IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

An automatic solver for very large jigsaw puzzles using genetic algorithms

[1]Meerja Maqbul Baig & [2]Shahela Osmani

[1]M-Tech, Dept. of computer science, Blogger and web developer

Jyothishmathi Institute of Technology and Science.

[2]Sr Analyst -Support & SEO,

Abstract

In this paper we propose the first effective genetic algorithm (GA)-based jigsaw puzzle solver. We introduce a novel crossover procedure that merges two ‘‘parent’’ solutions to an improved ‘‘child’’ configuration by detecting, extracting, and combining correctly assembled puzzle segments. The solver proposed exhibits state-of-the-art performance, as far as handling previously attempted puzzles more accurately and efficiently, as well as puzzle sizes that have not been attempted before. The extended experimental results provided in this paper include, among others, a thorough inspection of up to 30,745-piece puzzles (compared to previous attempts on 22,755-piece puzzles), using a considerably faster concurrent implementation of the algorithm. Furthermore, we explore the impact of different phases of the novel crossover operator by experimenting with several variants of the GA. Finally, we compare different fitness functions and their effect on the overall results of the GAbased solver.

1. Introduction

Introduction The problem domain of jigsaw puzzles is widely known to almost every human being from childhood. Given n different non-overlapping pieces of an potential in devising a genetic algorithm (GA)-based solver, the success of previous attempts was limited to 64-piece puzzles [23], most likely due to the enormous complexity associated with image, the player has to reconstruct the original image, taking advantage of both the shape and chromatic information of each piece. Although this popular game was proven to be an NP-complete problem [1, 7], it has been played successfully by children worldwide. Solutions to this problem might benefit the fields of biology [13], chemistry [24], literature [15], speech

Descrambling [26], archeology [2, 12], image editing [5] and the recovery of shredded documents or photographs [3, 6, 11, 14]. Besides, as Goldberg et al. [10] have noted, the jigsaw puzzle problem may and should be researched for the sole reason that it stirs pure interest. Recent years have witnessed a vast improvement in the research and development of automatic jigsaw puzzle solvers, manifested in both puzzle size, solution accuracy, and amount of manual human intervention required. Reported state-of-the-art solvers are fully automated and can handle puzzles of up to 9000 pieces. Most, if not all, of the aforementioned solvers are greedy, and thus are at great risk of converging to local optima. Despite the great

This version, where piece locations are the only unknowns, is called a ‘‘Type 1’’ puzzle [9]. The solver has no knowledge of the original image and may not make any use

of it. In this paper we introduce a novel crossover operator, enabling for the first time an effective GA-based puzzle solver. Our solver compromises neither speed nor size as it outperforms, for the most part, state-of-the-art solvers, tackling successfully up to 30,745-piece size puzzles (i.e. more than three times the number of pieces that has ever been attempted/reported), within a reasonable time frame (see e.g. Fig. 1 for 10,375- and 22,755-piece puzzles). Our contribution should benefit research regarding EC in general, and the jigsaw puzzle problem, in particular. From an EC perspective, our novel techniques could be used for solving additional problems with similar properties. As to the jigsaw puzzle problem, our proposed framework could prove useful for solving more advanced variants, such as puzzles with missing pieces, unknown piece orientation, mixed puzzles, and more. Finally, we assemble a new benchmark, consisting of sets of larger images (with varying degrees of difficulty), which we make public to the community [18]. Also, we share all of our results (on this benchmark and other public datasets) for future testing and comparative evaluation of jigsaw puzzle solvers. This paper is an extended version of our previously presented work [19]. We lay out the requirements from an effective (GA-based) jigsaw puzzle solver, and provide a more detailed description of our crossover operator, as part of our proposed solution. Furthermore, the paper includes new empirical results of an extended set of experiments aimed at a more exhaustive performance evaluation of our presented solver, in general, and its novel crossover operator, in particular. We formed an extended benchmark of up to 30,745-piece puzzles and tested our solver's performance multiple times on each image. Specifically, we have pursued the following empirical issues: (1) Explored the relative impact of the different phases of our 3-phase crossover operator; (2) tested our solver's performance also with a different fitness function; (3) investigated further the shifting anomaly discovered in our early work; (4) introduced a concurrent crossover version to reduce significantly the overall run-time of the GA (without affecting the solver's accuracy).

2. Previous work

Jigsaw puzzles were first introduced around 1760 by John Spilsbury, a Londonian engraver and mapmaker. Nevertheless, the first attempt by the scientific community to computationally solve the problem is attributed to Freeman and Garder [8] who in 1964 presented a solver which could handle up to nine-piece problems. Ever since then, the research focus regarding the problem has shifted from shape-based to merely color-based solvers of square-tile puzzles. In 2010 Cho et al. [4] presented a probabilistic puzzle solver that could handle up to 432 pieces, given some a priori knowledge of the puzzle. Their results were improved a year later by Yang et al. [25] who presented a particle filter-based solver. Furthermore, Pomeranz et al. [16] introduced that year, for the first time, a fully automated square jigsaw puzzle solver that could handle puzzles of up to 3000 pieces. Gallagher [9] has further advanced this by considering a more general variant of the problem, where neither piece orientation nor puzzle dimensions are known. In its most basic form, every puzzle solver requires an estimation function to evaluate the compatibility of adjacent pieces and a strategy for placing the pieces as accurately as possible. Although much effort has been invested in perfecting the compatibility functions, recent strategies tend to be greedy, which is known to be problematic when encountering local optima. Thus, despite achieving very good—if not perfect—solutions for some puzzles, supplementary material provided by Pomeranz et al. [17] suggests there is much room for improvement for many other puzzles. Comparative studies conducted by Gallagher ([9]; Table 4), regarding the benchmark of 432-piece images, reveal only a slight improvement in accuracy relative to Pomeranz et al. (95.1 vs. 95.0 %). To the best of our knowledge, no additional runs on other benchmarks have been reported by Gallagher. Interestingly, despite the availability of puzzle solvers for 3000- and 9000-piece puzzles, there exists no image set, for the purpose of benchmark testing, containing puzzles with more than 805 pieces. Current state-of-the-art solvers were tested only on very few large images, and those tested contained an extreme variety of textures and colors, which renders them, admittedly, as ''easier'' for solving [9]. We assume that as with the smaller images, the accuracy of current solvers on some large puzzles could be greatly improved by using more sophisticated algorithms. New attempts were made to solve the jigsaw puzzle problem since our original publication [19]; see e.g. [22] which is based on the so-called loop constraints. To the best of our knowledge, our GA-based solver exhibits comparable or superior performance (in most cases) to other ''Type 1'' solvers reported in the literature. Furthermore, the proposed GA framework has been adapted successfully to solve considerably harder puzzle variants, e.g. where the pieces are taken from different puzzles and where the piece orientations and puzzle dimensions are not known. See [20, 21] for further details.

3. Proposed method

3.1 The fitness function

Each chromosome represents a suggested placement of all pieces. The problem variant at hand assumes no knowledge whatsoever of the original image and thus, the correctness of the absolute location of puzzle pieces cannot be estimated in a simple manner. Instead, the pairwise compatibility (defined below) of every pair of adjacent pieces is computed. A measure which ranks the likelihood of two pieces in the original image as adjacent is called compatibility. Let Cðxi; xj; RÞ denote the compatibility of two puzzle pieces xi; xj, where R 2 fl;r; a; bg indicates the spatial relation between these pieces, i.e. whether xj lies, respectively, to the left/right of xi, or above or below it. Cho et al. [4] explored five possible compatibility measures, of which the dissimilarity measure of Eq. (1) was shown to be the most discriminative. Pomeranz et al. [16] further investigated this issue and chose a similar dissimilarity measure with some slight optimizations. 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 color bands) should be minimal. Assuming pieces xi, xj are represented in normalized L*a*b* space by a K K 3 matrix, where K is the height/width of a piece (in pixels), their dissimilarity (if xj is to the right of xi, for example) is

3.2 The crossover operator

As noted above, a chromosome is represented by anðN MÞ matrix, where each matrix entry corresponds to the number of a puzzle tile. This representation is straightforward and lends itself easily to the evaluation of the fitness function described above. The main issue concerning this representation is the design of an appropriate crossover operator. A naive crossover operator with respect to the given representation will create a new child chromosome at random, such that each entry of the resulting matrix is the corresponding cell of the first or second parent. This approach yields usually a child chromosome with duplicate and/or missing puzzle pieces, which is of course an invalid solution to the problem. The inherent difficulty surrounding the crossover operator may well have played a critical role in delaying thus far the development of a state-of-the-art solution to the problem, based on evolutionary computation. Once the validity issue is rectified, one needs to consider very carefully the crossover operator. Recall crossover is applied to two chromosomes selected due to their high fitness values, where the fitness function used is an overall pairwise compatibility measure of adjacent puzzle pieces. At best, the function rewards a correct placement of neighboring pieces next to each other, but it has no way of identifying the actual correct location of a piece. Since the population starts out from a random piece placement and then improves gradually, it is reasonable to assume that some correctly assembled puzzle segments emerge over the generations. Taking into account the fitness function's inability to reward a correct position, we expect such segments to appear most likely at incorrect locations. A crossover operator should pass on ''good traits'' from the parents to the child, thereby creating possibly a better solution. Discovering a correct segment is not trivial; it should be regarded as a good trait that needs to be exploited. Namely, it should not be discarded, but rather trickled to the next generations. Thus, the crossover operator must allow for position independence, i.e. the ability of shifting entire correctly-assembled segments, in an attempt to place them correctly in their true location in the child chromosome. Once the position-independence issue is taken care of, one should address the issue of detecting correctly-assembled segments, which are possibly misplaced. What segment should the crossover operator pass on to an offspring? A random approach might seem appealing, but in practice it could be infeasible due to the enormous size of the solution domain of the problem. Some heuristics may be applied to distinguish correct segments from incorrect ones. In summary, a good crossover operator should address the issues of validity of child chromosomes, detection of supposedly correctly-assembled puzzle segments in parents, and position independence of these segments when passed on to an offspring.

Given two parent chromosomes, i.e. two complete different arrangements of all puzzle pieces, the crossover operator constructs a child chromosome in a kernelgrowing fashion, using both parents as ''consultants''. The operator starts with a single piece and gradually joins other pieces at available kernel boundaries. New pieces may be joined only adjacently to existing ones, so that the emerging image is always contiguous. For example, let A be the first piece chosen by the operator. The second piece B must be placed in a neighboring slot, i.e. left/right of, above or below piece A. Assuming it is placed just above A, then a third piece C must be assigned to one of the empty, previously mentioned slots (i.e. left/right of, or below A), or to any of the newly available slots that are left/right of, or above B. See Fig. 2 for an illustration of the above scenario. The operator keeps adding tiles from a bank of

available puzzle pieces until no pieces are left. Hence, every piece will appear exactly once in the resulting image.

Since the image size is known in advance, the operator can ensure that there is no boundary violation. Thus, since the operator uses every piece exactly once inside a bounded frame of the correct size, achieving a valid image is guaranteed. Figure 3 illustrates the above described kernel-growing process. A key property of the kernel-growing technique is that the location of every piece is determined only when the kernel reaches its final size and the child chromosome is complete. Before that, all pieces might be shifted, depending on the kernel's growth vector. The initial piece, for example, might end up at the lower-left corner of the image, if the kernel should grow only upwards and to the right, after this piece is first assigned. On the other hand, the very same piece might be placed ultimately at the center of the image, its upper-right corner, or any other location, for that matter. This change in a file's location is illustrated in Figs. 2 and 3; see, in particular, Fig. 3(f)–(g) of the kernel-growing process, where pieces keep shifting to the right due to the insertion of new pieces on the left. It is this important property which enables the position independence of image segments. Namely, a correctly assembled puzzle segment in a parent chromosome, possibly misplaced with respect to its true location, could be copied over during the crossover operation (while preserving its structure) to its correct location in an offspring. Having described how to construct a child chromosome as a growing kernel, we now turn to the following remaining issues: (1) Which piece to select from the available bank of tiles and (2) where to place it in the child (i.e. at which available neighboring slot). Given a kernel, i.e. a partial image, we can mark all the boundaries where a new piece might be placed. A piece boundary is denoted by a pair ðxi; RÞ, consisting of the piece number and a spatial relation. The operator invokes a three-phase procedure. First, given all existing boundaries, the operator checks whether there exists a piece boundary which both parents agree on, i.e. whether there exists a piece xj that is in the spatial direction R of xi in both parents. If so, then xj is added to the kernel in the appropriate location. If the parents agree on two or more boundaries, one of them is chosen at random and the corresponding piece is assigned. Obviously, if the piece is already in use, it cannot be (re)assigned, i.e. it is ignored as if the parents do not agree on that particular boundary. If there is no agreement between the parents on any piece at any boundary, the second phase begins. To understand this phase, we briefly review the concept of a best-buddy piece, first introduced by Pomeranz et al. [16]. Two pieces are said to be best buddies if each piece considers the other as its most compatible piece. Pieces xi and xj are said to be best buddies if where Pieces is the set of all given image pieces and R1 and R2 are ''complementary'' spatial relations (e.g. if R1 = right, then R2 = left and vice versa). In the second phase, the operator checks whether one of the parents contains a piece xj in a spatial relation R of xi which is also a best buddy of xi with respect to that relation. If so, the piece is chosen and assigned. As before, if multiple best-buddy pieces are available, one of them is chosen at random. If a best-buddy piece is found that is already assigned, it is ignored and the search continues for other best-buddy pieces. If no best-buddy piece exists, the operator proceeds to the final third phase, where it selects a boundary at random and assigns to it the most compatible piece available. To introduce mutation, the operator places, with low probability, in the first and last phase, an available piece at random, instead of the most compatible relevant piece available.

In summary, the operator repeatedly uses a three-phase procedure of piece selection and assignment, placing first an agreed-upon piece, then a best-buddy piece, and finally the most compatible piece available (i.e. the most compatible piece not yet assigned). An assignment is only considered at relevant boundaries to maintain the contiguity of the kernel-growing image. The procedure returns to the first phase after every piece assignment due to the creation of new prospective boundaries. Algorithm 2 provides a simplified description of the crossover operator (without mutation).

3.3 Rationale

We expect to see child chromosomes inherit ''good'' traits from their parent chromosomes in an effective GA framework. Since the algorithm encourages piece position independence, the trait of interest is captured by sets of neighboring pieces, rather than the locations of particular pieces. Correct puzzle segments correspond to the correct placement of pieces relatively to each other. The notion, for example, that piece xi is in a spatial relation R to piece xj is key to solving the jigsaw puzzle problem. Thus, although every chromosome accounts for a complete placement of all the pieces, it is the internal relative piece assignments that are examined and exploited. We assume that a trait common to both parents under consideration has propagated through the generations and is essentially the reason for their survival and selection. In other words, if both parents agree on a spatial relation, the algorithm would consider it true, and attempt to draw on it, with high

probability. Given the highly randomized nature of the first few generations, agreed-upon pieces selected during this stage could persist incorrectly through the latter generations. Note, however, that due to the kernel-growing algorithm, some agreed-upon pieces might be selected as most compatible pieces at other boundaries and thus be discarded for later use. For example, let pieces A and B be adjacent in both parents;

If the parents agree on no piece, the algorithm could place a piece from one of the parents (picked randomly)that is compatible with a kernel piece at an available boundary. Another possibility would be to place a best-buddy piece (if one exists) with respect to a kernel piece at an available boundary. Since neither two adjacent pieces in a parent (which could have been assigned at random) nor two best-buddy pieces alone are guaranteed to capture a correct match, the idea in the second phase is to combine these two elements. In other words, the fact that two pieces are both adjacent in a parent and are also best buddies is a good indication that they probably match. The rationale of the above two phases is based on the assumption that every chromosome contains some correct segments. Propagating such segments from parents to their children is at the heart of the GA. More specifically, if two parents each contain a correct segment, with an overlap between the segments, the idea is to copy the joint sub-segment to the child, in the first phase, and add pieces from both parents, independently, in the second phase, to obtain the union of the segments. Combining the segments into a larger correct segment would yield an enhanced child chromosome, thereby advancing the overall correct solution. The third phase ensures that the algorithm can always add a piece to the growing kernel (in case the conditions for phase 1 or phase 2 are not met), by placing the most compatible piece left with respect to an available boundary. In essence, this phase enables the GA to try many different greedy placements across each generation; placements that seem correct are likely to persist through the latter generations. This exemplifies the principle of propagation of good traits in the spirit of the theory of natural selection.

4. Results

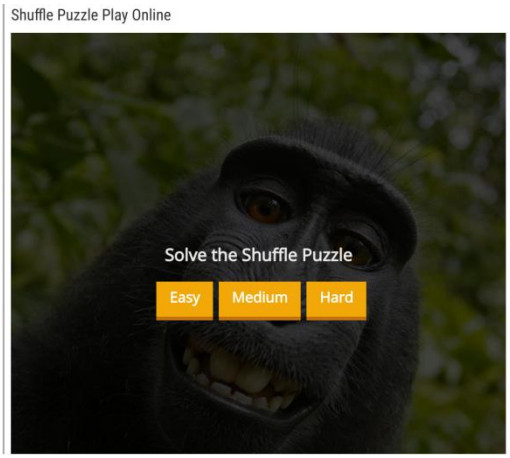

Fig.1 First step in the shuffle puzzle play online

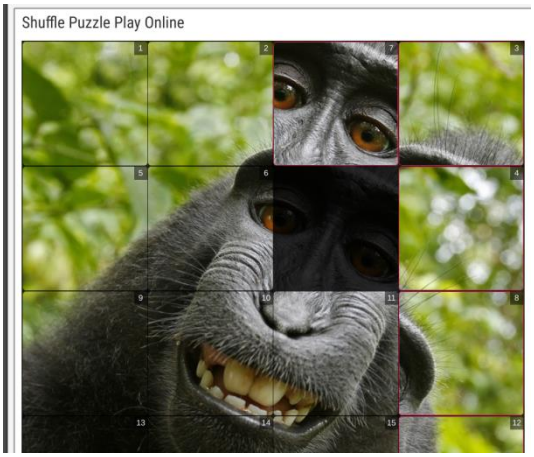

Fig.2 Arranging all the small parts into one image

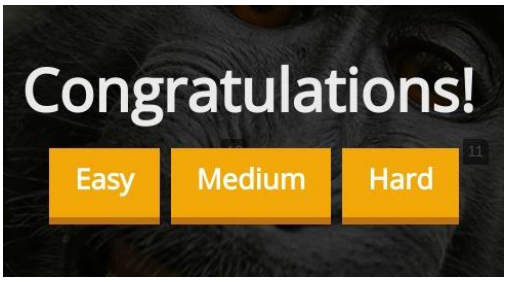

Fig.3 Message after completion of the puzzle

5. Conclusion

We introduced a new formulation for solving image jigsaw puzzle problems, the method PSQP – Puzzle Solving by Quadratic Programming. In our formulation, a solved puzzle is a one-to-one assignment of tiles to locations, according to an energy function. Since this is a hard combinatorial problem, we reformulate it as a quadratic programming approach, where we can find an approximate

solution by means of a gradient ascent algorithm. We compared PSQP to the current state-of-the-art and it provided superior results according to the used metrics. PSQP also has some advantages. First, it can solve puzzles not only with square tiles, but also with rectangular ones. Second, it is deterministic and although several parameter sets have to be tested, the method always yields the same results, while the current state-of-the-art method has to be executed several times to attain a certain accuracy. For the size of the puzzles tested, PSQP is faster, considering all the necessary executions in both methods. By analyzing the results, we observed that image puzzles that contain constant tiles are a weakness of PSQP. Constant tiles are difficult to order in a global sense, so we cannot consider them as a normal piece. We also observed that the right parameter set for each image may be determined a priori by analyzing the image and tiles properties. These two observations will be included in future studies.

References

1. T. Altman, Solving the jigsaw puzzle problem in linear time. Appl. Artif. Intell. Int. J. 3(4), 453–462 (1989)

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

3. S. Cao, H. Liu, S. Yan, Automated assembly of shredded pieces from multiple photos, in IEEE International Conference on Multimedia and Expo (2010), pp. 358–363

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

5. T. Cho, M. Butman, S. Avidan, W. Freeman, The patch transform and its applications to image editing, in IEEE Conference on Computer Vision and Pattern Recognition (2008), pp. 1–8

6. A. Deever, A. Gallagher, Semi-automatic assembly of real cross-cut shredded documents, in IEEE International Conference on Image Processing (2012), pp. 233–236

7. E. Demaine, M. Demaine, Jigsaw puzzles, edge matching, and polyomino packing: connections and complexity. Graphs Comb. 23, 195–208 (2007)

8. H. Freeman, L. Garder, Apictorial jigsaw puzzles: the computer solution of a problem in pattern recognition. IEEE Trans. Electron. Comput. EC–13(2), 118–127 (1964)

9. A. Gallagher, Jigsaw puzzles with pieces of unknown orientation, in IEEE Conference on Computer Vision and Pattern Recognition (2012), pp. 382–389

10. D. Goldberg, C. Malon, M. Bern, A global approach to automatic solution of jigsaw puzzles. Comput. Geom. Theory Appl. 28(2–3), 165–174 (2004)

11. E. Justino, L. Oliveira, C. Freitas, Reconstructing shredded documents through feature matching. Forensic Sci. Int. 160(2), 140–147 (2006)

12. D .Koller, M. Levoy, Computer-aided reconstruction and new matches in the forma urbisromae. BullettinodellaCommissioneArcheologicaComunale di Roma. Supplementi. 15, 103–125 (2006)

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

14. M. Marques, C. Freitas, Reconstructing strip-shredded documents using color as feature matching, in ACM Symposium on Applied Computing (2009), pp. 893–894

15. A.Q. Morton, M. Levison, The computer in literary studies, in IFIP Congress (1968), pp. 1072–1081

16.https://dailypuzzlecheats.com/shuffle-puzzle-play-online

17. https://dailypuzzlecheats.com/