Multi-Phase Relaxation Labeling for Square Jigsaw Puzzle Solving

Ben Vardi[1][a], Alessandro Torcinovich[2][b], Marina Khoroshiltseva[2,3][c], Marcello Pelillo[2,4][d], Ohad Ben-Shahar[1][e]

[1]*Ben-Gurion University of the Negev, Be'er-Sheva, Israel*

[2]*Ca' Foscari University of Venice, Venice, Italy*

[3]*Istituto Italiano di Tecnologia, Genoa, Italy*

[4]*European Centre of Living Technology, Venice, Italy*

{*benva@post, ben-shahar@cs*}*.bgu.ac.il,* {*ale.torcinovich, m.khoroshiltseva, pelillo*}*@unive.it*

Keywords: Puzzle Solving, Square Jigsaw Puzzles, Relaxation Labeling.

Abstract: We present a novel method for solving square jigsaw puzzles based on global optimization. The method is fully automatic, assumes no prior information, and can handle puzzles with known or unknown piece orientation. At the core of the optimization process is nonlinear relaxation labeling, a well-founded approach for deducing global solutions from local constraints, but unlike the classical scheme here we propose a multi-phase approach that guarantees convergence to feasible puzzle solutions. Next to the algorithmic novelty, we also present a new compatibility function for the quantification of the affinity between adjacent puzzle pieces. Competitive results and the advantage of the multi-phase approach are demonstrated on standard datasets.

1 INTRODUCTION

The jigsaw puzzle game is a well-known and time-honored pastime for children and adults. Given n non-overlapping image fragments, commonly referred to as puzzle pieces, the goal is to assemble them into a coherent visual image, preferably reconstructing the original (possibly unknown) image itself. Although large jigsaw puzzles are successfully solved by humans, the problem posed by this popular game is rather difficult, as it was shown to be NP-complete (Demaine and Demaine, 2007). Algorithmic capability to solve jigsaw puzzles has important applications in several fields, such as information security (Zhao, Su, Chou and Lee, 2007), assembly of shredded documents and photos (Deever and Gallagher, 2012; Liu, Cao and Yan, 2011) and archaeology (Koller and Levoy, 2006).

A particular type of jigsaw puzzles that has raised interest in the scientific community deals with *square jigsaw puzzles* (e.g., Cho, Avidan and Freeman, 2010; Pomeranz, Shemesh and Ben-Shahar, 2011), where all pieces have identical square shape, geometric information is unavailable, and only pictorial information may be used for driving the solution. This is also why solving square jigsaw puzzles marks an upper bound of jigsaw puzzle reconstruction performance when geometric cues are diminishing. Such an understanding is a key step in developing solvers for real-world puzzles, in which both pictorial and geometric information are usually available in varying degrees.

Square jigsaw puzzle solvers may be distinguished by two criteria. The first is the evaluation method for potential piece matchings, which defines the compatibility of placing any two puzzle pieces in one of the possible spatial 4-neighborhood adjacency relations. The measure used for this evaluation is commonly referred to as *piece compatibility measure*. The second criterion is the algorithmic method used to process these compatibilities in order to infer a solution for the full puzzle. In other words, one may argue that square jigsaw puzzle solvers use local constraints (piece compatibilities) in order to deduce a global solution (piece placements that seek to restore the original image).

While several attempts were made to solve square jigsaw puzzles and impressive results were achieved (e.g., Pomeranz et al., 2011; Andaló, Taubin and Goldenstein, 2016; Son, Hays and Cooper, 2018), most of the suggested algorithms use greedy meth-

[a] https://orcid.org/0000-0002-6950-8297
[b] https://orcid.org/0000-0001-8110-1791
[c] https://orcid.org/0000-0003-0424-0661
[d] https://orcid.org/0000-0001-8992-9243
[e] https://orcid.org/0000-0001-5346-152X

ods rather than global optimization methods. Nevertheless, global optimization algorithms are generally preferred over greedy algorithms for their ability to produce solutions which simultaneously take into consideration all the local potential piece matchings. Moreover, global optimization approaches may often facilitate additional mathematical analysis (Floudas, 2013) and thus a better understanding of the dynamics of the problem and solution process.

A well-founded global optimization approach that handles problems where local constraints are given and global interpretation should be found is *relaxation labeling* (RL). The approach, first introduced by Rosenfeld, Hummel and Zucker (1976), is a class of iterative procedures for refining label assignments based on contextual constraints. RL has been applicable in solving many global optimization problems, most commonly from the computer vision field (e.g., Zucker, Hummel and Rosenfeld, 1977). Although iterative, and reminiscent of gradient ascent, it can be formulated without any step size (Pelillo, 1997; Rosenfeld et al., 1976) while guaranteeing the optimization of its objective function (Hummel and Zucker, 1983; Pelillo, 1997).

Recently, preliminary results of using RL for square jigsaw puzzle solving (Khoroshiltseva et al., 2021) demonstrated the potential of this approach, but nevertheless suffered from several limitations such as occasional failures to converge to a feasible solution or being limited to puzzles in which piece orientations are known (known as Type 1 puzzles; Gallagher, 2012). In this work we further explore the potential of RL for square jigsaw puzzle solving and present a new RL algorithm that does guarantee feasible solutions, performs better than the preliminary approach, and handles both Type 1 but also Type 2 puzzles (whose piece orientations are *unknown*). In addition to the algorithmic novelty, another contribution of this paper is the proposal of a new piece compatibility measure.

2 RELATED WORK

2.1 Puzzle Solving

The topic of puzzle solving has long been of interest in the scientific community. The first to investigate puzzles were Freeman and Garder (1964), focusing on the geometric information of irregularly shaped puzzle pieces to suggest the first computational *apictorial* solver. Three decades later, Kosiba, Devaux, Balasubramanian, Gandhi and Kasturi (1994) were the first to also consider *pictorial* puzzles in their classical "toy" jigsaw puzzle solver.

While the effort to combine pictorial and geometric information was followed also by others, the line of research that has been dominating the literature focused on square jigsaw puzzles in which all pieces have an identical square shape, with puzzle dimensions being often known. While the objective of all different square jigsaw puzzle solvers is the same, solution approaches vary greatly. A first notable attempt was made by Cho et al. (2010) who employed loopy belief propagation algorithm for Type 1 puzzles. Although being able to solve the biggest puzzle at the time, their algorithm was only semi-automatic, requiring at least some ground truth information. A year later, Pomeranz et al. (2011) published the first fully automatic solver, requiring no prior knowledge. Their iterative greedy Type 1 solver is based on three steps: pieces placement according to the piece compatibility measure, segmentation of the proposed solution into coherently assembled segments, and shifting segments to improve the current solution. Despite being simpler, this approach provided not only a leap in performance, but also the ability to solve puzzles an order of magnitude larger than before.

Following Pomeranz et al. (2011), Andaló et al. (2012; 2016) were the first to formulate the problem as a global optimization problem using a constrained quadratic objective function. Gallagher (2012) used a tree-based greedy algorithm inspired by the well-known Kruskal's algorithm for finding minimum spanning trees. Gallagher (2012) was the first to introduce and solve Type 2 puzzles, namely puzzles in which piece orientations are unknown. Shortly after, Sholomon, David and Netanyahu (2013) devised a genetic algorithm-based solver for Type 1 puzzles, and later extended it for Type 2 puzzles (Sholomon, David and Netanyahu, 2014).

A different approach for the problem was proposed by Son et al. (2014; 2018) who devised a Type 1 and 2 solver that utilizes loops of pieces that form consistent cycles. More recently, Huroyan, Lerman and Wu (2020) proposed a Type 2 method based on recovering piece orientations by estimating the graph connection Laplacian. Of special importance to our work is the global optimization solver by Khoroshiltseva et al. (2021), which is based on RL. Due to its relevance, we dedicate Section 2.3 to that solver.

Several neural network methods have been suggested to solve square jigsaw puzzles, most of them applicable only for small puzzles. Paumard, Picard and Tabia (2020) offered a convolutional neural network-based solver that predicts neighboring relations in 9 pieces puzzles. Li, Liu, Wang, Liu and Zeng (2021) devised a generative adversarial network

(GAN) pipeline for training a classification network to predict pieces permutation in puzzles with up to 16 pieces. Talon, Del Bue and James (2022) suggested another GAN-based approach that is capable of solving bigger puzzles, but suffers from diminishing accuracy as puzzle size gets bigger.

Along with the different algorithmic approaches to the square puzzle problem, different piece compatibility measures have been also developed over time. Such measures define the compatibility of placing any two puzzle pieces in any of the 4-neighborhood adjacency relations. They are most commonly defined as a function $C(b_i, b_j, R)$ that specifies how compatible it is to place piece b_j in relation $R \in \{right, down, left, up\}$ to piece b_i. In virtually all solvers, piece compatibilities are based on a more fundamental measure of piece dissimilarity that usually compares the pictorial information along the boundaries of pieces. This measure is later mapped to a score of compatibility.

2.2 Relaxation Labeling

Relaxation labeling (RL) is a class of processes for ambiguity reduction of labeling assignments, done by iteratively refining assignments based on contextual information (Rosenfeld et al., 1976). Among these processes, in the current work we focus on *nonlinear relaxation labeling*, where problems involve four elements:

(1) Set of n objects $B = \{b_1, ..., b_n\}$.
(2) Set of m possible labels $\Lambda = \{\lambda_1, ..., \lambda_m\}$.
(3) Four-dimensional matrix of *compatibility coefficients* $\{r_{ij}(\lambda, \mu)\} \in \mathbb{R}^{n^2 m^2}$, expressing how compatible are the two hypotheses of "labeling object b_i with label λ" and "labeling object b_j with label μ".
(4) *Initial labeling* $\bar{p}^{(0)} \in \mathbb{R}^{nm}$, containing prior probabilities for labeling each object b_i with each label λ.

The purpose of nonlinear relaxation labeling (henceforth referred to as just RL) is to provide a consistent labeling assignment $\bar{p} \in \mathbb{R}^{nm}$ that represents a viable solution to the problem. An assignment is a set of probabilities for assigning each label λ to each object b_i. While the initial labeling $\bar{p}^{(0)}$ is already qualified as such, it may not be "good enough" in terms of satisfying the local constraints represented by the compatibility coefficients. An assignment that does satisfy these constraints is called *consistent*, and RL is the optimization process that provides such consistent assignments given the four elements above.

A labeling assignment $\bar{p}$, usually referred to just as labeling, is represented in the current work as *labeling matrix* of n rows and m columns. The probability of object b_i to be labeled with label λ is notated as $p_i(\lambda)$, and is indicated by the matrix entry in row i and column λ. Since each row i represents the probabilities of all label assignments to object b_i, the space of all possible labelings is

$$\mathbb{K} = \left\{ \bar{p} \in \mathbb{R}^{nm} \mid p_i(\lambda) \geq 0,\ \forall i \in B, \lambda \in \Lambda \right. \\ \left. \text{and } \sum_{\lambda=1}^{m} p_i(\lambda) = 1,\ \forall i \in B \right\}. \tag{1}$$

A particular important subset of $\mathbb{K}$ is the set of all possible *unambiguous labelings*:

$$\mathbb{K}^* = \left\{ \bar{p} \in \mathbb{K} \mid p_i(\lambda) \in \{0,1\},\ \forall i \in B, \lambda \in \Lambda \right\}, \tag{2}$$

where each object is assigned one label with full confidence.

In order to refine the initial (likely inconsistent) labeling, RL algorithm involves the computation of two terms in each iteration k of the process, the *support* and a new more "refined" labeling. While not necessarily the only way to do so, we adapt the common terms from the RL literature (Pelillo, 1997; Rosenfeld et al., 1976):

(1) The support matrix $\bar{q}^{(k)} \in \mathbb{R}^{nm}$ represents the contextual "support" that each object b_i gets for labeling it with each label λ. This matrix, referred to as *support*, is computed by a *support function*:

$$q_i^{(k)}(\lambda) = \sum_{j=1}^{n} \sum_{\mu=1}^{m} r_{ij}(\lambda, \mu) p_j^{(k)}(\mu). \tag{3}$$

(2) The refined labeling $\bar{p}^{(k+1)}$ that represents the new "better" labeling, is computed by a *nonlinear update rule*:

$$p_i^{(k+1)}(\lambda) = \frac{p_i^{(k)}(\lambda) q_i^{(k)}(\lambda)}{\sum_{\mu=1}^{m} p_i^{(k)}(\mu) q_i^{(k)}(\mu)}. \tag{4}$$

The entire algorithm is best viewed as a continuous mapping of $\mathbb{K}$ onto itself through Eq. 4, and it is expected to converge to a fixed point if applied successively. Theoretically, such a fixed point iteration stops when $p_i^{(k+1)}(\lambda) = p_i^{(k)}(\lambda),\ \forall i, \lambda$ for some iteration k. In practice, we may (heuristically) determine convergence if the increase of the global objective function (see below in Eq. 5) in two successive iterations is smaller than a given threshold ε.

The theoretical foundations for RL were laid by Hummel and Zucker (1983) who have proved several useful theoretical properties for an alternative RL process. Pelillo (1997) later proved similar dynamics for the original RL algorithm from Rosenfeld

et al. (1976). Of particular relevance to us is that under symmetric non-negative compatibility coefficients (i.e., $r_{ij}(\lambda,\mu) = r_{ji}(\mu,\lambda) \geq 0$, $\forall i,j,\lambda,\mu$), each iteration of the process strictly increases a global consistency measure known as *average local consistency* (ALC; Hummel and Zucker, 1983):

$$A(\bar{p},\bar{q}) = \sum_{i=1}^{n}\sum_{\lambda=1}^{m} p_i(\lambda)q_i(\lambda)\,. \tag{5}$$

Both contributions (Hummel and Zucker, 1983; Pelillo, 1997) showed that their respective process eventually approaches and converges to a consistent solution near the initial labeling assignment. Importantly, however, unlike other optimization algorithms such as gradient ascent, the original iteration due to Rosenfeld et al. (1976) does so without the need to determine a step size.

2.3 Puzzle Solving with "Balanced" RL

To our best knowledge, the only work that employs RL for square jigsaw puzzle solving is by Khoroshiltseva et al. (2021). In their formulation for Type 1 puzzles, the set of objects is the set of puzzle pieces and the set of labels is the set of possible positions on the puzzle grid. In these settings $n = m$, so the space $\mathbb{K}$ is the space of stochastic matrices, and the space $\mathbb{K}^*$ is the space of binary stochastic matrices. As no better prior information can be assumed, the initial labeling is a uniform distribution of labels across each object. The compatibility coefficients $r_{ij}(\lambda,\mu)$ are set to piece compatibility scores $C(b_i,b_j,R)$ for all pairs of different pieces b_i,b_j and positions λ,μ that represent neighboring positions on the puzzle grid in relation R. In all other cases, $r_{ij}(\lambda,\mu)$ is 0.

Unfortunately, applying the RL update rule from Eq. 4 using such definitions does not guarantee, and in fact often does not result in a *feasible* puzzle solution, defined as an assignment where each piece is placed at a unique position on the puzzle grid and all positions are occupied with some piece. Such feasible solutions, represented by *permutation labeling matrices*, are not always achieved because the theory just guarantees that the update rule will map a stochastic matrix into another stochastic matrix, with the additional hope that the process converges to an unambiguous assignment, namely a binary stochastic matrix. But even if the process converges to a binary stochastic matrix (as opposed to a non-binary one), this does not prohibit situations where two objects (pieces) obtain the same label (position), or that a given label (position) is not assigned to any object (piece). To remedy this situation, Khoroshiltseva et al. (2021) extended the update rule from Rosenfeld et al. (1976) with two versions of a matrix balancing component that seeks to map a stochastic labeling matrix to a *doubly stochastic* matrix. Since the latter are easier to binarize into permutation matrices, this should help the process converge to the space of all feasible puzzle solutions.

Khoroshiltseva et al. (2021) showed empirically that their approach can perform better than a plain RL without balancing. At the same time, while the method encourages feasible solutions, it cannot guarantee them. Testing on natural images indeed showed vulnerability to certain common conditions, and in particular the presence of constant pieces, namely puzzle pieces of identical appearance of some constant value throughout. Such failure cases can be explained by the fact that the rows of the labeling matrix that refer to constant pieces are identical in all steps of the process (both the update rule and balancing component map identical rows into identical rows), thus the process cannot converge to a permutation matrix.

In the current work we suggest an alternative and improved puzzle solver based on a novel multi-phase RL scheme. Our method is applicable for both Type 1 and 2 puzzles, addresses the aforementioned problems and facilitates further exploration of RL as a computational powerhouse for puzzle solving.

3 MULTI-PHASE RL PUZZLE SOLVERS

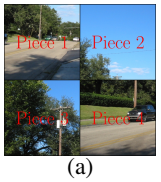
3.1 Puzzles with Known Piece Orientation

Type 1 puzzles, possibly the most elementary variation of the puzzle problem, constitute unordered set of square pieces whose orientation is given but their organization in an $N \times M$ array of positions is not. Treating this problem as an RL problem can be done in the following way, similar to Khoroshiltseva et al. (2021):

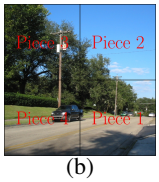
(1) The objects $B = \{b_1,\ldots,b_n\}$ are puzzle pieces.
(2) The labels $\Lambda = \{\lambda_1,...,\lambda_m\}$ are all possible positions (x_λ, y_λ) on the two-dimensional $N \times M$ puzzle grid.
(3) The compatibility coefficients are defined by

$$r_{ij}(\lambda,\mu) = r_{ij}((x_\lambda,y_\lambda),(x_\mu,y_\mu)) = \begin{cases} C(b_i,b_j,R), & \text{if } i \neq j \text{ and } (x_\lambda,y_\lambda),(x_\mu,y_\mu) \text{ are} \\ & \text{neighboring positions in relation } R, \\ 0, & \text{otherwise}\,, \end{cases} \tag{6}$$

where $R \in \{right, down, left, up\}$ represents a 4-neighborhood spatial relation and C is a non-



(a)



(b)

	(1,1)	(1,2)	(2,1)	(2,2)
Piece 1	0	0	0	1
Piece 2	0	1	0	0
Piece 3	1	0	0	0
Piece 4	0	0	1	0

(c)

Figure 1: Type 1 puzzle feasible labeling structure. (a) is a 2×2 puzzle and (b) is its perfect solution. This perfect solution is represented by the permutation labeling matrix (c). Piece IDs in red were added for demonstration purposes.

negative and symmetric piece compatibility function (see Section 3.3 for the function definition).

(4) The initial labeling $\bar{p}^{(0)}$ is the barycenter of the search space, i.e., $p_i^{(0)}(\lambda) = \frac{1}{m}$, $\forall i, \lambda$. With no prior information, this is the only rational choice one can make.

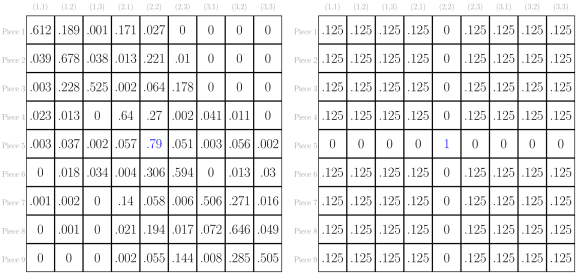
As mentioned above, a feasible Type 1 reconstruction is a solution in which each piece is placed at a unique position on the puzzle grid, and the space of such feasible labelings is the space of permutation matrices. Figure 1 exemplifies feasible labeling structure for the Type 1 problem.

Since C is non-negative and symmetric, the same applies to $r_{ij}(\lambda, \mu)$. Consequently, the process is guaranteed to strictly increase the ALC in each iteration, and to converge to a consistent assignment (Pelillo, 1997). For puzzle solving this turns the ALC to an intuitive objective function and the relaxation process to one that seeks to maximize the sum of compatibilities between all adjacent pieces, similar in spirit to other global optimizers for puzzle solving (Andaló et al., 2016; Sholomon et al., 2013). For the justification of this claim please refer to the supplementary material in the project webpage (*https://icvl.cs.bgu.ac.il/multi-phase-rl-for-jigsaw-puzzle-solving*).

3.1.1 Type 1 Solver

With the elements of the RL process now set for puzzle solving, and the observation that preliminary attempts (Khoroshiltseva et al., 2021) could not guarantee feasible solutions, we suggest an alternative algorithm that does guarantee this property. Central to our approach is a novel multi-phase scheme where each phase uses RL to determine the position of a *single* puzzle piece at a *non-occupied* position. As explained below, this enables to change the structure of the labeling assignment towards permutation labeling, as required for feasible solutions.

More specifically, in each phase of the algorithm we let the standard RL algorithm (Pelillo, 1997; Rosenfeld et al., 1976) run and intervene only when the process is confident enough about placing a certain piece b_i at some position λ, namely that the process converges to a consistent labeling and/or at least one label at one object satisfies $p_i(\lambda) \geq \alpha$, where α is a pre-defined threshold. This intervention marks the end of the current phase, and involves three steps:



	(1,1)	(1,2)	(1,3)	(2,1)	(2,2)	(2,3)	(3,1)	(3,2)	(3,3)
Piece 1	.612	.189	.001	.171	.027	0	0	0	0
Piece 2	.039	.678	.038	.013	.221	.01	0	0	0
Piece 3	.003	.228	.525	.002	.064	.178	0	0	0
Piece 4	.023	.013	0	.64	.27	.002	.041	.011	0
Piece 5	.003	.037	.002	.057	.79	.051	.003	.056	.002
Piece 6	0	.018	.034	.004	.306	.594	0	.013	.03
Piece 7	.001	.002	0	.14	.058	.006	.506	.271	.016
Piece 8	0	.001	0	.021	.194	.017	.072	.646	.049
Piece 9	0	0	0	.002	.055	.144	.008	.285	.505

	(1,1)	(1,2)	(1,3)	(2,1)	(2,2)	(2,3)	(3,1)	(3,2)	(3,3)
Piece 1	.125	.125	.125	.125	0	.125	.125	.125	.125
Piece 2	.125	.125	.125	.125	0	.125	.125	.125	.125
Piece 3	.125	.125	.125	.125	0	.125	.125	.125	.125
Piece 4	.125	.125	.125	.125	0	.125	.125	.125	.125
Piece 5	0	0	0	0	1	0	0	0	0
Piece 6	.125	.125	.125	.125	0	.125	.125	.125	.125
Piece 7	.125	.125	.125	.125	0	.125	.125	.125	.125
Piece 8	.125	.125	.125	.125	0	.125	.125	.125	.125
Piece 9	.125	.125	.125	.125	0	.125	.125	.125	.125

Figure 2: The first phase of our Type 1 solver on a 3×3 puzzle, done with $\alpha = 0.7$. The RL phase begins from the initial labeling and terminates once some labeling value crosses α (left matrix, obtained for $p_5((2,2))$ after 4 RL iterations). On the right is the labeling after anchoring piece b_5 to position $(2,2)$.

(1) *Anchoring* piece b_i to position λ and disallowing placing it from now on at any other position. This is done by setting $p_i(\lambda) = 1$ and $p_i(\mu) = 0, \forall \mu \neq \lambda$.

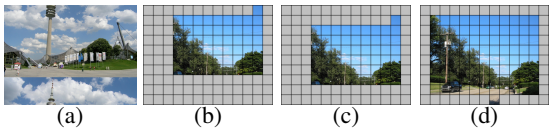
(2) Preventing the future placement of other pieces at position λ. This is done by setting $p_j(\lambda) = 0, \forall j \neq i$.

(3) Resetting all entries $p_j(\mu)$ that involve pieces b_j and positions μ for which past phases have not made a decision yet. This step is done by setting these entries to the barycenter of the remaining search *subspace*, i.e. to the reciprocal of the number of remaining unassigned objects/positions. This allows subsequent phases to start afresh without bias from prior phases.

Figure 2 demonstrates these steps after the completion of the first RL phase on a 3×3 puzzle.

The manipulated labeling serves as the initial labeling for the next phase of the algorithm, which anchors another piece at another position. The algorithm finally terminates after anchoring all pieces, namely after n phases. Importantly, although no special measures are taken, the process does not modify entries of pieces and positions for which previous phases have already made a decision since Eq. 4 does not (indeed, cannot) modify binary (i.e., 0 or 1) probabilities. Thanks to this property, each phase takes a decision about a single piece and position by binarizing their respective row and column in the labeling matrix, all while utilizing the information obtained in previous phases. This way, each phase updates the labeling so it becomes progressively "closer" to a permutation labeling. After n phases convergence to a permutation labeling is guaranteed.

Operating as described above, the multi-phase



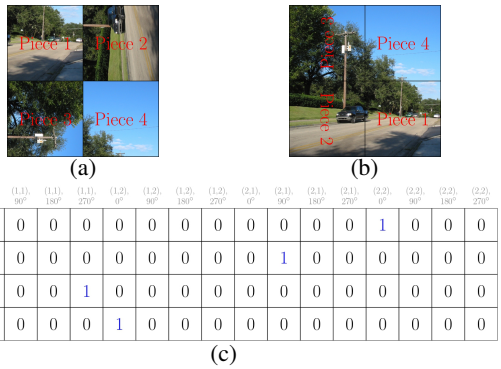
(a) (b) (c) (d)

Figure 3: The shifted reconstruction phenomenon and its solution. (a) is a reconstruction where rows are shifted. To prevent this, we translate the whole anchored block when the anchoring of some piece reaches the edge of the puzzle grid, as in (b) (gray areas represent non-occupied positions). At this point we translate the block downwards to obtain (c) and allow subsequent phases to grow in all directions. (d) is a case a translation will not necessarily benefit the situation as translating the block upwards will also limit the growth. At this point we branch the computation to try both options, translating the block upwards or leaving the block as is.

solver will normally attach new pieces to the block of pieces already anchored in the past. However, there is no guarantee for this behavior. To prevent the solver from generating isolated "islands" of solutions that would be difficult to "stitch" together properly across the puzzle grid, we allow anchoring only at positions adjacent to previously anchored ones, thus guaranteeing a single contiguous block of anchored pieces. Moreover, if the α threshold is met simultaneously by more than one combination of piece b_i and position λ, we select a single combination by an internal sorting of relevant combinations. Please see the supplementary material for more details.

The algorithm just described guarantees *feasible* solutions although, of course, it cannot guarantee *correct* solutions. In practice, many solutions are perfect (or near perfect) up to a circular shift of the rows and/or columns, as shown in Figure 3(a). This phenomenon is best explained as a lack of awareness of the solver to the puzzle dimensions, as it emerges from the likely event of assigning the very first piece to a wrong position in the puzzle grid. This problem was also observed in past (non-RL) solvers (e.g., Pomeranz et al., 2011), but is easy to alleviate by incorporating such awareness in the process. In particular, each time a phase is completed, if the anchored block abuts the edge of the puzzle grid, we translate it one column or row inside (see Figures 3(b)-3(c)). This translation is done by applying a proper transformation on the labeling matrix, and it allows subsequent phases to again grow the block in all directions.

Of course, these translations of the anchored block will cease once the solution is short of one row or column to the vertical or horizontal dimension of the puzzle (e.g., Figure 3(d)). At this point it is yet impossible to know what side the coming phases should grow the last row or column and thus we branch the algorithm to try both options. Doing this for both the vertical and horizontal growth, we end up with 4 final reconstructions, from which the final solution is the one with the largest ALC.



(a) (b)

	(1,1), 0°	(1,1), 90°	(1,1), 180°	(1,1), 270°	(1,2), 0°	(1,2), 90°	(1,2), 180°	(1,2), 270°	(2,1), 0°	(2,1), 90°	(2,1), 180°	(2,1), 270°	(2,2), 0°	(2,2), 90°	(2,2), 180°	(2,2), 270°
Piece 1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
Piece 2	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
Piece 3	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Piece 4	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

(c)

Figure 4: Type 2 puzzle feasible labeling structure. (a) is a 2×2 puzzle and (b) is its perfect solution. The corresponding type 2 permutation labeling matrix is shown in (c).

3.2 Puzzles with Unknown Piece Orientation

Type 2 puzzles extend the elementary case to situations where piece orientation is unknown. Since the final orientation of the pieces needs to be determined by the process, the setup of the problem as an RL process requires some adjustments. As before, the objects $b_1, \ldots, b_n$ still represent the NM puzzle pieces. However, the labels $\lambda_1, ..., \lambda_m$ now need to represent all combinations of piece positions and orientations. Each label λ is a pair $((x_\lambda, y_\lambda), \theta_\lambda)$, where (x_λ, y_λ) is a position on the $N \times M$ grid and $\theta_\lambda \in \{0°, 90°, 180°, 270°\}$ is one of 4 possible piece orientations.

Clearly, the change to labels requires a corresponding adjustment to the compatibility coefficients:

$$r_{ij}(\lambda,\mu) = r_{ij}(((x_\lambda,y_\lambda),\theta_\lambda),((x_\mu,y_\mu),\theta_\mu)) = \begin{cases} C(\theta_\lambda(b_i),\theta_\mu(b_j),R), & \text{if } i \neq j \text{ and } (x_\lambda,y_\lambda),(x_\mu,y_\mu) \\ & \text{are neighboring positions} \\ & \text{in relation } R, \\ 0, & \text{otherwise}, \end{cases} \tag{7}$$

where function $\theta_\lambda(b_i)$ represents the clockwise-rotation of piece b_i by θ_λ degrees.

Under this formulation several complications immediately arise. First, the labeling matrix is no longer square as $m = 4n$, and thus the solution *cannot* be a permutation matrix. Second, a feasible Type 2 reconstruction requires to place each piece in *some* orientation at a unique position on the puzzle grid. In fact, in order to represent feasible solutions, the labeling matrix should include a single 1 in each row (just as in Type 1 puzzles), but at the same time a single 1 in every group of 4 columns that represent the different orientations of the same position. A labeling for which the matrix representation meets this description

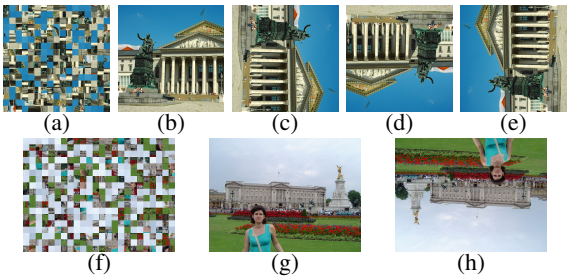


Figure 5: Possible reconstructions for Type 2 puzzles. (a) is a square-shaped puzzle, for which (b), (c), (d) and (e) are four perfect reconstructions. (f) is a rectangular-shaped puzzle, for which (g) and (h) are two perfect reconstructions.

is referred to as *type 2 permutation labeling* (see Figure 4). We note that similar to the Type 1 case, in the space of Type 2 feasible solutions the RL optimization seeks to maximize the sum of compatibilities between adjacent pieces.

To complete the description of Type 2 problems as RL processes, we need to define the initial labeling. The situation here is again somewhat more complicated since unlike for Type 1 puzzles, that have a single perfect reconstruction, any Type 2 puzzle has at least 2 perfect reconstructions due to global image rotation (see Figure 5). In order to encourage convergence to one of these solutions, the dynamics should be biased towards a specific solution rather than "favoring" all solutions equally. To address this, we set the initial labeling as the barycenter of the search space, while limiting some piece b_{i_1} to some orientation θ_1:

$$p_i(\lambda) = p_i(((x_\lambda, y_\lambda), \theta_\lambda)) = \begin{cases} \frac{1}{m}, & \text{if } i \neq i_1, \\ \frac{1}{n}, & \text{if } i = i_1, \theta_\lambda = \theta_1, \\ 0, & \text{if } i = i_1, \theta_\lambda \neq \theta_1 \,. \end{cases} \tag{8}$$

When puzzles are square in shape (i.e., $N = M$), the orientation θ_1 is chosen randomly among the 4 possible ones and serves as an "angular anchor" that pulls the entire solution towards the proper global orientation. Doing the same for rectangular $N \times M$ puzzles is problematic because the selected orientation of piece b_{i_1}, and thus the preferred global orientation of the solution, might conflict the rectangular dimensions of the puzzles. To cope with this problem we run the algorithm twice, first with an initial state derived from a randomly selected orientation θ_1, and once again with an initial state derived from $\theta_1 + 90°$. This ensures that one of the computations could converge to a correct reconstruction, which is identified and selected by its larger ALC. Finally, in order to maximize the influence of the bias obtained from the initial state, piece b_{i_1} is heuristically selected as the one that maximizes the sum of potential compatibilities in its 4-neighborhood.

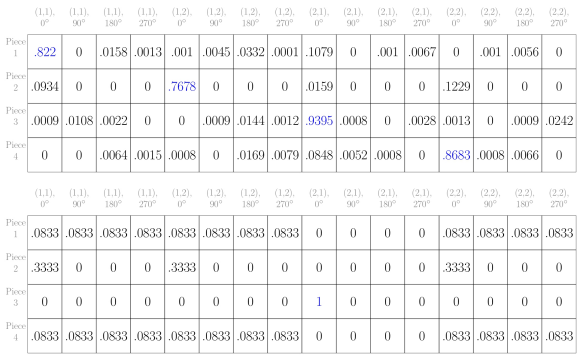


Figure 6: The first phase of our Type 2 solver on a 2×2 puzzle, with $\alpha = 0.7$. The RL phase begins from the initial labeling and terminates once labeling values cross α (top matrix, obtained after 3 RL iterations). The best suprathreshold entry at this point is $p_3(((2,1),0°))$. After anchoring piece b_3 to position $(2,1)$ at orientation $0°$ we obtain the bottom matrix. Note that b_2 is allowed only at orientation $0°$ since the initial labeling was set this way.

3.2.1 Type 2 Solver

Not unlike the Type 1 solver that directs the process to a permutation labeling, the Type 2 solver needs to lead the process to a type 2 permutation labeling. However, to "anchor" an entry $p_i(((x_\lambda, y_\lambda), \theta_\lambda))$ requires to revise the intervention at the end of each RL phase to manipulate the labeling in a slightly different way:

(1) Anchoring piece b_i to position (x_λ, y_λ) in orientation θ_λ and preventing its placement at any other position (regardless of orientation). Formally we set $p_i(((x_\lambda, y_\lambda), \theta_\lambda)) = 1$ and $p_i(\mu) = 0, \forall\, \mu \neq ((x_\lambda, y_\lambda), \theta_\lambda)$.

(2) Preventing a future placement of other pieces at position (x_λ, y_λ) regardless of their orientation. This is done by setting $p_j(((x_\lambda, y_\lambda), \theta)) = 0, \forall\, \theta, \;\; j \neq i$.

(3) Resetting entries $p_j(((x_\mu, y_\mu), \theta_\mu))$ that involve pieces b_j and positions (x_μ, y_μ) for which anchoring is yet made. This is done by setting these entries to the reciprocal of the number of combinations of unassigned positions and possible orientations (i.e., the number of unassigned positions multiplied by 4). If piece b_{i_1} (used for setting the initial labeling) is not anchored, we reset its values so only orientation θ_1 is allowed.

Figure 6 demonstrates the effect of these 3 steps after the completion of the first RL phase on a 2×2 Type 2 puzzle. The shifting of the anchored block at the end of each phase is handled as before.

3.3 Piece Compatibility Measure

At the foundations of any puzzle solver is some measure of affinity between puzzle pieces that allows to

quantify the quality of immediate neighborhood relationships. In our algorithm it is integrated in RL compatibility coefficients.

As mentioned earlier, piece compatibility measures are conventionally based on a more basic measure of piece dissimilarity. While various dissimilarity measures have been proposed, we use the Mahalanobis gradient dissimilarity proposed by Son et al. (2018). This dissimilarity measure evaluates the disagreement of placing two adjacent pieces by computing the Mahalanobis distances between the actual gradients and expected gradients with respect to the distribution of predicted gradients, where the gradients considered are both across and along the adjoining boundaries of pieces. Please refer to Son et al. (2018) for a full description.

Given the dissimilarity, it is possible to convert it to a compatibility function in various ways, the most immediate of which is simply negating the dissimilarities. Here we take a more elaborate approach that is based on the observation that our solver, and RL processes in general, perform better when the compatibility values are well spaced and the difference between compatible and incompatible values is significant. This was already observed in other solvers, and here we follow Andaló et al. (2016) who improved the basic idea from Pomeranz et al. (2011) and defined

$$C(b_i,b_j,R) = exp\left(-\varphi_{i,R}(j) - \frac{D(b_i,b_j,R)}{quartile(i,R)} \right), \quad (9)$$

where $quartile(i,R)$ is the quartile dissimilarity among all dissimilarities in relation R to piece b_i, and $\varphi_{i,R}(j)$ is the rank of the dissimilarity $D(b_i,b_j,R)$ in the increasingly ordered dissimilarities in relation R to piece b_i. The $quartile(i,R)$ term is included to consider the scattering of dissimilarity values (and not just the values themselves), and the $\varphi_{i,R}(j)$ rank term results in "spacing" the computed compatibility values.

Inspired by such ideas, we propose a new compatibility measure:

$$C(b_i,b_j,R) = \begin{cases} 1, & \text{if } D(b_i,b_j,R) = p_{avg}(i,R;k) = 0, \\ \left(1 - \frac{D(b_i,b_j,R)}{p_{avg}(i,R;k)}\right)^{\varphi_{i,R}(j)}, & \text{if } D(b_i,b_j,R) \le p_{avg}(i,R;k) \\ & \text{and } p_{avg}(i,R;k) > 0, \\ 0, & \text{otherwise}, \end{cases} \quad (10)$$

where $p_{avg}(i,R;k)$ is the average dissimilarity from the first value till the k percentile among all dissimilarities in relation R to piece b_i.

This measure offers two advantages over the measure from Eq. 9: (1) dissimilarities scattering information is determined by multiple values rather than one value (the quartile dissimilarity). Moreover, we introduce k as a parameter to avoid limitation to $k = 25$ (the quartile). (2) It better balances the $D(b_i,b_j,R)$ and $\varphi_{i,R}(j)$ terms since high $\varphi_{i,R}(j)$ does not necessarily guarantee low compatibility score.

An important challenge that most global optimization solvers face is due to *constant pieces*, or puzzle pieces that have the same constant color value throughout, which happens mostly in sky regions that turned white due to pixel saturation. Constant pieces maintain maximal compatibility with other constant pieces (and possibly with more pieces), thus it becomes difficult for the RL process to converge to a labeling that unambiguously determines their positions. To alleviate this, we slightly adjust the compatibility for puzzles with more than two constant pieces:

$$C'(b_i,b_j,R) = \begin{cases} max\{0,X\}, & \text{if there exists constant} \\ & \text{piece } b_k \text{ s.t. } C(b_i,b_k,R) = 1 \\ & \text{and } C(b_k,b_j,R) = 1, \\ C(b_i,b_j,R), & \text{otherwise}, \end{cases} \quad (11)$$

where $X \sim U(-4,1)$ is a random variable. This adjustment re-draws all maximal compatibilities for constant pieces (and after rectification set it to 0 with probability of 0.8), which sparsifies the compatibility coefficients and assists convergence.

4 EXPERIMENTAL RESULTS

To test the suggested puzzle solvers, we used the 20 puzzles of 432 pieces from the MIT data set proposed by Cho et al. (2010) and the 20 puzzles of 540 pieces from the McGill data set proposed by Pomeranz et al. (2011). All puzzles contain pieces of size of 28×28 pixels. We evaluated performance with the following three performance measures: (1) *Direct comparison* (DC; Cho et al., 2010) is the fraction of pieces that are in the same position (and orientation for Type 2 puzzles) in the assembly solution and the ground truth solution. (2) *Neighbor comparison* (NC; Cho et al., 2010) is the fraction of pairwise piece matchings that exist in the ground truth solution. (3) *Perfect reconstruction* (PR; Gallagher, 2012) is a binary indication of whether the assembly solution is identical to the ground truth. For an entire data set, PR becomes the number of perfectly solved puzzles.

The only random part in our solvers is the constant pieces compatibility adjustment (Eq. 11). Otherwise, solvers are fully deterministic for a given input. For this reason, the reported performance for puzzles with constant pieces is the average of 10 runs while the rest of the puzzles were solved once. In all experiments, the RL convergence threshold ε was set to 10^{-4}, while

Table 1: Comparison of Type 1 solvers.

Method	The MIT data set			The McGill data set		
	DC	NC	PR	DC	NC	PR
Andaló et al. (2016)	96%	95.6%	13	90.8%	95.3%	13
Gallagher (2012)	95.3%	95.1%	12	-	-	-
Paikin and Tal (2015)	96.2%	95.8%	13	93.2%	96.1%	13
Pomeranz et al. (2011)	94%	95%	13	83.5%	90.9%	9
Sholomon et al. (2013)	86.2%	96.2%	9	92.8%	96%	8
Son et al. (2018)	95.8%	95.6%	-	95.4%	97%	-
Khoroshiltseva et al. (2021)	79.1%	87.9%	6	25.6%	64.5%	0
Ours	95.5%	95.4%	13	62.9%	87.8%	6

Table 2: Comparison of Type 2 solvers.

Method	The MIT data set			The McGill data set		
	DC	NC	PR	DC	NC	PR
Gallagher (2012)	82.2%	90.4%	9	-	-	-
Huroyan et al. (2020)	94.8%	95.2%	13	88.3%	92.2%	13
Paikin and Tal (2015)	95.4%	95.4%	-	-	-	-
Sholomon et al. (2014)	94.6%	94.9%	10	89.6%	92%	8
Son et al. (2018)	95.8%	95.6%	12	92.3%	94.5%	11
Ours	87.1%	93.9%	5	25.8%	78%	0

the α anchoring threshold was set to 0.7. Dissimilarity computations are based on the CIELAB color space. Compatibility computations (Eq. 10) use $k = 3$ and $k = 1.5$ for Type 1 and 2 solvers, respectively.

Table 1 presents the Type 1 solver performance. Our main goal in this paper is a progression in performance of RL solvers and indeed the multi-phase method performs better than Khoroshiltseva et al. (2021). Equally important, results are highly competitive with many prior methods on the MIT data set and approach the MIT data set upper bounds (DC, NC, and PR upper bounds are 96.7%, 96.4% and 15, respectively; Son et al., 2014), which are below perfect since no solver is expected to internally order identical constant pieces correctly. The results of prior methods were taken from their papers, while the results of Khoroshiltseva et al. (2021) were obtained by our implementation for their solver with Sinkhorn-Knopp balancing. Table 2 presents our Type 2 results compared to other methods. Because Type 2 solutions may be rotated compared to the ground truth (see Figure 5), we evaluate only the rotation that achieves the highest DC. Figure 7 shows Type 1 and 2 results.

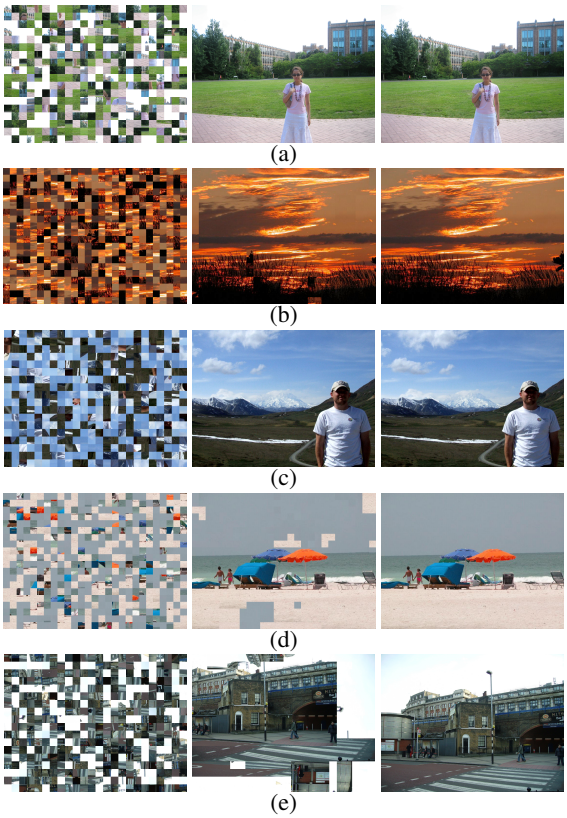


Figure 7: Input puzzle, RL solver result, and ground truth image are shown left to right. (a) shows a typical result on a Type 1 puzzle with constant pieces (sky pieces are constant white). While the solution is visually perfect, its DC and NC scores are relatively low (83.8% and 81.8%, respectively) due to wrong internal ordering of constant pieces. (b) shows a Type 1 solution with a very low DC score (6.3%) even though the solution is visually excellent, due to offset of one column. (c) shows a perfectly solved Type 2 puzzle. (d) and (e) exemplify an occasional failure of the Type 1 and 2 solvers (respectively) in which sub-blocks of pieces are assembled correctly, but in wrong position (and rotation for Type 2) relative to each other.

5 CONCLUSION

This paper suggests a step forward in RL puzzle solving based on a novel multi-phase RL optimization approach that is fully automatic, uses no prior information, guarantees feasible solutions, and is applicable for both Type 1 and 2 puzzles. Future work should handle failure cases (see Figures 7(d)-7(e)), which could, we believe, be alleviated within the RL framework. For example, by identifying correctly assembled block (such as done by Pomeranz et al., 2011), modifying the compatibility coefficients accordingly and running the RL process again.

We believe that the proposed multi-stage approach could be used as a general RL-based scheme for other hard optimization problems (not necessarily puzzles) for solving permutation challenges (cf. Type 1 cases) or even more complicated permutation constraints (cf. Type 2 cases). For example, it is likely that the multi-phase approach could be applied to much larger scale instances of the traveling salesman problem than previously attempted with RL (Pelillo, 1993).

ACKNOWLEDGEMENTS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 964854, the

Helmsley Charitable Trust through the ABC Robotics Initiative, and the Frankel Fund of the Computer Science Department at Ben-Gurion University.

REFERENCES

Andaló, F. A., Taubin, G., and Goldenstein, S. (2012). Solving image puzzles with a simple quadratic programming formulation. In *2012 25th SIBGRAPI Conference on Graphics, Patterns and Images*, pages 63–70.

Andaló, F. A., Taubin, G., and Goldenstein, S. (2016). PSQP: Puzzle solving by quadratic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(2):385–396.

Cho, T. S., Avidan, S., and Freeman, W. T. (2010). A probabilistic image jigsaw puzzle solver. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 183–190.

Deever, A. and Gallagher, A. (2012). Semi-automatic assembly of real cross-cut shredded documents. In *2012 19th IEEE International Conference on Image Processing*, pages 233–236.

Demaine, E. D. and Demaine, M. L. (2007). Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity. *Graphs and Combinatorics*, 23(1):195–208.

Floudas, C. A. (2013). *Deterministic global optimization: Theory, methods and applications*. Springer Science & Business Media.

Freeman, H. and Garder, L. (1964). Apictorial jigsaw puzzles: The computer solution of a problem in pattern recognition. *IEEE Transactions on Electronic Computers*, EC-13(2):118–127.

Gallagher, A. C. (2012). Jigsaw puzzles with pieces of unknown orientation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 382–389.

Hummel, R. A. and Zucker, S. W. (1983). On the foundations of relaxation labeling processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(3):267–287.

Huroyan, V., Lerman, G., and Wu, H.-T. (2020). Solving jigsaw puzzles by the graph connection laplacian. *SIAM Journal on Imaging Sciences*, 13(4):1717–1753.

Khoroshiltseva, M., Vardi, B., Torcinovich, A., Traviglia, A., Ben-Shahar, O., and Pelillo, M. (2021). Jigsaw puzzle solving as a consistent labeling problem. In *International Conference on Computer Analysis of Images and Patterns*, pages 392–402. Springer.

Koller, D. and Levoy, M. (2006). Computer-aided reconstruction and new matches in the forma urbis romae. *Bullettino Della Commissione Archeologica Comunale di Roma*, pages 103–125.

Kosiba, D. A., Devaux, P. M., Balasubramanian, S., Gandhi, T. L., and Kasturi, K. (1994). An automatic jigsaw puzzle solver. In *Proceedings of 12th International Conference on Pattern Recognition*, volume 1, pages 616–618.

Li, R., Liu, S., Wang, G., Liu, G., and Zeng, B. (2021). Jigsawgan: Auxiliary learning for solving jigsaw puzzles with generative adversarial networks. *IEEE Transactions on Image Processing*, 31:513–524.

Liu, H., Cao, S., and Yan, S. (2011). Automated assembly of shredded pieces from multiple photos. *IEEE Transactions on Multimedia*, 13(5):1154–1162.

Paikin, G. and Tal, A. (2015). Solving multiple square jigsaw puzzles with missing pieces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4832–4839.

Paumard, M.-M., Picard, D., and Tabia, H. (2020). Deepzzle: Solving visual jigsaw puzzles with deep learning and shortest path optimization. *IEEE Transactions on Image Processing*, 29:3569–3581.

Pelillo, M. (1993). Relaxation labeling processes for the traveling salesman problem. In *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*, volume 3, pages 2429–2432.

Pelillo, M. (1997). The dynamics of nonlinear relaxation labeling processes. *Journal of Mathematical Imaging and Vision*, 7(4):309–323.

Pomeranz, D., Shemesh, M., and Ben-Shahar, O. (2011). A fully automated greedy square jigsaw puzzle solver. In *CVPR 2011*, pages 9–16.

Rosenfeld, A., Hummel, R. A., and Zucker, S. W. (1976). Scene labeling by relaxation operations. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(6):420–433.

Sholomon, D., David, O., and Netanyahu, N. S. (2013). A genetic algorithm-based solver for very large jigsaw puzzles. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1767–1774.

Sholomon, D., David, O. E., and Netanyahu, N. S. (2014). A generalized genetic algorithm-based solver for very large jigsaw puzzles of complex types. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28.

Son, K., Hays, J., and Cooper, D. B. (2014). Solving square jigsaw puzzles with loop constraints. In *European Conference on Computer Vision*, pages 32–46.

Son, K., Hays, J., and Cooper, D. B. (2018). Solving square jigsaw puzzle by hierarchical loop constraints. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2222–2235.

Talon, D., Del Bue, A., and James, S. (2022). Ganzzle: Reframing jigsaw puzzle solving as a retrieval task using a generative mental image. *arXiv preprint arXiv:2207.05634*.

Zhao, Y.-X., Su, M.-C., Chou, Z.-L., and Lee, J. (2007). A puzzle solver and its application in speech descrambling. In *Proceedings of the 2007 Annual Conference on International Conference on Computer Engineering and Applications*, pages 171–176.

Zucker, S. W., Hummel, R. A., and Rosenfeld, A. (1977). An application of relaxation labeling to line and curve enhancement. *IEEE Transactions on Computers*, 26(04):394–403.