate facilities and clients, respectively), find a set $P' \subseteq P$ (facilities) of minimum cardinality such that $\mathrm{Disk}(P')$ covers Q, where $\mathrm{Disk}(A)$ denotes the set of unit disks centred on points in set A. In this work, we consider the *line-separable discrete unit disk cover* (LSDUDC), where P and Q are separated by a line l.

The DUDC problem is NP-hard [11]. In a recent result, Carmi et al. [4] describe a polynomial-time 38-approximate solution, improving on earlier 108-approximate [6] and 72-approximate solutions [16]. We present an $O(m^2n)$-time algorithm that returns an exact solution to the LSDUDC problem, as well as a thorough proof of correctness of the technique. By combining the LSDUDC algorithm with techniques from the algorithm of Carmi et al. [4], we present a 22-approximation algorithm to the DUDC problem, improving on the best previous practical polynomial-time approximation factor of 38.

1.1 Related Work

Line-Separable Discrete Unit Disk Cover. A solution to the LSDUDC problem was independently discovered and published by [3, Lemma 1], where they propose a dynamic programming algorithm with a time bound of $O(m^2n)$ but whose correctness is not straightforward nor is it formally argued. This paper presents an alternative algorithm together with a proof of correctness. Both algorithms follow natural approaches, yet a full proof of correctness is not immediate.

ε-nets for Geometric Hitting Problems. Using ϵ-nets, Mustafa and Ray [15,14] have recently presented a $(1+\epsilon)$-approximation to the DUDC problem. Their algorithm runs in $O(m^{2(c/\varepsilon)^2+1}n)$ time, where $c \leq 4\gamma$ [14]. Their γ value can be bounded from above by $2\sqrt{2}$ [8,12]. The fastest operation of this algorithm is obtained by setting $\varepsilon = 1$ for a 2-approximation, and this will run in $O(m^{2\cdot(8\sqrt{2})^2+1}n) = O(m^{257}n)$ time in the worst case. Clearly, this algorithm will not be practical for large values of m.

Minimum Geometric Disk Cover. In the minimum geometric disk cover problem, the input consists of a set of points in the plane, and the problem is to find a set of unit disks of minimum cardinality whose union covers the points. Unlike our problem, disk centres are not constrained to be selected from a given discrete set, but rather may be centred at arbitrary points in the plane. Again, this problem is NP-hard [7,17] and has a PTAS solution [9]. Of course the problem can be generalized further: see [5] for a discussion of geometric set cover problems.

Discrete k-Centre. Also related is the discrete Euclidean k-centre problem: given a set P of m points in the plane, a set Q of n points in the plane, and an integer k, find a set of k disks centred on points in P whose union covers Q such that the radius of the largest disk is minimized. Observe that set Q has a discrete unit disk cover consisting of k disks centred on points in P if and only if Q has a discrete k-centre centred on points in P with radius at most one. This problem is NP-hard if k is an input variable [2]. When k is fixed, Hwang et al. [10] give a $m^{O(\sqrt{k})}$-time algorithm, and Agarwal and Procopiuc [1] give an $m^{O(k^{1-1/d})}$-time algorithm for points in $\mathbb{R}^d$.

2 Overview of the Algorithm

In this section we describe a polynomial-time algorithm for the line-separable discrete unit disk cover (LSDUDC) problem and prove its correctness. Details of the algorithm and its running time will be discussed in Section 3. Recall that we have two sets $P = \{p_1, p_2, \dots, p_m\}$ and $Q = \{q_1, q_2, \dots, q_n\}$ of points in the plane that are separated by a line l. We want to find a subset $P' \subseteq P$ of minimum cardinality such that all points of Q are covered by unit disks centred at the points of P'. An instance of the problem is shown in Figure 1. Without loss of generality we assume that l is a horizontal line and points of P are above l. Let d_i denote the unit disk that is centred at p_i, for $i \in \{1, 2, \dots, m\}$, and let D denote the set of these disks. We use p_i and d_i interchangeably, e.g., our solution can be considered both as a set of points (a subset of P) and as a set of disks. Further, when we discuss the intersection of a line with a disk, we are referring to the intersection of the line with the boundary of the disk.

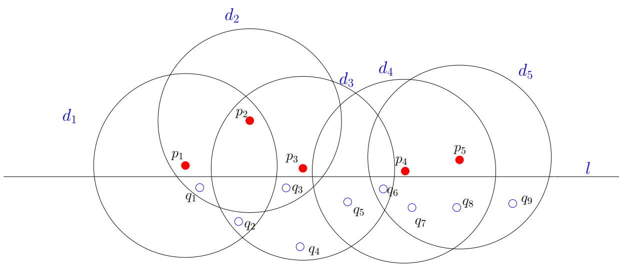

Fig. 1. An instance of the line-separable discrete unit disk cover problem

During the execution of our algorithm, it may be determined that a disk $d \in D$ should be added to the solution or that it is not relevant for the remainder of the computation of the solution set. When this occurs, we remove disk d from the problem. Similarly, we remove a point $q \in Q$ if this point is not relevant for the remainder of the computation (i.e., point q is covered by a disk in the partial solution being constructed). Our algorithm relies on the following three observations:

1. If a disk d_1 covers no points from Q, we remove it.
2. If a disk d_1 is *dominated* by a disk d_2, then we can remove d_1 from the problem instance. Disk d_2 dominates d_1 if it covers all points of Q covered by d_1. If two disks cover the same subset of points from Q, we designate the dominating disk as that whose left intersection with l is rightmost.

3. If a point $q_1 \in Q$ is only covered by a disk d_1, then d_1 must be part of the solution. We also remove d_1 together with all points of Q covered by d_1.

These three observations give us three Simplification rules. The idea is to apply these rules to as many disks as possible and simplify the problem. For example, consider the problem instance shown in Figure 1. Initially no disk dominates another, thus we cannot apply the second rule. Disk d_3 is the only disk that covers q_4 and, similarly, disk d_5 is the only disk that covers q_9. Thus we add d_3 and d_5 to the (initially empty) solution and remove them together with the points that are covered by them, namely $\{q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9\}$. Now disk d_4 covers no point and can be removed. There is only one remaining point (q_1) and it is covered by the two remaining disks (d_1 and d_2). According to our convention, d_1 is dominated by d_2 and is removed. Now d_2 is the only disk covering q_1. We add d_2 to the solution and remove d_2 and q_1. No disks or points remain and we are done. Thus the Simplification rules suffice for this instance and give an optimal solution $\{d_2, d_3, d_5\}$. This example also illustrates that an optimal solution is not necessarily unique, as $\{d_1, d_3, d_5\}$ is also an optimal solution. In general, however, these Simplification rules do not suffice to obtain an optimal solution.Referring to Fig. 1, if given only disks d_1, d_2 and d_3 and points q_1, q_2 and q_3, then no point $q \in Q$ is covered by only one disk and no disk dominates any other one.

We augment the Simplification rules with a simple greedy step to solve the problem. We rename the disks so that the left intersection of d_i with l is to the left of the left intersection of d_{i+1} with l. We say that d_i precedes d_{i+1} in the ordering (the disks in Figure 1 follow this ordering). This combined algorithm, Greedy, works by first applying the Simplification rules as many times as possible. Next we find the first remaining disk in the left-to-right order, say d_j. We add d_j to our solution and remove d_j from D and all points covered by d_j from Q. We apply the Simplification rules followed by the greedy step repeatedly until all disks have been removed. Since we remove at least one disk at each greedy step, the algorithm terminates after at most m iterations. See Algorithm 1 for the corresponding pseudocode.

Algorithm 1. Greedy (D, Q)

```
D ←sortLeftToRight(D) //sort in increasing order of left intersection with l
S ← ∅
while D ≠ ∅ do do
   Simplification (D, Q, S) //Simplification possibly modifies D, Q and S
   d_ℓ ← leftmost disk in D
   S ← S ∪ {d_ℓ}
   D ← D \ d_ℓ
   Q' ← {q ∈ Q | q is contained in d_ℓ}
   Q ← Q \ Q'
end while
return S
```

2.1 Correctness of Greedy

We now prove the correctness of the algorithm by proving that Greedy gives a minimum LSDUDC solution. Assume for the sake of contradiction that there is an algorithm Opt that gives a cover with fewer disks than Greedy. Let d_1 be the first disk in the ordering that is selected by Greedy but not by Opt. Let C be the set of points in Q that are covered by d_1 (we consider only the remaining points and disks, i.e., those that have not been removed by the algorithm). First assume that C is covered by a single disk d_0 in the solution of Opt. Since d_1 is not removed in the Simplification step, it is not dominated by any other disk. Thus the only possibility is that d_0 and d_1 cover exactly the same set of (remaining) points (i.e., set C) and d_0 precedes d_1 in the ordering. In this case, we replace d_0 with d_1 in Opt, pushing the first difference between the solution of Greedy and Opt to the right. Otherwise, C is covered by at least two disks in the solution of Opt. Let d_2 and d_3 be two disks in the solution of Opt such that each of them cover a strict subset of C. Without loss of generality assume that d_2 precedes d_3 in the ordering. We prove that $d_1 \cup d_3$ covers all points of Q covered by $d_2 \cup d_3$.

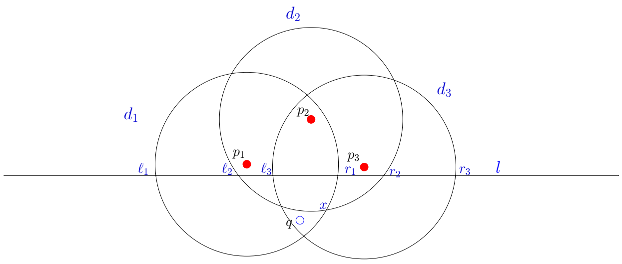

Fig. 2. Proof of correctness of Greedy. If d_1 is the first disk selected by Greedy and not by Opt, then Opt must have d_2 and d_3 in its solution.

Let ℓ_i and r_i denote the respective left and right intersection points of the boundary of the unit disk d_i with the line l, for $i \in \{1, 2, 3\}$. If d_2 precedes d_1 in the ordering, d_1 dominates d_2 (otherwise, Greedy would select d_2 and not d_1 at this step). In this case we replace d_2 with d_1 in Opt, pushing the difference between the two algorithms to the right. Hence we are left with the case in which d_1 precedes d_2 and d_2 precedes d_3 in the ordering. Thus the points are ordered ℓ_1, ℓ_2, ℓ_3, r_1, r_2, r_3 along line l (see Figure 2). Note that we cannot have a pair of disks nested below l, otherwise the nested disk is dominated by the other. Furthermore, we know that $(d_1 \cap d_3) \setminus d_2 \neq \varnothing$. Let $R = d_2 \setminus d_1$. It suffices to prove that R is completely contained in d_3.

Proposition 1. *Region R is contained in disk d_3.*

Proof. Since points r_1 and r_2 both lie between ℓ_3 and r_3 on line l, both points r_1 and r_2 are in disk d_3. Let x denote the rightmost point of the intersection of the boundaries of disks d_1 and d_2. Observe that x lies on the boundary of region $(d_1 \cap d_3) \setminus d_2$. Consequently, $x \in d_3$. Since the boundary of R consists of arcs of unit disks joining the points x, r_1, and r_2, it follows that R is contained in the 1-hull of $\{x, r_1, r_2\}$, where the 1-hull of $\{x, r_1, r_2\}$ is the intersection of all unit disks that contain $\{x, r_1, r_2\}$, denoted 1-$H(\{x, r_1, r_2\})$. Since $\{x, r_1, r_2\} \subseteq d_3$, it follows that $R \subseteq$ 1-$H(\{x, r_1, r_2\}) \subseteq d_3$. □

Thus by removing d_2 from the solution of Opt and adding d_1 to it we will have a feasible solution with the same number of disks. This pushes the first difference between the solution of Greedy and Opt to the right. By continuing this argument we can prove that the solution returned by Greedy uses the same number of disks as Opt and therefore Greedy is an optimal algorithm.

3 Implementation Details and Analysis

We construct a graph $G = (V, E)$, where each node $v_i \in V$ corresponds to disk d_i for $i \in \{1, \ldots, m\}$ (recall that d_i is the i^{th} disk sorted according to its left intersection with l). We also associate a counter c_{v_i} to each node v_i that stores the number of points in Q contained in disk d_i that have not yet been covered by the algorithm. Similarly, we associate with each edge $e = (v_{i_1}, v_{i_2})$ a counter c_e that represents the number of points contained in $d_{i_1} \cap d_{i_2}$. This graph can be constructed in $O(m^2 n)$ time by checking which points are contained in the intersection of each pair of disks, adding the corresponding edges, and updating the node and edge counters. The algorithm Greedy-Graph starts by traversing the nodes in order $v_1, v_2, \ldots, v_m$. At each node v_i, there are three possible cases: (1) The counter c_{v_i} is 0; in this case d_i does not contain any points or is dominated by a set of disks that has already been added to the solution. This disk will not be in the solution set, so we can ignore this node and continue with the next one. This is analogous to the first Simplification rule. (2) There is an edge $e = (v_i, v_k),\ k > i$, such that $c_e = c_{v_i}$; in this case we know that d_i is dominated by disk d_k. Again, we ignore this node and continue. This corresponds to an application of the second Simplification rule. (3) Every edge $e = (v_i, v_k),\ k > i$, satisfies $c_e < c_{v_i}$; that means that disk d_i is not dominated by any disk to its right. In this case we add d_i to the solution set and we eliminate all remaining points contained by this disk from the graph. We continue with the next node in the graph. Note that this is an application of the third rule of Simplification and the greedy step.

In order to identify the appropriate case above we traverse the adjacency list of each node we visit. This requires $O(m)$ time in the worst case. When a disk is added to the solution in the third case, all points contained in the disk must be eliminated. Consider the elimination of a point p in disk d_i. Let $N(v_i) = \{v_k \mid c_{(v_i, v_k)} > 0\}$. For all $v_k \in N(v_i)$, we decrease c_{v_k} and $c_{(v_i, v_k)}$ by one. In addition, for each pair of elements $\{v_{k_1}, v_{k_2}\} \subseteq N(v_i)$, we check whether the point is contained by both disks, and if this is case we decrease $c_{(v_{k_1}, v_{k_2})}$ by one. This can take at most $O(m^2)$ time per point, thus the time required for eliminating all points is bounded by $O(m^2n)$ time. Since the time required to construct the graph is $O(m^2n)$, the overall process takes $O(m^2n)$ time.

3.1 Correctness of Greedy-Graph

We now demonstrate that the Greedy-Graph algorithm is optimal by showing that the set of disks returned by this algorithm has the same cardinality as that returned by the Greedy algorithm presented in Section 2.

Lemma 1. *If S is the disk cover returned by* Greedy-Graph*, and S' is the disk cover returned by* Greedy*, then $|S| = |S'|$.*

Proof. Assume for the sake of contradiction that $|S| \neq |S'|$. Recall that Greedy is optimal, therefore $|S'|$ and $|S|$ can only differ if $|S| > |S'|$. Let d_1 be the first disk in the left-to-right order that is present in the solution of Greedy-Graph, and not in the solution of Greedy. At some point during its execution, Greedy must have decided to discard disk d_i. The only mechanisms in Greedy for discarding disks are the first and second Simplification rules. Recall that the first rule removes a disk if it contains no points, and the second rule discards a disk if it is dominated by some other disk. We now show that for any of the following possible events, Greedy-Graph will discard the same disk d_1.

– **Empty** - Suppose d_1 contains no points. In this case, Greedy-Graph will find that $c_{v_1} = 0$. Therefore, d_1 will be discarded by Case 1, in contradiction to our assumption.
– **Dominance (right)** - Now suppose d_1 is dominated by some disk to the right, d_r. In this case, we will encounter d_1 first during our walk, and we will have that $c_{v_1} = c_{(v_1,v_r)}$. Therefore, Greedy-Graph will remove d_1 by Rule 2, in contradiction to our assumption.
– **Dominance (left)** - Suppose d_1 is dominated by some disk to the left, d_ℓ. In this case, we will have encountered d_ℓ first during our walk. There are two possible cases in this scenario:
  (i) If $c_{v_\ell} > c_{(v_\ell,v_k)}$ for all d_k, d_ℓ is added to S by Rule 3 of Greedy-Graph. All points covered by d_ℓ are removed, leaving no points covered by d_1. This is now an instance of the Empty case.
  (ii) Otherwise, $c_{v_\ell} = c_{(v_\ell,v_k)}$ for some d_k. This means that d_ℓ is dominated by d_k. Greedy-Graph would discard d_ℓ by Rule 3. By transitivity, d_k also dominates d_1. If d_k is to the right of d_1, then this is now an instance of Dominance (right), and thus we reach a contradiction. If d_k is to the left of d_1, then this is again an instance of Dominance (left), so we apply this same argument recursively. The recursion stops either when we reach an instance of Dominance (right) or case (i) of Dominance (left). □

We have shown that the solution of Greedy-Graph has the same cardinality as the solution of Greedy, and since Greedy is optimal, so is Greedy-Graph.

Theorem 1. *Given sets P of m points and Q of n points in the plane, where P and Q can be separated by a line l, LSDUDC can be solved in $O(m^2n)$ time.*

4 Approximate Discrete Unit Disk Cover

We now show that our algorithm for the line-separable discrete unit disk cover (LSDUDC) problem leads to a 22-approximation algorithm for the discrete unit disk cover (DUDC) problem. The approximation algorithm is based on a suitable adaptation of the 38-approximation algorithm of Carmi et al. [4].

For simplicity, we use the notation and assumptions of [4]. In that work, the DUDC problem is supported by a variant of the LSDUDC problem: suppose are given a set of disks $D = L \cup U$. The disks in U are centred above a line l while the disks in L are centred below l. We are also given a set of points Q covered by U. The goal is to obtain the subset G of D of smallest cardinality such that every point in Q is covered by a disk in G.

Note that our line-separable algorithm does not immediately result in a straightforward improvement to the approximation factor of the algorithm of Carmi et al. Their proof of correctness uses the fact that their 2-approximation to the LSDUDC problem consists of disks forming the lower boundary of U, where the lower boundary is the union of all disk boundary arc segments below l not contained in other disks. This is not necessarily the case in our solution.

Instead, we first solve the LSDUDC problem optimally using our algorithm on the set of disks U to obtain a disk set H and then use the greedy *minimum assisted cover* algorithm (see Carmi et al. [4, §2] for the formal definition) over the sets H and L to obtain an improved solution E. Now we wish to compare the cardinality of E with that of the global minimum disk cover G.

Consider the upper and lower components of the solutions E and G, i.e., $E_U = E \cap U$, $E_L = E \cap L$, $G_U = G \cap U$, and $G_L = G \cap L$. Note that $|G| \leq |E|$ since G is the global minimum. Similarly, since E is the minimum assisted cover based on H, it follows that $|E| = |E_U| + |E_L| \leq |H/G_L| + |G_L|$, where H/G_L is the smallest subset of H that forms an assisted cover with G_L.

Now we will show that $2|G_U| \geq |H/G_L|$. Given a disk d in G_U, there are two cases: either d lies above the lower boundary of H/G_L, i.e., d is contained in the union of all the disks in H/G_L, or d contains one or more arc segments of the lower boundary of H/G_L. In the first case, Carmi et al. show that at most two disks in H/G_L suffice to cover d and, hence, for every such disk in the global optimum solution G there are most two disks in H/G_L. In the second case, let V denote the subset of disks that have lower boundary segments that are contained in d. The set of arc segments of the disks in V consists, from left to right, of a partially-covered arc segment of the lower boundary, zero or more fully-covered arc segments, and a partially-covered arc segment. Let W denote the disks whose arcs are partially covered together with d. W dominates V and hence there is at most one arc of the lower boundary fully contained in d; otherwise replacing V with W results in a cover of smaller cardinality, deriving a contradiction, since $V \subset H$, and H is the optimal LSDUDC solution[1]. Furthermore, observe that the partially-covered arc disks must contain points not contained in the fully-covered disk; otherwise they can also be eliminated while reducing the

[1] Recall that all disks in V and U are centred above l, and all points in Q are below l.

cardinality of the cover. As those disks contain other points, each of the disks is partially covered by at least one other disk in G. We arbitrarily associate each disk covered more than once to its leftmost disk in G. Thus, of the (at most) three disks in V, at most two are associated to d. In sum, in either case each disk in G_U has at most two associated disks in H/G_L from which it follows that $2|G_U| \geq |H/G_L|$. Hence, $2|G| = 2\,(|G_U| + |G_L|) \geq 2|G_U| + |G_L| \geq |H/G_L| + |G_L| \geq |E_U| + |E_L| = |E|$, which gives the approximation factor of two as desired. Carmi et al. [4] prove that any disk can be used in up to eight applications of the assisted LSDUDC algorithm, for which they have a 4-approximation. These operations, followed by a 6-approximation for any remaining disks results in an $8 \times 4 + 6 = 38$-approximation for the general DUDC problem. As we have shown that our technique provides a 2-approximation for the assisted LSDUDC problem, we now have an approximation ratio of $8 \times 2 + 6 = 22$ for DUDC.

4.1 Algorithm Analysis

There are essentially two main components to the algorithm for solving DUDC by Carmi et al. [4]. First, they apply a grid of size $3/2\times 3/2$ to the input data. Our LSDUDC algorithm supplemented by their assisting disk technique is run on all grid lines. Note that the number of relevant grid lines is $O(n)$. Our technique runs in $O(m^2 n)$, and the assisting disk operation is easily implementable in $O(mn)$, so the running time of the first component is dominated by our step.

The second major component to their technique is finding the 6-approximation for the DUDC of all disk centres and points contained in each of the $3/2 \times 3/2$ squares of the grid. Their technique is based on the application of a subset of nine properties depending on where the disk centres are located. First, they determine whether a solution exists using one or two centres by brute force, which is easily done in $O(m^2 n)$ time. The determination of which properties may be applied can be done in $O(m)$ time, and there are only two expensive steps that may be used in any of the procedures, each of which may only be used a constant number of times. First is the assisted LSDUDC technique, whose running time is $O(m^2 n)$, as we just discussed. The second technique that may be required is to determine the optimal disk cover of a set of points using centres contained in one of the $1/2 \times 1/2$ squares, which can be solved in $O(m^2 n^4)$ time using the technique presented in [13]. The centre of each disk can only be contained in one square, and so this operation is never performed twice for any given disk. Therefore, the complete DUDC algorithm achieves worst-case performance when all of the disk centres in the plane are confined to a single $1/2 \times 1/2$ square, so that the $O(m^2 n^4)$ operation is performed over the entire data set.

5 Conclusions

This paper presents a polynomial-time algorithm that returns an exact solution to the LSDUDC problem, as well as a proof of correctness of the approach. This algorithm for the line-separable problem allows us to improve the approximation algorithm of Carmi et al. [4], resulting in a 22-approximate solution to the general DUDC problem, which runs in $O(m^2n^4)$ time in the worst case.

Theorem 2. *Given sets P of m points and Q of n points in the plane, we can compute a 22-approximation of the DUDC problem in $O(m^2n^4)$ time in the worst case.*

Acknowledgements. The authors wish to thank Paz Carmi for sharing his insights and discussing details of his results on the discrete unit disk cover problem [4]. In addition, the authors acknowledge Sariel Har-Peled with whom a preliminary problem was discussed that inspired our examination of the disk cover problem.

References

1. Agarwal, P., Procopiuc, C.: Exact and approximation algorithms for clustering. Alg. 33, 201–226 (2002)
2. Agarwal, P., Sharir, M.: Efficient algorithms for geometric optimization. ACM Comp. Surv. 30, 412–458 (1998)
3. Ambühl, C., Erlebach, T., Mihal'ák, M., Nunkesser, M.: Constant-factor approximation for minimum-weight (connected) dominating sets in unit disk graphs. In: Proc. of Approx., Rand., and Comb. Opt. Alg. and Tech., pp. 3–14 (2006)
4. Carmi, P., Katz, M., Lev-Tov, N.: Covering points by unit disks of fixed location. In: Proc. Int'l Symp. on Alg. and Comp., pp. 644–655 (2007)
5. Clarkson, K., Varadarajan, K.: Improved approximation algorithms for geometric set cover. Disc. and Comp. Geom. 37(1), 43–58 (2007)
6. Călinescu, G., Măndoiu, I., Wan, P.J., Zelikovsky, A.: Selecting forwarding neighbours in wireless ad hoc networks. Mob. Net. and Appl. 9(2), 101–111 (2004)
7. Fowler, R., Paterson, M., Tanimoto, S.: Optimal packing and covering in the plane are NP-complete. Inf. Proc. Lett. 12(3), 133–137 (1981)
8. Frederickson, G.: Fast algorithms for shortest paths in planar graphs, with applications. SIAM J. on Comp. 16(6), 1004–1022 (1987)
9. Hochbaum, D., Maass, W.: Approximation schemes for covering and packing problems in image processing and VLSI. J. ACM 32, 130–136 (1985)
10. Hwang, R., Lee, R., Chang, R.: The generalized searching over separators strategy to solve some NP-hard problems in subexponential time. Alg. 9, 398–423 (1993)
11. Johnson, D.: The NP-completeness column: An ongoing guide. J. of Alg. 3(2), 182–195 (1982)
12. Koutis, I., Miller, G.: A linear work, $o(n^{1/6})$ time, parallel algorithm for solving planar Laplacians. In: Proc. Symp. Disc. Alg., pp. 1002–1011 (2007)
13. Lev-Tov, N.: Algorithms for Geometric Optimization Problems in Wireless Networks. Ph.D. thesis, Weizmann Institute of Science (2005)
14. Mustafa, N., Ray, S.: Improved results on geometric hitting set problems (2009), http://www.mpi-inf.mpg.de/~saurabh/Papers/Hitting-Sets.pdf
15. Mustafa, N., Ray, S.: PTAS for geometric hitting set problems via local search. In: Proc. Symp. on Comp. Geom., pp. 17–22 (2009)
16. Narayanappa, S., Voytechovsky, P.: An improved approximation factor for the unit disk covering problem. In: Proc. Can. Conf. Comp. Geom. (2006)
17. Supowit, K.: Topics in Computational Geometry. Ph.D. thesis, University of Illinois at Urbana-Champaign (1981)