Search Time when Solving Random Jigsaw Puzzles with a Planted Solution

by Jeff Edmonds,[1] *Alex Edmonds,*[2] *and Ian Mertz*[3]

Abstract

John Tsotsos asked his AI students to work on having an AI solve a jigsaw puzzle, though it is NP-complete. This paper considers the search time of the recursive backtracking algorithm for solving it when the puzzle pieces are generated randomly with a planted solution. Feeling that a jigsaw puzzle should have a particular image when completed, we set the probability that two pieces fit together locally to be just small enough that with high probability the planted solution is unique. We were surprised to see that in this case the recursive backtracking algorithm which expands a rectangle from a corner has expected $1 + \Theta(\epsilon)$ branching width. In contrast, this width is exponential if it attempts to put together just the edge pieces into a frame, build a triangle from a corner, or build a block in the middle. If the probability of pieces fitting together is increased just to the point that there is an exponential number of complete solutions, then no matter which order the pieces are put together it takes an exponential amount of time to find one of these exponentially many solutions. It is also interesting that for the different orders of completing the puzzle, there are very different "reasons" for the exponential blowup. We coded the algorithm and it ran just as described.

Keywords: jigsaw puzzle, random planted solution, recursive backtracking algorithm

Introduction

The problem is to put together a Jigsaw puzzle. The player is given $n = W \times L$ unit square pieces with an integer assigned to each edge of each piece. The pieces must be assembled on a $W \times L$ grid so that for any two adjacent squares the integers on the edges connecting the squares are equal[4]. Erik and Martin Demaine[5] proved that this problem is NP-complete. For this reason, we particularly consider puzzles which are randomly generated with a planted solution. More specifically, starting with $n = W \times L$ pieces already put together in a $W \times L$ grid, we randomly choose for each pair of connected pieces the same one of $q = \sqrt{\frac{4n}{\epsilon}}$ integers so that they do fit together. Then we mix up these pieces, knowing that there is at least one solution. We will see that locally each piece connects to $\Theta(\sqrt{\epsilon n})$ others while globally there is a $1 - \Theta(\frac{\epsilon^{1.5}}{\sqrt{n}})$ probability that the planted solution is the unique solution. The algorithm considered is recursive backtracking to fill the board in a prescribed order $L_n = \langle \ell_1, \ell_2, \ldots, \ell_n \rangle$, where $\ell_t = \langle i, j \rangle$ gives the coordinates on the table to fill a piece at time t. When multiple pieces legally fit a location, the algorithm branches. This forms a *branching tree* where the nodes at depth t have an outgoing edge for each piece that legally fits into location ℓ_t given that pieces in the path from the root to the node have been placed in $L_{t-1} = \langle \ell_1, \ell_2, \ldots, \ell_{t-1} \rangle$. Given how many pieces fit locally together, this tree at each node will branch a lot. A branch terminates when a mistake is found because there are no remaining pieces that fit the current location. For most orders L_n of doing this, the width of this tree grows exponentially with t for much of the algorithm and then resolves to the unique solution in the end with $t = n$. This is the case if the recursive backtracking algorithm first attempts to put together just the edge pieces into a frame, if it builds a block in the middle, or if it grows a

[1]York University, Canada. jeff@cse.yorku.ca. Supported in part by NSERC Canada.
[2]University of Toronto. edmonds@cs.toronto.edu
[3]University of Toronto. mertz@cs.toronto.edu
[4]It would have been equivalent to have pieces connect with a male/female connection. See [5]

triangle from a corner of the puzzle. These three different orders have three different reasons for this growth. Surprisingly however, if L_n is the order which starts in a corner and fills in a growing rectangle, then the expected total width of the branching tree at each point in time is only $1+\Theta(\epsilon)$. This means that one about every ϵ steps of t, an incorrect piece locally fits into L_t, but the mistake is quickly resolved leaving only the branching tree consisting of only a single path towards the construction of the unique solution. With a good data structure for finding matches, this gives a linear expected time algorithm for solving this random instance. This we coded and it ran just as described.

For completion we should consider what happens when the probability of pieces fitting together is increased just so that there is an exponential number of complete solutions. No matter which order the pieces are put together the branching tree width is going to be exponential. Instead of trying all possibilities in parallel, it is better to use recursive backtracking to do a depth first search of this branching tree in hopes that it will find one of these many solutions quickly. Still it take an exponential amount of time to find one of these exponentially many solutions. This expected time decreases as the number of complete solutions increases further.

1 Previous Work

Jigsaw puzzles [19, 20] are perhaps the most popular form of puzzle. The original jigsaw puzzles of the 1760s were cut from wood sheets using a hand woodworking tool called a jig saw, which is where the puzzles get their name.

The shape-matching and the image processing communities, when considering jigsaw puzzles, focus exclusively on how to determine if two pieces fit together locally [1, 3, 4, 7, 14, 13, 15, 18, 21]. They rely on the assumption that if two shapes locally fit together, then they will be together in the final global solution. In contrast, this paper assumes that the problem of which pieces fit together locally has already been solved, assumes that each piece locally fits with many pieces, and worries about combinatorial blowup in searching for a global solution.

It is taught in graduate courses in complexity theory and discussed in Martin Gardner's book of games [9] that the solving a jigsaw puzzle is undecidable if the pieces can be used an arbitrarily large number of times. This was proved in 1966 by Berger [2]. The basic idea is to have the square solution encode an accepting computation of a TM. Simply having this TM nondeterministically solve SAT, the problem is shown to be NP-complete if it is changed so that the instance specifies $\Theta(n)$ pieces and the goal is to put together n of them into a square. This result is also in an unpublished 1977 work of Garey, Johnson, and Papadimitriou [12], and used in Levins theory of average-case completeness [16]. It is a bigger challenge to prove that the problem is NP-complete even when each piece must be used exactly once. A proof given in [1] reduces it to the set partition problem [11]. Given integers $x_1, \ldots, x_n$, we want to learn whether they can be partitioned into two sets that sum to the same value. To do this we make for each $i \in [n]$ a puzzle piece that has height 1 and length x_i. The partitioning is possible iff these can be put into a frame of height 2 and length $\frac{1}{2}\sum_i x_i$. This is unsatisfying for two reasons. The first is that set partition can be approximated to an accuracy of $1+\epsilon$ in time $\Theta(\frac{n^3}{\epsilon})$. The second is that though the size of the description of the input $\langle x_1, \ldots, x_n \rangle$ is $\sum_i \log(x_i)$, the actual size of the puzzle instance is exponentially bigger, namely $\sum_i x_i$. In contrast, in 2007, Erik and Martin Demaine [5] proved that the version of the jigsaw puzzle problem considered here is NP-complete, namely each of the n pieces is a unit square and all n pieces need to be placed so that their sides match. They reduce it to the 3-partition problem, which is strongly NP-complete [10, 11]. Given $3n$ integers, the task is to partition them into triples each with the same sum.

The problem of solving an NP-complete problem on random instances with a planted solution has been considered in many other settings. Just to mention two, Frieze and Kannan [8] consider the planted clique problem and Flaxman [6] planted 3-SAT.

2 The Puzzle Problem

Given parameters n and ϵ, our puzzle is randomly generated with a planted solution as follows. We start with a square grid of $n = \sqrt{n} \times \sqrt{n}$ unit square pieces[5]. Let $m = 4\sqrt{n}$ be the number of edge pieces. For each pair of adjacent pieces, one of $q = \sqrt{\frac{4n}{\epsilon}} = \frac{m}{2\sqrt{\epsilon}}$ integers is assigned randomly to both of the *connection-sides*. The assigned integer represents the connecting feature between the two sides which could arise from the shape or from the picture on the pieces. (Nothing significant would change if we matched a male edge with its female complement.) We chose q so that, with high probability, the planted solution is the unique solution. Once the pieces are constructed, they are randomly permuted and rotated and presented to the algorithm. The algorithm must arrange the pieces on the $W \times L$ grid so that the assigned integers of connected sides match.

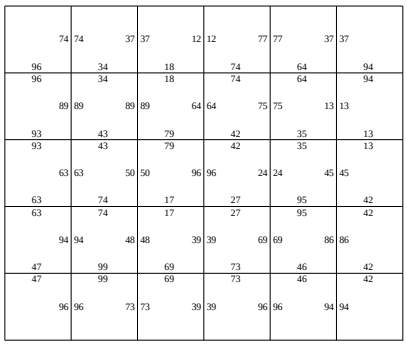

Figure 1: A jigsaw puzzle with random connection between the pieces.

3 The Algorithm

The following parallel or recursive backtracking algorithm is considered. The algorithm depends on a predetermined ordering $L_n = \langle \ell_1, \ell_2, \ldots, \ell_n \rangle$ in which the locations on the board are filled. Let $L_t = \langle \ell_1, \ell_2, \ldots, \ell_t \rangle$ denote the prefix filled at time t.

We prove the following.

Theorem 1 (Unique Solution): *The expected number of pieces that connect to a given side of a non-edge piece is $2\sqrt{\epsilon n}$. However, with probability $1 - \Theta(\epsilon)$, the planted solution is the unique solution. With probability $\Theta(\epsilon)$, a single piece can be rotated. With probability $\Theta(\epsilon^2)$, there will be identical pieces. If we consider two solutions to be identical if their only differences are these or you condition the distribution to have no rotatable or identical pieces, then the solution is unique with probability $1 - \Theta(\frac{\epsilon^{1.5}}{\sqrt{n}})$. Lemma 2 and Figure 3.A show the types changes that will most likely appear.*

[5]We could more generally consider the rectangular grid $n = W \times L$, but then notation and constants would be unnecessarily complicated. As long as $W, L \geq 3$, the results would be the same except that the number of edge pieces $m = 2W + 2L - 4$ would be different.

$\ell_1, \ldots, \ell_n$ specifies the order locations are filled.
loop
    $\langle$***loop−inv***$\rangle$**:** You have grown a subsolution for locations specified in $L_{t-1} = \langle \ell_1, \ell_2, \ldots, \ell_{t-1} \rangle$.
    The algorithm attempts to place a piece in the location given by ℓ_t.
    If there is a unique piece that fit in that location
        place it and continue
    else if there are more than one such pieces
        branch and continue with each option in parallel.
    else if no piece fits in this spot,
        terminate this computational branch.
end loop

Figure 2: We choose to present this algorithm using parallel threads but the same branching tree could be traversed using recursive backtracking. The number of threads/stackframes at level t of this tree will be the expected number ways $N(t)$ of legally filling in the locations $L_t = \langle \ell_1, \ell_2, \ldots, \ell_t \rangle$ with pieces. If there is only one valid full solution, then one expects to have to search half the tree even if this is done with recursive backtracking.

Theorem 2 (Build the Edge Alg): *If this parallel/recursive backtracking algorithm begins by filling in the edge pieces along one side of the puzzle, then the search space of possible subsolutions blows up at a rate of* $2^{\Theta(\sqrt{\epsilon}t)}$ *and then the number of valid ways to complete the edge pieces is* $e^{\Theta(\sqrt{\epsilon}m)}$ *where* $m = \Theta(\sqrt{n})$ *is the number of edge pieces. See Figure* **??***.A.*

Theorem 3 (Shape in Middle Alg): *If this algorithm begins with filling in some shape in the middle of the puzzle, then the size of this search space is* $n^{\Theta(\log n)}$*. In fact, when the shape has grown to being a* $\log n \times \log n$ *square, there will be this many subsolutions with none of the connections being the planted one. See Figure 4.B*

Theorem 4 (Triangle in Corner Alg): *If this algorithm grows a triangle from a corner piece, then the size of this search space is* $2^{\Theta(\epsilon\sqrt{n})}$*. See Figure 3.B. More specifically, if the shape grown from the top-left corner of the puzzle has* $w' \leq \mathcal{O}(\sqrt{n})$ *bottom-right corners, then we expect* $\Theta(\epsilon w')$ *of them to have two pieces fit in them (see Figure 3.B.h′) giving* $2^{\Theta(\epsilon w')}$ *ways of choosing which combination of these pieces to use.*

Theorem 5 (Rectangle in Corner Alg): *If this algorithm grows a rectangle (of width and height at least two) from a corner piece, then the expected number of threads/stackframes at any one level* t *will be* $1 + \Theta(\epsilon)$*. Lemma 2 and Figure 3.A.g & h. show the types of changes that will most likely appear. This theorem is still true if in addition to the planted solution,* $\Theta(n)$ *more pieces are choose randomly and pieces can be reused.*

Theorem 6 (Many Solutions): *If the probability of two pieces connecting is* $\frac{1}{q'} = \frac{2}{\epsilon} \cdot \frac{1}{q}$*, where* $\frac{1}{q}$ *is that defined above at which the planted solution is w.h.p. unique, then the number of solutions becomes exponential. Instead of trying all possibilities in parallel, it is better to use recursive backtracking to do a depth first search of this branching tree in hopes that it will find one of these many solutions quickly. Still it take an exponential amount of time to find one of these exponentially many solutions. This expected time decreases as the number of complete solutions increases further.*

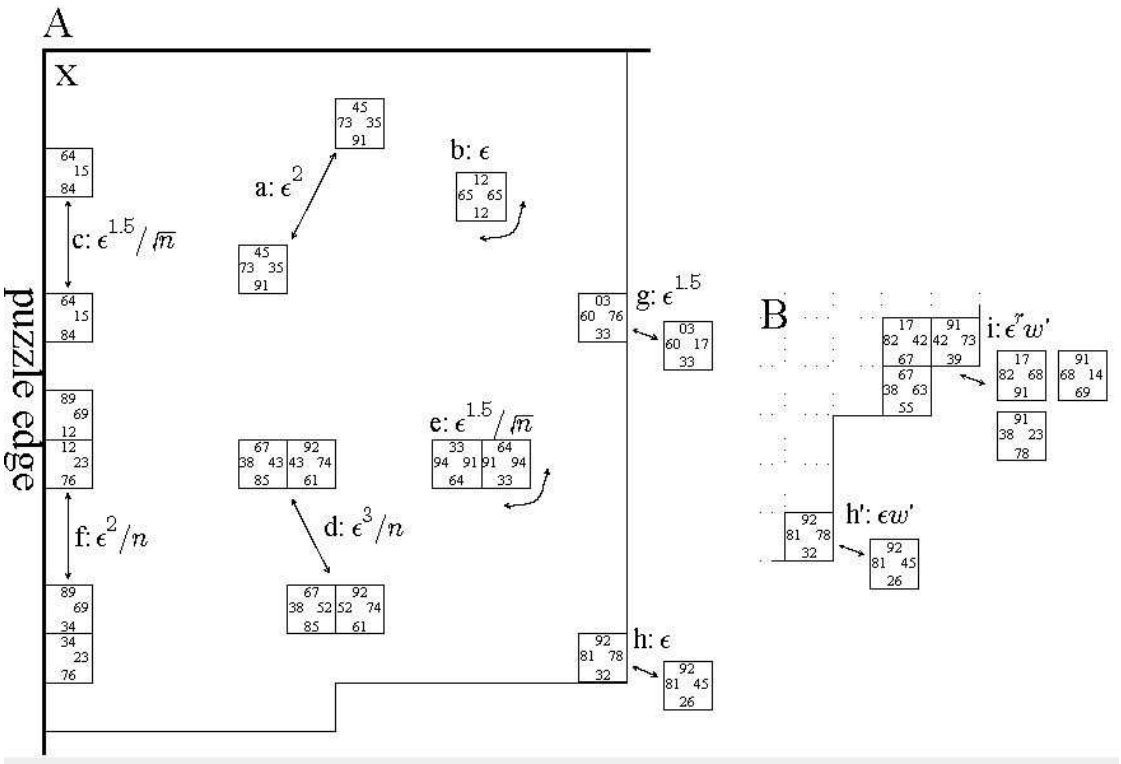

Figure 3: (A) shows the most likely local changes producing different subsolutions for a rectangle being grown from a corner. (B) the same for a triangle being grown from the corner. For each type of change, the the probability of it occurring is given. See Lemma 2 for the proof.

4 Running Time and Number of Subsolutions

The algorithm's running time is $Time(n) = \sum_{t=1}^{n} N(t) \cdot Find(n)$ where $N(t)$ is the expected number of threads/stackframes at level t of this tree, which is the number ways of legally filling in the locations $L_t = \langle \ell_1, \ell_2, \ldots, \ell_t \rangle$ with pieces and $Find(n)$ is the time to check which of the unused pieces fit into a given location. For example, in the case that two pieces fit only if they are supposed to then all subsolutions are unique, i.e. $N(t) = 1$ and $Time(n) = \sum_{t=1}^{n} 1 \cdot \mathcal{O}(t) = \mathcal{O}(n^2)$. In our case, we will have $N(t) = 1 + \Theta(\epsilon)$ and assume that we have a data structure so that $Find(n) = \mathcal{O}(1)$, giving $Time(n) = \Theta(n)$.

5 Dynamics of the Recursive Backtracking Algorithm

Lemma 1 *The following table gives the expected number of pieces C that can be correctly placed in a given location given the stated neighboring pieces have already been placed correctly/incorrectly. See Figure* **??**.

	Type	*Neighbors*	*Previous*	*Math*	*Expected Number*
a:	*nonedge*	*right of A*		$1 + 4n \cdot \frac{1}{q}$	$\Theta(\sqrt{\epsilon n})$
b:	*edge*	*right of A*		$1 + m \cdot \frac{1}{q}$	$1 + 2\sqrt{\epsilon}$
c:	*nonedge*	*right of A & below B*	*correct*	$1 + 4n \cdot \frac{1}{q^2}$	$1 + \epsilon$
d:	*nonedge*	*right of A & below B*	*incorrect*	$4n \cdot \frac{1}{q^2}$	ϵ

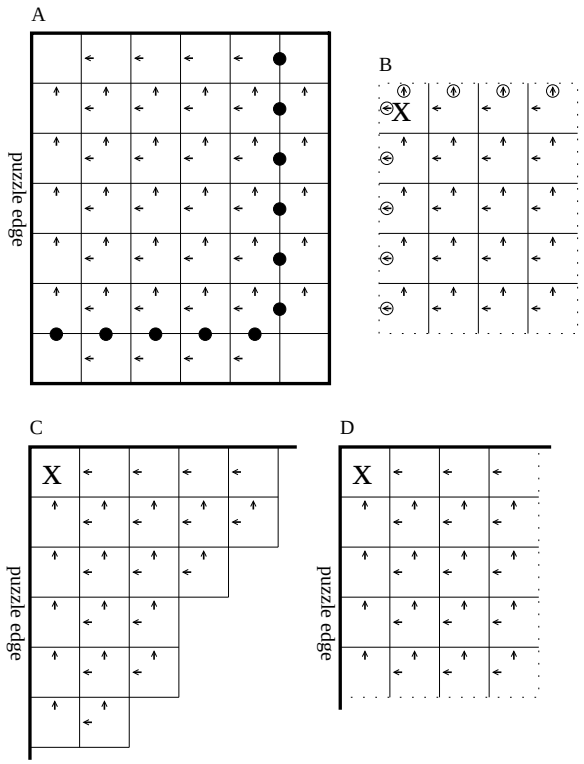

Figure 4: Each of the four figures specifies the set of t locations L_t that the algorithm has filled during the first t time steps. The X specifies the location of the first piece placed. This piece is assumed to be placed correctly. Each piece location in L_t "owns" the connection-side to its left and above it as indicated by the arrows. The edge piece locations only "own" one if its sides. The circled arrows are connection-sides that were over counted. In contrast, the solid circles are ones that were not counted. Fig (A) shows the entire puzzle, (B) is a L_t growing in the middle of the puzzle, (C) is a triangle growing from a corner, and (D) is a rectangle growing from a corner.

Proof of Lemma 1: For situation (a), let $\alpha \in \mathcal{F}$ denote the connecting feature on the right side of A. We must find a piece B that has the mirrored α feature on one of its four sides. There are n such pieces B, 4 ways to rotate it, and one connection to be matched, giving the expected number $4n \cdot \frac{1}{q} = 4n \cdot \sqrt{\frac{\epsilon}{4n}} = \Theta(\sqrt{\epsilon n})$. The plus one is because the planted solution provides one guaranteed connection. The computations for the others are similar except for (d). Here if A and B are not placed as in the planted solution, then there is not the plus one guaranteed planted piece. ■

Having $\Theta(\sqrt{\epsilon n})$ pieces fitting locally to a given piece implies lots of branching for the recursive backtracking algorithm. However, if the algorithm fills in the puzzle row by row, our next step places a piece on a location to the right of some piece A and below some piece B. See Figure **??**.B. If both A and B are placed correctly, then because of the planted solution, there is definitely the planted piece C that can go there next. More over Lemma 1.c gives that with probability ϵ there is an addition incorrect piece C' that also fits. Hence, for approximately $\frac{1}{\epsilon}$ such steps, the algorithm forks. Because C' is placed incorrectly, there is not a planted piece that fits and with only probability ϵ there is a piece D' that happens to fit. In this way, algorithm quickly detects that this branch is incorrect and terminates it. This is why the number of living branches hovers around one. A first problem is that completing the first row requires exponential growth. A solutions to

this is to instead grow a rectangle. A second problem is that when the last piece placed in the current row has an error, this error may not be detected until one (and possibly more) additional rows have been completed. It turns out that when one builds a rectangle from a corner, this is not a problem, but when one builds it in the middle of the puzzle then even when there is not planted solution, there is exponential branching growth.

Most people when putting together a jigsaw puzzle build a line following an edge of the solution. The dynamics in this case are different. See Figure **??**.A. As before, after placing piece A, there is definitely the planted piece C that can go next to it and with probability $\Theta(\sqrt{\epsilon})$ there is an additional piece C'. But with a line of edge pieces it is harder to detect when there is an error. The reason is that both C and C' have a planted piece that belongs next to it. Branching every approximately $\Theta(\frac{1}{\sqrt{\epsilon}})$ such pieces placed, grows the tree at the rate of $2^{\Theta(\sqrt{\epsilon}t)}$. If this would continue for all m edge pieces the branching width would be $2^{\Theta(\sqrt{\epsilon}m)}$. We would think the main reason for finding a mistake would be that the piece you want to place next has already be placed earlier. If one was needing the m edge pieces in a random order, then one would expect to wait $\sqrt{m}$ until a piece is needed twice. This suggests growth to $2^{\Theta(\sqrt{\epsilon m})}$. If these were the only dynamics, then provably there would be a unique way of putting together the edge pieces. But there is an additional dynamic that allows this number be at least $e^{\Theta(\sqrt{\epsilon}m)}$. See Theorem 2.

This dynamic is that there are dependencies between the events that non-planted pieces fit together. Suppose, for example,that $A \cdot BCDE \cdot F$ and $W \cdot XY \cdot Z$ are two sequences of edge pieces within the planted solution. Then with probability $\frac{1}{q^2}$, the connections happen to match so that $A \cdot XY \cdot F$ is also a valid sequence. It follows for free that $W \cdot BCDE \cdot Z$ is also valid sequence. This cycle could also have lengths longer than two. The graph in Figure 5.D helps us to better understand these dependencies.

6 The Number of Subinstances without a Planted Solution

When there is not a planted solution, the distribution independently selects four integers for each puzzle piece instead of selecting the same integer for two pieces if they touch in the planted solution. Let $N'(L_t)$ denote the expected number of valid subsolutions, i.e. number of valid ways of arranging t of the n pieces into the t locations, $L_t = \langle \ell_1, \ell_2, \ldots, \ell_t \rangle$. Computing $N'(L_t)$ is fairly straightforward, namely multiply the number of possible subsolutions with the probability that such a subsolution is valid. The first is $\binom{n}{t} t! 4^t \approx (4n)^t$ because this is the number of ways of choosing, arranging, and rotating the t pieces. Note that each such location $\ell_i \in L_t$ contributes a factor of $4n$ to this count. The probability that such subsolution is valid is $(1/q)^{\#\text{ of integers to match}}$. We will say that each location "owns" its left and top connection-sides which are both valid with probability $\frac{1}{q^2}$. It's bottom and right connection-sides are counted by the location below and that to its right. See Figure 4. Hence, its total contribution to the expected number of valid subsolutions $N'(L_t)$ is the multiplicative factor $4n \times \frac{1}{q^2} = 4n \times \sqrt{\frac{\epsilon}{4n}}^2 = \epsilon$, causing $N'(L_t)$ to decreases with ϵ^t as the number of locations filled grows. Each edge piece location contributes m to the number of solutions, because this is the number of edge pieces and their rotation is fixed. We will say that an edge location "owns" only one of its connection-sides, i.e. adjacent to the edge of the puzzle. Its other two side connections will be counted by other locations. Again see Figure 4. Hence, an edge piece location contributes a multiplicative factor of $m \cdot \frac{1}{q} = 2\sqrt{\epsilon}$ to $N'(L_t)$. When growing a shape from the corner (see Figure 4.C and D) all the required connections have been counted exactly once each. However, for the full puzzle, the $W + L$ connections between the edge pieces on the bottom and right edge of the puzzle and their interior pieces still need to be included in the probability calculation. See

Figure 4.A. This gives $N'(L_n) \approx (\epsilon)^n \times (2\sqrt{\epsilon})^m \times \left(\frac{1}{q}\right)^{W+L} = \epsilon^{\Theta(n)} n^{-\Theta(m)}$. On the other hand, if a shape is grown in the middle of the puzzle (see Figure 4.B) then we over counted factors of $\frac{1}{q}$ when computing the probability.

Proof of Theorem 3 (Shape in Middle Alg): Consider the recursive backtracking algorithm which grows a $t = w \cdot \ell$ piece rectangle in the middle of the puzzle. As seen in Figure 4.b, the number of connections over counted is $Over(L_t) = w + \ell \geq 2\sqrt{t}$. This gives $N(L_t) \geq N'(L_t) \geq \epsilon^t \cdot q^{w+\ell} = \epsilon^t \cdot (4n)^{2\sqrt{t}}$ which is maximized to $n^{\Theta(\log n)}$ by setting $t = \Theta(\log^2 n)$ giving the stated result. Note that when the square grows any larger, the ϵ^t effect of the area dominates killing any reasonable possibility of a non-planted solution. ■

7 Counting the Number of Objects that can be Swapped.

Having the planted solution means that there maybe many others that are similar. Hence, it is important to count the number of local changes that can be made to the planted subsolution. These prove the lower bounds on the number of valid subsolutions needed for Theorems 1 and 5.

Lemma 2 *Figure 3.A and B shows a number of shapes of blocks and for each gives the probability that there is such a block within the planted solution that can be swapped with another or rotated.*

Proof of Lemma 2: For Figure 3.A.a, we consider the probability that two pieces of the puzzle are identical and hence can be swapped. There are at most n^2 such pairs of pieces and 4 ways of rotating the second to make it the same as the first and four edge connections that must be the same giving the expected number identical pairs to be $4n^2 \cdot \frac{1}{q^4} = 4n^2 \cdot \left(\frac{\epsilon}{4n}\right)^2 = \frac{\epsilon^2}{4}$. For (g), there are at most m locations along the side of L_t and at most $4n$ pieces to rotate to place there and three edges to match giving the expected number of such swaps be $4mn \cdot \frac{1}{q^3} = 16\sqrt{n}n \cdot \left(\frac{\epsilon}{4n}\right)^{1.5} = \Theta(\epsilon^{1.5})$. Similarly, (b): $n \cdot \frac{1}{q^2} = \Theta(\epsilon)$; (c): $m^2 \cdot \frac{1}{q^3} = \Theta(\frac{\epsilon^{1.5}}{\sqrt{n}})$; (d): $8n^2 \cdot \frac{1}{q^6} = \Theta(\frac{\epsilon^3}{n})$; (e): $2n \cdot \frac{1}{q^3} = \Theta(\frac{\epsilon^{1.5}}{\sqrt{n}})$; (f): $m^2 \cdot \frac{1}{q^4} = \Theta(\frac{\epsilon^2}{n})$; (g): $4mn \cdot \frac{1}{q^3} = \Theta(\epsilon^{1.5})$; (h): $4n \cdot \frac{1}{q^2} = \Theta(\epsilon)$; (h′): $4w'n \cdot \frac{1}{q^2} = \Theta(\epsilon w')$; (i): $r = w'(4n)^r \cdot \frac{1}{q^{2r}} = \Theta(\epsilon^r w')$. Here $w' \leq \mathcal{O}(\sqrt{n})$ is the number of bottom-right corners in the triangle shape grown from the top-left corner of the puzzle and r is the number of pieces in the structure that we are trying to place into such a corner. ■

8 An Upper Bound on the Number of Subsolutions

This section proves the upper bounds needed for Theorems 1 and 5, by bounding the expected number $N(L_t)$ of subsolution allocations of pieces to the locations specified in L_t.

Lemma 3 *a) When L_t is a rectangle grown from the corner like Figure 4.c, $N(L_t) \leq 1 + \mathcal{O}(\epsilon)$.*
b) When L_t is the full puzzle with no duplicate pieces or self similar pieces, then $N(L_n) \leq 1 + \mathcal{O}(\frac{\epsilon^{1.5}}{\sqrt{n}})$.

To prove this, one has to make sure that there are not many complex methods for forming a new subsolution from the planted solution. Such a method would not break the planted solution into individual pieces from which to build the subsolution, because this requires far too many edge connections to be correct. Instead the planted solution is broken into what are called *meta-pieces*, which are then rearranged to form the new subsolution. Because the inner meta-piece connection-sides are guaranteed to be valid by the planted solution, we only have to worry about the connection-sides between the meta-pieces. The locations in L_t of these connection-sides will be called the solution's *framework*. See Figure 5. Stated in a slightly different way, we construct (count)

subsolutions for locations L_t as follows. The edge between two locations for pieces in L_t we will refer to as a *connection-location*. Two *connection-sides* of pieces will meet here in a solution. First we form a *framework* F consisting of a set of connection-locations within L_t that partitions its t piece locations into connected components that we will refer to as *meta-locations*. A solution S is formed by cutting out of the planted solution a *meta-piece* for each such meta-location. The expected number of subsolutions for L_t can then be computed as $N(L_t) = \sum_R N_F(R) \cdot N_S(F) \cdot prob(S)$, where the sum is over the parameters R of a subsolution, $N_F(R)$ is the number of frameworks F consistent with these parameters R, $N_S(F)$ is the number of placements S of the pieces within L_t consistent with F, and $prob(S)$ is probability that all the necessary connection-sides in S are valid. For placement S to be valid, each of the e non-planted connections between sides must happen to be correct. Assuming independence the probability of this is $\frac{1}{q^e}$. However, there is some dependencies between these events.

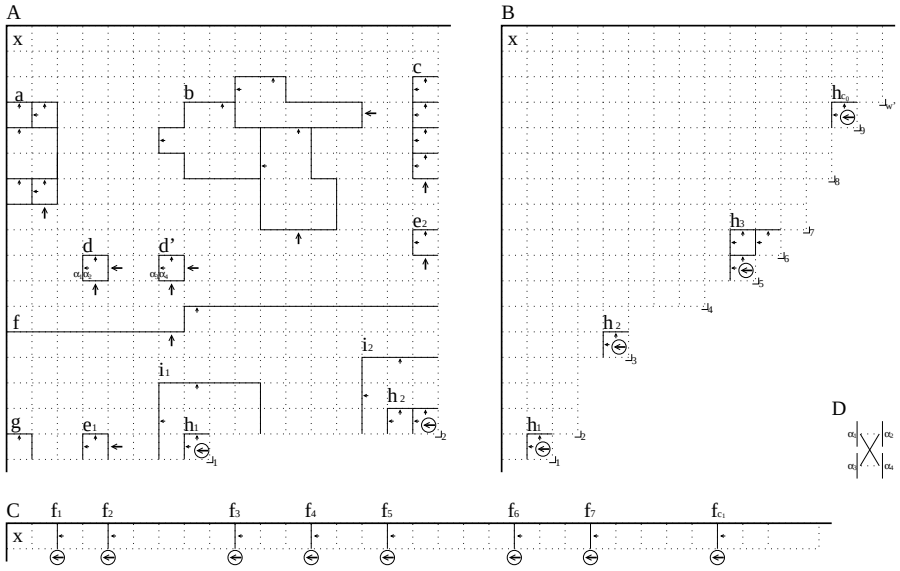

Figure 5: Each of the three figures uses dotted lines to mark off the set of t locations L_t that the algorithm has filled during the first t time steps. Figure A is for a rectangle growing from the corner, (B) a triangle from the corner, and (C) an edge from the corner. The X specifies the location of the first piece placed. The solid lines specifies a framework within this L_t. Pairs of pieces placed across such a line need to be not next to each other in the planted solution, while the pieces connected by dotted lines merge into meta-pieces pieces which appear together in the planted solution. The small arrows specify for each meta-piece the sides that the piece "owns". The large arrows specify for each framework-component which sides it "owns". An arrow is circled if being in c_0 it moves but does not have an edge to pay for it. Luckily with (A) $c_0 \leq 2$. (D) shows a graph with nodes being connection-sides, solid edge between two if they touch in the planted solution, and a dotted edge if they touch in the subsolution.

Consider some framework F. Let e denote the number of connection-locations in it. Partition these into connected *framework-components* by viewing them as edges of a graph. For example in Figure 5.A, the framework-components are labeled A.a through A.i_2. Note the sides of the puzzle do not connect these components. Components A.i_1 and h_1 are interesting in that one is contained in the other. Note that component A.a separates out five meta-locations, A.b three, and the piece locations between these components A.a and A.b form one big meta-location whose bottom edge

is the framework-component A.f. A framework-component will be classified with $d \in \{0.0, 0, 1, 2\}$ if it can be translated within L_t through at most d dimensions once their shape and rotations have been fixed and hence can be in at most 1, 2, $\sqrt{|L_t|}$, or $|L_t|$ places. Then let c_d denote the number of such framework-components. For example, $c_{0.0} = 3$ includes A.g, A.i$_1$ and A,i$_2$ because they cannot be placed anywhere else, while $c_0 = 2$ amusingly includes A.h$_1$ and h$_2$ because these two components could be swapped with each other. $c_1 = |\{$A.a, c, e$_1$, e$_2$, and f$\}|$, and $c_2 = |\{$A.b, d, and d'$\}|$. We can now bound the number of frameworks.

Lemma 4 *If L_t is a rectangle growing from a corner, the number of frameworks F consistent with parameters $R = \langle e, c_{0.0}, c_0, c_1, c_2\rangle$ is $N_F(R) \leq 8^e \cdot 2^{c_0}\ \sqrt{n}^{c_1}\ n^{c_2} \cdot \frac{1}{c_0!c_1!c_2!}$.*

Proof of Lemma 4: We bound the number of frameworks by describing one with $\log_2(8^e \cdot 2^{c_0}\ \sqrt{n}^{c_1}\ n^{c_2})$ bits. This encoding will first tell the shapes and rotations of each framework-component as a rigid structure and then translate each of them within L_t. Consider a component with e' of the connection-locations. Define shape and rotation by starting one of its connection-locations placed in L_t and growing it in a breadth first search order. For each of the e' connection-locations in the framework, we use 3 bits to specify which of the remaining 3 Manhattan sides adjacent to this side are also sides of the framework. Doing this for each of the components uses at most $3e = log_2 8^e$ bits. The next step is to place each component within L_t. As described when defining these counts, $c_{0.0}$ is the number of components than can go in at most one place within L_t, c_0 in at most two places, c_1 at most $\sqrt{n}$, and c_1 at most n. This is why $\log_2(2^{c_0}\ \sqrt{n}^{c_1}\ n^{c_2})$ more bits specifies the locations of these framework-components. The extra $\frac{1}{c_0!c_1!c_2!}$ factor arises because our description leads to the same framework F if the components were presented in any of the $c_0!c_1!c_2!$ orderings. ■

Given a framework F giving a set of meta-locations, a placement S of pieces within L_t is formed by cutting out of the planted solution a meta-piece for each such meta-location. First the meta-locations will be classified by the number $d \in \{0, 1, 2\}$ of dimensions through which they can be translated and hence can be in rotated and moved to at most 4, $4\sqrt{n}$, or $4n$ places when choosing where to cut them out of the planted solution. More specifically, a meta-location will be considered to move in $d = 0$ dimensions if it contains a corner piece, $d = 1$ if it contains an edge piece but no corner piece, and $d = 2$ if it contains no edge pieces. Let r_0, r_1, and r_2 denote the number of each type. For example, there are $r_1 = 6$ $d = 1$ meta-locations in Figure 5.A and they all touch the left edge of the puzzle, while framework-component (b) separates out three $d = 2$ meta-locations. Assuming independence in choosing where to cut out the meta-pieces, the number of placements S of the pieces within L_t consistent with F is $N_S(F) \leq 4^{r_0}\ (4\sqrt{n})^{r_1}\ (4n)^{r_2}$. Dependencies between the choices, however, allow us to obtain a slightly better upper bound. Because of its complexity, this will be delayed until Lemma 7 below. We will first give some relationships between the parameters.

Lemma 5 *Suppose L_t is a rectangle growing from a corner with width and height at least two. If $R = \langle e, r_0, r_1, r_2, c_{0.0}, c_0, c_1, c_2\rangle$ is possible for a framework then*
a) If $e = 0$, then $r_0 = 1$ and $r_1 = r_2 = c_0 = c_1 = c_2 = 0$,
b) else $e \geq 2$, $r_0 \leq 4$, and $c_0 \leq 2$.
c) $e \geq 2(r_2+c_2) + r_1+c_1$.
d) If L_t is the full puzzle with no duplicate pieces or self similar pieces, then $e \geq 2(r_2+c_2)+r_1+c_1+1$.

Proof of Lemma 5: When there are $e = 0$ connection-locations defining the framework, then clearly all the pieces located in L_t form one meta-piece and it contains the start corner marked x. Otherwise, $e \geq 2$ because these edges must cut between two meta-pieces and the width and height of L_t is at least two. $r_0 \leq 4$ because there are at most this many corners. Actually, $r_0 = 1$ until close to the end. In contrast, $c_{0.0}$ could be big because, the framework-components can be

arbitrarily nested as shown with A.i_1 and i_2. However, the rotation of these framework-components place each in a unique location. Hence, having a large $c_{0.0}$ does not matter. On the other hand, because L_t is a rectangle growing from a corner, there can be two bottom right corners, namely the right most in the last completed row and the right most in the row currently being filled. This allows the framework-components A.h_1 and h_2 to be swapped. But because one of these corners has height one, Figure 5.A.h_1 and h_2 can be swapped but i_1 and i_2 cannot and hence $c_0 \leq 2$. In contrast, if L_t is grown as a triangle as in Figure 5.B, then c_0 could be $\Omega(\sqrt{n})$.

The next thing to prove is that $e \geq 2(r_2+c_2) + r_1+c_1$ by having each of the r_2 meta-piece with $d = 2$ privately own one of its top most and one of its left most connection-locations; each of the r_1 meta-piece with $d = 1$ own either its top or its left most depending on where the edge of the puzzle is; each of the c_2 framework-components with $d = 2$ own one of its bottom most and one of its right most connection-locations; and each of the c_1 owns one or the other depending on where the edge of the puzzle is. See respectively the small and larger arrows in Figure 5.A. The component depicted in Figure 5.A.f spans from one side of L_t to the other. Both it and the meta-piece it cuts off need to own connection-location (large and small arrow). This is not a problem because the theorem requires that L_t has width and height at least two. (This is in contrast growing the edges seen in Figure 5.C.f_i.)

When L_t is the full puzzle, we need to prove that the number of connection-locations e is at least one bigger than previously stated. With no duplicate pieces or self similar pieces, component like Figure 5.A.d don't happen and those like b and f contain at least two puzzle locations, and hence either its width or height can "own" not one but two connection-locations. ■

We will now bound the probability that the solution formed is valid.

Lemma 6 *The probability that all the necessary side connections in the pieces placement S are valid is $prob(S) \leq \frac{1}{q^{2(r_2+c_2)+r_1+c_1-a}}$, where a is defined both within this proof and that for Lemma 7.*

Proof of Lemma 6: For such a subsolution S to be valid, each of the e non-planted connections between sides must happen to be correct. Assuming independence, the probability of this is $\frac{1}{q^e}$. However, there is some dependencies between these events. For example, if two identical pieces are being swapped as in Figure 3.A.a and Figure 5.A.d vs d′, then the piece at location d′ in the planted solution fits into location d in the subsolution with probability $\frac{1}{q^4}$, but then for free you get that d fits in d′. Hence, the calculation $\frac{1}{q^e}$ is *double counting* the probability of these events. To better understand these dependencies form the graph as seen in Figure 5.D. The proof of Lemma 5 has each meta-piece and each framework-components privately own one or two connection-locations for a total of $2(r_2+c_2) + r_1+c_1$. Denote this number by $e' \leq e$. The solution S places a puzzle piece in each location in L_t and as such puts a connection-side of a piece on either side of each connection-location. The graph will have $2e'$ node for each of these connection-sides. Put a dotted edge between two such sides if we need their features to be the same for the subsolution to be valid because they touch at one of these e' connection-location within the solution S. Put an solid edge between them if their features are guaranteed by the distribution to be the same because they touch in the planted solution. Note each node has degree at most two making the graph a collection of cycles and paths. For each such cycle or path, all the connection-sides in it by transitivity need the same feature $\alpha \in \mathcal{F}$. Each of the e' connection-location in the framework corresponds to a dotted edge, those features match with probability $\frac{1}{q}$. Traversing around a cycle/path, each such dotted edge is independent and hence can be count when computing our probability that the subsolution is valid until we reach the last dotted edge in a cycle. Because this last dotted edge connects the two ends of a path that we already know all have the same feature, we already know its two features are the same. Let a denote the number of such cycles. The result that $prob(S) \leq \frac{1}{q^{e'-a}}$ follows. ■

Now that we understand the dependency better, we can get a better lower bound the number of potential subsolutions.

Lemma 7 *The number of placements S of the pieces within L_t consistent with F is $N_S(F) \leq 4^{r_0}\ (4\sqrt{n})^{r_1-a}\ (4n)^{r_2}$, where a is defined both within this proof and that for Lemma 6.*

Proof of Lemma 7: We bound the number of different placement of the pieces S by counting the number of bits needed to communicate one given a framework F. The same dependencies that arose in the probabilities allows us to save bits. Again, suppose two identical pieces are being swapped as in Figure 3.A.a. The description of the solution S begins as described above, i.e. that the meta-location Figure 5.A.d will get the meta-piece cut out of meta-location A.d′. Specifying d′ requires $\log(4n)$ bits. Instead of using a another $\log(4n)$ bits to specify that the meta-location d' will get the meta-piece at d, we use one bit to state that this is a swap. In general, we will follow the cycles in the probability graph define in the proof of Lemma 6. Given the placement of the pieces S, we describe it as follows. We start with the placement of the top corner piece marked x. We assume that it is correct. For example, in Figure 5.A, this specifies all the pieces being placed in the meta-location from the corner x all the way down to framework-component A.f, excluding the meta-pieces cut out of these pieces. When placing a meta-piece that lies on a cycle in the graph from the proof of Lemma 6, we follow the cycle. For each edge of the cycle, one additional bit is used to specify whether or not the cycle is over. If not, $\log(4n)$ bits (or $\log(4\sqrt{n})$ bits for $d = 1$) are used as before to specifying where in the planted solution to cut it out. If this is the end of the cycle, then it is already known where the last meta-piece comes from. It comes from where the first meta-piece in the cycle was put. Lemma 6 used a to denote the number of these cycles. For $d = 2$ meta-pieces, there were two connection-locations associated with them and hence two such cycles. Hence, each of these two cycle can be paid for by the savings of these $\log(4n)$ bits, for a total of $\log(4\sqrt{n})$ bits of savings for each of the a cycles. ■

We can now prove $N(L_t) \leq 1 + \mathcal{O}(\epsilon)$ as follows.

Proof of Lemma 3: If L_t is a rectangle growing from a corner, the expected number of subsolutions is

$$N(L_t) = \sum_R N_F(R) \cdot N_S(F) \cdot prob(S)$$

$$\leq \sum_R \left(8^e \cdot 2^{c_0}\ \sqrt{n}^{c_1}\ n^{c_2} \frac{1}{c_0!c_1!c_2!}\right) (4^{r_0}\ (4\sqrt{n})^{r_1-a}\ (4n)^{r_2}) \left(\frac{1}{q^{2(r_2+c_2)+r_1+c_1-a}}\right).$$

In case (a) of Lemma 5, $e = r_1 = r_2 = c_0 = c_1 = c_2 = 0$ and the term in the sum is one. This counts the subsolution consistent with the planted solution. Otherwise, plugging in $e \geq 2$, $r_0 \leq 4$, $c_0 \leq 2$, and $q = \sqrt{\frac{4n}{\epsilon}}$ and simplifying gives

$$N(L_t) \leq \sum_{R, e \geq 2} \frac{1}{c_0!c_1!c_2!} \mathcal{O}(\sqrt{\epsilon})^{2(r_2+c_2)+r_1+c_1}$$

Using $e \geq 2$, gives that the next term is $\mathcal{O}(\epsilon)$. Hence, the geometric sum over $R = \langle r_1, r_2, c_1, c_2 \rangle$ gives $N(L_t) \leq 1 + \mathcal{O}(\epsilon)$. When L_t is the full puzzle with no duplicate pieces or self similar pieces, then Lemma 5.e gives that e' is one bigger than stated. This improves the result to $N(L_n) \leq 1 + \mathcal{O}(\frac{\epsilon^{1.5}}{\sqrt{n}})$. ■

9 Subsolutions for Edge Pieces from the Planted Solution

We now consider putting together the edge pieces of the puzzle.

Proof of Theorem 2 (Build the Edge Alg): Section 5 argued that the branching tree width grows at the rate of $2^{\Theta(\sqrt{\epsilon}t)}$. If this was able continue for all m edge pieces the branching width would be $2^{\mathcal{O}(\sqrt{\epsilon}m)}$ where $m = \Theta(\sqrt{n})$ is the number of edge pieces. We now prove that there are in fact this number of valid ways to complete the edge pieces. The proof is very similar to that in Lemma 3. We form an exponential number of valid subsolutions starting by breaking the planted rectangle of m edge pieces into r meta-pieces (frameworks 5.C). There are $\binom{m}{r}$ ways of doing this. Then we choose how to order these r pieces into a valid subsolution. If we stated that the probability of being valid is $\frac{1}{q^r}$ because each of these r non-planted connections is independently valid with probability $\frac{1}{q}$, then the expected number of valid subsoltions would be computed as $\binom{m}{r}r!\frac{1}{q^r} = \Theta(1)$. However, depending on this ordering there can be dependencies between the events of these meta-pieces correctly fitting together. To get a large number of valid subsolutions, we will only consider those permutations for which this dependency is maximized, i.e. the cycles in Figure 5.D all have length two. We define now an algorithm for merging pairs of the meta-pieces together until there is one solution containing all of the edge pieces. The loop invariant is that there are $r' \leq r$ such unmerged meta-pieces left and that if we wanted we could order these meta-pieces so that each new connection is between pieces that belong together in the planted solution, i.e. for each one $BC..DE$, there is a meta-piece ending in A and one beginning with F, where A and F are the pieces placed next to B and E in the planted solution. We maintain this loop invariant while making progress as follows. Let $BC..DE$ be the meta-piece that we have been growing. We choose one of the other $r'-1$ other meta-pieces $X..Y$. By the loop invariant, there are other meta-pieces $F..G$ and $V..W$ such that merging them as $BC..DE \cdot F..G$ and $V..W \cdot X..Y$ would place together pieces that are next to each other in the planted solution. To obtain a different solution, we swap these and merge them as $BC..DE \cdot X..Y$ and $V..W \cdot F..G$. The probability that the new $E \cdot X$ connection matches is $\frac{1}{q}$. But given that they do match, we get that $W \cdot F$ match for free. This maintains the loop invariant, multiplies the number of subsolutions by $\frac{r'-1}{q}$, and decreases r' by two. The expected number of valid permutation produced in this way is $\frac{r'-1}{q}\frac{r'-3}{q}\frac{r'-5}{q}\cdots\frac{1}{q} \geq (\frac{r}{eq})^{\frac{r}{2}}$. Another potential way in which the above considered subsolutions may be invalid is that the corner pieces may not have distance L or W between them as required. But these are the most likely distances and hence the probability that this is correct is at least $\frac{1}{m^3}$. Combining these gives that the number of valid subsolutions is at least $N(L_t) \geq \max_r \binom{m}{r}(\frac{r}{eq})^{\frac{r}{2}}\frac{1}{m^3} \geq \max_r(\frac{m}{er})^r(\frac{2\sqrt{\epsilon}r}{em})^{\frac{r}{2}} \geq \max_r(\frac{2\sqrt{\epsilon}m}{e^3 r})^{\frac{r}{2}} \geq e^{\frac{\sqrt{\epsilon}m}{e^4}}$ by setting $r = \frac{2\sqrt{\epsilon}m}{e^4}$. ■

10 Exponential Time with Many Solutions

For completion we should consider what happens when the probability of pieces fitting together is increased just so that there is an exponential number of complete solutions. Instead of trying all possibilities in parallel, it is better to use recursive backtracking to do a depth first search of this branching tree in hopes that it will find one of these many solutions quickly. Theorem 6 proves that it still take an exponential amount of time to find one of these exponentially many solutions.

Proof of Theorem 6 (Many Solutions): Let the probability of two pieces connecting be $\frac{1}{q'} \geq \frac{1}{\sqrt{n}}$, which is at least $\frac{2}{\epsilon}$ times that at which the planted solution is unique. Split the sequence

of locations to be filled into two phases $L_{\langle n-n' \rangle} = \left\langle \ell_1, \ell_2, \ldots, \ell_{\langle n-n' \rangle} \right\rangle$ and the rest $L_{\langle last\ n' \rangle} = \langle \ell_{n-n'+1}, \ldots, \ell_n \rangle$, where the size of the second phase is $n' = \frac{1}{8}q'^2 \leq \frac{n}{8}$. Similarly partition the pieces so that there is not a dependence between them. Colour the planted solution like a chess board and let $S_{\langle last\ n' \rangle}$ be a subset of n' of the black puzzle pieces so that having a planted solution has no effect on putting these together. Let $S_{\langle n-n' \rangle}$ be the remaining $n-n'$ pieces. Lets assume that the recursive backtracking algorithm when trying pieces cycles through them $S_{\langle n-n' \rangle}$ followed by $S_{\langle last\ n' \rangle}$. In Section 6, we proved that in the unplanted distribution, the number of valid ways of placing n pieces in t locations is $\left(\frac{4n}{q^2}\right)^t$. The same argument proves that the number of valid subsolution in which the pieces in $S_{\langle n-n' \rangle}$ are placed in $L_{\langle n-n' \rangle}$ is at least $\left(\frac{4(n-n')}{q'^2}\right)^{\langle n-n' \rangle} \geq 2^{n/2}$. The recursive algorithm will try each of these subsolution. Then for each, the probability (expected number) that there is a valid way of completing the solution by placing the pieces in $S_{\langle last\ n' \rangle}$ in $L_{\langle last\ n' \rangle}$ is at most $\left(\frac{4n'}{q'^2}\right)^{n'} = \left(\frac{1}{2}\right)^{n'}$. Hence, the algorithm likely must try $2^{n'}$ of the $L_{\langle n-n' \rangle}$ subsolution until it finds one that leads to a valid solution. At our smallest connection probability $\frac{1}{q'} = \frac{1}{\sqrt{n}}$, this number is $2^{q'^2/8} = 2^{n/8}$ and it decreases as the connection probability increases. ■

11 Thanks

I would like to thank John Tsotsos for his enjoyable AI class in which he got me thinking about this problem and Russell Impagliazzo for advising me on the problem.

References

[1] T. Altman, Solving the Jigsaw Puzzle Problem in Linear Time, *Applied Artificial Intelligence, 3:4, 453-462, 1989.*

[2] Robert Berger, The undecidability of the domino problem, *Memoirs of the American Mathematical Society, 66, 1966.*

[3] C. Cheng, Object-based place recognition and loop closing with jigsaw puzzle image segmentation algorithm, *Robotics and Automation, ICRA 2008. IEEE International Conference on p. 557 - 562.*

[4] T. S. Cho, S. Avidan and W. T. Freeman, A Probabilistic Image Jigsaw Puzzle Solver, *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2010.*

[5] E. D. Demaine and M. L. Demaine, Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity, *Graphs and Combinatorics, 23, 2007.*

[6] A. Flaxman, Random Planted 3-SAT, *Encyclopedia of Algorithms 2008.*

[7] H. Freeman and L. Gardner, Apictorial jigsaw puzzles: the computer solution of a problem in pattern recognition, *IEEE Transactions on Electronic C omputers, 13:1 18127, 1964*

[8] A. Frieze, R. Kannan, A new approach to the planted clique problem, *FSTTCS 2008: 187-198.*

[9] Martin Gardner, *Mathematical games Scientific American,* 110121, 1977. Reprinted as Penrose Tiling, chapter 1 of Penrose Tiles to Trapdoor Ciphers, published by W. H. Freeman (1989) and The Mathematical Association of America (1997).

[10] M. R. Garey and D. S. Johnson, Complexity results for multiprocessor scheduling under resource constraints, *SIAM Journal on Computing, 4(4):397411, 1975.*

[11] M. Garey and David S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, *W. H. Freeman & Co., 1979.*

[12] M. Garey, D. Johnson, and C. Papadimitriou, Unpublished.

[13] F. Gindre, D, Pizzo, G. Barrera, and M. Lopez De Luise, A Criterion-based Genetic Algorithm Solution to the Jigsaw Puzzle NP-Complete Problem, *Pro. of the World Congress on Eng. and Comp. Sci. 2010*

[14] David Goldberg, Christopher Malon, and Marshall Bern, A global approach to automatic solution of jigsaw puzzles, *In Proceedings of the 18th Annual Symposium on Computational Geometry, 8287, 2002.*

[15] Weixin Kong and Benjamin B. Kimia, On solving 2D and 3D puzzles using curve matching, *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Hawaii, December 2001.*

[16] Leonid A Levin, Average case complete problems, *SIAM Journal on Computing, 15(1):285 286, 1986.*

[17] Stephen R. Mahaney, Sparse complete sets for NP: solution of a conjecture of Berman and Hartmanis, *Journal of Computer and System Sciences, 25(2):130143, 1982.*

[18] G. M. Radack and N. I. Badler, Jigsaw puzzle matching using a boundary-centered polar encoding, *Computer Graphics and Image Processing, 19:117, 1982.*

[19] Anne D. Williams, Jigsaw Puzzles: An Illustrated History and Price Guide, Wallace- Homestead Book Co, Radnor, PA, 1990.

[20] Anne D. Williams, The Jigsaw Puzzle: Piecing Together a History, *Berkley Books, New York, 2004.*

[21] H. Wolfson, A. Kalvin, E. Schonberg, and Y. Lambdan, Solving jigsaw puzzles by computer, *Annals of Operation Research, 12(14):5164, February 1988.*