LA-UR 96-1375

CONF-960679--3

TITLE: **APPROXIMATION ALGORITHMS FOR MAXIMUM TWO-DIMENSIONAL PATTERN MATCHING**

AUTHOR(S): S.R. Arikati, Anders Dessmark, Andrzej Lingas, Madhav Marathe

SUBMITTED TO: 7th Annual Symposium on Combinatorial Pattern Matching (CPM
Laguna Beach, CA
June 10-12, 1996

Approximation Algorithms for Maximum Two-dimensional Pattern Matching

SRINIVASA R. ARIKATI[1] *, ANDERS DESSMARK[2] **, ANDRZEJ LINGAS[2] ** and MADHAV MARATHE[3] ***

[1] Department of Mathematical Sciences, University of Memphis, Memphis, TN 38152. Email: arikatis@next1.msci.memphis.edu.

[2] Department of Computer Science, Lund University, Box 117, S-221 00 Lund, Sweden. Email {Anders.Dessmark, Andrzej.Lingas}@dna.lth.se

[3] Los Alamos National Laboratory P.O. Box 1663, MS K990, Los Alamos NM 87545. Email: madhav@c3.lanl.gov

Abstract. We introduce the following optimization version of the classical pattern matching problem (referred to as the *maximum pattern matching problem*). Given a two-dimensional rectangular text and a 2-dimensional rectangular pattern find the maximum number of non-overlapping occurrences of the the pattern in the text.

Unlike the classical 2-dimensional pattern matching problem, the maximum 2-dimensional pattern matching problem is NP-complete. We devise polynomial time approximation algorithms and approximation schemes for this problem. We also briefly discuss how the approximation algorithms can be extended to include a number of other variants of the problem.

1 Introduction

Given a *pattern string* PAT and a text T over a finite alphabet Σ, the classical pattern matching problem is to find all occurrences of PAT in T. In the recent years there has been growing interest in finding efficient algorithms for multi-dimensional pattern matching problems (see [2, 22, 10, 1, 5, 25] and the references therein.). Consider the following optimization variant of the classical pattern matching problem: Given a text T and a pattern PAT over a finite alphabet Σ, find the maximum number of *non-overlapping* occurrences of the pattern PAT in T. We call this problem the **maximum pattern matching** problem and use MPM_d to denote the maximum d-dimensional pattern matching problem. Maximum pattern matching problem arises naturally in the areas of automated digital image processing. For example, researchers at the

* Part of the work was done while the author was visiting Department of Computer Science, Lund University, Sweden; and Max-Planck-Institut fuer Informatik, Saarbruecken, Germany.

** The work is supported by TFR under Contract 93-159.

*** Research supported by the Department of Energy under Contract W-7405-ENG-36.

Los Alamos National Laboratory are currently developing CANDID, the Comparison Algorithm for Navigating Digital Image Databases, which facilitates a query-by-example approach to image retrieval [8, 23, 24]. A user poses queries such as, "Show me all or the the maximum number of non-overlapping images in the database that contain textures similar to those in this example image". Such queries are useful in a variety of settings such as analysis of the images sent by remote sensing satellites and medical diagnostics (See [8, 23, 24] and the references therein). For other applications of two-dimensional matching and a general survey, we refer the reader to [15, 22].

2 Summary of Results

Here, for the first time in the literature, we study the problem MPM_d and several of its variants. In the one dimensional case (i.e., the problem MPM_1), the maximum solution can be easily found by successively taking the leftmost non-overlapping (with those already selected) location, if all possible locations are precomputed. In the case of tree matching the intersection graph corresponding to the set of matching locations is chordal [16]. Therefore, the maximum number can be found in time linear in the size of the graph and the size of the text, by combining the results in [17] and [30]. For $d \geq 2$, MPM_d becomes harder to solve [9]. Specifically, we observe that a known NP-completeness result on planar geometric packing [14] implies the NP-completeness of the problem of maximum two-dimensional pattern matching (MPM_2). In Section 4, we give a simple and efficient approximation algorithm with performance guarantee of 2 for the problem MPM_2. If the set of the so-called periods of the pattern is appropriately restricted, our simple approach yields maximum solutions. In Section 5, we present our first involved approximation algorithm for MPM_2, based on good separation properties of the intersection graph of the pattern locations. Our proof of these properties might be of independent interest. The separator-based approximation algorithm yields a solution of relative error $O(1/\sqrt{\log\log n})$ for constant size patterns, and runs in $O(n \log n)$ time, on an input of size n. In Section 6, we present our second approximation algorithm for MPM_2 based on the *shifting strategy* introduced by Baker [4] and by Hochbaum and Maass [20, 21]. Specifically, when patterns are of fixed size, we obtain NC-approximation schemes for MPM_2. In the last Section we briefly describe various extensions of our results for MPM_2.

3 Preliminaries

Following [1], the *two-dimensional exact pattern matching* is defined as follows.
Input: A text matrix $T[1,\ldots,n][1,\ldots,n']$, and a pattern matrix $PAT[1,\ldots,m]$ $[1,\ldots,m']$ over a finite alphabet Σ.
Output: The set L of all location $[i,j]$ in T such that $T[i+k-1, j+l-1] = PAT[k,l]$, $1 \leq k \leq m$ and $1 \leq l \leq m'$.

For two-dimensional pattern matching, since there are known linear-time algorithms that find all possible locations of PAT in T [2, 3, 6], we assume that the set L of all such locations is known. Following standard convention, the size of a pattern PAT is the number of characters in it. Thus the size of a $m \times m'$ pattern is $O(mm')$ and the size of the $n \times n'$ text is $O(nn')$. Finally, we assume a RAM model of computation with uniform cost criterion.

We shall adhere to a standard notation for undirected graphs [19]. An *independent set* in a graph $G = (V, E)$ is a subset S of vertices such that no two vertices of S are adjacent in G. S is *maximal* if every vertex in $V - S$ is adjacent to some vertex in S. S is a *maximum independent set* if it has the maximum size among all independent sets of G. An *α-approximate independent set* is an independent set of size at least $(1/\alpha)$ times the maximum independent size. Also recall that an approximation algorithm for a maximization problem Π has a performance guarantee of ρ, if for every instance I of Π, the solution value returned by the approximation algorithm is at least $\frac{1}{\rho}$ of an optimal solution for I.

Let $a < 1$, $f : N \to N$, and $d > 0$. A class F of graphs has an $(a,\ f,\ d)$-*separator* if for each n-vertex $G \in F$ either $n \leq d$ or there is a a subset S of the set of vertices of G whose removal disconnects G into two subgraphs G_1 and G_2 in F such that:

1. Both G_1 and G_2 have at most an vertices each; and
2. S has at most $f(n)$ vertices.

We sometimes identify the notion of an (a, f, d)-separator with the separation subsets S. Consequently, we say that an (a, f, d)-separator is constructible in time t if such S are computable in time t.

Given T, PAT and the set L, we say that two locations $[i_1, j_1]$ and $[i_2, j_2]$ in L overlap, if and only if $|i_1 - i_2| < m$ and $|j_1 - j_2| < m'$. Let $G_L = (L, E_L)$ denote the intersection graph corresponding to L, i.e., for $l_1, l_2 \in L$, (l_1, l_2) is an edge of G_L if and only if the locations l_1 and l_2 of PAT overlap in T. The set L can also be thought of as defining a set of intersecting isothetic rectangles of size $m \times m'$ as follows. The isothetic equisized rectangles R are in one-to-one correspondence with the set of locations in L. A rectangle $r \in R$ corresponding to a location $(i, j) \in L$ is placed with its lower left lower corner at (i, j). It is clear that two rectangles in R intersect if and only if the corresponding locations overlap. Now, we can apply the well known methods for reporting intersections of isothetic rectangles in order to construct G_L. By Theorem 8.9 in [28], we have the following lemma.

Lemma 1. *G_L can be constructed from L in $O(|L| \log |L| + |E_L|)$ time.*

It can be easily verified that the problem MPM_2 reduces to finding a maximum independent set in G_L. Note that in general G_L corresponds to the intersection graph of equisized isothetic rectangles. Moreover, α-approximate independent sets in G_L are precisely α-approximate solutions to MPM_2.

The NP-hardness of MPM_2 immediately follows from the NP-hardness of the *planar geometric packing problem*, given in [14]. An instance of this problem consists of a set of m isothetic squares laid out in the plane. The question is to decide if it is possible to find k isothetic, pairwise disjoint locations of a given square (of integer side length) within an isothetic polygon with holes on an integer grid. To obtain the NP-hardness of MPM_2 we simply set PAT to the square filled with 0's, and model the input polygon P by setting the entries of T corresponding to the grid points inside P to 0 and the remaining entries to 1. Importantly, the area of the integer grid containing the instance of the packing problem, modeling an instance of $3SAT$ in [14], is polynomial in the size of the instance of $3SAT$. Thus MPM_2 is NP-hard. The graph representation G_L yields the membership of MPM_2 in NP.

Theorem 2. *The maximum two-dimensional pattern matching problem is NP-complete.*

4 Simple Approximations

Consider a maximum set S of non-overlapping locations of PAT in T. By a simple packing argument, it follows that any location in a maximal set of non-overlapping occurrences of PAT in T can overlap with at most four locations in S. Hence, the maximal set contains at least $|S|/4$ elements. The discussion also implies that the intersection graph G_L is 5-claw free graph. (A d-claw is the graph $K_{1,d}$, i.e., a star with d independent neighbors. A graph is a d-claw free graph if it has no induced d-claw.) For the maximum independent set problem for d-claw free graphs, Halldórsson [18] gives a local improvement heuristic with performance guarantee of $\frac{d}{2} + \epsilon$, for any $\epsilon > 0$. Since the intersection graphs associated with the problem MPM_2 are 5-claw free, the result in [18] can be used to obtain an algorithm for MPM_2 with asymptotic performance guarantee of 2. We can obtain an alternative heuristic which is more efficient and has a performance guarantee 2 by observing the following. An extreme location of PAT in T in one of the four directions can overlap with at most two other independent locations. Let ML be a maximal independent set in G_L constructed by repeatedly taking the vertex corresponding to the leftmost location of PAT and removing all its neighbors in the current graph. Then, we have the following lemma.

Lemma 3. *The maximal independent set ML yields a 2-approximate solution to MPM_2.*

Theorem 4. *A 2-approximate solution to MPM_2 can be computed in $O(|nn'| + |L|\log|L| + |E_L|)$ time.*

Proof. By Lemma 3, it suffices to construct the set ML within the stated time. To achieve this, we build the graph G_L and sort L by X-coordinate. The operation of extracting the leftmost location takes $O(1)$ time. The operation of deleting the overlapping location takes time proportional to the degree of corresponding vertex in G_L. Finally, recall that L can be constructed in $O(nn')$ time [9], and G_L in $O(|L|\log|L| + |E_L|)$ time by Lemma 1.

4.1 Periods of Pattern

A *period* of the $m \times m'$ pattern array PAT is a non-null vector (r, s) such that $-m < r < m$, $0 \leq s < m'$, and $PAT[i, j] = PAT[r + i, s + j]$ whenever both sides are within PAT. There are two classes of periods depending on whether r is negative or not.

If the pattern array has periods of only one class, a simple algorithm for optimally solving MPM_2 can be designed based on the following lemma.

Lemma 5. *If PAT has only nonnegative (negative) periods, no two locations corresponding to two vertices in the same connected component in G_L are such that one lies to the right and over (respectively, under) the other.*

Proof. The proof is by contradiction. Let u, v respectively denote the vertices of G_L corresponding to two locations contradicting the lemma, e.g., in the nonnegative case. Clearly, they cannot be neighbors in G. Consider the shortest path P in G_L connecting u and v. Let l be the first location corresponding to a vertex in P such that the locations l_1 and l_2 corresponding to the neighbors in P are both to the right or both to the left of l. Note that both l_1 and l_2 have to cover the left-upper or the right-lower corner of l. Hence, there is an edge connecting the two neighboring vertices in G. We obtain a contradiction to the optimality of P. The proof in the negative case is symmetrical.

By Lemma 5, we can order the vertices in each connected component of G_L according to their relative position in T, from the upper left or lower left corner depending on the class of periods. Now we can refine the 2-approximation algorithm given in the proof of Theorem 4 by giving preference to the vertex corresponding to the uppermost or the lowermost location respectively in a sweep from left to right. In result, we obtain the following theorem.

Theorem 6. *If PAT has only nonnegative periods (or, only negative periods), then MPM_2 can be solved in time $O(|E_L| + nn' + |L| \log |L|)$.*

It follows from Lemma 5 that G_L is a unit interval graph. Hence, G_L is in fact a chordal graph and a maximum independent set in G_L can be found in time linear in the size of G_L by [30] and [17]. This yields an alternative proof of Theorem 6.

5 Separator-based Approximation

In case PAT is of small size compared with T, e.g., constant size, we show below that an efficient approximation to MPM_2 exists and the approximation can be made arbitrarily close to the optimal solution. Our approach is inspired by the Lipton-Tarjan's method [26] of computing approximate independent sets in planar graphs. From the sophisticated randomized and deterministic methods for constructing separators for geometric graphs given in [27] and [11] respectively, it follows that G_L has a good separator. Independently of [11, 27], we show that an equally good separator for G_L is simply induced by $m-1$ consecutive columns and/or $m'-1$ consecutive rows in T. This very simple separator construction is the basis of our sophisticated approximation algorithm for MPM_2.

Lemma 7. *The class of graphs G_L has an $(5/6, O(\sqrt{mm'|L|}), O(mm'))$-separator constructible in $O(|L| + n/m + n'/m')$ time.*

Proof. It is sufficient to prove the following under the assumption of $|L| > (48)^2mm'$. In time $O(|L| + n/m + n'/m')$ one can find a sequence of $m-1$ consecutive columns or rows of T such that the locations of PAT in T with the left-upper corner in the sequence correspond to a subset of $O(\sqrt{mm'|L|})$ vertices of G_L disconnecting G_L into two subgraphs none of which has more than $5|L|/6$ vertices.

For convenience, we shall say that a vertex of G_L belongs to a subset S of entries of T if the left-upper corner of the location of PAT corresponding to this vertex is in S.

Group the n columns of T into supercolumns, each consisting of $m-1$ consecutive columns of T (possibly but for the last one). Similarly, group the n' rows of T into superrows, each consting of $m'-1$ consecutive rows of T (possibly but for the last one).

Note that the removal of all vertices of G_L belonging to a single supercolumn disconnects the two subgraphs of G_L induced by the vertices belonging respectively to the part of T to the left and to the part of T to the right of the supercolumn. A similar observation holds for the superrows.

Let C_l be the leftmost supercolumn such that there are at least totally $|L|/6$ vertices in C_l and to the left of C_l in T. Symmetrically, let C_r be the rightmost supercolumn such that there are at least totally $|L|/6$ vertices in C_r and to the right of C_r in T. Clearly, both C_l and C_r are well defined and C_l cannot lie to the right of C_r. Let BC be the block of consecutive supercolumns starting from C_l and ending with C_r.

If BC contains a supercolumn different from C_l and C_r with $\leq \sqrt{4mm'|L|}$ vertices we are done. Note that otherwise BC contains less than $\sqrt{|L|/(4mm')}+2$ supercolumns.

Similarly, we define the analogous block BR of superrows. Similarly, if BR contains a superrow with less than $\sqrt{4mm'|L|}$ vertices we are done, and otherwise BR contains less than $\sqrt{|L|/(4mm')}+2$ superrows.

To prove that BC or BR always contains a supercolumn (or superrow, respectively) with $\leq \sqrt{4mm'|L|}$ vertices, we argue as follows.

Let B be the intersection of BC with BR in T. Note that B has at least $|L|/3$ vertices. On the other hand, since B has both width $< (\sqrt{|L|/(4mm')}+2)m'$ and height $< (\sqrt{|L|/(4mm')}+2)m$, it cannot contain $|L|/3$ vertices if $|L| > (48)^2mm'$. We thus obtain a contradiction.

To find the number of vertices in each supercolumn in BC and each superrow in BR, we search the graph G_L. While visiting a vertex v, we identify the supercolumn and the superrow it belongs to, increasing the counters associated with them by one. It takes $O(|L|)$ time. To find the number of vertices to the left and to the right of each supercolumn (or, below or above each superrow, respectively), we apply prefix sums. It takes $O(n/m + n'/m')$ time.

For simplicity, we put $N = |L| + n/m + n'/m'$ and $d = (48)^2 mm'$.

Theorem 8. *For any $k > d$, G_L has a set of vertices C of size $O(|L|\sqrt{mm'}/\sqrt{k})$ whose removal from G_L leaves no connected component with more than k vertices. Furthermore C can be found in $O(N \log |L|)$ time.*

Proof. Initialize $C := \emptyset$, and construct C as follows.

```
while there is a connected component $H$ of $G - C$ with more than $k$
vertices do
find a separator $C'$ of $H$ and set $C := C \cup C'$.
```

The construction of C may be visualized by means of a tree, whose vertices represent subgraphs of G (the root represents G) that are encountered during the execution of the procedure; the leaves correspond to the components of G with at most k vertices. Define the *level* of a vertex v in the tree as the height of the full subtree rooted in v. Clearly, any two subgraphs on the same level are vertex-disjoint. By induction it follows that each i-th level ($i \geq 1$) subgraph has at least $(1/a)^{i-1}k$ vertices for some constant $a < 1$. Thus the number of i-th level subgraphs is at most $a^{i-1}|L|/k$. Since $k > 1$, the number of levels is $O(\log |L|)$. Further, we spend $O(N)$ time at each level, by Lemma 7. Hence the above procedure runs in $O(N \log |L|)$ time.

To bound the size of C, let $n_1, \dots, n_\ell$ be the sizes of the subgraphs at some level $i \geq 1$. The total number of vertices added to C by splitting these subgraphs is at most $O(\sum_{j=1}^{\ell} \sqrt{mm'n_j})$. This number is $O(a^{(i-1)/2}|L|\sqrt{mm'}/\sqrt{k})$, since $\ell \leq a^{i-1}|L|/k$ and $\sum_{j=1}^{\ell} n_j \leq |L|$. Hence $|C| = O(|L|\sqrt{mm'}/\sqrt{k})$.

Theorem 9. *In $O(\max\{N \log |L|, 2^k|L|\})$ time, we can find an independent set I in G_L such that the relative error in the size of I is $O((mm')^{3/2}/\sqrt{k})$.*

Proof. Apply Theorem 8 to G_L and find the set C. In each connected component of $G - C$, find a maximum independent set by an exhaustive search. Let I be the union of all such independent sets. Consider any maximum independent set I^* in G. Observe that $|I^*| = \Omega(|L|/(mm'))$, since every vertex in G_L has degree $O(mm')$. Notice that the restriction of I^* to any connected component cannot be larger than the restriction of I to the same component. Thus, the difference in the sizes of I and I^* is at most the size of C, which is $O(|L|\sqrt{mm'}/\sqrt{k})$. Consequently, the relative error in the size of I is $(|I^*| - |I|)/|I^*| = O((mm')^{3/2}/\sqrt{k})$.

To bound the time complexity, observe that the exhaustive search in each component takes $O(k \cdot 2^k)$ time. Thus the search over all components takes time $O(2^k|L|)$. Finally, by Theorem 8, C can be found in $O(N \log |L|)$ time.

Theorem 9 gives a trade-off between the running time of the algorithm and the quality of the solution. For small size and constant-size patterns, we have the following result by taking $k = \lfloor \log \log |L| \rfloor$.

Corollary 10. *If PAT is of size* $o((\log \log |L|)^{1/3})$, *then a solution to* MPM_2 *of relative error* $o(1)$ *can be constructed in* $O(N \log |L|)$ *time.*

Corollary 11. *If PAT is of constant size, then a solution to* MPM_2 *of relative error* $O(1/\sqrt{\log \log |L|})$ *can be constructed in* $O(N \log |L|)$ *time.*

6 An Approximation Scheme for MPM_2

6.1 Basic Technique

The *shifting strategy* was used by Baker [4] for obtaining polynomial time approximation schemes (PTAS) for problems restricted to planar graphs, by Hochbaum and Maass [20, 21] for devising PTAS for certain covering and packing problems in the plane, and by Feder and Greene [13] for obtaining a PTAS for a certain location problem.

We outline the basic technique by discussing our approximation scheme for MPM_2. Without loss of generality, we may assume that the intersection graph G_L of the set L of locations of PAT T is connected. As in the previous section, we divide T into supercoloumns composed of $m-1$ consecutive coloumns of T (except the last one). For an $\epsilon > 0$, we calculate the smallest integer k such that $(\frac{k}{k+1}) \geq 1 - \epsilon$. Next, for each i, $0 \leq i \leq k$, we disconnect G_L into l subgraphs $G_1, \cdots G_l$ by removing the vertices of G_L corresponding to the locations of L in supercoloums with number congruent to $i \mod (k+1)$. (A location is said to lie in a given subarray if its left-upper corner lies in that subarray). For each subgraph G_p, $1 \leq p \leq l$, we find an optimal independent set in G_p. The independent set for this partition is just the union of independent sets for each G_p. By an argument similar to the shifting lemma in [20], it follows that the iteration in which the partition yields the largest solution value contains at least $(\frac{k}{k+1}) \cdot OPT(G_L)$ vertices, where $OPT(G_L)$ denotes the size of a maximum independent set in G_L. (For simplicity, we also denote the cardinality of a maximum independent set in G_L by $OPT(G_L)$.) The algorithm takes $O(n)$ work. It is easy to see that the algorithm admits an NC implementation. We are now ready to give our approximation scheme for MPM_2. The algorithm is outlined in Figure 1.

6.2 Finding an optimal solution in Step 3(c)

We now discuss how to obtain an optimal solution for the independent set problem in Step 3(c) of the algorithm MAX-IS. For each fixed $k > 0$, the subgraph

```
Algorithm: MAX-IS
Input: A pattern $PAT$ of size $m \times m'$, a text array $T$ of size $n \times n'$ and the intersection
graph $G_L$ of the locations of $PAT$ in $T$.
Output: An independent set in $G_L$ with at least $(\frac{k}{k+1}) \cdot OPT(G_L)$ vertices.

 1. Find the smallest integer $k$ such that $(\frac{k}{k+1}) \geq 1 - \epsilon$.
 2. Divide $T$ into supercolumns of width $m - 1$.
 3. For each $i$, $0 \leq i \leq k$ do
    (a) Disconnect $G_L$ into $r_i$ disjoint subgraphs $G_{i,1} \cdots G_{i,r_i}$ by removing all the
        vertices corresponding to locations of $PAT$ in supercolumns with numbers
        congruent to $i$ mod $(k + 1)$;
    (b) $G_i \leftarrow \bigcup_{1 \leq j \leq r_i} G_{i,j}$;
    (c) Compute an optimal independent set $IS(G_{i,j})$ in $G_{i,j}$.
    (d) $IS(G_i) \leftarrow \bigcup_{1 \leq j \leq r_i} IS(G_{i,j})$
 4. $IS(G_L) \leftarrow \max_{0 \leq i \leq k} IS(G_i)$
```

Algorithm 1: Details of the approximation scheme for the maximum independent set problem for equisized-rectangle intersection graphs.

$G_{i,j}$ obtained in Step 3(b) has treewidth $\leq ck$, for some constant $c > 0$. Given this we can use the sequential (or NC-algorithms) for computing the maximum independent set in treewidth bounded graphs [7, 29]. Thus the optimal independent set in Step 3(c) can be found by using $O(|L_{i,j}|)$ work. Here $L_{i,j}$ denotes the vertex set of the graph $G_{i,j}$.

6.3 Performance Guarantee

We next prove that the algorithm given above indeed computes a near optimal independent set. That is, given any $\epsilon > 0$ the algorithm will compute an independent set whose size is at least $(1 - \epsilon)$ times that of an optimal independent set.

We first prove that of all the different iterations for i, at least one iteration has the property that the number of vertices that are not considered in the independent set computation is a small fraction of an optimal independent set.

Recall that for each i we did not consider the vertices in the subgraphs $G_{j_1}, G_{j_2} \cdots G_{j_{p_i}}$ such that $j_l = i \; mod(k + 1)$, $1 \leq l \leq p_i$. For each i, $0 \leq i \leq k$, let S_i be the set of vertices of G_L which were not considered in the i-th iteration. Let $IS_{opt}(S_i)$ denote the vertices in the set S_i which were chosen in the maximum independent set $OPT(G_L)$.

Lemma 12.

$$\max_{0 \leq i \leq k} |OPT(G_i)| \geq \frac{k}{(k+1)} |OPT(G_L)|$$

Proof. First observe that the following equation holds:

$$0 \le i, j \le k,\ i \ne j,\ S_i \cap S_j = \phi;$$

since different subgraphs are considered in different iteration. It now follows that

$$|IS_{opt}(S_0)| + |IS_{opt}(S_1)| + \cdots + |IS_{opt}(S_k)| = |OPT(G_L)|.$$

Therefore,

$$\min_{0\le t\le k} |IS_{opt}(S_t)| \le |OPT(G_L)/(k+1)|,$$

$$\max_{0\le i\le k} |OPT(G_i)| \ge |OPT(G_L)| - \min_{0\le t\le k} |IS_{opt}(S_t)| \ge \frac{k}{(k+1)} |OPT(G_L)|.$$

Theorem 13. $|IS(G_L)| \ge (\frac{k}{k+1}) \cdot |OPT(G_L)|$.

Proof. We consider the iteration when the value of i is such that $|OPT(G_i)| \ge (\frac{k}{k+1})|OPT(G_L)|$. By Lemma 12 such an i exists. Fix the iteration i.

$$|OPT(G_i)| = \textstyle\sum_{j=1}^{j=r} |OPT(G_{i,j})|$$

Using the above equations we get that

$$\begin{aligned}
|IS(G_L)| = & \quad \max\nolimits_{0\le i\le k} |IS(G_i)| \\
= & \quad \max\nolimits_{0\le i\le k} \textstyle\sum_{j=1}^{j=r} |IS(G_{i,j})| \quad \text{(By Step 3(b))} \\
\ge & \quad \max\nolimits_{0\le i\le k} \textstyle\sum_{j=1}^{j=r} |OPT(G_{i,j})| \quad \text{(By Step 3(c))} \\
\ge & \quad \max\nolimits_{0\le i\le k} |OPT(G_i)| \quad \text{(By Step 3(c))} \\
\ge & \quad (\tfrac{k}{k+1}) \cdot |OPT(G_L)| \quad \text{(By Lemma 12)}
\end{aligned}$$

The time required for each iteration of the For loop is $\Sigma_{j=1}^{j=r_i} O(|L_{i,j}|) = O(|L|)$. Hence the total running time of our algorithm is $\Sigma_{i=0}^{i=k} O(|L|) + O(n/m) = O(|L|) + O(n/m)$ (in case of the NC-algorithm the total amount of work is $O(|L|) + O(n/m)$.) Moreover, the algorithm has a performance guarantee of $(k+1)/k$.

7 Extensions

We briefly outline the possible extensions of our ideas presented in the previous sections.

Higher Dimensional Matching Problems Our approximation algorithms for MPM_2 directly extend to solve the problems MPM_d for any fixed $d > 2$. This can be seen by oberserving the following. For each fixed $d > 0$ there is an $r > 0$ such that, the intersection graph associated with the problem MPM_d is r-claw free. Also note that the d-dimensional geometric graphs have also good separator properties [27, 11]. Finally, note that the shifting strategy can be easily extended to apply to d-dimensional rectangles. The performance guarantee of the algorithm based on shifting strategy for solving MPM_d is $(\frac{k+1}{k})^{d-1}$.

Multiple Matchings Idury and Schäffer [22] consider a variant of the classical matching problem in which we are given a set of patterns instead of single pattern. Our results extend to handle the optimization version of the multiple pattern matching problem studied in [22]. In particular, we obtain two types of results depending on the size and the number of patterns. If the number of patterns and the size of each pattern is fixed, our approximation schemes can be extended to obtain approximation schemes. To see this, note that although the rectangle graph induced now does not have equisized rectangles, we can subdivide the plane with respect to the largest rectangle. Furthermore, since the rectangles are of fixed size, for each $\epsilon > 0$, the treewidth of the subgraphs obtained as a result of decomposition is still a constant. With these two observations in mind the extension is fairly straight forward. In the second case, when the shapes and the sizes of the patterns are not fixed, we can obtain a 4-approximation by modifying the algorithm in Section 4 to choose the smallest rectangle instead of choosing the leftmost rectangle. Since the graph induced by the smallest rectangle and its neighbors is $K_{1,4}$ free, the performance guarantee follows immediately by an inductive argument.

Non-Rectangular shapes As pointed out in Amir and Farach [1], several practical applications require us to match *non-rectangular* shapes. Using ideas similar to those outlined for the Multiple matching case, the approximation schemes for MPM_2 can also be extended to the case when we have fixed sized patterns that are non-rectangular, e.g., an L-shaped patterns.

Allowing Mismatches Amir and Farach [1] also study the two dimensional pattern matching problem in which we are allowed certain number of mismatches. Our approximation algorithms extend to finding a maximum number of non-overlapping patterns such that no more than k mismatches are allowed per matched location.

References

1. A. Amir, and M. Farach, "Efficient 2-dimensional Approximate Matching of Non-Rectangular Figures," *Proc. 2nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA),* 1992, pp. 212-223.
2. A. Amir, G. Benson, G. Benson and M. Farach. An Alphabet-independent Approach to Two-Dimensional Matching. *Proc. 24th ACM Symposium on Theory of Computing,* 1992, pp. 59-68. Journal version to appear in *SIAM J. Computing.*
3. T. P. Baker. A technique for extending rapid exact-match string matching to arrays of more than one dimension. *SIAM J. Computing,* No. 7, 1978, pp. 533-541.
4. B.S. Baker. "Approximation Algorithms for NP-complete Problems on Planar Graphs," *24th IEEE Symposium on Foundations of Computer Science (FOCS),* 1983, pp 265-273. (Journal version in *J. ACM,* Vol. 41, No. 1, 1994, pp. 153-180.)
5. B. Baker, "A Theory of Parameterized Pattern Matching: Algorithms and Applications," *Proc. 25th ACM Symposium on Theory of Computing,* 1993, pp. 71-80. Journal version to appear in *Journal of Computer and System Sciences (JCSS).*
6. R. S. Bird. "Two-dimensional pattern matching," *Information Processing Letters* No. 6, 1977, pp. 168-170.
7. H. L. Bodlaender, "Dynamic programming on graphs of bounded treewidth," *Proc. 15th International Colloquium on Automata Languages and Programming (ICALP),* LNCS Vol. 317, 1988, pp. 105-118.
8. "CANDID Project," Los Alamos National Laboratory, 1993.
9. M. Crochemore and W. Rytter. *Text Algorithms,* Oxford University Press, New York, 1994.
10. A. Czumaj, Z. Galil, L. Gasieniec, K. Park and W. Plandowski, "Work-Time-Optimal Parallel Algorithms for String Problems," *27th ACM Symposium on Theory Of Computing (STOC),* pp. 713-722, 1995.
11. D. Eppstein, G.L. Miller, S.H. Teng. "A Deterministic Linear Time Algorithm for Geometric Separators and its Application," *9th ACM Symposium on Computational Geometry,* pp 99-108, 1993.
12. M. Farach and M. Thorup, "String Matching in Lempel-Ziv Compressed Strings," *27th ACM Symposium on Theory Of Computing (STOC),* pp. 703-712, 1995.
13. T. Feder and D. Greene. "Optimal Algorithms for Approximate Clustering", *20th ACM Symposium on Theory Of Computing (STOC),* pp. 434-444, 1988.
14. R.J. Fowler, M.S. Paterson and S.L. Tanimoto. "Optimal Packing and Covering in the Plane are NP-Complete," *Information Processing Letters,* Vol 12, No.3, June 1981, pp. 133-137.
15. G.H. Gonnet and R. Baeza Yates, *Handbook of Algorithms and Data Structures,* Addison-Wesley, Reading, MA, 1991.
16. F. Gavril, "The intersection graphs of subtrees in trees are exactly the chordal graphs," *J. Combin. Theory B,* 16, 1974, pp. 47-56.
17. F. Gavril, "Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph," *SIAM J. Computing,* 1, 1972, pp. 180-187.
18. M.M. Halldórsson, "Approximating Discrete Collections via Local Improvement," *Proc. 6th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA),* 1995, pp. 160-169

19. F. Harary, *Graph Theory,* Addison-Wesley, Reading, Massachusetts, 1979.
20. D.S. Hochbaum and W. Maass, "Approximation Schemes for Covering and Packing Problems in Image Processing and VLSI," *J. ACM,* Vol 32,No. 1, 1985, pp 130-136.
21. D.S. Hochbaum and W. Maass, "Fast Approximation Algorithms for a Nonconvex Covering Problem," *Journal of Algorithms,* Vol. 8, 1987, pp. 305-323.
22. R. Idury and A. Schäffer, "Multiple Matching of Rectangular Figures," *Proc. 25th ACM Symposium on Theory of Computing,* 1993, pp. 81-89.
23. P.M. Kelly, T.M. Cannon, and D.R. Hush, "Query by image example: the CANDID approach," *SPIE Vol. 2420 Storage and Retrieval for Image and Video Databases III,* pages 238-248, 1995.
24. P.M. Kelly and T.M. Cannon, "CANDID: Comparison Algorithm for Navigating Digital Image Databases," *In Proc. of the 7th International Working Conference on Scientific and Statistical Database Management, pages 252-258. Charlottesville, VA,* Sept., 1994.
25. S. R. Kosaraju, "Faster Algorithms for the Construction of Parameterized Suffix Trees," *36th IEEE Symposium on Foundations of Computer Science (FOCS),* 1995, pp 631-637.
26. R. J. Lipton and R. E. Tarjan, "Applications of a planar separator theorem," *SIAM J. Computing,* 9, 1980, pp. 615-627.
27. G.L. Miller, S.-H. Teng and S.A. Vavasis, "A Unified Geometric Approach to Graph Separators," *Proc. of the 32nd IEEE Symposium on Foundations of Computer Science,* 1991, pp. 538-547.
28. F. P. Preparata and M. I. Shamos, *Computational Geometry,* Springer Verlag, New York, 1985.
29. N. Robertson and P. Seymour, "Graph Minors II. Algorithmic aspects of tree-width," *J. Algorithms,* No. 7 (1986), 309-322.
30. D. J. Rose, R. E. Tarjan and G. S. Lueker, "Algorithmic aspects of vertex elimination on graphs," *SIAM J. Computing,* 5, 1976, pp. 266-283.