Maximum Matchings on co-Unit Disk Graphs

Shuhao Tan
Department of Computer Science, University of Maryland, College Park, USA

David M. Mount
Department of Computer Science, University of Maryland, College Park, USA

1 Introduction

The unit disk graph is a classical object of interest in computational geometry. It has many applications, notably in the field of communication [3], where each edge models ability to transmit a signal locally. Computing matchings on unit disk graphs is a topic of fundamental interest [2, 5]. We consider the problem of computing a maximum cardinality matching on the *complement* of a unit disk graph, or *co-unit disk graph*. A motivating application is to pair-up people "diversely", in the sense that we want to avoid matchings between individuals that are too similar. Other motivating applications include distributing backup data to distant servers and allocating resources to non-interfering agents.

We consider the problem in a multi-dimensional setting. Let $d(\cdot, \cdot)$ denote the Euclidean distance function. Given a point set $P = \{p_1, p_2, \ldots, p_n\} \subset \mathbb{R}^d$, where d is a constant, the objective is to compute a maximum cardinality matching M on P such that for any matched pair (p_i, p_j), $d(p_i, p_j) > 1$. Here, we consider an approximate variant: Given a constant $0 < \varepsilon < 1$, compute a matching M of cardinality no less than the optimal (in the exact case), such that for any matched pair (p_i, p_j), $d(p_i, p_j) > 1 - \varepsilon$.

Prior Work

Maximum matching on general graphs has been extensively studied. The first breakthrough of the problem was Edmond's blossom algorithm [4], which computes successive augmenting paths in the graph as if it is bipartite and contracts any odd-length cycles it encounters.

The current best bound for general graphs is due to Micali and Vazirani [7, 8]. The algorithm is very similar to the bipartite matching algorithm of Hopcroft and Karp [6], where a maximal partial augmenting matching is found in each iteration. The algorithm runs in time $O(|E|\sqrt{|V|})$, where V and E are the vertex and edge sets, respectively.

The line of study on matchings has generally focused on blossom-based approaches, but Norbert Blum proposed an alternative algorithm by duplicating the graph to form a directed bipartite graph [1]. Finding an augmenting path reduces to a simple reachability problem, but with some additional constraints. A modified depth-first search (DFS) is employed to solve the problem. To achieve the best running time, this is combined with a Hopcroft-Karp style approach.

Matching for geometric objects is also well-studied, but the focus is more on minimizing the weight of matching [9–11] or on geometric intersection graphs [2]. The longest perfect matching problem [5], in which the goal is to maximize the minimum distance between matched pairs, is closely related to the problem we consider.

Our Contribution

Our technique involves computing the well-separated pair decomposition and obtaining a compressed representation of the ε-approximation of the co-unit disk graph. We then modify a reachability-based matching algorithm by Norbert Blum [1] to run on the compressed representation.

The standard quadtree-based WSPD construction yields a quadtree with a list of pairs of nodes representing the well-separated pairs. We consider the general case where a tree and a list of pairs of nodes are given. The corresponding graph is obtained by expanding a pair to a bipartite clique between the leaves of the two subtrees rooted at the two nodes in the pair. We show that a matching corresponds to a flow-network-like structure on the tree, and finding an augmenting path corresponds to finding an augmenting flow.

We then modify Blum's matching algorithm on this structure. In Blum's algorithm, the original graph is duplicated to represent inner/outer vertices, and directed edges between them are added to represent matched/unmatched edges. Blum showed that an alternating path in the original corresponds to a simple path in the modified graph that visits at most one copy of each duplicated vertex. Augmentation reduces to finding such a path. Our duplication is more involved since our modified graph is not bipartite, the tree structure is preserved in both copies and edges corresponding to a pair of nodes are added between copies. We also realize that in our model, the same vertex from both copies can be both visited in a valid augmenting flow. We generalize Blum's definition of strong simplicity to restrict double visit to a given set of edges. We show that the invariants for Blum's algorithm still hold, and adapting the algorithm solves the matching problem on our compressed representation.

▶ **Theorem 1.** *An ε-approximation of maximum cardinality matching on a co-unit disk graph of n points in $\mathbb{R}^d$ can be computed in $O(n^2/\varepsilon^d)$ time.*

Remark Solving the exact problem using Micali-Vazirani algorithm yields a running time of $O(n^{2.5})$. We believe further adapting the algorithm to Hopcraft-Karp style could result in a $O(n^{1.5}/\varepsilon^d)$ algorithm.

2 Hierarchical Bipartite Clique Decomposition

A *hierarchical bipartite clique decomposition* (HBCD hereafter) is defined as (T, B) where

1. T is a tree.
2. B is a set of edges between tree nodes such that neither node is an ancestor of the other. We will call them *bridges*.
3. If $(u, v) \in B$, then for any $x \in S(u), y \in S(v)$, where $S(u)$ is the subtree rooted at u, $(x, y) \neq (u, v) \implies (x, y) \notin B$.

Given an HBCD (T, B), the corresponding general graph $G(V, E)$ is constructed as follows:

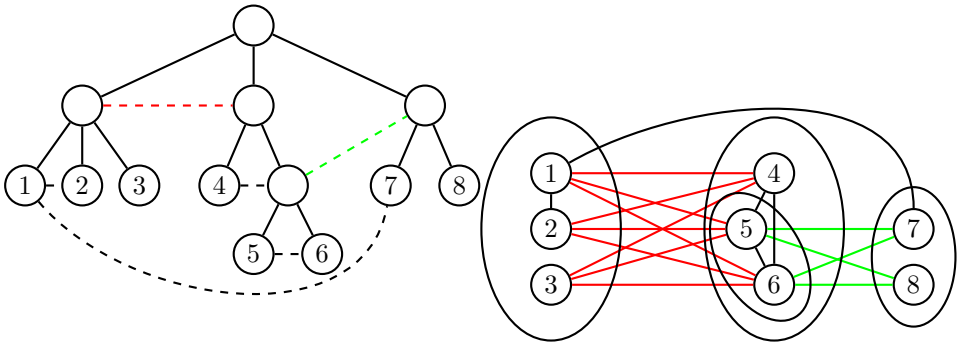
1. V is the set of leaves of T.
2. $(u, v) \in E \iff \exists x \in P(u), \exists y \in P(v), (x, y) \in B$. e.g., there exists a bridge connecting u and v via their ancestor.

A *matching* in an HBCD is a pairing relation between leaves such that the relation is a matching in the corresponding general graph. The *cardinality* of a matching is the number of pairs.

A *flow network* on an instance of HBCD (T, B) is a function $f : E(T) \cup B \to \mathbb{N}$ where $E(T)$ is the edge set of T such that:

1. Flow conservation: For each leaf node u, $\sum_{e \in E(u)} f(e) \leq 1$ where $E(u)$ denotes the set of edges incident to u.
2. Uncapacitated: For every other node u, let A be the set of edges that connect u and its children, and let B be the remaining edges incident to u, $\sum_{e \in A} f(e) = \sum_{e \in B} f(e)$.

Intuitively, we have an undirected flow network where Property 2 mandates that flows are generated at leaf nodes and each leaf node can only contribute to no flow or a single unit



(a) HBCD, dashed edges are bridges **(b)** Equivalent general graph

Figure 1 Correspondence between HBCD and general graph

of flow in the network. Property 1 is the usual flow conservation property where we treat each node as a tunnel between edges from children and other edges.

We call a leaf node *exposed* if there is no flow attached to it.

▶ **Lemma 2.** *Let (T, B) be an instance of HBCD. There is a matching of cardinality n if and only if there is a flow network with $2n$ non-exposed leaf nodes.*

Like an augmenting path for a matching on general graph, we have similar definition for the flow network. The augmenting path in a flow network is a path equipped with ± 1 values, starting and ending at two distinct exposed leaf nodes, and preserve flow conservation locally and validity of flow network globally when added to the network.

Remark In contrast to an augmenting path for general graph, an augmenting path for flow network is not necessarily simple. Moreover, same edge could appear multiple times in a path. However, we will show that it is almost simple after some simplification.

It is also easy to see that adding the values of an augmenting path to a flow network f should produce a flow network that is still valid, but with two fewer exposed leaf nodes.

▶ **Definition 3.** *A simplified augmenting path for a flow network f is a path $\mathcal{P}$ such that:*

1. *$\mathcal{P}$ is an augmenting path.*
2. *Consider the edges on $\mathcal{P}$ to be directed, any edge (u, v) appears at most once in $\mathcal{P}$.*
3. *Consider the edges on $\mathcal{P}$ to be directed, if (u, v) and (v, u) both appear in $\mathcal{P}$, their associated values are the same.*

▶ **Lemma 4.** *If there exists an augmenting path $\mathcal{P}$ in a flow network f, then there also exists a simplified augmenting path $\widehat{\mathcal{P}}$*

▶ **Lemma 5.** *A flow network f is maximum if and only if there is no augmenting path for f.*

3 Algorithm to Find an Augmenting Path

We will modify the maximum cardinality matching algorithm by Blum [1]. The distinct feature of Blum's algorithm is that it does not explicitly deal with blossoms from the traditional Edmond's blossom algorithm. Instead, he reduced the problem to a reachability problem with constraints on paths constructed. We will follow the same idea here.

The main difference between our algorithm from Blum's algorithm is that a valid augmenting path in our model could contain the same edge twice. We need to relax the notion of

"strongly simple" path from Blum's algorithm (meaning that the final path does not contain duplicate vertices). We tackle this by modifying the algorithm with additional notion of one-way/two-way edges.

The general idea is to naturally add tree structure on the two copies that correspond to the possible flow on the edge. We identify pairs of edges that correspond to -1 on an edge with exactly 1 flow to be "one-way", so that such edges cannot be both traversed in the two copies. Blum's algorithm is modified in a way that instead of avoiding visiting the same vertex in both copies, it avoids visiting the same "one-way" edges in both copies.

Due to space limitations, we omit the presentation of the algorithm. These details will be presented in the full version. We obtain the following result.

▶ **Theorem 6.** *Maximum cardinality matching on HBCD can be solved in* $O(|T|(|T| + |B|)$.

We can now prove Theorem 1. Given a point set P, we first compute a quadtree based $(4/\varepsilon)$-WSPD of P, then prune the pairs with closest distance smaller than $1 - \varepsilon$, and perform the maximum cardinality matching described in the previous section. Computing the WSPD takes $O(n/\varepsilon^d + n \log n)$ time, producing a quad tree of size $O(n)$ and a list of pairs of size $O(n/\varepsilon^d)$. The matching takes time $O(n^2/\varepsilon^d)$, and so the overall running time is $O(n^2/\varepsilon^d)$.

References

1 N. Blum. A new approach to maximum matching in general graphs. *Proc. 17th Internat. Colloq. on Autom., Lang. and Program.*, pages 586–597, 1990. Note: A newer and more complete version can be found at https://arxiv.org/abs/1509.04927. doi:10.1007/BFb0032060.

2 É. Bonnet, S. Cabello, and W. Mulzer. Maximum Matchings in Geometric Intersection Graphs. *Proc. 37th Internat. Sympos. on Theoret. Aspects of Comp. Sci.*, 154:31:1–31:17, 2020. URL: https://drops.dagstuhl.de/opus/volltexte/2020/11892, doi:10.4230/LIPIcs.STACS.2020.31.

3 B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Math.*, 86(1):165–177, 1990. URL: https://www.sciencedirect.com/science/article/pii/0012365X90903580, doi:https://doi.org/10.1016/0012-365X(90)90358-O.

4 J. Edmonds. Paths, trees, and flowers. *Canad. J. Math*, 17:449–467, 1965. doi:10.4153/CJM-1965-045-4.

5 A. Efrat, A. Itai, and M. J. Katz. Geometry helps in bottleneck matching and related problems. *Algorithmica*, 31(1):1–28, September 2001. doi:10.1007/s00453-001-0016-8.

6 J. E. Hopcroft and R. M. Karp. A $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. In *12th Annual Symposium on Switching and Automata Theory (swat 1971)*, pages 122–125, 1971. doi:10.1109/SWAT.1971.1.

7 S. Micali and V. V. Vazirani. An $O(\sqrt{|v|} \cdot |E|)$ algoithm for finding maximum matching in general graphs. In *21st Annual Symposium on Foundations of Computer Science (sfcs 1980)*, pages 17–27, 1980. doi:10.1109/SFCS.1980.12.

8 P. A. Peterson and M. C. Loui. The general maximum matching algorithm of Micali and Vazirani. *Algorithmica*, 3(1):511–533, Nov 1988. doi:10.1007/BF01762129.

9 R. Sharathkumar and P. K. Agarwal. A near-linear time ϵ-approximation algorithm for geometric bipartite matching. *Proc. 44th Annu. ACM Sympos. Theory Comput.*, page 385–394, 2012. doi:10.1145/2213977.2214014.

10 P. M. Vaidya. Approximate minimum weight matching on points in k-dimensional space. *Algorithmica*, 4(1):569–583, June 1989. doi:10.1007/BF01553909.

11 K. R. Varadarajan and P. K. Agarwal. Approximation algorithms for bipartite and non-bipartite matching in the plane. *Proc. Tenth Annu. ACM-SIAM Sympos. Discrete Algorithms*, page 805–814, 1999. doi:10.5555/314500.314918.