Finding Complete Bipartite Subgraphs in Bipartite Graphs

Mark de Berg Mark Overmars Marc van Kreveld

RUU-CS-89-30
December 1989

Utrecht University

Department of Computer Science

Padualaan 14, P.O. Box 80.089,
3508 TB Utrecht, The Netherlands,
Tel. : ... + 31 - 30 - 531454

Finding Complete Bipartite Subgraphs in Planar Graphs*

Mark de Berg Mark Overmars Marc van Kreveld

Abstract

Given a bipartite graph G, we investigate the problem of determining whether G contains a $K_{l,m}$ (a complete bipartite graph with l nodes in one node set and m nodes in the other node set) as a subgraph. Using a transformation from a geometrical setting, we solve this problem in time $O(n^{2-1/l})$ if $m \geq l!$, where n is the number of edges of G and $l = \min(l, m)$.

1 Introduction

Given a graph G, a natural question is to ask whether G contains some given subgraph H as a subgraph. For example, one might want to find out if G contains a cycle of certain length (i.e. $H = C_k$) or a clique of certain size ($H = K_l$).

In this paper we present an algorithm that tests efficiently whether a bipartite graph G contains a $K_{l,m}$ as a subgraph. ($K_{l,m}$ is a complete bipartite graph with l nodes in one node set and m nodes in the other node set.) For bipartite graphs the problem is NP-complete (see Garey and Johnson [2], problem GT24), so we consider l and m to be constants. For the case $l = m = 2$ efficient algorithms already exist: Chiba and Nishizeki [1] and van Kreveld and de Berg [4] give algorithms that work in time $O(n\sqrt{n})$, where n is the number of edges in G. (In fact, Chiba and Nishizeki solve the more general problem of finding cycles of length four in an arbitrary graph.) Our new algorithm works for arbitrary l and m and takes time $O(n^{2-1/l})$ if $m \geq l!$, where $l = \min(l, m)$. For $l = m = 2$ this is only slightly worse than the previous results.

The method we use is similar to the one used in [4]. We transform the graph problem into a geometrical setting, where it becomes a pattern recognition problem: the set of points in the plane, determine whether this set contains a subset of points that form the vertices of an l-convex polygon. (A polygon is l-convex if ...)

Dept. of Computer Science, Utrecht University, P.O. Box 80.089, TB Utrecht, the Netherlands.

This research was partially supported by the ESPRIT II Basic Research Actions Program of the EC under contract No. 3075 (project ALCOM). The work of the first author was also supported by the Dutch Organization for Scientific Research (N.W.O.).

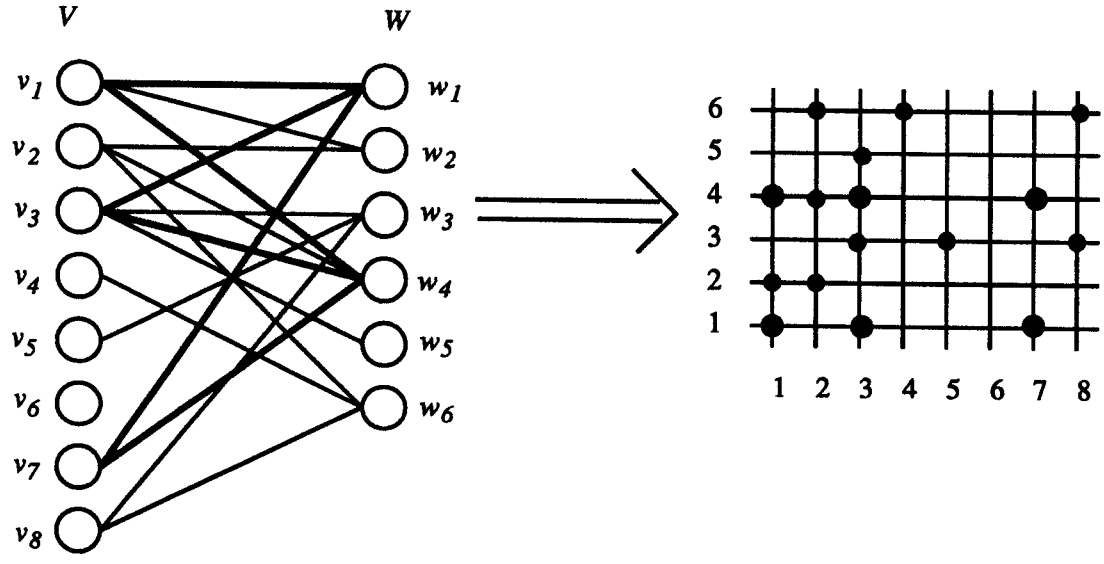

Figure 1: Transforming a bipartite graph to a set of points. Observe that the graph contains a $K_{2,3}$ (the thick edges) and hence the set of points contains a 2×3 subgrid (the thick points).

problem lies in the fact that a set of points cannot contain too many subgrids of size $l \times l$ without containing a subgrid of size $l \times m$.

The sequel of this paper is organised as follows.

First we give the transformation from the graph problem to the geometrical setting and we give an algorithm that counts the number of $l \times l$ subgrids in a set of points. Then we show how to use this algorithm to determine whether there is a $l \times m$ subgrid.

2 The algorithm

Let $G = (V \cup W, E)$ be a bipartite graph. The nodes in V are labeled $v_1, \ldots, v_{|V|}$ and the nodes in W are labeled $w_1, \ldots, w_{|W|}$. The edge between v_i and w_j is denoted (v_i, w_j). With this graph G we associate a set S_G of points in the plane as follows: $S_G = \{(i,j) | (v_i, w_j) \in E\}$. Thus there is a one-to-one correspondence between the edges of G and the points in S_G, and two points in S_G have the same x–coordinate (y–coordinate) if and only if the corresponding edges are incident to the same node of V (W). Hence, under this transformation a $K_{l,m}$ corresponds to a set of points that lie on an axis-parallel grid of size $l \times m$ (see Figure 1). A $K_{2,2}$, for example, corresponds to four points that are the vertices of an axis-parallel rectangle. This transformation is also used in [4].

In the remainder of this paper we will take the geometrical point of view and show how to determine whether a set S of n points in the plane contains a subset of size $l \times m$ whose points lie on an axis-parallel grid of size $l \times m$. We denote such a grid simply by $K_{l,m}$.

2.1 Counting $K_{l,l}$'s

We first turn our attention to the counting (and reporting) of the $K_{l,l}$'s in a set S of points in the plane. The idea is to partition S into subsets whose points have equal first coordinates and distinguish between small subsets and large subsets. Let a *column* (of size l) be a set of l points that lie on a common vertical line; similarly, a *row* is a set of points that lie on a common horizontal line. We identify a column C consisting of the points $(x, y_1), \ldots, (x, y_l)$, where $y_1 < y_2 < \cdots < y_l$, with the point $(y_1, \ldots, y_l)$ in l–dimensional space. Thus to find a $K_{l,l}$ we have to look for l columns that are identical (under this identification). Because of the size of the small subsets we can afford to enumerate all columns of size l in the small subsets. Then we search with these columns in the large subsets. To do this efficiently we make a distinction between large subsets that are relatively small and large subsets that are relatively large. This way we count the number of times such a column is present. Then we can calculate the number of $K_{l,l}$'s. The only $K_{l,l}$'s that we have not counted so far are those having all their columns in large subsets. These $K_{l,l}$'s are counted by reversing the roles of x– and y–coordinates.

We next give a more detailed description of this algorithm.

1. Partition S into subsets whose points have equal first coordinate. Let $S_1, \ldots, S_a$ be the subsets of size at most n^α (the *small* subsets). Let $L_1, \ldots, L_b$ be the sets of size between n^α and n^β, and let $L'_1, \ldots, L'_{b'}$ be the sets of size greater than n^β (the *large* subsets that are relatively small and the ones that are relatively large). α and β are parameters to be chosen later (with $\beta \geq \alpha$).

2. For each S_i, enumerate all possible columns of size l having the points in S_i. Let $C_1, \ldots, C_d$ be the different columns thus obtained. (Recall that we identify columns with points in l–dimensional space and that two columns are equal if the set of y–coordinates of the points in one column equals the set of y–coordinates of the points in the other column.) Store the multiplicity of the columns (the number of times a column occurs) in an array $M_S[1..d]$. E.g., if some C_i occurs in S_j and $S_{j'}$ but in no other small subset we set $M_S[i] := 2$. The multiplicities are found by sorting the columns lexicographically. Now columns that are identical are adjacent in this sorted list.

3. Search for the occurences in the large subsets of columns found in step 2 in the small subsets as follows. Initialize all entries in an array $M_L[1..d]$ to 0.

(i) Count the number of times a C_i occurs in an L_j as follows. Enumerate all columns in every L_j and sort them lexicographically. Now walk simultaneously along this sorted list and the sorted list of C_i's. If a C_i occurs in c L_j's, set $M_L[i] := c$.

(ii) Count the number of times a C_i occurs in an L'_j as follows. Build a search tree T on the y–coordinates of the points in $\bigcup_{1\le j\le b'} L'_j$. Store at each leaf γ in T a list containing all j–values such that L'_j contains a point whose y–coordinate is equal to γ_y, the y–coordinate corresponding to γ. With this tree T we can find in $O(\log n)$ time a list of all L'_j's that contain a point with some specific y–coordinate by searching in T with this y–coordinate.

For each C_i do the following. For every coordinate y of C_i (these are the y–coordinates of the points in the column) search in T and mark each L'_j that contains a point whose y–coordinate is equal to y. (The names of these subsets are stored at the leaf γ with $\gamma_y = y$.) Check each L'_j to see if it is marked l times and if this is the case (every point of C_i has been found in L'_j, i.e., C_i occurs in L'_j) increment $M_L[i]$.

4. To count the number of $K_{l,l}$'s with all columns in large subsets we partition $\bigcup_{1\le j\le b} L_j \cup \bigcup_{1\le j\le b'} L'_j$, the set of all points in large subsets, into new subsets whose points have equal y–coordinate. We then enumerate for each new subset all possible rows having their points in this subset and we compute $M'[1..d']$, an array containing the multiplicities of the d' different rows, by sorting the rows (as M was computed in step 2).

5. The number of $K_{l,l}$'s is now given by

$$\sum_{i=1}^{d} \binom{M_S[i]+M_L[i]}{l} + \sum_{j=1}^{d'} \binom{M'[j]}{l} - \sum_{i=1}^{d} \binom{M_L[i]}{l},$$

where we define $\binom{r}{t} = 0$ for $r < t$. (The last summation gives the number of $K_{l,l}$'s having all columns in large subsets which have a corresponding column in a small subset. These $K_{l,l}$'s are counted twice, so we have to subtract this number.)

This leads to:

Lemma 1 *The number of $K_{l,l}$'s in a set of n points can be counted in time $O(n^{\frac{1}{2}l+\frac{3}{4}-\frac{1}{8l-4}})$.*

Proof: We first note that all the coordinates of the points lie in the range $1 \ldots n$. Hence, the sorting steps that are needed in the algorithm only take time linear in the number objects to be sorted (plus $O(n)$, of course). Thus the first step of the

algorithm only takes $O(n)$ time and the second step takes time linear in the number of columns in small subsets, which is

$$\sum_{i=1}^{a}\binom{|S_i|}{l} \leq \sum_{i=1}^{a}|S_i|^l \leq (n^\alpha)^{l-1}\sum_{i=1}^{a}|S_i| \leq n^{\alpha l-\alpha+1}.$$

Step 2(i) takes time linear in the number of columns in large subsets that are relatively small plus the number of columns in the small subsets. Hence, this takes $O(n^{\beta l-\beta+1})$ time.

Step 2(ii) takes time $O(\sum_{i=1}^{d} l(\log n + n^{1-\beta}))$ (with every point in each C_i we have to search, which takes $O(\log n)$ time, and increment at most $n^{1-\beta}$ counters) plus $O(n)$ (to build the tree). Because $d = O(n^{\alpha l-\alpha+1})$, the time for the second step is $O(n^{2+\alpha(l-1)-\beta})$ (plus $O(n)$).

The time spent in the third step is bounded by $O(n^{\alpha(1-l)+l})$, the maximal number of rows in large subsets.

Hence, to obtain the best time bound we have to minimize

$$\max(n^{\alpha l-\alpha+1}, n^{\beta l-\beta+1}, n^{2+\alpha(l-1)-\beta}, n^{\alpha(1-l)+l}).$$

Note that the first term is never greater than the second term since $\beta \geq \alpha$. Also observe that if we choose $\alpha = \frac{1}{2} - \varepsilon$ and $\beta = \frac{1}{2} + \varepsilon$ the second and fourth term are equal. If we let $\varepsilon = \frac{1}{4l-2}$, then we even have that the second and fourth term are equal to the third. Since the second term increasing in β, the third term is decreasing in β and increasing in α and the fourth term is decreasing in α, this must be the optimal solution. Substituting the values $\alpha = \frac{1}{2} - \frac{1}{4l-2}$ and $\beta = \frac{1}{2} + \frac{1}{4l-2}$ yields the claimed time bound. □

2.2 Finding $K_{l,m}$'s

We will now show how to use the algorithm of the previous section to determine efficiently whether a set of points contains a $K_{l,m}$. Assume w.l.o.g. that $l \leq m$.

Lemma 2 *If a set S of n points in the plane contains more than $2\binom{m-1}{l}n^{\frac{1}{2}l+\frac{1}{2}}$ $K_{l,l}$'s then S contains at least one $K_{l,m}$.*

Proof: Suppose that S contains no $K_{l,m}$. Partition S into subsets whose points have equal x–coordinate. In other words, two points are in the same subset iff they have the same x–coordinate. Distinguish between subsets that contain at most $\sqrt{n}$ points (the *small* subsets) and subsets that contain more than $\sqrt{n}$ points (the *large* subsets).

Let $S_1, \ldots, S_a$ be the collection of small subsets. Consider all columns of size l in the small subsets. Clearly, the number of these columns is

$$\sum_{i=1}^{a}\binom{|S_i|}{l} \leq \sum_{i=1}^{a}|S_i|^l \leq \sqrt{n}^{l-1}\sum_{i=1}^{a}|S_i| \leq n^{\frac{1}{2}l+\frac{1}{2}}.$$

Now the number of occurences of some particular column is smaller than m (otherwise there would be a $K_{l,m}$). Thus any particular column can take part in at most $\binom{m-1}{l}$ $K_{l,l}$'s. Since there were at most $n^{\frac{1}{2}l+\frac{1}{2}}$ columns in the small subsets, there are no more than $\binom{m-1}{l} n^{\frac{1}{2}l+\frac{1}{2}}$ $K_{l,l}$'s having at least one column in a small subset.

To count the number of $K_{l,l}$'s having all their columns in large subsets, we note that there cannot be more than $\sqrt{n}$ large subsets. Hence, the number of rows in large subsets is bounded by $n^{\frac{1}{2}l+\frac{1}{2}}$. The same argument now shows that there are at most $\binom{m-1}{l} n^{\frac{1}{2}l+\frac{1}{2}}$ $K_{l,l}$'s having all rows, and thus all columns, in large subsets. □

Our strategy to decide upon the existence of a $K_{l,m}$ will thus be to count the number of $K_{l,l}$'s (where $l \leq m$). If this number is greater than $2\binom{m-1}{l} n^{\frac{1}{2}l+\frac{1}{2}}$ then we know by the above lemma that there must also be a $K_{l,m}$. If the number is smaller then we sort all the $K_{l,l}$'s we have found lexicographically; a $K_{l,m}$ would give $m-l+1$ $K_{l,l}$'s that are adjacent in this sorted list. (Observe that the algorithm that counts the $K_{l,l}$'s can easily be adapted to report the $K_{l,l}$'s. Thus what we do is start reporting them until their number becomes too large. Then we know that there must be a $K_{l,m}$. If this does not happen we have all $K_{l,l}$'s available for the last step.)

The counting of the $K_{l,l}$'s is done using the algorithm presented in the previous section. We now describe the last step (determining whether there is a $K_{l,m}$ if the number of $K_{l,l}$'s is small) in a little more detail. Let K be a $K_{l,l}$. Let $x_1, \ldots, x_l$ and $y_1, \ldots, y_l$ be the different x–coordinates resp. y–coordinates of the points of K in sorted order. We identify K with the point $(x_1, \ldots, x_l, y_1, \ldots, y_l)$ in $2l$–dimensional space. This induces an ordering on the $K_{l,l}$'s that corresponds to the lexicographical ordering on the corresponding points in $2l$-dimensional space. It is easily seen that if there is a 'vertical' $K_{l,m}$ (a $K_{l,m}$ with l columns and m rows) then there must be at least $m-l+1$ $K_{l,l}$'s that are adjacent in the above ordering whose points together form a $K_{l,m}$. (Notice that not every $K_{l,m}$ has this property. However, if there are vertical $K_{l,m}$'s, then at least one has.) Thus by walking along the ordered list of $K_{l,l}$'s we can check for the existence of a 'vertical' $K_{l,m}$ in time linear in the length of this list. To find 'horizontal' $K_{l,m}$'s we just reverse the roles of the x– and y–coordinates.

This leads to:

Theorem 1 *Given a bipartite graph G with n edges, it can be decided in time $O(n^{\frac{1}{2}t+\frac{3}{4}-\frac{1}{8t-4}})$ whether G contains a $K_{l,m}$ as a subgraph, where $t = \min(l,m)$.*

Proof: The fact that the algorithm presented above correctly solves the (equivalent geometrical) problem immediately follows from Lemma 2. Let $l \leq m$. The time taken by the algorithm that counts the number of $K_{l,l}$'s is $O(n^{\frac{1}{2}l+\frac{3}{4}-\frac{1}{8l-4}})$ (Lemma 1) and the time needed to test if there is a $K_{l,m}$ when the number of $K_{l,l}$'s is smaller

than $2\binom{m-1}{l}n^{\frac{1}{2}l+\frac{1}{2}}$ is linear in this number since, as we already noted, the sorting takes only linear time. □

References

[1] Chiba, N., and T. Nishizeki, Arboricity and Subgraph Listing Algorithms, *SIAM J. on Comput.* 14 (1985), pp. 210-223.

[2] Garey, M.R., and D.S. Johnson, *Computers and Intractability, A Guide to the Theory of NP–Completeness*, W.H. Freeman and Company, New York, 1979.

[3] Itai, A., and M. Rodeh, Finding a Minimum Circuit in a Graph, *SIAM J. on Comput.* 7 (1978), pp.413-423.

[4] van Kreveld, M., and M. de Berg, Finding Squares and Rectangles in Sets of Points, to appear in *Proc. Workshop on Graph-Theoretic Concepts in Computer Science WG '89.*

[5] Papadimitriou, C.H., and M. Yannakakis, The Clique Problem for Planar Graphs, *Inform. Proc. Lett.* 13 (1981), pp. 131-133.

[6] Richards, D., Finding Short Cycles in Planar Graphs Using Separators, *J. of Algorithms* 7 (1986), pp. 382-394.

[7] Richards, D., and A.L. Liestman, Finding Cycles of a Given Length, *Ann. Discrete Math.* 27 (1985), pp. 249-256.

Finding Complete Bipartite Subgraphs in Bipartite Graphs

Mark de Berg Mark Overmars Marc van Kreveld

RUU-CS-89-30
December 1989

Utrecht University

Department of Computer Science

Padualaan 14, P.O. Box 80.089,
3508 TB Utrecht, The Netherlands,
Tel. : ... + 31 - 30 - 531454

Finding Complete Bipartite Subgraphs in Bipartite Graphs

Mark de Berg Mark Overmars Marc van Kreveld

Technical Report RUU-CS-89-30
December 1989

Department of Computer Science
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands

Finding Complete Bipartite Subgraphs in Bipartite Graphs*

Mark de Berg Mark Overmars Marc van Kreveld

Abstract

Given a bipartite graph G, we investigate the problem of determining whether G contains a $K_{l,m}$ (a complete bipartite graph with l nodes in one node set and m nodes in the other node set) as a subgraph. Using a transformation to a geometrical setting, we solve this problem in time $O(n^{\frac{1}{2}t+\frac{3}{4}-\frac{1}{8t-4}})$, where n is the number of edges of G and $t = \min(l, m)$.

1 Introduction

Given a graph G, a natural question to ask is whether G contains some specific graph H as a subgraph. One could, for example, ask whether G contains a cycle of certain length ([1, 3, 6, 7]) or a clique of certain size ([1, 5]).

In this paper we present an algorithm that tests efficiently whether a bipartite graph G contains a $K_{l,m}$ as a subgraph. (A $K_{l,m}$ is a complete bipartite graph with l nodes in one node set and m nodes in the other node set.) For variable l and m this problem is NP–complete (see Garey and Johnson [2], problem GT24), so we consider l and m to be constants. For the case $l = m = 2$ efficient algorithms already exist; Chiba and Nishizeki [1] and van Kreveld and de Berg [4] give algorithms that work in time $O(n\sqrt{n})$, where n is the number of edges in G. (In fact, Chiba and Nishizeki solve the more general problem of finding cycles of length four in an arbitrary graph.) Our new algorithm works for arbitrary l and m and takes time $O(n^{\frac{1}{2}t+\frac{3}{4}-\frac{1}{8t-4}})$, where $t = \min(l, m)$. For $l = m = 2$ this is only slightly worse than the previous results.

The method we use is similar to the one used in [4]. We transform the graph problem to a geometrical setting, where it becomes a pattern recognition problem: Given a set of points in the plane, determine whether this set contains a subset of points that form the vertices of an $l \times m$ subgrid. The key to our solution to this

*Dept. of Computer Science, Utrecht University, P.O.Box 80.089, 3508 TB Utrecht, The Netherlands. This research was partially supported by the ESPRIT II Basic Research Actions Program of the EC under contract No. 3075 (project ALCOM). The work of the first author was also supported by the Dutch Organisation for Scientific Research (N.W.O.).

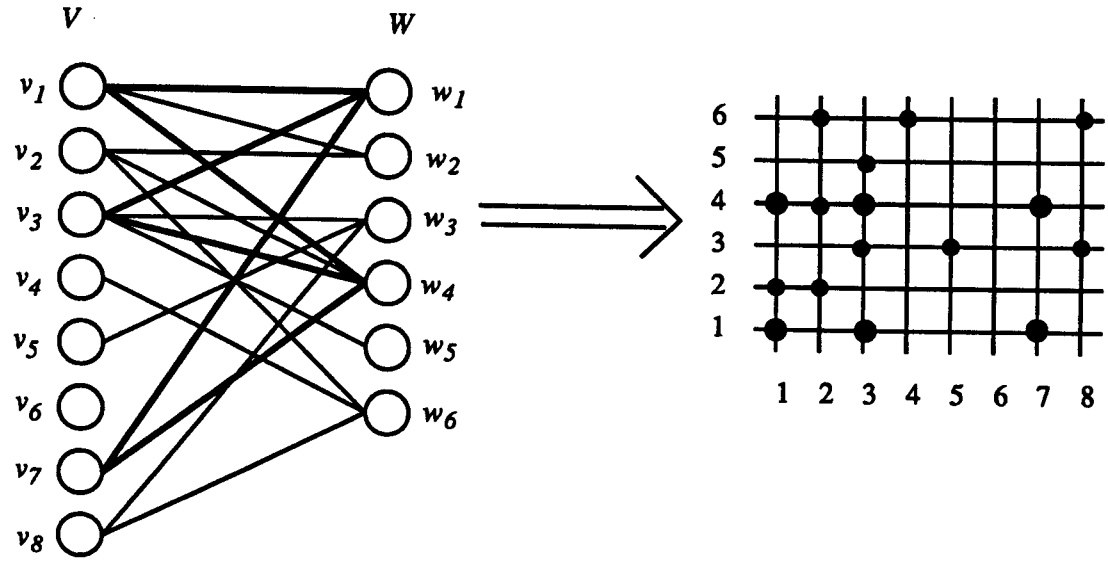

Figure 1: Transforming a bipartite graph to a set of points. Observe that the graph contains a $K_{2,3}$ (the thick edges) and hence the set of points contains a 2×3 subgrid (the thick points).

problem lies in the fact that a set of points cannot contain too many subgrids of size $l \times l$ without containing a subgrid of size $l \times m$.

The sequel of this paper is organised as follows.

First we give the transformation from the graph problem to the geometrical setting and we give an algorithm that counts the number of $l \times l$ subgrids in a set of points. Then we show how to use this algorithm to determine whether there is a $l \times m$ subgrid.

2 The algorithm

Let $G = (V \cup W, E)$ be a bipartite graph. The nodes in V are labeled $v_1, \ldots, v_{|V|}$ and the nodes in W are labeled $w_1, \ldots, w_{|W|}$. The edge between v_i and w_j is denoted (v_i, w_j). With this graph G we associate a set S_G of points in the plane as follows: $S_G = \{(i,j) | (v_i, w_j) \in E\}$. Thus there is a one-to-one correspondence between the edges of G and the points in S_G, and two points in S_G have the same x–coordinate (y–coordinate) if and only if the corresponding edges are incident to the same node of V (W). Hence, under this transformation a $K_{l,m}$ corresponds to a set of points that lie on an axis-parallel grid of size $l \times m$ (see Figure 1). A $K_{2,2}$, for example, corresponds to four points that are the vertices of an axis-parallel rectangle. This transformation is also used in [4].

In the remainder of this paper we will take the geometrical point of view and show how to determine whether a set S of n points in the plane contains a subset of size $l \times m$ whose points lie on an axis-parallel grid of size $l \times m$. We denote such a grid simply by $K_{l,m}$.

2.1 Counting $K_{l,l}$'s

We first turn our attention to the counting (and reporting) of the $K_{l,l}$'s in a set S of points in the plane. The idea is to partition S into subsets whose points have equal first coordinates and distinguish between small subsets and large subsets. Let a *column* (of size l) be a set of l points that lie on a common vertical line; similarly, a *row* is a set of points that lie on a common horizontal line. We identify a column C consisting of the points $(x, y_1), \ldots, (x, y_l)$, where $y_1 < y_2 < \cdots < y_l$, with the point $(y_1, \ldots, y_l)$ in l–dimensional space. Thus to find a $K_{l,l}$ we have to look for l columns that are identical (under this identification). Because of the size of the small subsets we can afford to enumerate all columns of size l in the small subsets. Then we search with these columns in the large subsets. To do this efficiently we make a distinction between large subsets that are relatively small and large subsets that are relatively large. This way we count the number of times such a column is present. Then we can calculate the number of $K_{l,l}$'s. The only $K_{l,l}$'s that we have not counted so far are those having all their columns in large subsets. These $K_{l,l}$'s are counted by reversing the roles of x– and y–coordinates.

We next give a more detailed description of this algorithm.

1. Partition S into subsets whose points have equal first coordinate. Let $S_1, \ldots, S_a$ be the subsets of size at most n^α (the *small* subsets). Let $L_1, \ldots, L_b$ be the sets of size between n^α and n^β, and let $L'_1, \ldots, L'_{b'}$ be the sets of size greater than n^β (the *large* subsets that are relatively small and the ones that are relatively large). α and β are parameters to be chosen later (with $\beta \geq \alpha$).

2. For each S_i, enumerate all possible columns of size l having the points in S_i. Let $C_1, \ldots, C_d$ be the different columns thus obtained. (Recall that we identify columns with points in l–dimensional space and that two columns are equal if the set of y–coordinates of the points in one column equals the set of y–coordinates of the points in the other column.) Store the multiplicity of the columns (the number of times a column occurs) in an array $M_S[1..d]$. E.g., if some C_i occurs in S_j and $S_{j'}$ but in no other small subset we set $M_S[i] := 2$. The multiplicities are found by sorting the columns lexicographically. Now columns that are identical are adjacent in this sorted list.

3. Search for the occurences in the large subsets of columns found in step 2 in the small subsets as follows. Initialize all entries in an array $M_L[1..d]$ to 0.

(i) Count the number of times a C_i occurs in an L_j as follows. Enumerate all columns in every L_j and sort them lexicographically. Now walk simultaneously along this sorted list and the sorted list of C_i's. If a C_i occurs in c L_j's, set $M_L[i] := c$.

(ii) Count the number of times a C_i occurs in an L'_j as follows. Build a search tree T on the y–coordinates of the points in $\bigcup_{1\le j\le b'} L'_j$. Store at each leaf γ in T a list containing all j–values such that L'_j contains a point whose y–coordinate is equal to γ_y, the y–coordinate corresponding to γ. With this tree T we can find in $O(\log n)$ time a list of all L'_j's that contain a point with some specific y–coordinate by searching in T with this y–coordinate.

For each C_i do the following. For every coordinate y of C_i (these are the y–coordinates of the points in the column) search in T and mark each L'_j that contains a point whose y–coordinate is equal to y. (The names of these subsets are stored at the leaf γ with $\gamma_y = y$.) Check each L'_j to see if it is marked l times and if this is the case (every point of C_i has been found in L'_j, i.e., C_i occurs in L'_j) increment $M_L[i]$.

4. To count the number of $K_{l,l}$'s with all columns in large subsets we partition $\bigcup_{1\le j\le b} L_j \cup \bigcup_{1\le j\le b'} L'_j$, the set of all points in large subsets, into new subsets whose points have equal y–coordinate. We then enumerate for each new subset all possible rows having their points in this subset and we compute $M'[1..d']$, an array containing the multiplicities of the d' different rows, by sorting the rows (as M was computed in step 2).

5. The number of $K_{l,l}$'s is now given by

$$\sum_{i=1}^{d} \binom{M_S[i]+M_L[i]}{l} + \sum_{j=1}^{d'} \binom{M'[j]}{l} - \sum_{i=1}^{d} \binom{M_L[i]}{l},$$

where we define $\binom{r}{t} = 0$ for $r < t$. (The last summation gives the number of $K_{l,l}$'s having all columns in large subsets which have a corresponding column in a small subset. These $K_{l,l}$'s are counted twice, so we have to subtract this number.)

This leads to:

Lemma 1 *The number of $K_{l,l}$'s in a set of n points can be counted in time $O(n^{\frac{1}{2}l+\frac{3}{4}-\frac{1}{8l-4}})$.*

Proof: We first note that all the coordinates of the points lie in the range $1 \ldots n$. Hence, the sorting steps that are needed in the algorithm only take time linear in the number objects to be sorted (plus $O(n)$, of course). Thus the first step of the

algorithm only takes $O(n)$ time and the second step takes time linear in the number of columns in small subsets, which is

$$\sum_{i=1}^{a}\binom{|S_i|}{l} \leq \sum_{i=1}^{a}|S_i|^l \leq (n^\alpha)^{l-1}\sum_{i=1}^{a}|S_i| \leq n^{\alpha l-\alpha+1}.$$

Step 2(i) takes time linear in the number of columns in large subsets that are relatively small plus the number of columns in the small subsets. Hence, this takes $O(n^{\beta l-\beta+1})$ time.

Step 2(ii) takes time $O(\sum_{i=1}^{d} l(\log n + n^{1-\beta}))$ (with every point in each C_i we have to search, which takes $O(\log n)$ time, and increment at most $n^{1-\beta}$ counters) plus $O(n)$ (to build the tree). Because $d = O(n^{\alpha l-\alpha+1})$, the time for the second step is $O(n^{2+\alpha(l-1)-\beta})$ (plus $O(n)$).

The time spent in the third step is bounded by $O(n^{\alpha(1-l)+l})$, the maximal number of rows in large subsets.

Hence, to obtain the best time bound we have to minimize

$$\max(n^{\alpha l-\alpha+1}, n^{\beta l-\beta+1}, n^{2+\alpha(l-1)-\beta}, n^{\alpha(1-l)+l}).$$

Note that the first term is never greater than the second term since $\beta \geq \alpha$. Also observe that if we choose $\alpha = \frac{1}{2} - \varepsilon$ and $\beta = \frac{1}{2} + \varepsilon$ the second and fourth term are equal. If we let $\varepsilon = \frac{1}{4l-2}$, then we even have that the second and fourth term are equal to the third. Since the second term increasing in β, the third term is decreasing in β and increasing in α and the fourth term is decreasing in α, this must be the optimal solution. Substituting the values $\alpha = \frac{1}{2} - \frac{1}{4l-2}$ and $\beta = \frac{1}{2} + \frac{1}{4l-2}$ yields the claimed time bound. □

2.2 Finding $K_{l,m}$'s

We will now show how to use the algorithm of the previous section to determine efficiently whether a set of points contains a $K_{l,m}$. Assume w.l.o.g. that $l \leq m$.

Lemma 2 *If a set S of n points in the plane contains more than $2\binom{m-1}{l-1}n^{\frac{1}{2}l+\frac{1}{2}}$ $K_{l,l}$'s then S contains at least one $K_{l,m}$.*

Proof: Suppose that S contains no $K_{l,m}$. Partition S into subsets whose points have equal x–coordinate. In other words, two points are in the same subset iff they have the same x–coordinate. Distinguish between subsets that contain at most $\sqrt{n}$ points (the *small* subsets) and subsets that contain more than $\sqrt{n}$ points (the *large* subsets).

Let $S_1, \ldots, S_a$ be the collection of small subsets. Consider all columns of size l in the small subsets. Clearly, the number of these columns is

$$\sum_{i=1}^{a}\binom{|S_i|}{l} \leq \sum_{i=1}^{a}|S_i|^l \leq \sqrt{n}^{l-1}\sum_{i=1}^{a}|S_i| \leq n^{\frac{1}{2}l+\frac{1}{2}}.$$

Now the number of occurences of some particular column is smaller than m (otherwise there would be a $K_{l,m}$). Thus any particular column can take part in at most $\binom{m-1}{l}$ $K_{l,l}$'s. Since there were at most $n^{\frac{1}{2}l+\frac{1}{2}}$ columns in the small subsets, there are no more than $\binom{m-1}{l} n^{\frac{1}{2}l+\frac{1}{2}}$ $K_{l,l}$'s having at least one column in a small subset.

To count the number of $K_{l,l}$'s having all their columns in large subsets, we note that there cannot be more than $\sqrt{n}$ large subsets. Hence, the number of rows in large subsets is bounded by $n^{\frac{1}{2}l+\frac{1}{2}}$. The same argument now shows that there are at most $\binom{m-1}{l} n^{\frac{1}{2}l+\frac{1}{2}}$ $K_{l,l}$'s having all rows, and thus all columns, in large subsets. □

Our strategy to decide upon the existence of a $K_{l,m}$ will thus be to count the number of $K_{l,l}$'s (where $l \leq m$). If this number is greater than $2\binom{m-1}{l} n^{\frac{1}{2}l+\frac{1}{2}}$ then we know by the above lemma that there must also be a $K_{l,m}$. If the number is smaller then we sort all the $K_{l,l}$'s we have found lexicographically; a $K_{l,m}$ would give $m-l+1$ $K_{l,l}$'s that are adjacent in this sorted list. (Observe that the algorithm that counts the $K_{l,l}$'s can easily be adapted to report the $K_{l,l}$'s. Thus what we do is start reporting them until their number becomes too large. Then we know that there must be a $K_{l,m}$. If this does not happen we have all $K_{l,l}$'s available for the last step.)

The counting of the $K_{l,l}$'s is done using the algorithm presented in the previous section. We now describe the last step (determining whether there is a $K_{l,m}$ if the number of $K_{l,l}$'s is small) in a little more detail. Let K be a $K_{l,l}$. Let $x_1, \ldots, x_l$ and $y_1, \ldots, y_l$ be the different x–coordinates resp. y–coordinates of the points of K in sorted order. We identify K with the point $(x_1, \ldots, x_l, y_1, \ldots, y_l)$ in $2l$–dimensional space. This induces an ordering on the $K_{l,l}$'s that corresponds to the lexicographical ordering on the corresponding points in $2l$-dimensional space. It is easily seen that if there is a 'vertical' $K_{l,m}$ (a $K_{l,m}$ with l columns and m rows) then there must be at least $m-l+1$ $K_{l,l}$'s that are adjacent in the above ordering whose points together form a $K_{l,m}$. (Notice that not every $K_{l,m}$ has this property. However, if there are vertical $K_{l,m}$'s, then at least one has.) Thus by walking along the ordered list of $K_{l,l}$'s we can check for the existence of a 'vertical' $K_{l,m}$ in time linear in the length of this list. To find 'horizontal' $K_{l,m}$'s we just reverse the roles of the x– and y–coordinates.

This leads to:

Theorem 1 *Given a bipartite graph G with n edges, it can be decided in time $O(n^{\frac{1}{2}t+\frac{3}{4}-\frac{1}{8t-4}})$ whether G contains a $K_{l,m}$ as a subgraph, where $t = \min(l, m)$.*

Proof: The fact that the algorithm presented above correctly solves the (equivalent geometrical) problem immediately follows from Lemma 2. Let $l \leq m$. The time taken by the algorithm that counts the number of $K_{l,l}$'s is $O(n^{\frac{1}{2}l+\frac{3}{4}-\frac{1}{8l-4}})$ (Lemma 1) and the time needed to test if there is a $K_{l,m}$ when the number of $K_{l,l}$'s is smaller

than $2\binom{m-1}{l}n^{\frac{1}{2}l+\frac{1}{2}}$ is linear in this number since, as we already noted, the sorting takes only linear time. □

References

[1] Chiba, N., and T. Nishizeki, Arboricity and Subgraph Listing Algorithms, *SIAM J. on Comput.* 14 (1985), pp. 210-223.

[2] Garey, M.R., and D.S. Johnson, *Computers and Intractability, A Guide to the Theory of NP–Completeness*, W.H. Freeman and Company, New York, 1979.

[3] Itai, A., and M. Rodeh, Finding a Minimum Circuit in a Graph, *SIAM J. on Comput.* 7 (1978), pp.413-423.

[4] van Kreveld, M., and M. de Berg, Finding Squares and Rectangles in Sets of Points, to appear in *Proc. Workshop on Graph-Theoretic Concepts in Computer Science WG '89.*

[5] Papadimitriou, C.H., and M. Yannakakis, The Clique Problem for Planar Graphs, *Inform. Proc. Lett.* 13 (1981), pp. 131-133.

[6] Richards, D., Finding Short Cycles in Planar Graphs Using Separators, *J. of Algorithms* 7 (1986), pp. 382-394.

[7] Richards, D., and A.L. Liestman, Finding Cycles of a Given Length, *Ann. Discrete Math.* 27 (1985), pp. 249-256.