A PTAS for the Square Tiling Problem

Amihood Amir[1,⋆], Alberto Apostolico[2,⋆⋆], Gad M. Landau[3,⋆⋆⋆], and Oren Sar Shalom[4]

[1] Bar-Ilan University, Israel and Johns Hopkins University, USA
amir@cs.biu.ac.il
[2] Georgia Tech, USA and University of Padova, Italy
axa@cc.gatech.edu
[3] University of Haifa, Israel, and NYU-Poly, USA
landau@cs.haifa.ac.il
[4] Bar-Ilan University, Israel
oren.sarshalom@gmail.com

Abstract. The Square Tiling Problem was recently introduced as equivalent to the problem of reconstructing an image from patches and a possible general-purpose indexing tool. Unfortunately, the Square Tiling Problem was shown to be $\mathcal{NP}$-hard. A 1/2-approximation is known.

We show that if the tile alphabet is fixed and finite, there is a Polynomial Time Approximation Scheme (PTAS) for the Square Tiling Problem with approximation ratio of $(1 - \frac{\epsilon}{2 \log n})$ for any given $\epsilon \leq 1$.

1 Motivation

Recently [2] Amir and Parienty introduced the *Patches Model* as a possible abstraction of ad-hoc techniques that have been used to index various domains. The idea is to slice the images that we want to index into many small overlapping *patches*, or *tiles*. Patterns whose patches match a large number of patches in an indexed object, are likely to appear in the image. Similar methods have been used in Computational Biology (e.g. [15,7,3,12,6,5]), Linguistics (e.g. [8,1]), Image Processing (e.g. [14,11,4]), or Audio Indexing (e.g. [9]).

The first task tackled in [2] was reconstructing an image from its patches – the *Square Tiling Problem.* Unfortunately, it was proven that this problem is $\mathcal{NP}$-hard. An image constructed from $n \times n$ tiles has $2n^2 - 2n$ "*seams*" between tiles. If it is correctly constructed then two adjacent tiles *match* at the seams, i.e., have equal alphabet symbols on their adjacent edges. We then say that the seam is *correct*, otherwise there is an *error* at the seam. We count a single error

⋆ Partly supported by NSF grant CCR-09-04581, ISF grant 347/09, and BSF grant 2008217.

⋆⋆ Partly supported by BSF grant 2008217.

⋆⋆⋆ Partly supported by the National Science Foundation Award 0904246, Israel Science Foundation grants 35/05 and 347/09, the Israel-Korea Scientific Research Cooperation, Yahoo, Grant No. 2008217 from the United States-Israel Binational Science Foundation (BSF) and DFG.

at the seam whether one or two symbols are not equal at the adjacent tiles. An *approximation* in that context is an $n \times n$ square where "many" seams match. In [2] a polynomial-time algorithm that constructs a square with at least $\frac{1}{2}$ of the seams being correct, was shown. If the idea of patch indexing is to have any chance of applicability, much better approximations are necessary.

In this paper we present a Polynomial Time Approximation Scheme (PTAS) for the square tiling problem with a fixed finite alphabet. We show that for such finite alphabets, for any given ϵ and n^2 tiles, there is an algorithm polynomial in n and ϵ that constructs a square with at least $(1 - \frac{\epsilon}{2\log n})n^2$ correct seams.

The paper is constructed as follows. We begin with background and definitions in Section 2. In Section 3 we show how to reconstruct a rectangle of size $n \times \log n$ tiles in polynomial time (or, for a given c an $n \times c$ rectangle). We then show in Section 4 how multidimensional knapsack techniques can be used to approximate a square tiling.

2 Background and Definitions

The definition below was used in [2] to combinatorially describe the patches concept.

Definition 1. *Given matrix M,* $\begin{pmatrix} m_{0,0} \cdots\cdots\cdots m_{0,n} \\ \cdots\cdots\cdots\cdots \\ \cdots\cdots\cdots\cdots \\ m_{n,0} \cdots\cdots\cdots m_{n,n} \end{pmatrix}$
A is a division of M to patches *if* $A = \{a_{0,0}, \cdots a_{0,n-1}, \cdots\cdots a_{n-1,0}, \cdots a_{n-1,n-1}\}$ *and*
$\forall i,j\ a_{i,j} = \begin{bmatrix} m_{i,j}, m_{i,j+1} \\ m_{i+1,j}, m_{i+1,j+1} \end{bmatrix}$.

The problem we are concerned with is the converse.

Definition 2. *The problem of* constructing an image from patches *is defined as follows:*

INPUT: $A = \{a_0, \cdots, a_{n^2-1}\}$ *be a set of* 2×2 *matrices over alphabet* Σ.

OUTPUT: Construct an $(n+1) \times (n+1)$ *matrix* $M = \begin{pmatrix} m_{0,0} \cdots\cdots\cdots m_{0,n} \\ \cdots\cdots\cdots\cdots \\ \cdots\cdots\cdots\cdots \\ m_{n,0} \cdots\cdots\cdots m_{n,n} \end{pmatrix}$
such that A is the division of M to patches, if such a matrix exists. Otherwise report that no matrix can be constructed from the input.

In [2] it was proven that the problem of constructing an image from overlapping patches is equivalent to the square tiling problem, where we are given 2×2 patches over alphabet Σ and we are only allowed to place patches next to each other if their symbols match. Formally:

Definition 3. *A patch A may be* correctly placed *to the right (left, top, bottom) of patch B if the pair of letters on the right (left, top, bottom) side of B are the same as the pair on the left (right, bottom, top) of patch A. The common edge of two patches is called a* seam. *The seam is* correct *if the adjacent tiles are correctly placed. Otherwise, it is an* error.

Definition 4. *The* Square Tiling Problem *is defined as follows:*

INPUT: A multiset S of n^2 tiles. Each tile is a 2×2 matrix over alphabet Σ
DECIDE: Whether there exists a square of tiles correctly placed next to each other, whose tiles are exactly those in the multiset.

In [2] it was proven that the Square Tiling Problem is $\mathcal{NP}$-hard.

We seek a polynomial time algorithm that approximates the square. The approximation means reconstructing an image from all of the tiles, with the minimum amount of seam errors.

Our goal is to improve this result, and in fact we even generalize that. Assume that the maximum number of correct seams that can be achieved by *any* tiling of a given set of patches S is s_{max}. Then for any given ϵ we can tile, in time polynomial in the size of S and ϵ, the patches of S achieving at least $(1 - \frac{\epsilon}{\log n})s_{max}$ correct seams.

3 Rectangle Tiling

We begin by showing that the $\mathcal{NP}$-hardness is dependent on the dimensions of the square. For some rectangle dimensions, the Tiling Problem is polynomial-time computable.

Definition 5. *The* entropy *of multiset m of size $a \cdot b$ is the minimum number of errors obtained by tiling m as an $a \times b$ rectangle.*

Theorem 1. *The entropy of all multisets of size $n \times \frac{\log n}{\epsilon}$ can be computed in time polynomial in ϵ and n.*

Proof: We want to compute the entropy for **all** multisets, so we must first make sure that there are not too many multisets.

Claim. There is a polynomial number of multisets of size $n \times \frac{\log n}{\epsilon}$.

Proof: The alphabet is fixed and finite. Let us assume that $\Sigma = \{1, ..., c\}$, So there are $O(c^4)$ types of tiles.

A multiset is a histogram of the types of tiles that compose it. There are at most $n \cdot \frac{\log n}{\epsilon}$ tiles of each type, while the sum of all types is also $n \cdot \frac{\log n}{\epsilon}$. Combinatorially, the number of different multisets it is equal to the number of combinations to insert n' balls into k' bins, when $n' = n \cdot \frac{\log n}{\epsilon}$ and $k' = c^4$. There are $\binom{n'+k'-1}{n'} = O(n'^{k'-1}) = O((n \cdot \frac{\log n}{\epsilon})^{c^4-1})$ multisets. □

We now show an exact polynomial-time algorithm for a fixed finite alphabet that calculates the entropy for each multiset m of size $n \cdot \frac{\log n}{\epsilon}$. we will need an auxiliary data structure to keep track of some values for the multisets constructed during the execution of the algorithm. In particular, consider a possible tiling of multiset of size $i \cdot \frac{\log n}{\epsilon}$ into a rectangle of i columns and $\frac{\log n}{\epsilon}$ rows. The rightmost column, rc, of such a tiling is composed of $\frac{\log n}{\epsilon}$ 2×2 patches. For the sake of improving the time complexity, we consider the column composed only of the $\frac{2 \log n}{\epsilon}$ symbols on the *right side* of the patches in the rightmost column. It is possible that different columns of $\frac{\log n}{\epsilon}$ patches have the same right side. For any multiset m and right side r, choose a tiling with the lowest entropy. For that tiling, record in addition to the right side r, also the $\frac{2 \log n}{\epsilon}$ symbols on the *left side* of the patches in the rightmost column.

Auxiliary Data
Define a 2-dimensional array M with the following values:

At iteration i, for each multiset m of size $i \cdot \frac{\log n}{\epsilon}$, that represents a rectangle of i columns and $\frac{\log n}{\epsilon}$ rows, and for each r – the right side of the rightmost column of multiset m:

– $M[m, r].entropy$ holds the minimum entropy of m subject to the constraint that r is the right side column of the $\frac{\log n}{\epsilon} \times i$ rectangle tiling of m.
– $M[m, r].left$ is the left side of the rightmost column in a tiling that obtained the minimum entropy.
  This field's goal is to enable us to reconstruct the optimal tiling of m.

The size of array M is bounded by the product of the number of multisets and the number of possibilities for the right side of column r. The number of multisets for the last iteration was calculated above as $O((n^{\frac{\log n}{\epsilon}})^{c^4-1})$.

Calculation of the number of different possibilities for the right side of a column:

There are c^2 possibilities for each tile, and $\frac{\log n}{\epsilon}$ tiles altogether. Therefore, the right side of a column can be arranged in

$(c^2)^{\frac{\log n}{\epsilon}} = (2^{\log c^2})^{\frac{\log n}{\epsilon}} = (2^{\log n})^{\frac{log c^2}{\epsilon}} = n^{\frac{\log c^2}{\epsilon}} = n^{\frac{2 \log c}{\epsilon}} = z$ different ways.

Thus the size is clearly polynomial.

Algorithm Outline
Initialization phase: initialize all of the *entropy* values of M to ∞.

A column can be considered a Cartesian product of two sides $< l, r >$, so there are z^2 different columns.

Count the number of errors of every possible column c, i.e, an ordered $\frac{\log n}{\epsilon}$ tiles placed one above the other. In this case, m is the multiset comprised by c and r is the right side of column c.

Assume we have computed array M for the first i iterations.

Iteration step: For each combination of (m, r) computed in the previous iteration, the algorithm attaches to r all possibilities as column c. Let C be the multiset of the elements of c. Then each such column c creates a new multiset $m \leftarrow m \cup C$, and new right side $r \leftarrow$ right side of column$c$. The new entropy e' the entropy of the old entry + the number of errors introduced by attaching c to the right of the old entry.

Thus, $M[m, r] \leftarrow \min(M[m, r], e')$

Correctness:
We prove by induction that for each combination of multiset m of size $i \cdot \frac{\log n}{\epsilon}$ and right side of column r, the algorithm finds the entropy of (m, r).

Base Case: For $i = 1$, the assumption holds, because the algorithm goes through all columns and picks the minimum for each combination.

Inductive Step: Assume correctness for i and prove for $i + 1$: Let m_{i+1} be a multiset of size $(i + 1) \cdot \frac{\log n}{\epsilon}$ and r_{i+1} a right side of a column. Any optimal tiling of m_{i+1} as a rectangle such that r_{i+1} is its right side of the rightmost column, is a selection among an optimal tiling of multiset m_i as a rectangle of size $i \times \frac{\log n}{\epsilon}$ such that the right side of the rightmost column is r_i adjacent to column $< l_{i+1}, r_{i+1} >$, such that the number of errors of m_i and $< l_{i+1}, r_{i+1} >$ is smallest. By the induction assumption, the algorithm finds m_i as iteration i. Iteration $i + 1$ attaches all possible columns, particularly $< l_{i+1}, r_{i+1} >$.

Therefore the algorithm finds (m_{i+1}, r_{i+1}). □

Claim. The algorithm's complexity is polynomial.

Proof:
Initialization phase: Considering all combinations of right side of column r and column $< l', r' >$ is $O(z^3) = O(n^{\frac{6 \log c}{\epsilon}})$.

Column construction: We seek after the number of multisets of size $i \cdot \frac{\log n}{\epsilon}$ $\forall i \leq n$. There are $O((n \cdot \frac{\log n}{\epsilon})^{c^4 - 1})$ multisets of size exactly $n \cdot \frac{\log n}{\epsilon}$. Therefore, there are $O(n \cdot (n \cdot \frac{\log n}{\epsilon})^{c^4 - 1})$ multisets altogether. For each multiset we go through all m^2 columns and perform a constant time work.
Therefore, the algorithm's running time is $O(n \cdot (n \cdot \frac{\log n}{\epsilon})^{c^4 - 1} \cdot n^{\frac{4 \log c}{\epsilon}})$.

Finding the optimal tiling: Once a multiset $m =< t_1, ..., t_{c^4} >$ of size $n \cdot \frac{logn}{\epsilon}$, is constructed, it's trivial to reverse the algorithm and tile it as a rectangle with the minimum number of errors. □

4 The Approximation

In the previous section we enumerated all multisets and their entropies. That action could be referred as a pre-processing action, because it does not use the input of the problem, but only exploits the problem size n.

We now partition the n^2 input patches into $\frac{n \cdot \epsilon}{\log n}$ sets of $n \cdot \frac{\log n}{\epsilon}$ tiles, such that the sum of their entropies is minimal. This will also be done by partitioning *all* multisets of size n^2 into such sets.

Algorithm Outline:

Step 1 - Find the entropy for all multisets of size $n \cdot \frac{\log n}{\epsilon}$.

Step 2 - Select a set of multisets, consisting exactly of the input tiles, with minimal entropy.

Implementation:
Step 1: The implementation of step 1 was shown in Section 3.

Step 2: This problem is similar to another $\mathcal{NP}$-hard problem - The *Multidimensional Knapsack problem.* It needs to select a multiset of given objects (or items) in such a way that the total profit of the selected objects is maximized while a set of knapsack constraints are satisfied.

Unfortunately, MDK is $\mathcal{NP}$-hard. It was also shown that finding an FPTAS even for a special case where all profits are the same and equal to 1 and $m = 2$ is $\mathcal{NP}$-hard [10]. Moreover, the problem is $\mathcal{NP}$-hard in the strong sense and thus any dynamic programming approach would result in strictly exponential time bounds [13].

We want an exact algorithm to be polynomial on the one hand, and on the other hand to support multiple knapsacks, utilizing the unique characteristics of our problem.

Let $MS = \{c_1, \ldots, c_x\}$, be the set of all multisets of size $n \cdot \frac{\log n}{\epsilon}$, where x is of size $O((n \cdot \frac{\log n}{\epsilon})^{c^4-1})$. Multiset c_i can be represented by $< w_{i1}, \ldots, w_{ic^4} >$, where w_{ij} is the number of patches of type j there are in multiset c_i. Let e_i be the entropy of multiset c_i. We assume that MS is sorted in non-decreasing order of its entropy, i.e., $\forall i \le m-1$, $e_i \le e_{i+1}$.

The input S of the tiling problem is a set of patches S of size n^2. S can be represented by the tuple $< S_1, \ldots, S_{c^4} >$, where $_i$ is the number of input tiles of type i. Our task is to find a multiset of multisets $C \subseteq MS$, such that $\bigcup\limits_{c \in C} c = S$, i.e. $\forall j$, $1 \le j \le c^4$, $\sum o_i \cdot w_{ij} = S_j$, where o_i is the number of occurrences of c_i in C.

The Dynamic Programming Matrix:
$T[1..x \ ; \ 1..(n^2)^{c^4}]$ is a matrix with the following values:

Let $< b_1, \ldots, b_{c^4} >$ be a multiset of size between 0 and n^2, and let i be a number $1 \le i \le x$.

If there does not exist a multiset which is the union of sets from $\{c_1, \ldots, c_i\}$, and whose elements are exactly the patches $< b_1, \ldots, b_{c^4} >$, then $T(i, < b_1, \ldots, b_{c^4} >) = \infty$.
Otherwise, let L be such a multiset where $\sum\limits_{c \in L}$(entropy of c) $= E$ is smallest.
Set $T(i, < b_1, \ldots, b_{c^4} >) = E$.

Algorithm Outline:
The matrix is filled using dynamic programming:

Initialization: The first row can use only c_1, so $\forall \alpha \in \mathbb{N}_0$ every cell that represents $\alpha \cdot c_1$ will have the entropy $\alpha \cdot e_1$. The rest of the cells are infeasible and their entropies are ∞.

Filling the Matrix: For each column $(b_1, \ldots, b_{c^4})$ and row i there are two possibilities:

1. The optimal solution does not use c_i at all in order to achieve $(b_1, \ldots, b_{c^4})$. In that case $T(i, < b_1, \ldots, b_{c^4} >) = T(i-1, < b_1, \ldots, b_{c^4} >)$
2. The optimal solution uses c_i at least one time. In that case $T(i, < b_1, \ldots, b_{c^4} >) = T(i, < b_1, \ldots, b_{c^4} > \setminus c_i) + e_i$

Therefore:

1. $T(i, < b_1, \ldots, b_{c^4} >) \leftarrow \min(T(i, < b_1, \ldots, b_{c^4} >), T(i-1, < b_1, \ldots, b_{c^4} >))$
2. $T(i, < b_1, \ldots, b_{c^4} > \cup\ \alpha \cdot c_i) \leftarrow \min(T(i, < b_1, \ldots, b_{c^4} > \cup c_i), T(i, < b_1, \ldots, b_{c^4} >) + e_i)$

Theorem 2. *The dynamic programming algorithm's running time is* $O((n \cdot \frac{\log n}{\epsilon})^{c^4-1} \cdot (n^2)^{c^4} / (n \cdot \frac{\log n}{\epsilon}))$

Proof: There are $m \cdot (n^2)^{c^4}$ cells in matrix M. The algorithm only handles columns representing multisets of size divisible by $n\frac{\log n}{\epsilon}$. The time to fill each cell is constant. Therefore, the total complexity is $O((n \cdot \frac{\log n}{\epsilon})^{c^4-1} \cdot (n^2)^{c^4} / (n \cdot \frac{\log n}{\epsilon}))$ $\square$

Theorem 3. *Let S be a set and let s be the number of correct seams in the dynamic programming construction of S. Let s_{max} be the maximum number of correct seams for any square tiling of S. Then $s \geq (1 - \frac{\epsilon}{2 \log n}) s_{max}$.*

Proof: Let M be an optimal tiling. Let X be the number of correct vertical seams, and Y the number of correct horizontal seams in M. $s_{max} = X + Y$. (If M has no errors, then $s_{max} = 2n^2 - 2n$).

Without loss of generality we may assume that $X \geq Y$ (otherwise, we rotate all tiles by 90^o). Since the dynamic programming algorithm provides the optimum tiling within the strips of size $\frac{\log n}{\epsilon} \times n$, and the only errors may occur when "putting together" these strips, then it is clear that the number of correct vertical seams decided by our algorithm is no less than the number of vertical seams in the optimum tiling. We need to consider only Y – the number of correct horizontal seams, since our algorithm makes no effort to match the rows between the strips.

We start by identifying the total number of correct horizontal seams at the bottom of the $\frac{\log n}{\epsilon} \times n$ strips of M. Call that number Y_0. Next we consider the strips as moved down by an offset of 1, i.e., assume the first strip is of only one row, and the following strips are of size $\frac{\log n}{\epsilon} \times n$. The total number of correct horizontal seams at the bottom of the $\frac{\log n}{\epsilon} \times n$ strips (with offset 1) we call Y_1. In general, let Y_i the total number of correct horizontal seams at the bottom of the $\frac{\log n}{\epsilon} \times n$ strips with offset i (i.e, the first strip has only i rows, followed by $\frac{\log n}{\epsilon} \times n$ strips.) Formally, $\forall\ \ 0 \leq i \leq \frac{\log n}{\epsilon} - 1$, define $Y_i = \sum_{j=1}^{\frac{n \cdot \epsilon}{\log n}}$(number of matches between row $j \cdot \frac{\log n}{\epsilon} + i$ and the row below it).

Let Y_{min} be such that $Y_{min} \leq Y_i \forall i,\ 0 \leq i \leq \frac{\log n}{\epsilon} - 1$. We will assume that the worst happened, and all the horizontal seams at the bottom of the $\frac{\log n}{\epsilon} \times n$ strips of the dynamic programming tiling of S are erroneous. However, this number of errors can not exceed Y_i, since the dynamic programming vertical tiling within the strips is *superior* to the optimal tiling. Therefore $s \geq X + Y - Y_{min}$.

However, because of averaging considerations, it is clear that $Y_{min} \leq Y/\frac{\log n}{\epsilon}$. Therefore

$s \geq s_{max} - Y_{min} \geq s_{max} - Y/\frac{\log n}{\epsilon}$. However, $s_{max} = X + Y \geq 2Y$, therefore $Y/\frac{\log n}{\epsilon} \leq s_{max}/\frac{2\log n}{\epsilon}$. Thus $s_{max} - Y/\frac{\log n}{\epsilon} \geq (1 - \frac{\epsilon}{2\log n})s_{max}$. $\square$

5 Conclusions and Open Problems

The idea of using patches for indexing, presented by Amir and Parienty [2], is not viable if tiling can not be done efficiently. In this paper we showed a PTAS for the square tiling problem over fixed finite alphabets. An interesting open question is whether square tiling over an infinite alphabet is also approximable.

An intriguing direction is, perhaps, using rectangles, rather than squares for indexing, since we have shown that rectangle tiling is polynomially computable for "long and skinny" rectangles over a finite fixed alphabet. Indeed, for indexing purposes, the entire square will rarely be sought. Thus the results of this paper bring encouraging evidence to the proposal of utilizing patches for indexing.

References

1. Amir, A., Apostolico, A., Landau, G.M., Satta, G.: Efficient text fingerprinting via Parikh mapping. J. of Discrete Algorithms 1(5-6), 409–421 (2003)
2. Amir, A., Parienty, H.: Towards a theory of patches. In: Karlgren, J., Tarhio, J., Hyyrö, H. (eds.) SPIRE 2009. LNCS, vol. 5721, pp. 254–265. Springer, Heidelberg (2009)
3. Bergeron, A., Corteel, S., Raffinot, M.: The algorithmic of gene teams. In: Guigó, R., Gusfield, D. (eds.) WABI 2002. LNCS, vol. 2452, pp. 464–476. Springer, Heidelberg (2002)
4. Epshtein, B., Ullman, S.: Identifying semantically equivalent object fragments. In: Proc. IEEE Conference on Computer vision and Pattern Recognition (CVPR), vol. 1, pp. 2–9 (2005)
5. Eres, R., Landau, G.M., Parida, L.: Permutation pattern discovery in biosequences. Journal of Computational Biology 11(6), 1050–1060 (2004)
6. He, X., Goldwasser, M.H.: Identifying conserved gene clusters in the presence of orthologous groups. In: Proc. 8th Annual International Conferences on Research in Computational Molecular Biology (RECOMB), pp. 272–280 (2004)
7. Heber, S., Stoye, J.: Finding all common intervals of k permutations. In: Amir, A., Landau, G.M. (eds.) CPM 2001. LNCS, vol. 2089, pp. 207–218. Springer, Heidelberg (2001)
8. Karlsson, F., Voutilainen, A., Heikkilä, J., Anttila, A.: Constraint Grammar. A Language Independent System for Parsing Unrestricted Text. Mouton de Gruyter (1995)
9. Lu, G.: Indexing and retrieval of audio: A survey. Multimedia Tools and Applications 15(3), 269–290 (2001)
10. Magazine, M.J., Chern, M.-S.: A note on approximation schemes for multidimensional knapsack problems. Mathematics of Operations Research 9(2), 244–247 (1984)

11. Vidal-Naquet, M., Ullman, S., Sali, E.: A fragment-based approach to object representation and classification. In: Arcelli, C., Cordella, L.P., Sanniti di Baja, G. (eds.) IWVF 2001. LNCS, vol. 2059, pp. 85–102. Springer, Heidelberg (2001)
12. Schmidt, T., Stoye, J.: Quadratic time algorithms for finding common intervals in two and more sequences. In: Sahinalp, S.C., Muthukrishnan, S.M., Dogrusoz, U. (eds.) CPM 2004. LNCS, vol. 3109, pp. 347–358. Springer, Heidelberg (2004)
13. Srisuwannapa, C., Chamsethikul, P.: An exact algorithm for the unbounded knapsack problem with minimizing maximum processing time. J. of Computer Science 3(3), 138–143 (2007)
14. Stricker, M., Swain, M.: The capacity of color histogram indexing. In: Proc. IEEE Conference on Computer vision and Pattern Recognition (CVPR), pp. 704–708 (1994)
15. Uno, T., Yagiura, M.: Fast algorithms to enumerate all common intervals of two permutations. Algorithmica 26(2), 290–309 (2000)