SIAM J. IMAGING SCIENCES
Vol. 8, No. 2, pp. 916–938
© 2015 Society for Industrial and Applied Mathematics

A Global Approach for Solving Edge-Matching Puzzles*

S. Z. Kovalsky†, D. Glasner‡, and R. Basri†

Abstract. We consider apictorial edge-matching puzzles, in which the goal is to arrange a collection of puzzle pieces with colored edges so that the colors match along the edges of adjacent pieces. We devise an algebraic representation for this problem and provide conditions under which it exactly characterizes a puzzle. Using the new representation, we recast the combinatorial, discrete problem of solving puzzles as a global, polynomial system of equations with continuous variables. We further propose new algorithms for generating approximate solutions to the continuous problem by solving a sequence of convex relaxations.

Key words. edge-matching puzzles, convex optimization, relaxation, polynomial systems

AMS subject classifications. 90C90, 13, 65

DOI. 10.1137/140987869

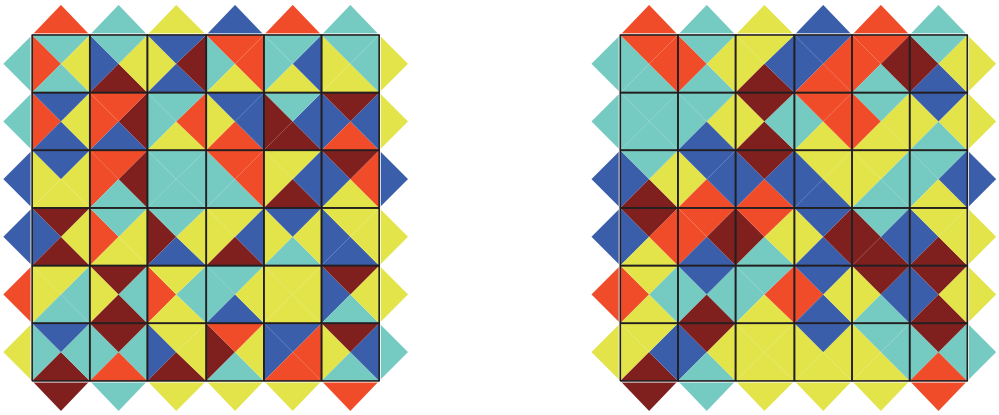
1. Introduction.

Jigsaw puzzles [47], dating back to the 1760s, are among the most popular single-player puzzles. The edge-matching puzzle, introduced in the 1890s, is a variation of the jigsaw puzzle in which the goal is to arrange a given collection of tiles with colored edges so that the colors match up along the edges of adjacent tiles. An example of an edge-matching puzzle with 36 square pieces is shown in Figure 1. Edge-matching puzzles are challenging compared to standard jigsaw puzzles as only an entire solution guarantees the correctness of any local part of the solution.

Despite recent breakthroughs in algorithmic solutions for pictorial puzzles [39, 20, 43], in which one aims to reorganize a scrambled image, relatively little attention had been given to apictorial edge-matching puzzles. This NP-hard problem [17] sparked a lot of interest following the launch of the Eternity puzzle challenges. The challenge posed in the Eternity I puzzle was to tile a large dodecagon with 209 irregularly shaped smaller polygonal pieces; it was marketed as being practically unsolvable but was solved within a year, for a prize of £1 million. For the Eternity II puzzle one must correctly place 256 square pieces, whose edges are marked with different patterns, into a 16×16 grid. The puzzle was launched in 2007 and to date remains unsolved. A prize of $ 2 million was on offer up until December 2010.

In this paper we propose a novel representation for apictorial edge-matching puzzle games in terms of algebraic varieties, i.e., as solutions of systems of polynomial equations derived from the pieces of the puzzle. We explain how to generate systems of polynomial equations which are

*Received by the editors September 19, 2014; accepted for publication (in revised form) February 9, 2015; published electronically April 21, 2015. This research was supported in part by the Israel Science Foundation grants 764/10 and 1265/14 and by the Minerva Foundation. The vision group at the Weizmann Institute is supported in part by the Moross Laboratory for Vision Research and Robotics.
http://www.siam.org/journals/siims/8-2/98786.html
†Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 76100, Israel (shahar.kovalsky@weizmann.ac.il, ronen.basri@weizmann.ac.il).
‡School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138 (dglasner@gmail.com).



(a) Scrambled (b) Solved

Figure 1. *An edge-matching puzzle with* 36 *square pieces and a frame. Square puzzle pieces are overlaid with four colored triangles, representing the colors of their four edges. The puzzle frame is similarly represented by colored triangles placed along its boundaries. Given a scrambled puzzle as shown on the left, the goal is to rearrange the pieces so that the colors match along adjacent edges. A solution generated using our method is shown on the right.*

satisfied by puzzle solutions. We refer to systems for which the converse also holds, that is, any solution of the system is a solution of the puzzle, as *complete* representations. We characterize and prove the completeness of representations for two-dimensional translation-only puzzles (i.e., puzzles with two-dimensional pieces which can be translated but not rotated).

Using our algebraic representation we devise new algorithms for solving edge-matching puzzles. We show that approximate solutions can be generated by solving a sequence of *continuous* and *global* convex optimization problems. Our motivation for seeking a global solution strategy is the unique characteristic of edge-matching puzzle problems: only an entire (global) solution provides a certificate of correctness for any local part of the solution. Convergence to a global solution of the original nonconvex problem, which corresponds to a solution of the puzzle, is achieved for some interesting puzzle instances.

Specifically, we propose two types of convex relaxations for two-dimensional translation only edge-matching puzzles. For puzzles where we know in advance that pieces may only be placed in a finite set of predetermined locations, we propose a *Vandermonde formulation*, where a solution to the puzzle is attained by approximately solving a constrained optimization problem over the manifold of Vandermonde matrices using linear programming. For the more general setting, we propose a *rank one formulation*, where we approximate a solution to a constrained optimization over rank one matrices using semidefinite programming (SDP).

Finally, we show how our computational framework and algorithms can be extended to various interesting variants of edge-matching puzzles, including higher-dimensional puzzles, puzzles in which the pieces have irregular shapes (e.g., see Figure 6) and puzzles for which not only the location but also the orientation of the pieces is unknown (e.g., see Figure 7).

Effective representations and algorithms for solving puzzle games have applications beyond mere theoretical interest. Computational solutions of puzzles are used in computer-aided

reconstruction in archaeology [23, 4, 14, 27], the recovery of shredded documents [22, 34] or photos [30], and in image editing [8, 9]. They are also relevant to speech descrambling [49], machine translation of text [28], and even biology (DNA sequence reconstruction can be viewed as a jigsaw puzzle) [33] and chemistry (determination of molecular conformation) [31].

2. Related work. Early work, beginning with Freeman and Gardner [19] in 1964, developed algorithmic solutions for jigsaw puzzle games based solely on their geometry; a detailed review can be found in [46]. Subsequent algorithms combined shape with the image content of the pieces [24, 10, 48, 32, 36].

A number of methods for solving pictorial puzzles with square pieces have been proposed. Cho, Avidan, and Freeman [7] present a probabilistic framework based on the patch transform proposed in [8, 9], which synthesizes an image from a set of image patches. Approximate puzzle reconstruction is achieved via loopy belief propagation on a suitable graphical model. More recently, Pomeranz, Shemesh and Ben-Shahar [39] and Gallagher [20] proposed greedy algorithms that solve puzzles comprising thousands of pieces. While [39] solves only for translations in the plane, [20] and [43] allow pieces to rotate and solve for their orientations as well. Methods based on nonconvex constrained quadratic programming [1] as well as genetic algorithms [41] have shown competitive results. All these approaches, however, rely on the statistics of natural images (either explicitly or implicitly). Therefore, they are not expected to perform well if directly applied to apictorial edge-matching puzzles, in which there is no image content.

Specific puzzles have also been studied: Deutsch and Hayes [18] suggest a heuristic approach for solving the tangram puzzle. A connectionist approach to solving the same puzzle was proposed by Oflazer [37]. Dattorro [15] proposed an interesting convex SDP approach for attempting to solve the Eternity II puzzle.

Demaine and Hearn [17] and Demaine and Demaine [16] studied jigsaw puzzles from the perspective of combinatorial game theory. In the latter, the decision problems corresponding to jigsaw puzzles and edge-matching puzzles (i.e., "does this puzzle have a solution") are shown to be NP-complete.

3. Problem statement. In this work, we focus on *ideal edge-matching framed* puzzles. That is, apictorial geometric puzzles in which a set of pieces of known shapes and edge colors is given. The puzzle is bounded by a frame and each edge must perfectly match (in its location, color, and orientation) an edge of either another piece or the frame.

We start by considering simple polygonal two-dimensional puzzles, where puzzle pieces are equilateral polygons with unit-length colored edges. Furthermore, we will assume that

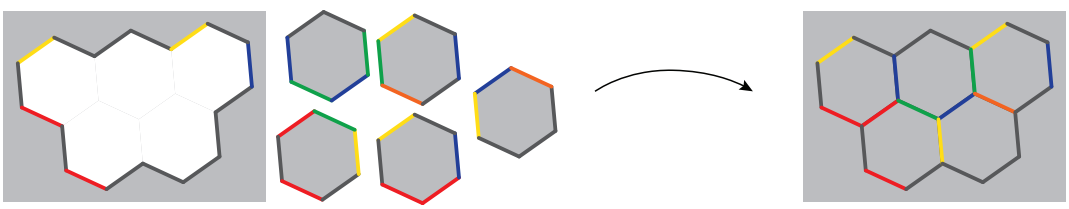


Figure 2. *Problem statement: place all pieces within a given bounding frame such that all edges pair with matching edges.*

puzzle pieces may only be translated (i.e., the rules of the game allow shifting the pieces but not rotating them).

The goal of the game is then to place all pieces within the given bounding frame such that all edges pair with matching edges. This, in turn, implies that the puzzle pieces cover the frame (for a properly designed puzzle); see Figure 2.

Next, we formally present an algebraic representation for these types of puzzles. In section 6 we show how to extend our framework to cases in which the above assumptions do not hold such as puzzles with irregularly shaped pieces, higher-dimensional puzzles, and puzzles in which rotating the pieces is also allowed.

3.1. Formulation. Consider a two-dimensional polygonal puzzle with N pieces. As illustrated in Figure 3, we describe the ith piece by the location $t_i \in \mathbb{R}^2$ of its center and its set of edge elements E_i (polygon sides). Each edge $j \in E_i$ is described by the relative location of its center $b_{i,j}$ with respect to the piece center its color $c_{i,j}$, and its orientation $\theta_{i,j}$. Therefore, the absolute location of the jth edge of the ith piece is given by the sum $t_i + b_{i,j}$.

We shall consider puzzles with a bounding frame, which can be seen as another puzzle piece, with the exception of being static. We denote the piece corresponding to the puzzle frame by $i = 0$ and the properties of its edge elements by $b_{0,j}$, $c_{0,j}$, and $\theta_{0,j}$.

Under the assumption that puzzle pieces may *only be translated*, the goal of the game amounts to finding $t_1, \ldots, t_N$ for which all edge elements pair with matching edge elements in their spatial locations, colors, and orientations (see Figure 2). We shall refer to a configuration $t_1, \ldots, t_N$ that satisfies this criterion as a *solution* of the puzzle.

4. Algebraic representation of a puzzle. In this section, we derive an algebraic representation of a puzzle. We show how, given a puzzle, one can construct an algebraic system of equations that characterizes the puzzle. Furthermore, we prove that this representation is complete, in the sense that it encapsulates all the information available in the given puzzle and its solutions are exactly the solutions of the puzzle.

Suppose $t_1, \ldots, t_N$ is a solution of the puzzle. By definition, every edge is paired with a matching edge. Consider such a pair of matching edges, edge j of the ith piece and edge $\hat{j}$ of the $\hat{i}$'th piece. By definition, these edges must spatially coincide, i.e.,

$$t_i + b_{i,j} = t_{\hat{i}} + b_{\hat{i},\hat{j}}. \tag{4.1}$$

They must have the same color,

$$c_{i,j} = c_{\hat{i},\hat{j}}, \tag{4.2}$$

and they must face opposite directions,

$$\theta_{i,j} \equiv \theta_{\hat{i},\hat{j}} + \pi \pmod{2\pi}. \tag{4.3}$$

See Figure 3 for an illustration. In particular, notice that (4.1) linearly constrains the relative locations of the two pieces. For simplicity of notation we will omit the congruent modulo notation for angles with the understanding that $\theta = \phi$ is shorthand for $\theta \equiv \phi \pmod{2\pi}$.

In general, however, we do not know in advance which edges pair with each other. Nevertheless, we notice that all edges of a certain color and orientation (c, θ) must pair with

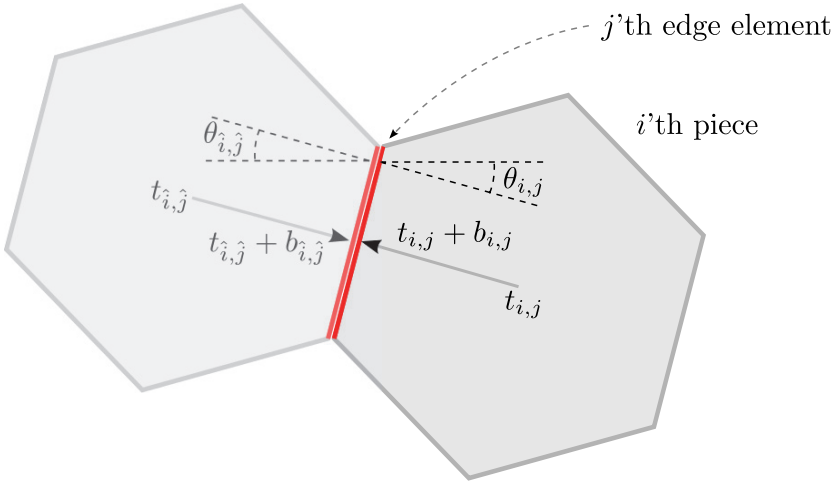


Figure 3. *The description of edge elements and the constraints implied by two matching edge elements.*

complementary edges of the same color and opposite orientation $(c, \theta + \pi)$. Therefore, for every (c, θ) we have an equality of sets

$$\left\{ t_i + b_{i,j} : \begin{array}{l} c_{i,j} = c \\ \theta_{i,j} = \theta \end{array} \right\} = \left\{ t_i + b_{i,j} : \begin{array}{l} c_{i,j} = c \\ \theta_{i,j} = \theta + \pi \end{array} \right\}. \tag{4.4}$$

Clearly, (4.4) holds for all (c, θ) if and only if $t_1, \ldots, t_N$ is a solution of the puzzle.

If we define the signed indicator function w.r.t. (c, θ) by

$$s_{i,j}(c, \theta) = \begin{cases} 1, & c_{i,j} = c \text{ and } \theta_{i,j} = \theta, \\ -1, & c_{i,j} = c \text{ and } \theta_{i,j} = \theta + \pi, \\ 0 & \text{otherwise,} \end{cases} \tag{4.5}$$

then by construction we have derived the following result.

Proposition 4.1. *If $t_1, \ldots, t_N$ is a solution of the puzzle, then*

$$\sum_{i,j} s_{i,j}(c, \theta) f\left(t_i + b_{i,j}\right) = 0 \tag{4.6}$$

for every (c, θ) and every real valued function $f : \mathbb{R}^2 \to \mathbb{R}$.

Note that (4.6) is an algebraic constraint on $t_1, \ldots, t_N$. This allows us to construct an algebraic *system* of constraints by taking all admissible (c, θ)'s and different f's. Different choices of a function f establish systems with possibly different properties.

4.1. Linear representation. A particular choice of f is $f_v(u) = v^T u$ for any $v \in \mathbb{R}^2$. This choice yields a linear representation of the puzzle. Namely, equations of the form

$$\sum_{i,j} s_{i,j}(c, \theta) \left(t_i + b_{i,j}\right) = 0 \tag{4.7}$$

for all (c, θ) define a *linear system of equations* in $t_1, \ldots, t_N$. Clearly, every solution of the puzzle must satisfy this system of equations. Usually, however, the converse does not hold, and this linear system does not determine the solutions of the puzzle.

4.2. Polynomial representation. In the rest of the paper, we use functions f which belong to a family of *exponential* functions. For a given $k \in \mathbb{R}^2$, we define $f_k(u) = e^{k^T u}$. Equation (4.6) then becomes

$$\sum_{i,j} s_{i,j}(c,\theta) e^{k^T(t_i + b_{i,j})} = 0. \tag{4.8}$$

To simplify notation we change variables $T_i = e^{t_i}$ (elementwise exponentiation) to obtain

$$\sum_i \alpha_i^{(k,c,\theta)} T_i^k = 0, \tag{4.9}$$

where $\alpha_i^{(k,c,\theta)} = \sum_j s_{i,j}(c,\theta) e^{k^T b_{i,j}}$, and T_i^k is defined according to the multi-index notation

$$T_i^k = (e^{t_i})^k = e^{k^T t_i}.$$

Note that (4.9) is simply a polynomial in (the entries of) $T_1, \ldots, T_N$. Collecting many equations of the form (4.9) for all (c,θ) and various choices of k establishes a *polynomial system of equations* in $T_1, \ldots, T_N$.

According to Proposition 4.1, if $t_1, \ldots, t_N$ is a solution of the puzzle, then $T_1, \ldots, T_N$ is a solution of any such polynomial system of equations. Next, we discuss the conditions under which the converse holds, so that every solution of the polynomial system corresponds to a valid solution of the puzzle. In what follows, we identify $t_1, \ldots, t_N$ with $T_1, \ldots, T_N$ and thus refer to the latter as a solution of the puzzle as well.

4.3. Completeness of polynomial representations. Naturally, we seek a complete representation, one that encapsulates all the information available in the given puzzle and exactly characterizes the solutions of the puzzle.

We consider the polynomial system constructed by collecting equations of the form (4.9) for different (c,θ) and various k values. Intuitively, if additional equations provide additional independent constraints, then we can hope that collecting many such equations will sufficiently constrain the unknown variables, exactly determining the solutions of the puzzle. Indeed, we can show that a system of sufficiently many such polynomial equations establishes an equivalent representation, in which $T_1, \ldots, T_N$ is a solution of the puzzle if and only if it is a root of this system.

More formally, consider the polynomial system $\mathcal{P}$ constructed by collecting equations as follows: for every possible type of edge (c,θ) add $K(c,\theta) \in \mathbb{N}$ equations of the form (4.9),

$$\sum_i \alpha_i^{(k,c,\theta)} T_i^k = 0,$$

for distinct values of k, where $K(c,\theta)$ is an integer which depends on the edge type (c,θ). This construction is well defined since, by our assumption of polygonal puzzle pieces, there exist finitely many distinct edge types (c,θ).

By construction, every solution of the puzzle is a solution of $\mathcal{P}$. The proposition below suggests that the converse also holds, provided that $\mathcal{P}$ is defined by sufficiently many equations.

In particular, Appendix A provides a *constructive* proof which shows that the number of equations required for each type of edge (c, θ) is exactly the number of puzzle edges of this type.

Proposition 4.2. *There exists (constructively) a polynomial system $\mathcal{P}$ with*

$$K(c,\theta) = \#\{\textit{edges of type } (c,\theta)\}$$

equations for each edge type (c, θ) that satisfies that $T_1, \ldots, T_N$ is a solution of $\mathcal{P}$ if and only if it is a solution of the puzzle.

The proof relies on the following observation. Consider *independently* all $K = K(c, \theta)$ edges of type (c, θ). Any pairing of these edges with corresponding opposite edges satisfies polynomial equations of the form (4.9) for any order k. There are exactly $K!$ such pairings, as the number of permutations of K elements. On the other hand, using Bézout's theorem [12, 44] we can show that a polynomial system of equations of the form (4.9) of orders $k = 1, \ldots, K$ has no more than $K!$ distinct solutions. Thus, such a system exactly characterizes all possible independent pairings of (c, θ) edges, in the sense that any subset of its equations is insufficient, whereas additional higher-order polynomials are redundant. The full proof (given in Appendix A) follows by generalizing this observation to the intersection of multiple similar systems, each corresponding to edges of a different type (c, θ), and the addition of constraints accounting for the geometry of the puzzle pieces.

5. Solving puzzles.

In previous sections we have shown that a puzzle problem can be faithfully represented as a system of polynomial equations

$$\left\{ \sum_i \alpha_i^{(k,c,\theta)} T_i^k = 0 \right\}_{k,c,\theta} \tag{5.1}$$

in the unknowns $T_1, \ldots, T_N$.

Next, we discuss several approaches for determining a solution $T_1, \ldots, T_N$ of this system, either directly or by approximation. Once such a solution is found, one may recover the piece locations $t_1, \ldots, t_N$ using the relation $T_i = e^{t_i}$. Assuming the conditions for Proposition 4.2 hold, these locations are guaranteed to solve the puzzle.

To simplify notations we present the derivations for one-dimensional puzzles (with coordinates over $\mathbb{R}$). Extending the results to the case of two-dimensional puzzles, discussed so far (or in fact arbitrary dimension), is straightforward; see Appendix B for the technical details.

5.1. Exact solutions.

Solving polynomial systems of equations is an important problem which is covered in a large body of literature; for example, see [11, 45, 13, 26] and the references therein. Existing methods for solving generic systems may be coarsely classified into two categories: symbolic (exact) and numerical methods.

Gröbner algorithms [5] are typical representatives of the class of symbolic methods. These algorithms seek to simplify a given polynomial system, to enable the extraction of its roots. This can be seen as a multivariate, nonlinear generalization of the Euclidean algorithm (for computation of univariate GCD) and Gaussian elimination for linear systems. However, the complexity of methods for calculating Gröbner bases may be extremely high.

A popular approach in the class of numerical methods is homotopy continuation. Such methods rely on Bertini's theorem, which introduces a continuous deformation between the

polynomial system whose solution we seek and a simpler polynomial system whose solutions are known; see [29, 42] for more details. Keeping track of the roots during this deformation can provide a solution to the system. However, a good initialization (i.e., an initial polynomial system with known solutions) is crucial to the success of the algorithm. Unfortunately, the combinatorial nature of the polynomial systems we are concerned with typically makes it hard to find a good initialization.

5.2. Reformulation and convex relaxation. Our attempts to solve very simple puzzles using generic polynomial system solvers met with limited success. Therefore, in what follows we describe a reformulation and optimization algorithms which try to take advantage of the unique properties of systems of the form (5.1).

Note that the polynomial system (5.1) has a special structure: each one of its equations includes only kth degree monomials in the unknown T_i's. We exploit this structure in order to restate the problem of solving puzzles as a linear system of equations over a (nonlinear) manifold of matrices. We discuss two alternative reformulations, relying on either Vandermonde or rank one matrices. We further discuss convex relaxations of the resulting problems, which can be used to solve certain types of puzzles.

5.2.1. Vandermonde reformulation and preset locations. Any polynomial system of equations can always be replaced with a linear system of equations of a higher dimension by considering each monomial as a variable, and adding simple polynomial constraints coupling the new variables. In the case of our polynomial systems, this can be conveniently stated using Vandermonde matrices. Recall that a matrix of the form

$$\begin{bmatrix} 1 & T_1 & T_1^2 & \cdots & T_1^K \\ 1 & T_2 & T_2^2 & \cdots & T_2^K \\ 1 & T_3 & T_3^2 & \cdots & T_3^K \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & T_N & T_N^2 & \cdots & T_N^K \end{bmatrix} \tag{5.2}$$

is known as a *Vandermonde* matrix. It is straightforward to see that the problem of solving (5.1) can be equivalently reformulated as

$$\begin{array}{ll} \text{find} & X = [x_{ij}] \\ \text{subject to} & X \text{ is Vandermonde of size } N \times K, \\ & \sum_i \alpha_i^{(k,c,\theta)} x_{ij} = 0 \qquad \forall j, c, \theta. \end{array} \tag{5.3}$$

To ensure that X contains the monomials necessary for a complete polynomial representation as in Proposition 4.2 we choose $K \geq \max_{(c,\theta)} K(c,\theta)$. Formulation 5.3 amounts to solving a linear system over the manifold of Vandermonde matrices. Of course this new problem is as hard to solve as the original problem itself. We will now show how to derive an approximation under an additional assumption.

Preset locations. In many puzzles, pieces may only be placed at a finite number of preset locations. This is the case, for example, in square tiling puzzles (e.g., see Figure 1). This additional information can be incorporated into the formulation (5.3).

Suppose that piece locations $t_1, \ldots, t_N$ must belong to a finite *known* set of feasible locations $\{s_1, \ldots, s_N\}$. This assumption can be formally stated by

$$\begin{bmatrix} t_1 \\ \vdots \\ t_N \end{bmatrix} = P \begin{bmatrix} s_1 \\ \vdots \\ s_N \end{bmatrix} \tag{5.4}$$

for some permutation matrix P. In turn, since $T_i = e^{t_i}$, this implies that

$$\begin{bmatrix} T_1 \\ \vdots \\ T_N \end{bmatrix} = P \begin{bmatrix} S_1 \\ \vdots \\ S_N \end{bmatrix}, \tag{5.5}$$

where $S_i = e^{s_i}$. We use the known preset piece locations to define the Vandermonde matrix Y by

$$Y = \begin{bmatrix} 1 & S_1 & S_1^2 & \cdots & S_1^K \\ 1 & S_2 & S_2^2 & \cdots & S_2^K \\ 1 & S_3 & S_3^2 & \cdots & S_3^K \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & S_N & S_N^2 & \cdots & S_N^K \end{bmatrix}. \tag{5.6}$$

Then, (5.5) implies that any solution of problem (5.3) must satisfy

$$X = PY. \tag{5.7}$$

Notice that $X = PY$ defines a Vandermonde matrix for any choice of *permutation* matrix P. Therefore, problem (5.3) can be replaced with the following equivalent problem:

$$\begin{array}{ll} \text{find} & P = [p_{i,j}] \\ \text{subject to} & P \text{ is a permutation,} \\ & X = [x_{ij}] = PY, \\ & \sum_i \alpha_i^{(k,c,\theta)} x_{i,j} = 0 \qquad \forall j, c, \theta. \end{array} \tag{5.8}$$

Thus, we have replaced the Vandermonde constraint with the restriction of the search space to the set of permutation matrices, which in turn corresponds to the set of feasible piece locations.

Convex relaxation. Problem (5.8) is a nonconvex feasibility problem. Nevertheless, we next show that it can be restated as a linearly constrained quadratic maximization problem and then *relaxed* into a sequence of linear programs (LPs).

The Birkhoff–von Neumann theorem [3] asserts that permutations are the extremal points of the set of bi-stochastic matrices. Moreover, the Frobenius norm of matrices in this set is maximized by permutations. Using this, (5.8) can be replaced with the following equivalent optimization problem:

$$\begin{array}{ll} \text{maximize} & \|P\|_F^2 \\ \text{subject to} & P\mathbf{1} = \mathbf{1},\ \mathbf{1}^T P = \mathbf{1}^T,\ P \geq 0, \\ & X = [x_{ij}] = PY, \\ & \sum_i \alpha_i^{(k,c,\theta)} x_{i,j} = 0 \qquad \forall j, c, \theta, \end{array} \tag{5.9}$$

ALGORITHM 1. ITERATIVE LP APPROXIMATION FOR PUZZLES WITH PRESET PIECE LOCATIONS.

Input: Polynomial representation coefficients $\alpha_i^{(k,c,\theta)}$
Preset locations matrix Y as in (5.6)
Output: Puzzle solution matrix X

Initialize $P^{(0)} = 0$; $n = 0$;
repeat
  $n = n + 1$;
  Solve the linear program (5.10) with $P^{(n-1)}$;
  Set $P^{(n)}$ to be the optimizer;
until $P^{(n)}$ *is a permutation*;
return $X = P^{(n)}Y$;

where $\mathbf{1} \in \mathbb{R}^N$ is the all ones vector. This is again a nonconvex optimization, as it maximizes norm.

We generate approximate solutions to (5.9) by linearizing its quadratic objective term and applying an iterative algorithm, in a similar spirit to reweighted optimization approaches (e.g., reweighted ℓ_1 minimization [6]).

Note that $\|P\|_F^2 = \operatorname{tr}\left(P^T P\right) = \langle P, P\rangle$, where $\langle A, B\rangle = \operatorname{tr}\left(B^T A\right)$ is the standard matrix inner product. Suppose that $\hat{P}$ is an approximate guess of P. We linearly approximate the quadratic objective as $\langle \hat{P}, P\rangle$ and iterate solving the following LP:

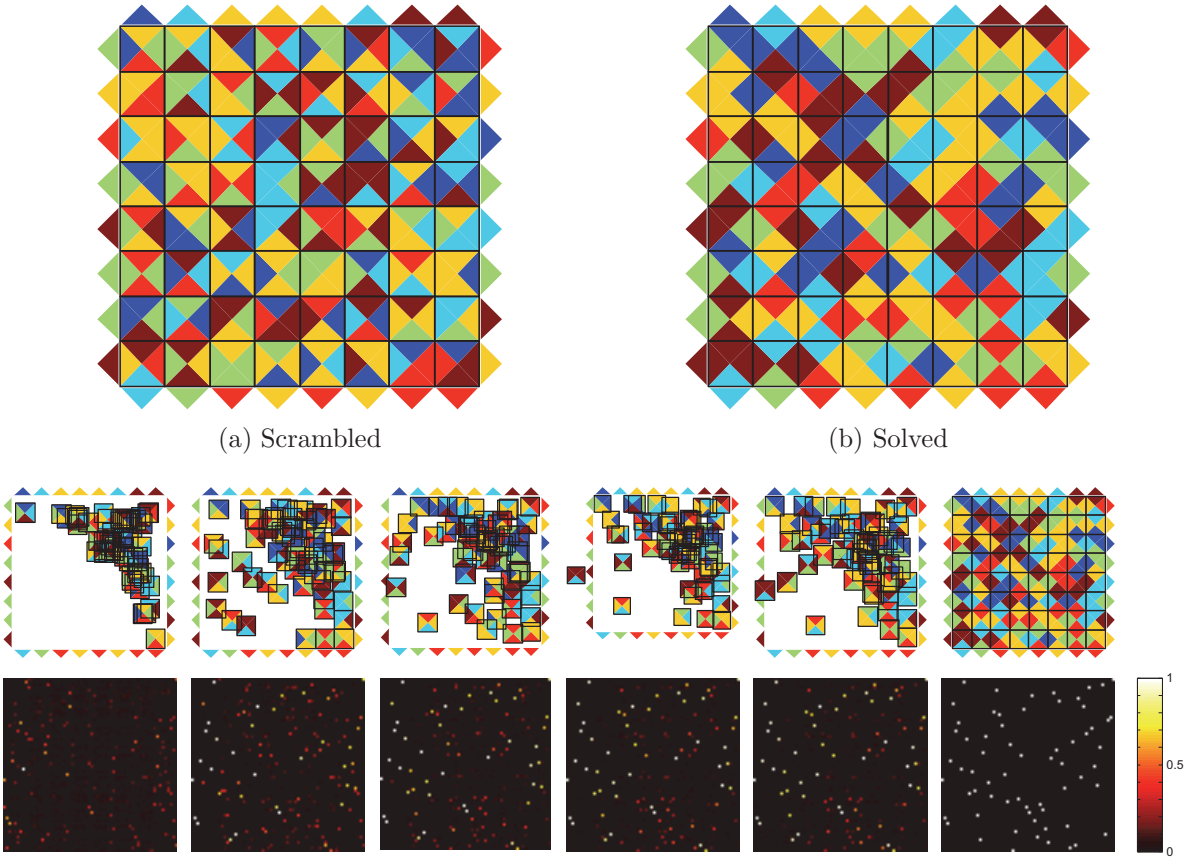
$$
\begin{array}{ll}
\text{maximize} & \left\langle P^{(n-1)}, P\right\rangle \\
\text{subject to} & P\mathbf{1} = \mathbf{1},\ \mathbf{1}^T P = \mathbf{1}^T,\ P \geq 0, \\
& X = [x_{ij}] = PY, \\
& \sum_i \alpha_i^{(k,c,\theta)} x_{i,j} = 0 \qquad \forall j, c, \theta,
\end{array}
\tag{5.10}
$$

where $P^{(n-1)}$ is the optimizer of the previous iteration. The algorithm is outlined in Algorithm 1.

Note that a global optimizer of (5.9) is a fixed point of (5.10). We initialize (5.10) with $P^{(0)} = 0$. With this initialization (5.10) is equivalent to the standard LP relaxation of the feasibility problem (5.8) over the convex hull of permutations. This sequence of optimization problems attains a bounded and monotonically nondecreasing objective. However, convergence to a permutation is not guaranteed.

Examples. We demonstrate the proposed approach by solving examples of edge-matching puzzles. Specifically, we generate two-dimensional square tiling puzzles of various sizes whose edge colors are drawn at random; see Figures 1 and 4.

We use the polynomial representation described in section 4.2 to calculate the coefficients of a corresponding polynomial system of equations and apply Algorithm 1. Figure 1 shows the method applied to a 6×6 puzzle. Only two iterations of (5.10) were required to attain a permutation, which corresponds to the sole solution of the puzzle.



(a) Scrambled (b) Solved

(c) Iterations of (5.10), left to right. Top row shows intermediate puzzle solutions. Bottom row shows the state of the variable P (a 64×64 matrix), converging to a permutation matrix. White entries indicate a value of 1, black is 0, and intermediate colors correspond to fractional values.

Figure 4. *Solution of an* 8×8 *puzzle using our method.*

Figure 4 depicts the method applied to an 8×8 puzzle. Six iterations of (5.10), shown at the bottom of the figure, were required to attain a permutation. In this case, the algorithm collapses into one of the two solutions of this puzzle.

5.2.2. Rank one reformulation. For certain types of puzzles the set of valid locations of the pieces in a solution is not known in advance. One such example, where due to the irregularity of piece shapes their possible locations cannot be predetermined, is the *tangram* puzzle (Figures 5 and 6). In this section we propose an alternative reformulation of (5.1) as an optimization over rank one matrices.

Optimization subject to rank one constraints, or more generally low-rank optimization, has recently received substantial interest (e.g., [38, 25, 35, 40]). We suggest a convex relaxation algorithm for generating approximate solutions to our rank one formulation, which does not require the piece locations to be known in advance.

Define N *Hankel* matrices $Z_1, \ldots, Z_N$ of the form

$$Z_i = \begin{bmatrix} 1 & z_{i,1} & z_{i,2} & \cdots & z_{i,K} \\ z_{i,1} & z_{i,2} & & \iddots & z_{i,K+1} \\ z_{i,2} & & \iddots & & \vdots \\ \vdots & \iddots & & \iddots & \\ z_{i,K} & z_{i,K+1} & \cdots & & z_{i,2K} \end{bmatrix}. \tag{5.11}$$

We state the following problem:

$$\begin{array}{lll} \text{find} & Z_1, \ldots, Z_N \text{ of the form (5.11)} & \\ \text{subject to} & \sum_i \alpha_i^{(k,c,\theta)} z_{i,j} = 0 & \forall j, c, \theta, \\ & z_{i,j} \geq 0 & \forall i, j, \\ & \operatorname{rank}(Z_i) = 1 & \forall i. \end{array} \tag{5.12}$$

It is clear that if $T_1, \ldots, T_N$ is a solution of the puzzle, then the assignment

$$Z_i = \begin{bmatrix} 1 & T_i & T_i^2 & \cdots & T_i^K \\ T_i & T_i^2 & & \iddots & T_i^{K+1} \\ T_i^2 & & \iddots & & \vdots \\ \vdots & \iddots & & \iddots & \\ T_i^K & T_i^{K+1} & \cdots & & T_i^{2K} \end{bmatrix} = \begin{bmatrix} 1 \\ T_i \\ T_i^2 \\ \vdots \\ T_i^K \end{bmatrix} \begin{bmatrix} 1 \\ T_i \\ T_i^2 \\ \vdots \\ T_i^K \end{bmatrix}^T \tag{5.13}$$

is a solution of problem (5.12). To show that (5.12) is in fact equivalent to the problem of solving the polynomial system (5.1) we must show that any solution to (5.12) is also a solution of the polynomial system (5.1).

Toward this end, we use the following simple lemma.

Lemma 5.1. $\operatorname{rank}(Z_i) = 1$ *and* $z_{i,j} \geq 0$ *if and only if* $z_{i,j} = z_{i,1}^j$ *for all* j.

The lemma follows immediately by noticing that Z_i must be an outer product of its first column with itself. This implies that under the constraints of problem (5.12), the matrix Z_i is simply the outer product of the ith row of the Vandermonde matrix (5.2) with itself. Thus, every solution of the low-rank problem (5.12) is a solution of the Vandermonde problem (5.3) and in turn of the original polynomial system (5.1).

Therefore, (5.12) is an alternative formulation to the problem of solving puzzles. It can be interpreted as finding a solution of a linear system over nonlinear manifolds of rank one matrices.

Convex relaxation. Problem (5.12) is a nonconvex feasibility problem. Note that its constraints $\operatorname{rank}(Z_i) = 1$ and $z_{i,j} \geq 0$ imply that Z_i is a positive semidefinite matrix, which we denote by $Z_i \succeq 0$. Namely, since Z_i is a symmetric rank one matrix, it admits a rank one eigendecomposition of the form $Z_i = \lambda_i u_i u_i^T$ for some vector u_i. Having nonnegative entries implies that $\lambda_i \geq 0$, and in turn that $Z_i \succeq 0$.

Dattorro [15] discusses an SDP heuristic for rank-constrained optimization. Inspired by his work, we propose an iterative approximate procedure. Intuitively, the idea is that a rank

ALGORITHM 2. ITERATIVE SDP APPROXIMATION FOR PUZZLES.

Input: Polynomial representation coefficients $\alpha_i^{(k,c,\theta)}$
tolerance ε

Output: Puzzle solution matrices $Z_1, \ldots, Z_N$

Initialize $W_1, \ldots, W_N = 0$; $n = 0$;
repeat
  $n = n + 1$;
  Solve the semidefinite program (5.14) with $W_1, \ldots, W_N$;
  Set $Z_1^{(n)}, \ldots, Z_N^{(n)}$ to be the optimizer;
  Calculate the eigendecompositions $Z_i^{(n)} = V_i \Lambda_i V_i^T$;
  Update $\{W_i\}$ according to (5.15);
until $\left\langle W_1, Z_1^{(n)} \right\rangle + \cdots + \left\langle W_N, Z_N^{(n)} \right\rangle \leq \varepsilon$;
return $Z_1 = Z_1^{(n)}, \ldots, Z_N = Z_N^{(n)}$;

one symmetric positive semidefinite matrix can be characterized as one that minimizes the sum of all eigenvalues but the largest. Thus, a feasibility problem of the form (5.12) can be cast as an eigenvalue minimization problem. Since the resulting problem is not convex we generate approximate solutions by employing local linearization and solving a sequence of optimization problems. We iteratively solve the following SDP:

$$\begin{array}{ll} \text{minimize} & \langle W_1, Z_1 \rangle + \cdots + \langle W_N, Z_N \rangle \\ \text{subject to} & Z_1, \ldots, Z_N \text{ are of the form (5.11)}, \\ & \sum_i \alpha_i^{(k,c,\theta)} z_{i,j} = 0 \qquad \forall j, c, \theta, \\ & Z_i \succeq 0 \qquad \forall i, \end{array} \tag{5.14}$$

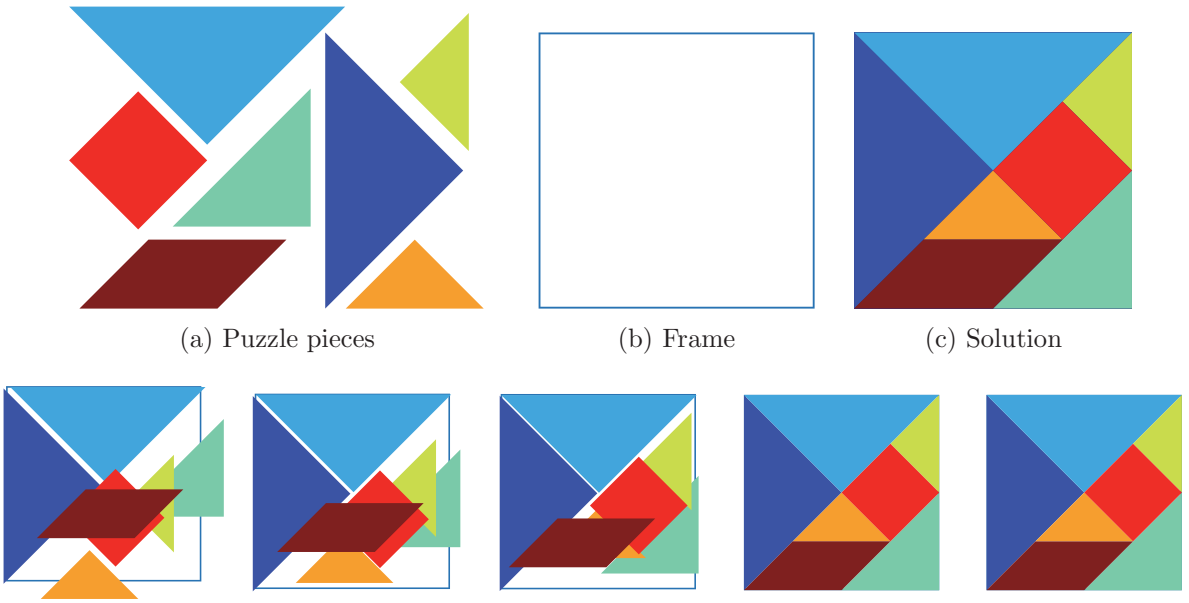
where W_i are fixed matrices, updated at each iteration according to

$$W_i = V_i \begin{bmatrix} 0 & & & 0 \\ & 1 & & \\ & & \ddots & \\ 0 & & & 1 \end{bmatrix} V_i^T, \tag{5.15}$$

and $Z_i^{(n-1)} = V_i \Lambda_i V_i^T$ is the eigendecomposition of $Z_i^{(n-1)}$, the optimizer of the previous iteration, with eigenvalues sorted in descending order.

Note that the functional of (5.14) is nonnegative, as $\{Z_i\}$ and $\{W_i\}$ are all positive semidefinite. Moreover, it vanishes if and only if the rank of each of the matrices Z_i is at most one. Therefore, it is easy to see that a solution of (5.12) is a global optimizer of (5.14), as well as a fixed point of the suggested iterative procedure (see [15] for additional details).

We initialize the iterative procedure with $W_i = 0$, which reduces (5.14) to a standard SDP relaxation for the rank one feasibility problem (5.12). The algorithm is outlined in Algorithm 2. As in section 5.2.1, convergence to a global minimum is not guaranteed.



(a) Puzzle pieces (b) Frame (c) Solution

(d) Iterations of (5.14), left to right. Converging to a solution of the tangram.

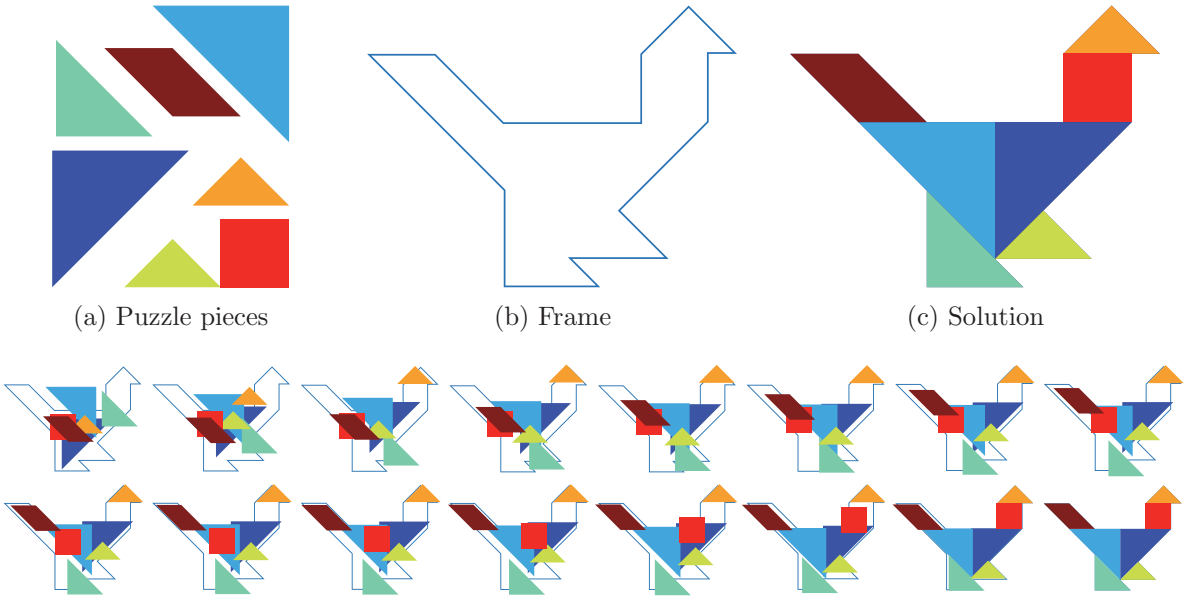
Figure 5. *Solution of a tangram puzzle using our method. In this edge-matching problem all edges share the same color; in the figure pieces are colored for visualization only.*

Examples. To demonstrate the proposed approach, we applied the iterative algorithm to the problem of solving tangram puzzles. Figures 5 and 6 show two instances of tangram puzzles. In these puzzles one aims to tile a given shape (b) with the puzzle pieces (a). Thus, it can be seen as an edge-matching problem in which all edges share the same color. In the figures, we give each piece in a different color for visualization only.

The coefficients of a corresponding polynomial system of equations were calculated as described in section 4.2. (The calculations were slightly adapted to allow for nonequilateral polygonal pieces; see section 6.2 for additional technical details.) Then, solution (c) was obtained by applying Algorithm 2. Panel (d) shows the iterations of the algorithm until convergence. Note that no assumptions on the locations of the puzzle pieces were made, in contrast to the algorithm derived in section 5.2.1, which constrains puzzle pieces to preset locations.

5.2.3. Implementation details and limitations. Algorithms 1 and 2 were both implemented in MATLAB. YALMIP was used for the modeling of the LP (5.10) and SDP (5.14). Gurobi [21] and MOSEK [2], correspondingly, were used for their optimization.

In theory both LPs and SDPs are solvable in polynomial time. In practice, Algorithm 1 scales moderately well with the number of pieces. We have successfully solved instances of 12×12 puzzles. We haven't had as much success with larger problems, such as the Eternity II puzzle, which pose a considerable combinatorial challenge. Algorithm 2 has limited scalability, which is dominated by the number and dimension of the positive definite constraints in SDP (5.14). These, in turn, depend on the number of puzzle pieces and edge colors,



(a) Puzzle pieces (b) Frame (c) Solution

(d) Iterations of (5.14), top left to bottom right. Converging to a solution of the tangram.

Figure 6. *Solution of a tangram puzzle using our method.*

respectively. Current leading SDP optimization engines employ second-order interior-point algorithms which limit the applicability of this approach to larger problems.

6. Extended framework.

In this section we describe extensions of our framework to (i) higher-dimensional puzzles, (ii) general polygonal pieces, and (iii) puzzles in which pieces can be rotated to one of a finite set of discrete orientations.

6.1. High-dimensional puzzles.

Thus far we have restricted our attention to two-dimensional puzzles. We can extend our framework to dimensions $d > 2$ as follows. Consider coordinates $t_i, b_{i,j} \in \mathbb{R}^d$ and real valued functions $f : \mathbb{R}^d \to \mathbb{R}$. Proposition 4.1 continues to hold, implying as in the two-dimensional case that if $t_1, \ldots, t_N$ is a solution of the puzzle, then

$$\sum_{i,j} s_{i,j}(c,\theta) f\left(t_i + b_{i,j}\right) = 0 \tag{6.1}$$

for every (c, θ) and every function f.

The polynomial representation proposed in section 4.2 takes exactly the same form, now by taking $f(u) = e^{k^T u}$, where k is a d-tuple determining the degree of the resulting multivariate polynomial.

The rest of the derivations introduced in the paper follow in the exact same manner with the exception of Proposition 4.2, whose proof is restricted to the two-dimensional case (as it relies on identifying $\mathbb{R}^2$ with $\mathbb{C}$.). We conjecture that a similar result holds in arbitrary dimension as well.

6.2. Shape. So far we have assumed puzzle pieces to be *equilateral* polygons. This assumption can be relaxed to the case of general polygons, with arbitrary edge length, by introducing path integrals over the boundary of pieces. When puzzle pieces are not equilateral, Proposition 4.1 may no longer hold since, for example, puzzle edges are no longer restricted to pair as a whole with a single other edge. That is, in the solution to the puzzle one edge may be paired with multiple edges or even parts of edges. Nevertheless, an analogous proposition holds provided that (4.6) is replaced with

$$\sum_{i,j} s_{i,j}(c,\theta) \int_{\gamma_{i,j}} f\left(t_i + z\right) dz = 0, \tag{6.2}$$

where $\gamma_{i,j}$ is the path corresponding to the jth edge of ith piece. The derivations of sections 3 and 4 follow by making the appropriate adaptations. Note that the tangram examples shown in Figures 5 and 6 involve nonequilateral polygons. Indeed the algebraic representation which we used to solve them was computed by calculating the path integrals of (6.2).

6.3. Orientations. Often one wishes to solve a puzzle for which the orientation of the pieces is also unknown. We now show how the proposed framework can be extended to address this case, under the assumption that finitely many orientations are admissible.

For simplicity, we focus on the two-dimensional case. Let φ_i denote the unknown orientation of the ith piece. Suppose that each orientation φ_i belongs to a finite cyclic group of r rotations, namely, $\varphi_i \in \left\{0, \frac{2\pi}{r}, \ldots, \frac{2\pi}{r}(r-1)\right\}$. This is a reasonable assumption for many puzzles (e.g., 90° rotations are sufficient for square puzzles, 60° rotations for hexagonal puzzles, etc.).

Denote by $R(\theta)$ the rotation matrix

$$R(\theta) = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}. \tag{6.3}$$

Next, we prove a claim that allows US to establish an algebraic representation in the presence of unknown orientations. The intuitive interpretation is that we duplicate the entire puzzle (frame and pieces) r times, rotated in each of the orientations $\left\{0, \frac{2\pi}{r}, \ldots, \frac{2\pi}{r}(r-1)\right\}$. Then, all r puzzles are simultaneously solved as a single augmented translation only puzzle, with the location $t_i^{(\rho)}$ of the ρ-rotated ith piece linearly constrained by $t_i^{(\rho)} = R\left(\frac{2\pi}{r}\rho\right) t_i$. A solution of this augmented puzzle, under the assumption that pieces may only translate, implies a solution of the original puzzle (with unknown orientations). Namely, a claim analogous to Proposition 4.2 holds, by summing over r rotated copies of the puzzle.

Proposition 6.1. *Suppose that a given puzzle is solved by placing the pieces at the locations* $\hat{t}_1, \ldots, \hat{t}_N$ *rotated by* $\varphi_1, \ldots, \varphi_N$. *Then,*

$$\sum_{\rho=0}^{r-1} \sum_{i,j} s_{i,j}(c, \theta - \frac{2\pi}{r}\rho) f\left(R\left(\frac{2\pi}{r}\rho\right)\left(t_i + b_{i,j}\right)\right) = 0 \tag{6.4}$$

for every (c,θ) *and every function* f, *with* $t_i = R(-\varphi_i)\,\hat{t}_i$.

Note that (6.4) yields a representation of the puzzle which is invariant to rotations of the pieces. The solution will consist of r copies of the puzzle. Each piece is associated with a possibly different rotation angle according to the copy of the puzzle to which it is translated, as illustrated in Figure 7.

Proof. Proposition 4.2 implies that if the puzzle is solved by placing the pieces at the locations $\hat{t}_1, \ldots, \hat{t}_N$ rotated by $\varphi_1, \ldots, \varphi_N$, then for every (c, θ) and every function f we have

$$\sum_{i,j} s_{i,j}(c, \theta - \varphi_i) f\left(\hat{t}_i + R(\varphi_i) b_{i,j}\right) = 0. \tag{6.5}$$

Set $t_i = R(-\varphi_i)\hat{t}_i$ to yield

$$\sum_{i,j} s_{i,j}(c, \theta - \varphi_i) f\left(R(\varphi_i)(t_i + b_{i,j})\right) = 0. \tag{6.6}$$

Next, sum over all admissible rotations

$$\sum_{\rho=0}^{r-1} \sum_{i,j} s_{i,j}(c, \theta - \varphi_i - \frac{2\pi}{r}\rho) f\left(R\left(\varphi_i + \frac{2\pi}{r}\rho\right)(t_i + b_{i,j})\right) = 0. \tag{6.7}$$

This equation can be understood as the summation over r rotated copies of the puzzle (as illustrated in Figure 7). The proposition follows by noticing that since $\varphi_i \in \left\{0, \frac{2\pi}{r}, \ldots, \frac{2\pi}{r}(r-1)\right\}$, all terms φ_i may be omitted. ∎

Proposition 6.1 provides a construction which enables us to apply the algorithms developed in sections 4 and 5 to solve puzzles with an unknown discrete set of orientations. In Figure 7 we show an example where we use this construction to solve a puzzle where pieces can be rotated into one of four different orientations.

7. Concluding remarks. In this paper, we propose a novel representation for apictorial edge-matching puzzles. We explain how to generate systems of polynomial equations which are satisfied by puzzle solutions. We further show how to construct systems which are *complete* representations for two-dimensional translation only puzzles. We prove that for these systems the converse also holds, i.e., any solution of the system is a solution of the puzzle.

Using this representation, we devise two algorithms for approximating solutions of edge-matching puzzles. Both algorithms rely on solving a sequence of *continuous* and *global* convex relaxations. An iterative algorithm based on linear programming relaxation is proposed for the case where we know in advance that pieces may only be placed in a finite set of predetermined locations. For the more general setting, we propose an iterative algorithm based on SDP relaxation.

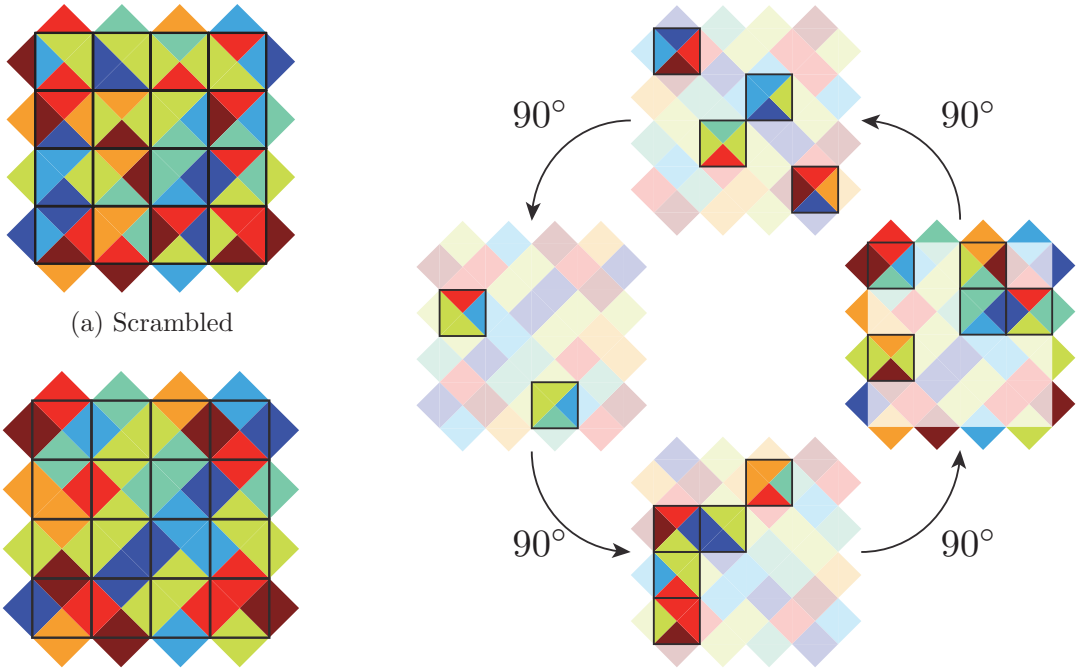
Finally, we extend our computational framework and algorithms to various interesting variants of edge-matching puzzles, including higher-dimensional puzzles, puzzles in which the pieces have irregular shapes, and puzzles in which pieces can be rotated to one of a finite set of discrete orientations.

Appendix A. Proof of Proposition 4.2 (complete polynomial representation).

Proposition 4.2. *There exists (constructively) a polynomial system $\mathcal{P}$ with*

$$K(c, \theta) = \#\{\text{edges of type } (c, \theta)\}$$

equations for each edge type (c, θ) that satisfies that $T_1, \ldots, T_N$ is a solution of $\mathcal{P}$ if and only if it is a solution of the puzzle.



(a) Scrambled

(b) Solution

(c) Solution of the augmented puzzle

Figure 7. *Unknown orientation addressed by solving an augmented puzzle, generated by duplicating the entire puzzle, rotated in each of the admissible orientations. Rotated pieces (and frames) are visualized by transparency.*

Proof. We prove the proposition in three main steps: we (i) consider each edge type (c, θ) separately and construct a polynomial system $\mathcal{P}_1$ that determines all admissible edge pairings; (ii) add constraints accounting for the geometry of puzzle pieces, thus achieving a polynomial system $\mathcal{P}_2$ which exactly determines the puzzle solutions; and (iii) show that $\mathcal{P}_3$, the polynomial representation of Proposition 4.2, is equivalent to $\mathcal{P}_2$ and is therefore a complete representation of the puzzle.

To prove the proposition we shall use complex numbers to represent two-dimensional coordinates, that is, we consider piece locations $t_1, \ldots, t_N \in \mathbb{C}$. This will both simplify notation and facilitate our derivations.

We begin by proving a simple lemma. Fix $v_1, \ldots, v_N \in \mathbb{C}$ and consider the polynomial system

$$\left\{ p_k(u_1, \ldots, u_N) = \sum_i u_i^k - \sum_i v_i^k = 0 \right\}_{k=1,\ldots,K} \tag{A.1}$$

in the variables $u_1, \ldots, u_N$.

Lemma A.1. *For $K \geq N$, the polynomial system (A.1) has exactly $N!$ solutions:*

$$(u_1, \ldots, u_N) = (v_{\sigma(1)}, \ldots, v_{\sigma(N)})$$

for all permutations σ of $1, \ldots, N$.

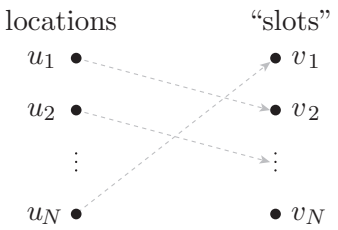


Figure 8. *Discrete piece location assignment as a bipartite matching problem.*

Proof. Denote by S the set of solutions of (A.1). By construction, $(v_{\sigma(1)}, \ldots, v_{\sigma(N)}) \in S$ for any permutation σ; therefore, $|S| \geq N!$.

On the other hand, if we assume that $K = N$, then Bézout's theorem [12, 44] asserts that

$$|S| \leq \prod_{i=1}^{N} deg\{p_i\} = \prod_{i=1}^{N} i = N!.$$

This bound clearly holds for $K > N$, thus concluding the proof. ∎

We shall refer to a system of the form (A.1) which satisfies the conditions of Lemma A.1 as a *bipartite system*, since it encodes all possible perfect matchings in the complete bipartite graph $\mathbf{K}_{N,N}$ whose vertices are the u_i's and v_i's; see Figure 8.

Consider the global coordinate $t_{i,j} = t_i + b_{i,j}$ of every edge element *independently*, and correspondingly, consider the exponential coordinates

$$T_{i,j} = e^{t_{i,j}} = e^{t_i + b_{i,j}} = e^{t_i} e^{b_{i,j}} = T_i B_{i,j}. \tag{A.2}$$

We first treat each edge type (c, θ) separately. Let

$$V(c,\theta) = \{(i,j) : c_{i,j} = c \text{ and } \theta_{i,j} = \theta\} \tag{A.3}$$

be the set of indices of (c, θ)-edges, and let

$$\overline{V}(c,\theta) = \{(i,j) : c_{i,j} = c \text{ and } \theta_{i,j} = \theta + \pi\} \tag{A.4}$$

be the set of indices of their complimentary $(c, \theta + \pi)$-edges. Consider the set of polynomial equations

$$\mathcal{P}_{(c,\theta)} = \left\{ \sum_{(i,j)\in V(c,\theta)} {T_{i,j}}^k = \sum_{(i,j)\in \overline{V}(c,\theta)} {T_{i,j}}^k \right\}_{k=1,\ldots,K(c,\theta)} \tag{A.5}$$

of degrees $1, 2, \ldots, K(c, \theta)$.

The sets $V(c, \theta)$ and $\overline{V}(c, \theta)$ are disjoint and $|V(c, \theta)| = |\overline{V}(c, \theta)|$; otherwise the puzzle is unsolvable. As such, we may consider the right-hand-side of (A.5) fixed and employ Lemma A.1.

Namely, for sufficiently many polynomials, specifically,

$$K(c,\theta) \geq |V(c,\theta)| = \#\left\{\text{edges of type } (c,\theta)\right\}, \tag{A.6}$$

the polynomial system (A.5) is a bipartite system; hence solutions of (A.5) are exactly the valid pairings of edges in $V(c,\theta)$ and $\overline{V}(c,\theta)$.

Next, we take the union of all such systems, over all types of edges, with $K(c,\theta)$ chosen according to (A.6):

$$\mathcal{P}_1 = \bigcup_{(c,\theta)} \mathcal{P}_{(c,\theta)}. \tag{A.7}$$

This is a polynomial system of equations, in variables $T_{i,j}$; in particular, *every edge* of the puzzle is constrained to satisfy the bipartite relation of the system it belongs to. That is, all edge elements of the puzzle are paired with appropriate edges.

However, employing independent coordinates for edge elements $T_{i,j}$ is insufficient. Since the internal geometry of a puzzle piece is not taken into account, a solution of $\mathcal{P}_1$ may imply an invalid puzzle assignment (e.g., two edge elements of the same piece can pair).

This, however, may be easily resolved by relating the coordinates of an edge to the piece it belongs to, namely, by adding the constraints $T_{i,j} = T_i B_{i,j}$ for all pieces. We therefore generate the augmented system

$$\mathcal{P}_2 = \mathcal{P}_1 \cup \left\{T_{i,j} = T_i B_{i,j}\right\}_{i,j}. \tag{A.8}$$

The solutions of $\mathcal{P}_2$ must therefore satisfy both the bipartite relations and the internal geometry of each piece, that is, a solution of $\mathcal{P}_2$ is a solution of the puzzle.

Last, we directly employ these linear constraints and substitute $T_{i,j} = T_i B_{i,j}$ into (A.5) to obtain the polynomial system

$$\widehat{\mathcal{P}}_{(c,\theta)} = \left\{ \sum_{(i,j)\in V(c,\theta)} {T_i}^k {B_{i,j}}^k = \sum_{(i,j)\in \overline{V}(c,\theta)} {T_i}^k {B_{i,j}}^k \right\}_{k=1,\ldots,K(c,\theta)}. \tag{A.9}$$

Using the notation $\alpha_i^{(k,c,\theta)} = \sum_j s_{i,j}(c,\theta) B_{i,j}^k$ of section 4.2, this can be equivalently written as

$$\widehat{\mathcal{P}}_{(c,\theta)} = \left\{ \sum_i \alpha_i^{(k,c,\theta)} T_i^k = 0 \right\}_{k=1,\ldots,K(c,\theta)}. \tag{A.10}$$

Again, taking the union over all types of edges yields the polynomial system

$$\mathcal{P}_3 = \bigcup_{(c,\theta)} \widehat{\mathcal{P}}_{(c,\theta)} \tag{A.11}$$

with $K(c,\theta)$'s chosen as before.

Notice that by construction the polynomial systems $\mathcal{P}_3$ and $\mathcal{P}_2$ are equivalent. Therefore, it is guaranteed that a solution of $\mathcal{P}_3$ corresponds to a valid solution of the puzzle. ∎

Appendix B. Solution of two-dimensional puzzles. The presentation in section 5 was simplified for notational purposes by considering one-dimensional puzzles (i.e., coordinates over $\mathbb{R}$). We now provide the technical details required for adapting these results to the case of two-dimensional puzzles. Similar modifications can be applied to address puzzles of arbitrary dimension, thereby completing the discussion of section 6.1 regarding puzzles in higher dimensions.

In section 5.2.1 the requirement that X is a Vandermonde matrix needs to be replaced with a multivariate analogue. Recall that in the two-dimensional case the variable associated with the ith piece is a 2-vector $T_i = [T_{ix}\ T_{iy}]^T$. Correspondingly, X in the optimization problem (5.3) should be of the form

$$\text{(B.1)}\qquad \begin{bmatrix} 1 & T_{1x} & T_{1y} & T_{1x}^2 & T_{1x}T_{1y} & T_{1y}^2 & \cdots & T_{1y}^K \\ 1 & T_{2x} & T_{2y} & T_{2x}^2 & T_{2x}T_{2y} & T_{2y}^2 & \cdots & T_{2y}^K \\ 1 & T_{3x} & T_{3y} & T_{3x}^2 & T_{3x}T_{3y} & T_{3y}^2 & \cdots & T_{3y}^K \\ \vdots & \vdots & \vdots & \ddots & \vdots & & & \\ 1 & T_{Nx} & T_{Ny} & T_{Nx}^2 & T_{Nx}T_{Ny} & T_{Ny}^2 & \cdots & T_{Ny}^K \end{bmatrix}.$$

That is, a matrix whose ith row entries are all the monomials in T_i of total degree up to K.

Similarly, in section 5.2.2 the requirement that each matrix Z_i is a Hankel matrix is replaced by constraining Z_i to be of the form corresponding to the outer product of all univariate monomials in the entries of T_i of degree up to K, namely, each Z_i should be of the form

$$\text{(B.2)}\qquad \begin{bmatrix} 1 \\ \hdashline T_{ix} \\ \vdots \\ T_{ix}^K \\ \hdashline T_{iy} \\ \vdots \\ T_{iy}^K \end{bmatrix} \begin{bmatrix} 1 \\ \hdashline T_{ix} \\ \vdots \\ T_{ix}^K \\ \hdashline T_{iy} \\ \vdots \\ T_{iy}^K \end{bmatrix}^T = \left[\begin{array}{c:ccc:ccc} 1 & T_{ix} & \cdots & T_{ix}^K & T_{iy} & \cdots & T_{iy}^K \\ \hdashline T_{ix} & T_{ix}^2 & & T_{ix}^{K+1} & T_{ix}T_{iy} & \cdots & T_{ix}T_{iy}^K \\ \vdots & & \iddots & & \vdots & & \vdots \\ T_{ix}^K & T_{ix}^{K+1} & \cdots & T_{ix}^{2K} & T_{ix}^K T_{iy} & \cdots & T_{ix}^K T_{iy}^K \\ \hdashline T_{iy} & T_{ix}T_{iy} & & T_{ix}^K T_{iy} & T_{iy}^2 & \cdots & T_{iy}^{K+1} \\ \vdots & & \iddots & & \vdots & & \vdots \\ T_{iy}^K & T_{ix}T_{iy}^K & \cdots & T_{ix}^K T_{iy}^K & T_{iy}^{K+1} & \cdots & T_{iy}^{2K} \end{array}\right].$$

Note that the main diagonal blocks of Z_i are now Hankel matrices; however, the entire matrix Z_i is no longer Hankel.

REFERENCES

[1] F. A. Andaló, G. Taubin, and S. Goldenstein, *Solving image puzzles with a simple quadratic programming formulation*, in Proceedings of SIBGRAPI, IEEE, 2012, pp. 63–70.

[2] E. D. Andersen and K. D. Andersen, *The MOSEK Interior Point Optimization for Linear Programming: An Implementation of the Homogeneous Algorithm*, Kluwer Academic Publishers, Norwell, MA, 1999, pp. 197–232.

[3] G. Birkhoff, *Three observations on linear algebra*, Univ. Nac. Tucumán. Rev. A, 5 (1946), pp. 147–151.
[4] B. J. Brown and C. Toler-Franklin, *A system for high-volume acquisition and matching of fresco fragments: Reassembling Theran wall paintings*, in Proceedings of SIGGRAPH, ACM, 2008.
[5] B. Buchberger, *A theoretical basis for the reduction of polynomials to canonical forms*, SIGSAM Bull., 10 (1976), pp. 19–29.
[6] E. J. Candes, M. B. Wakin, and S. P. Boyd, *Enhancing sparsity by reweighted l_1 minimization*, J. Fourier Anal. Appl., 14 (2008), pp. 877–905.
[7] T. S. Cho, S. Avidan, and W. T. Freeman, *A probabilistic image jigsaw puzzle solver*, Evaluation, (2010), pp. 1–8.
[8] T. S. Cho, M. Butman, S. Avidan, and W. T. Freeman, *The patch transform and its applications to image editing*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.
[9] T. S. Cho, S. Avidan, and W. T. Freeman, *The patch transform*, IEEE Trans. Pattern Anal. Mach. Intell., 32 (2010), pp. 1489–1501.
[10] M. G. Chung, M. M. Fleck, and D. A. Forsyth, *Jigsaw puzzle solver using shape and color*, in Proceedings of the International Conference on Signal Processing, Vol. 2, IEEE, 1998, pp. 877–880.
[11] A. M. Cohen, H. Cuypers, and H. Sterk, *Some Tapas of Computer Algebra*, Springer, Berlin, 1999.
[12] J. L. Coolidge, *A Treatise on Algebraic Plane Curves*, Dover, Mineola, NY, 2004.
[13] D. A. Cox, J. B. Little, and D. O'Shea, *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative algebra*, Springer, New York, 2007.
[14] H. C. da. G. Leito and J. Stolfi, *Automatic Reassembly of Irregular Fragments*, Technical report IC-98-06, Institute of Computing, University of Campinas, Brazil, 1998.
[15] J. Dattorro, *Convex Optimization & Euclidean Distance Geometry*, Meboo Publishing USA, 2005.
[16] E. D. Demaine and M. L. Demaine, *Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity*, Graphs Combin., 23 (2007), pp. 195–203.
[17] E. D. Demaine and R. A. Hearn, *Playing games with algorithms: Algorithmic combinatorial game theory*, in Proceedings of the 26th symposium on Mathematical Foundations in Computer Science, Lecture Notes in Comput. Sci., 2136, Springer, New York, pp. 18–32.
[18] E. S. Deutsch and C. Hayes, Jr., *A heuristic solution to the tangram puzzle*, Mach. Intell., 7 (1972), pp. 205–240.
[19] H. Freeman and L. Gardner, *Apictorial jigsaw puzzles: The computer solution of a problem in pattern recognition*, IEEE Trans. Electronic Computers, EC-13 (1964), pp. 118–127.
[20] A. C. Gallagher, *Jigsaw puzzles with pieces of unknown orientation*, in proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2012.
[21] *Gurobi Optimizer Reference Manual*, Gurobi Optimization, 2014.
[22] E. Justino, L. S. Oliveira, and C. Freitas, *Reconstructing shredded documents through feature matching*, Forensic Sci. Internat., 160 (2006), pp. 140–147.
[23] D. Koller and M. Levoy, *Computer-aided reconstruction and new matches in the Forma Urbis Romae*, in Bullettino Della Commissione Archeologica Comunale di Roma, 2006, pp. 103–125.
[24] D. A. Kosiba, *An automatic jigsaw puzzle solver*, in Proceedings of the IEEE International Conference on Pattern Recognition, 1994.
[25] B. Kulis, A. C. Surendran, and J. C. Platt, *Fast low-rank semidefinite programming for embedding and clustering*, in Proceedings of AISTATS, 2007, pp. 235–242.
[26] M. Laurent, *The approach of moments for polynomial equations*, in Semidefinite, Cone and Polynomial Optimization, Internat. Ser. Oper. Res. Management Sci. 166, Springer, New York, 2012, pp. 25–60.
[27] H. C. G. Leitão and J. Stolfi, *Measuring the information content of fracture lines*, Int. J. Comput. Vis., 65 (2005), pp. 163–174.
[28] M. Levison, *The computer in literary studies*, in Machine Translation, North-Holland, Amsterdam, 1967, pp. 173–194.
[29] T. Li, *Numerical solution of multivariate polynomial systems by homotopy continuation methods*, Acta Numeri., 6 (1997), pp. 399–436.
[30] H. Liu, S. Cao, and S. Yan, *Automated assembly of shredded pieces from multiple photos*, IEEE Trans. Multimedia Processing, 13 (2010), pp. 358–363.

[31] T. Lozano-Pérez and C. Wang, *Determining Molecular Conformation from Distance or Density Data*, Ph.D. thesis, Massachusetts Institute of Technology, 2000.
[32] M. Makridis and M. Papamarkos, *A new technique for solving a jigsaw puzzle*, in Proceedings of the IEEE Conference on Image Processing, 2006, pp. 2001–2004.
[33] W. Marande and G. Burger, *Mitochondrial DNA as a genomic jigsaw puzzle*, Science, 318 (2007), p. 415.
[34] M. A. O. Marques and C. O. A. Freitas, *Reconstructing strip-shredded documents using color as feature matching*, in Proceedings of the 2009 ACM symposium on Applied Computing, 2009, pp. 893–894.
[35] K. Mitra, S. Sheorey, and R. Chellappa, *Large-scale matrix factorization with missing data under additional constraints*, in NIPS 2010, Adv. Neural Inf. Process. Sys. 23, MIT Press, Cambridge, MA, 2010, pp. 1651–1659.
[36] T. Nielsen, P. Drewsen, and K. Hansen, *Solving jigsaw puzzles using image features*, Pattern Recognition Letters, 29 (2008), pp. 1924–1933.
[37] K. Oflazer, *Solving tangram puzzles: A connectionist approach*, International J. Intelligent Systems, 8 (1993), pp. 603–616.
[38] R. Orsi, U. Helmke, and J. B. Moore, *A Newton-like method for solving rank constrained linear matrix inequalities*, Automatica, 42 (2006), pp. 1875–1882.
[39] D. Pomeranz, M. Shemesh, and O. Ben-Shahar, *A fully automated greedy square jigsaw puzzle solver*, in Proceedings of Computer Vision and Pattern Recognition, IEEE, 2011, pp. 9–16.
[40] B. Recht, M. Fazel, and P. A. Parrilo, *Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization*, SIAM Rev., 52 (2010), pp. 471–501.
[41] D. Sholomon, O. David, and N. S. Netanyahu, *A genetic algorithm-based solver for very large jigsaw puzzles*, in Proceedings of CVPR, IEEE, 2013, pp. 1767–1774.
[42] A. J. Sommese and C. W. Wampler, *The Numerical Solution of Systems of Polynomials Arising in Engineering and Science*, World Scientific, River Edge, NJ, 2005.
[43] K. Son, J. Hays, and D. B. Cooper, *Solving square jigsaw puzzles with loop constraints*, in Proceedings of the European Conference on Computer Vision, 2014.
[44] F. Sottile, *From enumerative geometry to solving systems of polynomial equations*, in Computations in Algebraic Geometry with Macaulay 2, ACM, New York, 2002, pp. 101–128.
[45] B. Sturmfels, *Solving Systems of Polynomial Equations*, AMS, Providence, RI, 2002.
[46] R. Tybon, *Generating Solutions to the Jigsaw Puzzle Problem*, Ph.D. thesis, Griffith University, Australia, 2004.
[47] A. D. Williams, *The Jigsaw Puzzle: Piecing Together a History*, Berkley Books, New York, 2004.
[48] F.-H. Yao and G.-F. Shao, *A shape and image merging technique to solve jigsaw puzzles*, Pattern Recognition Lett., 24 (2003), pp. 1819–1835.
[49] Y. Zhao and M. Su, *A puzzle solver and its application in speech descrambling*, in Proceedings of the Annual Conference on International Conference on Computer Engineering and Applications, 2007, pp. 171–176.