Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence

A Generalized Genetic Algorithm-Based Solver for Very Large Jigsaw Puzzles of Complex Types

Dror Sholomon and **Omid E. David** and **Nathan S. Netanyahu***
Department of Computer Science
Bar-Ilan University
Ramat-Gan 52900, Israel
dror.sholomon@gmail.com, mail@omiddavid.com, nathan@cs.biu.ac.il

Abstract

In this paper we introduce new types of square-piece jigsaw puzzles, where in addition to the unknown location and orientation of each piece, a piece might also need to be flipped. These puzzles, which are associated with a number of real world problems, are considerably harder, from a computational standpoint. Specifically, we present a novel generalized genetic algorithm (GA)-based solver that can handle puzzle pieces of unknown location and orientation (Type 2 puzzles) and (two-sided) puzzle pieces of unknown location, orientation, and face (Type 4 puzzles). To the best of our knowledge, our solver provides a new state-of-the-art, solving previously attempted puzzles faster and far more accurately, handling puzzle sizes that have never been attempted before, and assembling the newly introduced two-sided puzzles automatically and effectively. This paper also presents, among other results, the most extensive set of experimental results, compiled as of yet, on Type 2 puzzles.

Introduction

Jigsaw puzzles are a popular form of entertainment, first produced around 1760 by John Spilsbury, a Londonian engraver and mapmaker. Given n different non-overlapping pieces of an image, the player has to reconstruct the original image, taking advantage of both the shape and chromatic information of each piece. Despite the popularity and vast distribution of jigsaw puzzles, their assembly is not trivial, from a computational standpoint, as this problem was proven to be NP-hard (Altman 1989; Demaine and Demaine 2007). Nevertheless, a computerized jigsaw solver may have applications in many real-world problems, arising in various domains such as archeology (Koller and Levoy 2006; Brown et al. 2008), biology (Marande and Burger 2007), chemistry (Wang 2000), literature (Morton and Levison 1968), speech descrambling (Zhao et al. 2007), art restoration (Fornasier and Toniolo 2005), image editing (Cho et al. 2008), and the recovery of shredded documents or photographs (Justino, Oliveira, and Freitas 2006; Marques and Freitas 2009; Cao, Liu, and Yan 2010; Deever and Gallagher 2012). Besides, as noted in (Goldberg, Malon, and Bern 2004), pursuing this topic may be justified solely due to its intriguing nature.

The first to tackle the jigsaw problem, computationally, were Freeman and Garder (1964). Their solver handles up to 9-piece puzzles, using only piece shape information. Kosiba et al. (1994) were the first to facilitate the use of image content by their solver. Subsequent research turned to color-based square-piece puzzles, instead of the earlier shape-based variants. Cho et al. (2010) presented a probabilistic puzzle solver that can handle up to 432 pieces, given some a priori knowledge of the puzzle (e.g., anchor pieces). Their results were further improved by (Yang, Adluru, and Latecki 2011), who presented a so-called particle filter-based solver and by (Pomeranz, Shemesh, and Ben-Shahar 2011), who made a major contribution to the field by introducing a fully automated jigsaw puzzle solver that handles puzzles of up to 3,000 (square) pieces, without any prior knowledge about the image. The latter solver treats puzzles with unknown piece location but assumes a known orientation. Gallagher (2012) was the first to handle puzzles with unknown piece location and orientation (Type 2 puzzles), where each piece might need to be translated and rotated (by 0, 90, 180, or 270 degrees). This solver was tested on 432- and 1,064-piece puzzles and a single 9,600-piece image. More recently, a solver based on *genetic algorithms (GA)* (Holland 1975), which can handle up to 22,755-piece puzzles, was presented by(Sholomon, David, and Netanyahu 2013). Although capable of solving considerably larger puzzles, their solver is restricted to known piece orientations (i.e., Type 1 puzzles).

In its most basic form, every puzzle solver requires an evaluation function for the compatibility of adjacent pieces and a tiling strategy for placing the pieces as accurately as possible. Recent tiling strategies tend to be greedy and thus are subject to the familiar disadvantages of greedy methods, i.e., they are more likely to be affected by local optima. While (Sholomon, David, and Netanyahu 2013) successfully employed a GA-based solver for Type 1 puzzles (unknown piece location only), the question remains whether GA-based solvers could be applied to more challenging types, namely Type 2 puzzles (unknown piece location and orientation) and Type 4 puzzles (two-sided pieces of unknown location and orientation).

*Nathan Netanyahu is also with the Center for Automation Research, University of Maryland, College Park, MD 20742 (e-mail: nathan@cfar.umd.edu).

In this paper, we continue in the same vein of employing genetic algorithms as a strategy for piece placement. We describe a detailed GA scheme for a solver capable of handling both Type 1 and Type 2 puzzles more than twice as large as puzzle sizes that have been attempted before, and present extensive empirical results which demonstrate the efficiency of the presented method in terms of speed and accuracy. We further advance the state-of-the-art with respect to the jigsaw puzzle problem by extending our solver to handle also two-sided puzzles (Type 4 puzzles), i.e., puzzles where the correct face of each piece is also unknown (adding another degree of freedom to each piece). Thus, the solver has to find the correct location for each piece, its correct orientation (out of four possible angles), and its correct face (out of two possible ones). This type of puzzle, which is representative of various real-world applications, is considerably more complex. We present an extensive set of empirical results for all available datasets, establishing the effectiveness of our solver in handling different images of different sizes.

Puzzle Types

The first discussion on different puzzle types appears in (Gallagher 2012). In all types, n different non-overlapping square pieces of an image are given and there exists a unique tiling (arrangement) which is considered correct. Type 1 is the most common variant handled; it refers to puzzles with only piece location unknown. In Type 2 puzzles, a piece orientation is also unknown, allowing each piece to be rotated by 0, 90, 180 or 270 degrees. As noted by (Gallagher 2012), this puzzle type increases the complexity in several ways. First, a pair of pieces can fit together in any of 16 possible configurations, multiplying the number of possible solutions by 4^n in comparison to Type 1 puzzles. Second, an algorithmic solver must consider both translations and rotations. Third, the puzzle reconstruction should be considered in both landscape and portrait orientations. Type 3 puzzles, consisting of pieces with known location and unknown orientation, are listed only for the sake of completeness.

We define here Type 4 puzzles, i.e., two-sided puzzles of two images, where each piece face belongs to one of the images. The solver has to determine the correct location of each piece, its correct orientation, and its relevant face (with respect to each of the two images). This problem version is motivated by real-world applications, e.g., a shredded document might have been printed on both sides before being shredded. (This is all the more applicable, in view of current global environmental trends, encouraging double-sided printing.) The computational complexity in this case increases in several ways. First, each pair of pieces might now fit together in any of 64 possible configurations, multiplying the number of possible solutions by 8^n, relatively to Type 1 puzzles. Second, the solver has to consider now the possibility of flipping a piece, in addition to its translation and rotation. Third, the solver must always consider the two images (being formed on the fly) when placing new pieces and assessing the results, e.g., whether to assemble each image separately or assemble them simultaneously.

For completeness, one can define additional puzzle types with all possibilities of (un)known piece location, orientation, and face.

Type	Unknown Location	Unknown Orientation	Unknown Face
1	✓		
2	✓	✓	
3		✓	
4	✓	✓	✓
5	✓		✓
6		✓	✓
7			✓

Table 1: Puzzle types according to different problem specifications.

Genetic Algorithms

In this section we provide a quick overview of the GA methodology. A GA is a search procedure within a problem's solution domain. Since examining all possible solutions of a specific problem is virtually infeasible, GAs offer an optimization heuristic inspired by biological natural selection.

In GA terms, a solution to the problem (e.g., a suggested arrangement of the puzzle's pieces) is represented as an individual "organism" (i.e., *chromosome*) of a large population. Essentially, a GA attempts to reach an optimal solution by mimicking the processes of natural selection and evolution; the "fittest" individuals of each generation reproduce, creating offspring chromosomes. If defined correctly, the *crossover* operator responsible for offspring creation should allow for "good" qualities to pass on from parents to children, in an attempt to create better offspring (i.e., solutions). In each iteration (i.e., one *generation* of the algorithm), the entire population is replaced by the many offspring created by the crossover operation. (The total population size remains fixed throughout all the generations.)

In order to imitate natural selection, a chromosome's reproduction rate, i.e., the number of times it is selected to reproduce (and hence the number of its offspring), is set directly proportionate to its *fitness*. The fitness, which is a score obtained by a *fitness function*, represents the quality of a given solution. The crossover should, thus, take place mainly between higher-rated solutions.

The successful performance of a GA depends mainly on the appropriate choice of chromosome representation, crossover operator, and fitness function. The chromosome representation and crossover operator should yield an enhanced solution by merging two "promising" chromosomes (i.e., chromosomes representing promising partial solutions) that are passed on to the next generation. Figure 1 provides a pseudo-code of a common GA framework.

Puzzle Solving

In this section we present our generalized GA-based solver, which is designed to handle more difficult puzzle types. The

```
population = create_random_population(POPULATION_SIZE);
for (i = 0; i < NUM_OF_GENERATIONS; ++i) {
    new_population = NULL;
    evaluate_population(&population);
    for (j = 0; j < POPULATION_SIZE; ++j) {
        parent1 = select_parent(population);
        parent2 = select_parent(population);
        child = crossover(parent1, parent2);
        add_child_to_population(&new_population, child);
    }
    population = new_population;
}
solution = get_best(population);
```

Figure 1: Pseudocode of GA framework

GA starts from a fixed-size population of randomly generated solutions. In each iteration, the entire population is evaluated using the fitness function described below, and a new population is (re)produced by employing the crossover operator to the selected chromosome pairs. We use the common selection method of *roulette wheel selection*, where each chromosome is selected to reproduce, with probability directly proportional to its fitness score.

We define each chromosome to be a complete solution to the problem, i.e., a suggested tiling of all pieces, including their orientation and face (if applicable). We now have to supply an appropriate fitness function and a crossover operator.

The Fitness Function

The fitness function determines the quality of each chromosome (i.e., each solution), and hence the expected number of its children. In every generation, all chromosomes are evaluated for the purpose of selection.

For fitness evaluation, we use the *dissimilarity* measure, which was investigated thoroughly in previous comparative studies (Cho, Avidan, and Freeman 2010; Pomeranz, Shemesh, and Ben-Shahar 2011) and found to be very effective. The dissimilarity measure relies on the premise that adjacent jigsaw pieces in the original image tend to share similar colors along their abutting edges, i.e., the sum (over all neighboring pixels) of squared color differences (over all three color bands) should be minimal. Let pieces p_i, p_j be represented in normalized L*a*b* space by corresponding $W \times W \times 3$ matrices, where W is the height/width of each piece (in pixels). Assuming, for example, that p_j is to the right of p_i, the piece dissimilarity in this case is given by:

$$D(p_i, p_j) = \sqrt{\sum_{k=1}^{W}\sum_{b=1}^{3}(p_i(k, W, b) - p_j(k, 1, b))^2}. \quad (1)$$

Obviously, to maximize the compatibility of two pieces, their dissimilarity should be minimized.

For Type 2 puzzles we set the fitness function of a given chromosome to be the sum of pairwise dissimilarities over all adjacent edges. Given that the puzzle consists of $(N \times M)$ tiles, we represent each chromosome by an $(N \times M)$ matrix, where a matrix entry $x_{i,j}(i = 1..N, j = 1..M)$ corresponds to a single puzzle piece and its orientation, we define its fitness as:

$$\sum_{i=1}^{N}\sum_{j=1}^{M-1}(D(x_{i,j}, x_{i,j+1})) + \sum_{i=1}^{N-1}\sum_{j=1}^{M}(D(x_{i,j}, x_{i+1,j})) \quad (2)$$

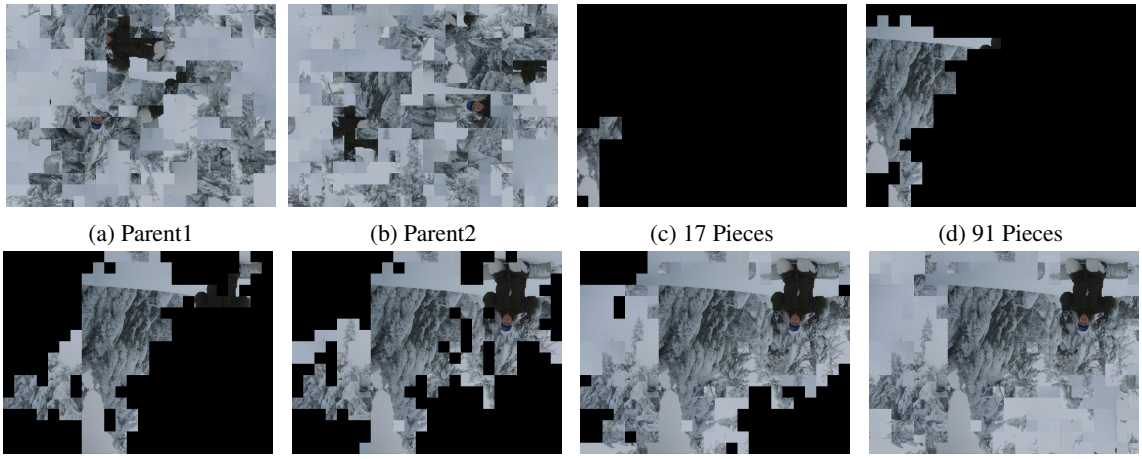
where the D term in each case is the dissimilarity of the two pieces in question about their joint edge, taking into account their actual rotations, as stored in the chromosome.

For Type 4 puzzles, we need to consider actually two abutting edges (i.e., an adjacent edge for each piece face), for every neighboring piece in a piece pair. We sum the pairwise dissimilarities over *all* adjacent edges, computing effectively a fitness score for each image of the two-sided puzzle. Thus, the GA must balance the dissimilarity minimization on both sides of the puzzle to obtain, hopefully, the correct reconstruction of both images.

Type 2 Crossover

Given a chromosome representation, the crossover operator receives two tile configurations (i.e., two parent chromosomes) and produces a new arrangement of the pieces (i.e., a new offspring chromosome). In general, the crossover operator should encourage "good" qualities to pass on from the parents to their child. In particular, we would like an effective operator to meet the following three requirements. First, it must create a valid child solution, where each piece appears exactly once (i.e., no missing or duplicate pieces). Second, the operator must support *position independence* of puzzle segments assembled correctly. Namely, if a parent assembles correctly part of the image (i.e., a segment), modulo some spatial offset, the operator should accommodate the required spatial translation and rotation of the entire segment in the resulting child. For example, Figure 2 features a parent chromosome containing a person's head correctly assembled, albeit spatially misplaced. The idea is to retain the correctly assembled head segment but allow for its translation and rotation. Third, a proper heuristic should be applied to detect correctly assembled segments (as the person's head in Figure 2).

Various works (Gallagher 2012; Sholomon, David, and Netanyahu 2014) use a weighted graph representation for the problem, where each node corresponds to a jigsaw piece and each (graph) edge corresponds to a joint edge of two adjacent puzzle pieces. We denote the edges of piece p_i (in a clockwise manner) as $p_i.a, p_i.b, p_i.c$ and $p_i.d$. For example, the graph edge $p_i.b - p_j.d$ denotes the geometric configuration where edge b of piece p_i is adjacent to edge d of piece p_j. The weight of each graph edge is the dissimilarity of the two-piece configuration (given by Eq. 1). For Type 2 puzzles, there are 16 possible edges between every two (graph) vertices. This representation lends itself easily to an effective crossover of correctly assembled segments in the parents. The geometric relation $p_i.b - p_j.d$ is invariant to both the absolute spatial location and orientation of the pieces (e.g., translating and/or rotating both pieces by 90 degrees will not affect this relation).



(a) Parent1 (b) Parent2 (c) 17 Pieces (d) 91 Pieces

(e) 147 Pieces (f) 229 Pieces (g) 347 Pieces (h) Child

Figure 2: Illustration of Type 2 crossover operation: (a) Parent1, (b) Parent2, (c)–(g) evolution of kernel growth until (h) a complete child is obtained. Note how the operator manages to locate and compose the skier's body from all differently located and oriented parts assembled in both parents.

Arriving at a piece arrangement can be viewed analogously to constructing a spanning tree of $n-1$ graph edges. Although each jigsaw piece is represented exactly once in this spanning tree, the correct dimensions and geometrical feasibility of the corresponding image might not be satisfied. We propose a crossover procedure, analogous to Prim's algorithm (Prim 1957), for finding a minimal spanning tree (MST). This procedure meets the required constraints while attempting to construct a tree representing a better child solution.

Prim's algorithm starts from a single vertex and grows the sub-tree, one edge (and vertex) at a time, selecting at each step a minimum-weight edge connecting a sub-tree vertex and an external one. Similarly, we start constructing the puzzle from one piece and grow it by adding a single jigsaw piece (a graph edge) at a time. At every iteration we review all edges emanating from the partially grown piece kernel. In addition to Prim's basic constraint (of avoiding edge cycles), the following requirements should be met upon adding a new piece (associated with a joint edge of an adjacent piece). First, the known image dimensions must not be breached. Second, each piece edge must be accounted for only once for all selected edges (thus avoiding infeasible geometrical configurations due to piece overlap, e.g., two competing pieces for the same edge of an adjacent third piece). Thus, we ensure that each piece appears exactly once and that the resulting image is geometrically feasible. Finally, the constructed tree is translated to a chromosome by recording the resulting locations and orientations of all the pieces. Also recorded are geometric configurations which are not part of the MST but are implied by it.

Unlike Prim, instead of selecting literally the minimum-weight edge in each iteration, we employ a heuristic based on intrinsic knowledge of the parents and edge weights. The procedure first selects an edge appearing in both parents; e.g., if p_3 is in the current kernel, both parents contain the edge $p_3.b - p_6.d$, and all of the above constraints are met, this edge will be selected. Second, if no common edge can be added to the kernel, the procedure seeks a *best-buddy edge* (described below) residing in either parent; e.g., if p_3 is in the current kernel, one parent contains the edge $p_3.b - p_6.d$, which is a best-buddy edge, and all other constraints are met, this edge will be picked. The notion of *best-buddy pieces* was first introduced by (Pomeranz, Shemesh, and Ben-Shahar 2011); two pieces are said to be best-buddies if each piece considers the other as its most compatible piece according to the compatibility measure defined. Extending this notion to best-buddy edges is straightforward. Finally, if none of the above holds, the procedure follows Prim's approach and chooses a minimal-weight edge, for which all of the required constraints are satisfied. Note that only edges emanating from the growing piece kernel are considered, in each iteration. Hence, each edge (jigsaw piece) addition introduces additional possible edges (i.e., edges shared amongst the parents or best-buddy edges) that should be considered next.

A subtle issue that requires careful consideration is that of image dimensions. Since a piece orientation is unknown, it cannot be determined, in advance, whether an $N \times M$ or an $M \times N$ frame should be used. Choosing either frame, in advance, could discard all chromosomes trying to assemble correctly rotated instances of the intended image by 90 or 270 degrees. We overcome this problem by maintaining initially a flexible frame. After each piece assignment, we check the farthest boundaries. Assuming $M < N$, once a length of $M+1$ is reached along one of the dimensions, the frame must grow up to N along this same dimension. This allows the images to be assembled in any direction, with no boundary violations. Thus, the crossover operator is invariant to the orientation chosen when creating a child. Experimental results show, indeed, that the GA assembles the images correctly in different orientations in each generation.

The above described procedure achieves all its prede-

fined goals. The resulting chromosome (image) is valid, each piece appears exactly once and both image dimensions and geometric feasibility are maintained. Furthermore, segments assembled correctly (through the addition at each step of shared piece edges between parents or best-buddy edges) are copied to the child solution. Since only piece adjacencies are copied, the segments might appear in different spatial positions and orientations in the two parents (see Figure 2), achieving position- and orientation-independence.

Type 4 Crossover

In this puzzle type each piece is two sided and might need to be flipped. Segments assembled correctly might appear in different absolute locations, different orientations, and opposite sides of the puzzle. For example, the same segment might be located on opposite sides of two different chromosomes while two segments belonging to the same image (e.g., a person's head and their body) might be located on opposite sides of the same chromosome.

Similarly to the discussion on Type 2 puzzles, eight edges are now associated with each piece, four edges on each piece face. We mark the piece edges as a, b, c, d and their opposite counterparts as a', b', c', d', respectively (e.g., if b is the right piece edge, b' is the left edge of the flipped face). All graph edges will be labeled using the augmented edge scheme. There are now 64 possible edges between every two vertices; each piece edge should be considered during every iteration of the crossover procedure (e.g., the procedure considers adding a piece p_i, possibly rotated and/or flipped, to the growing kernel). To maintain geometrical validity, we add the following constraint. The flipping side of a piece edge at the joint boundary of a two-piece configuration may not be selected for another two-piece configuration, e.g., if edge $p_i.b - p_j.a'$ is selected, then all MST edges incident on $p_i.b'$ or $p_i.a$ are prohibited.

Choosing an edge such as $p_i.b - p_j.a'$ effectively implies flipping one of the pieces. Hence, the operator easily merges correctly assembled segments residing in different image sides (e.g., copying the first segment, flipping one piece from the second segment, and copying and flipping the remaining pieces accordingly).

Interestingly, as a result of the above mentioned procedure, the crossover operator actually assembles both images concurrently (this approach is different from how a human would solve such a puzzle, and is considerably superior). Thus, we achieved a concurrent puzzle solver, that exploits "easier" segments in each image for achieving a better score. The solver is completely invariant to the orientation and side of the assembled images. Indeed, our experiments reveal that these parameters change constantly in the best chromosome of each generation.

Experimental Results

In all experiments we used the same GA parameters. Each generation consists of 1,000 chromosomes, selection is due to the roulette-wheel selection method (as previously mentioned), and the number of generations (for each run) is 30. (Based on our experience, the latter number seems to achieve a good balance between accuracy and efficiency.)

# Pieces	Direct	Neighbor	Perfect	Run-time
432	94.58%	94.86%	10	8 sec
540	89.57%	91.98%	8	11.9 sec
805	87.76%	92.07%	6	22.1 sec
5,015	93.24%	93.66%	8	9.74 min
10,375	96.18%	97.05%	4	35.12 min
22,755	77.43%	91.07%	1	3.48 hr

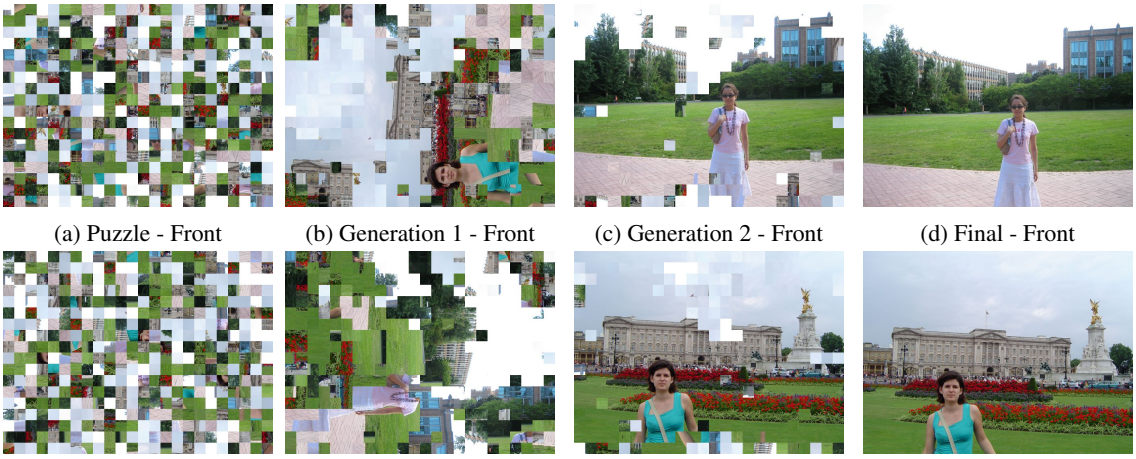
Table 2: Accuracy results (under direct and neighbor comparison) on Type 2 puzzles, running the generalized GA five times on each image of every 20-image set; average of best result (per image) is shown for each image set.

Following previous works (Cho, Avidan, and Freeman 2010; Pomeranz, Shemesh, and Ben-Shahar 2011; Gallagher 2012; Sholomon, David, and Netanyahu 2013), we evaluated our scores using the *direct comparison* and *neighbor comparison* measures. Direct comparison returns the fraction of pieces in the assembled puzzle that are in their correct absolute position. Neighbor comparison is the fraction of pairwise piece adjacencies that are correct. We also report the number of images reconstructed perfectly in every set.

For Type 2 puzzles we tested our solver on all previously established benchmarks (Pomeranz, Shemesh, and Ben-Shahar 2011; Sholomon, David, and Netanyahu 2013) using standard tile dimensions of 28 $\times$ 28 pixels. These benchmark datasets contain 20 images of 432-, 540-, 805-, 5,015-, 10,375-, and 22,755-piece puzzles. The largest Type 2 puzzle that has been attempted before is a single 9,600-piece puzzle, i.e., we have tackled 20 puzzles more than twice as large as this size. Due to the stochastic nature of GAs, we ran the solver five times on each image in every set, and recorded the best, worst, and average result over these five runs. The average best results for each image set (with respect to the direct comparison and neighbor comparison measures) is reported in Table 2. Comparing the results achieved on the 432-piece dataset to the best results reported in (Gallagher 2012, Table 6) reveals their algorithm yields a 90.4% accuracy, i.e., we obtained a significant improvement of over 4%.

For completeness, we also report set averages (according to the neighbor comparison) of the average and worst results obtained for each image (over its five runs). Table 3 contains these additional results for all image sets experimented with. Despite the random nature of GAs, the results are consistent (i.e., the small standard deviation suggests that a single run could have sufficed).

For Type 4 puzzles we did not experiment with the entire benchmark, as each puzzle requires two images so running the entire 400 possible pairs associated with each image set would have been very tedious, if not infeasible. Instead, we composed three two-sided puzzles (each containing two images) from the 5,015- and 10,375-piece puzzle sets. The first (two-sided) puzzle from each set contained two perfectly solved images as Type 2 puzzles, the second contained one perfectly solved image and a different image, and the third contained two images that were not solved perfectly as Type 2 puzzles. All puzzles with at least one perfectly solved side



(a) Puzzle - Front (b) Generation 1 - Front (c) Generation 2 - Front (d) Final - Front

(e) Puzzle - Back (f) Generation 1 - Back (g) Generation 2 - Back (h) Final - Back

Figure 3: Solution process of 432-piece Type 4 puzzle: (a) Given puzzle, and best chromosome obtained by the GA in the (b) first, (c) second, and (d) last generation; (e)–(f) opposite sides of exact same chromosomes. Final chromosome's accuracy is 100%. Largest Type 4 puzzle solved contains 10,375 pieces.

# of Pieces	Avg. Best	Avg. Worst	Avg. Avg.	Avg. Std. Dev.
432	94.86%	93.79%	94.44%	0.39%
540	91.98%	90.60%	91.33%	0.49%
805	92.07%	90.76%	91.45%	0.48%
5,015	93.66%	93.19%	93.42%	0.17%
10,375	97.05%	96.75%	96.91%	0.11%
22,755	91.07%	90.47%	90.80%	0.22%

Table 3: Accuracy results (under neighbor comparison) on Type 2 puzzles, running generalized GA five times on each image of every 20-image set; average of best, worst, and average score (and average standard deviation) per image is given for each image set.

# Pieces	Images	Direct	Neigh.	Run-time
5,015	09, 10	100%	100%	20.08 min
5,015	19, 07	100%	100%	17.88 min
5,015	05, 14	84.77%	85.73%	28.57 sec
5,015	03, doc	99.92%	99.86%	27.88 min
10,375	01, 04	100%	100%	76.02 min
10,375	15, 11	100%	100%	81.65 min
10,375	15, 19	99.35%	99.20%	85 min

Table 4: Results for Type 4 GA on selected puzzles, under direct and neighbor comparisons, including total run-time.

were again solved perfectly. This result is quite remarkable, as it attests to the GA's power to assemble concurrently the two images by focusing, presumably, on the "easier" one. Even the more challenging puzzles were solved with far greater accuracy than when solved separately, again due to the GA solver's ability to effectively incorporate the information to the opposite face of a given piece.

Finally, to further challenge our generalized GA solver, in terms of handling real-world applications, we also attempted a two-sided, 5,015-piece puzzle whose one side is an image scene and its other side is a scanned document. Trying to solve the document as a Type 1 puzzle, we obtained horrendous accuracy, probably due to the ineffectiveness of the compatibility measure used, as all pieces are mostly white. This experiment is the closest attempt known to us to automatically solve a large jigsaw shredded document. A scenario in which such a document contains an image on its other side is not very frequent but is definitely possible. Given such a puzzle, the generalized GA solver reaches 99.86% accuracy, reassembling almost completely the correct image(s). This was accomplished despite the possible interference of the white piece faces.

All test results regarding Type 4 puzzles can be viewed in Table 4. The tests were performed on a modern PC. Average run-time of the solver on the larger, 10,375-piece puzzles was 1.35 hours, a considerable improvement compared to the 23.5 hours reported by (Gallagher 2012) for a slightly smaller and far less complex, 9,600-piece Type 2 puzzle.

Conclusion

In this paper, we introduced a new puzzle type, the two-sided puzzle, where the location, orientation, and face of each piece is unknown. This type of puzzle is likely to stir great interest, as it is supposedly the most complex type known; more importantly, it is highly representative of real-world applications, such as the reconstruction of shredded documents.

In addition, we presented the first GA-based solver capable of solving both Type 2 and Type 4 puzzles. Our generalized GA solver outperforms, to a significant extent, current state-of-the-art solvers, as it is capable of solving Type 2 puzzles far more accurately and efficiently. Specifically, our solver can handle Type 2 puzzles of up to 22,755 pieces, i.e., more than twice as large as the size of Type 2 puzzles that

have been reported. In addition, our solver is the only Type 4 solver, as of yet, and it manages, among other tasks, to reconstruct perfectly two-sided puzzles of up to 10,375 pieces.

We believe our contributions can be further generalized to other, more difficult types of puzzles, in a similar manner.

References

Altman, T. 1989. Solving the jigsaw puzzle problem in linear time. *Applied Artificial Intelligence an International Journal* 3(4):453–462.

Brown, B.; Toler-Franklin, C.; Nehab, D.; Burns, M.; Dobkin, D.; Vlachopoulos, A.; Doumas, C.; Rusinkiewicz, S.; and Weyrich, T. 2008. A system for high-volume acquisition and matching of fresco fragments: Reassembling Theran wall paintings. *ACM Transactions on Graphics* 27(3):84.

Cao, S.; Liu, H.; and Yan, S. 2010. Automated assembly of shredded pieces from multiple photos. In *IEEE International Conference on Multimedia and Expo*, 358–363.

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

Cho, T.; Butman, M.; Avidan, S.; and Freeman, W. 2008. The patch transform and its applications to image editing. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1–8.

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

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

Fornasier, M., and Toniolo, D. 2005. Fast, robust and efficient 2D pattern recognition for re-assembling fragmented images. *Pattern Recognition* 38(11):2074–2087.

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. 2012. Jigsaw puzzles with pieces of unknown orientation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 382–389.

Goldberg, D.; Malon, C.; and Bern, M. 2004. A global approach to automatic solution of jigsaw puzzles. *Computational Geometry: Theory and Applications* 28(2-3):165–174.

Holland, J. H. 1975. *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.

Justino, E.; Oliveira, L.; and Freitas, C. 2006. Reconstructing shredded documents through feature matching. *Forensic Science International* 160(2):140–147.

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

Kosiba, D. A.; Devaux, P. M.; Balasubramanian, S.; Gandhi, T. L.; and Kasturi, K. 1994. An automatic jigsaw puzzle solver. In *International Conference on Pattern Recognition*, volume 1, 616–618. IEEE.

Marande, W., and Burger, G. 2007. Mitochondrial DNA as a genomic jigsaw puzzle. *Science* 318(5849):415–415.

Marques, M., and Freitas, C. 2009. Reconstructing strip-shredded documents using color as feature matching. In *ACM Symposium on Applied Computing*, 893–894.

Morton, A. Q., and Levison, M. 1968. The computer in literary studies. In *IFIP Congress*, 1072–1081.

Pomeranz, D.; Shemesh, M.; and Ben-Shahar, O. 2011. A fully automated greedy square jigsaw puzzle solver. In *IEEE Conference on Computer Vision and Pattern Recognition*, 9–16.

Prim, R. C. 1957. Shortest connection networks and some generalizations. *Bell system technical journal* 36(6):1389–1401.

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

Sholomon, D.; David, O. E.; and Netanyahu, N. S. 2014. Genetic algorithm-based solver for very large multiple jigsaw puzzles of unknown dimensions and piece orientation. In *Proceeding of the 16th Genetic and Evolutionary Computation Conference*. To appear.

Wang, C.-S. E. 2000. *Determining molecular conformation from distance or density data*. Ph.D. Dissertation, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science.

Yang, X.; Adluru, N.; and Latecki, L. J. 2011. Particle filter with state permutations for solving image jigsaw puzzles. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2873–2880.

Zhao, Y.; Su, M.; Chou, Z.; and Lee, J. 2007. A puzzle solver and its application in speech descrambling. In *WSEAS International Conference on Computer Engineering and Applications*, 171–176.