https://doi.org/10.26552/com.C.2013.1.49-53

Johan Oppen *

KEEPING JIGSAWS CONNECTED

This paper describes a combinatorial problem where the idea is to find out in how many ways a jigsaw puzzle can be built, piece by piece, in such a way that it stays connected at all times during the building phase. Computational methods, both exact and approximate, to count the number of such connected sequences are presented.

Keywords: *Combinatorics, puzzles, counting.*

1. Introduction

The idea to the work described here originates from puzzling a jigsaw on the web [1]. The player is supposed to place the 50 US states, one after the other, as they appear in random order, at their correct positions on a map. Fig. 1 shows a screenshot where the first state, Nevada, has been correctly placed. The player now has to place New Hampshire as the second state. The player of course needs some geographical skills to be able to complete the jigsaw correctly, but the focus in this paper is on combinatorial and algorithmic aspects of this type of game, rather than on geographical skills.

If one disregards Alaska and Hawaii, it is possible to keep the jigsaw connected all the time while building it. If the player is allowed to pick the pieces (states) as he or she wants, this is quite easy: start with any of the 48 states, then always pick the next state among the neighbors to the states that are already placed on the map, follow this strategy until the jigsaw is completed. Such a sequence will be referred to as a *connected sequence*. If the pieces appear in random order, however, it is very unlikely that the jigsaw will stay connected: most of the time it will become disconnected quite soon, typically when the second or third piece is added. In Fig. 1 we see that the connection will be broken when the second piece is added. A question that may emerge is then the following: exactly *how* likely (or unlikely) is it that the jigsaw will stay connected all the time? The probability is certainly not 0, but in this case it seems to be very low. It is quite easy to compute this probability: count the number of connected sequences, find out the total number of sequences, the ratio of connected (or "feasible") over total then gives the probability. The number of connected sequences is easily found by complete enumeration, the total number of sequences is 48!, the only problem is the time it takes to count the feasible sequences.

In the following, the problem described above will be referred to as the Connected Sequence Problem (CSP). The CSP can also be defined as a graph problem, but in this paper only the geographical version is discussed.

No research papers dealing with the particular problem discussed here have been found, but many more or less closely related problems have been described in the literature. While political districting, see, e.g., [2] deals with how a geographical region should be divided into smaller parts to meet certain criteria in the best possible way, the CSP deals with an already given structure in terms of boundaries and neighbors. Another well-known problem usually defined on a planar map is the four color problem, see [3]. The four color problem can also be viewed as a special case of the graph coloring problem, see [4], where the graph is restricted to be a planar map, and where it has been proven that any such map can be colored using at most four colors without using the same color for two neighboring sectors of the map. In addition to issues of connectivity and neighborships, the CSP also deals with order and sequencing, and may in this sense be viewed as a kind of scheduling problem, see [5]. While most scheduling problems deal with finding an optimal or at least feasible solution, the CSP asks for how many feasible or connected sequences there exist.

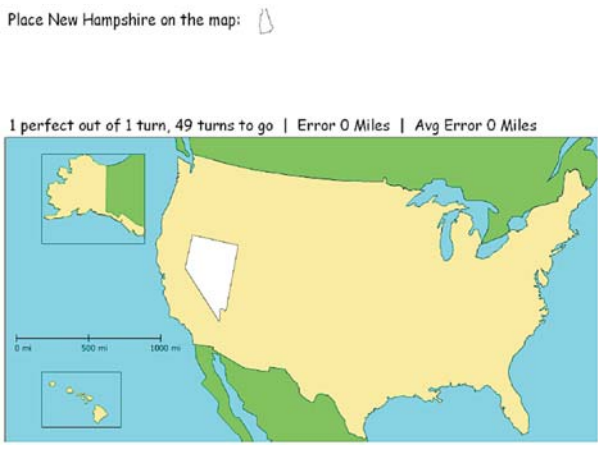

Fig. 1 Playing the jigsaw Place The State on the web; Nevada has been correctly placed and New Hampshire is the next

* **Johan Oppen**
Molde University College, 6402 Molde, Norway, E-mail: Johan.Oppen@hiMolde.no

This paper presents methods for counting connected sequences and finds more efficient ways than the brute force method of total enumeration, which can be done by a simple computer program. For large problems where finding the exact number of connected sequences seems impossible in reasonable time, randomly drawn sequences are used to get estimates.

Practical applications where this type of combinatorial computations might be of interest are not obvious, but any development of a transportation network where a geographical area has to be kept connected might lead to questions like those addressed in this paper.

The rest of the paper is organized as follows. Section 2 presents both exact and approximate methods for counting the number of feasible sequences, computational experiments are described in Section 3. Finally, conclusions can be found in Section 4.

2. Solution methods

Because the number of neighbors differs from one state or jigsaw piece to the next, and because the number of neighbors after adding one more piece depends on which of the current neighbors is added, it seems reasonable to start from a full enumeration of connected sequences. For small instances with no more than about 10 – 12 pieces, it only takes a few seconds of CPU time to enumerate all connected sequences. The solution time increases exponentially as the number of pieces grows, and the basic algorithm is too slow to be of any use for instances with more than about 15 pieces. As an example, South America with 13 countries has a total number of connected sequences of 620 534 320. In the following, two solution approaches will be explored. The first approach is to speed up the exact algorithm in order to count the number of connected sequences more efficiently; the second is to estimate the number of connected sequences by generating random sequences.

2.1 Speeding up the full enumeration

The basic version of the full enumeration proceeds as follows:

1. Set *count* = 0.
2. Generate all n partial sequences of length 1, where n is the number of jigsaw pieces, and store the sequences for further extension in a set E. This corresponds to constructing a set of initial, partial sequences where each sequence starts with a different piece.
3. Pick a partial sequence s from E and extend it in all feasible ways by adding a new piece to the sequence. The new piece must not already be part of the sequence, and it must be a neighbor of at least one of the pieces already present in the sequence. This generates one new sequence for each neighbor of s.
   - If the size of s before extension was $n - 2$, *count* is incremented for every feasible extension found, as this means there is only one piece left to add after the extension, this has to be neighbor and can be added in exactly one way.
   - Otherwise, every sequence corresponding to a feasible extension is added to E.
4. Remove s from E.
5. If E is empty, stop. If not, go back to 3.
6. When the algorithm stops, *count* holds the number of connected sequences.

This algorithm finds all connected sequences up to a point where there is only one piece left to add, so in practice it enumerates all connected sequences.

An observation that can be used to speed up the algorithm is the fact that all sets of 2 neighboring pieces can be constructed in exactly 2 ways. When such a partial, connected, sequence of size 2 is extended, it does not matter in which order the first two pieces were added. This means there is no need to generate both of them, and each successfully completed connected sequence originating from the initial pair of neighbors counts for two connected sequences.

This idea can be extended to partial, connected sequences of any size, but one then has to find a way of checking that the sequence is actually connected, and the number of feasible ways of building the set has to be computed. For size three, this is easily done: if all three pieces are neighbors to the other two, there are six ways to build the set, if one neighborship is missing the set can be built in four ways, and if two or more neighborships are missing the set is not connected at all. Figure 2 shows the states Utah, Colorado and Kansas placed on the map. This set of three pieces can be constructed in four feasible ways: Utah – Colorado – Kansas, Colorado – Utah – Kansas, Colorado – Kansas – Utah and Kansas – Colorado – Utah. The two sequences where Colorado is added last are not feasible. Starting from this set of states, every complete connected sequence found adds four to the total count.

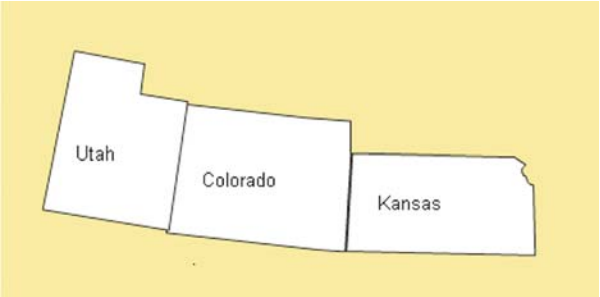

Fig. 2 Initial set of three pieces. Utah, Colorado and Kansas have been placed on the map, this can be done in four different ways without getting a disconnected sequence

For sets of larger sizes, checking connectedness and counting feasible ways of building them becomes more cumbersome. In addition, the number of sets to check grows very fast. For the jigsaw with the 48 lower states in the US, there are only $\binom{48}{3} = 17\,296$ sets of size three, but this grows to $\binom{48}{10} = 6\,540\,715\,896$ sets of size ten and $\binom{48}{24} = 32\,247\,603\,683\,100$ sets of size 24. It would

nevertheless be very useful to avoid extending the same partial sequences multiple times, and this can be done by joining sequences containing the same set of pieces or states.

There is, of course, a computational cost of comparing and joining sequences, but we save a lot more because we are left with a much smaller number of sequences to extend.

Another useful observation is that a partial sequence to which all the remaining m pieces are neighbors, can be completed in exactly $m!$ ways. Depending on the size of m, this can lead to substantial savings, as there is no need to extend such a sequence further. Note that this will occur at the latest when $m = 1$, most of the time at an earlier stage. An example of such a situation is shown in Fig. 3, where all the eight remaining states Nevada, Arizona, Wyoming, Illinois, Tennessee, Kentucky, Ohio and Pennsylvania are neighbors to the part of the jigsaw already completed. If the jigsaw has been kept connected so far, it can be completed in $8! = 40\,320$ ways and thus this number can be added to the total count without extending the sequence any further.

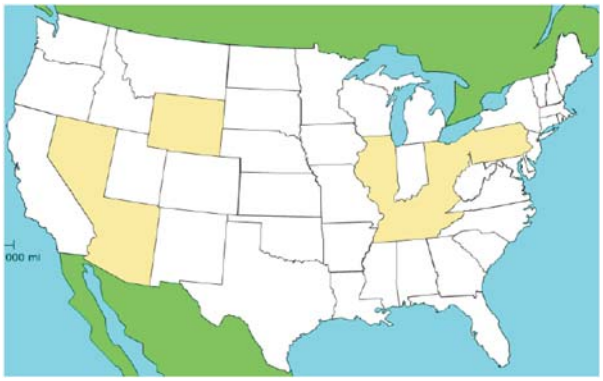

Fig. 3 Eight remaining pieces are all neighbors. The map can now be completed in 40 320 ways, it cannot be disconnected because each remaining state is a neighbor to at least one of the states already present in the puzzle

In the final version of the exact algorithm, the ideas of extending and joining partial sequences, and checking if all remaining pieces are neighbors, are combined. We then also have to deal with memory issues, as the amount of computer memory needed to store all partial sequences of a given size may extend the available amount of RAM. The memory need is approximately doubled during the extension and joining step. We handle this by estimating the memory need before the extension/joining step. If the amount of available memory is insufficient to do the next step, the set of sequences is split in two halves. The algorithm is then completed for the first half of the sequences before it is run for the second half. The problem may have to be split again several times to avoid running out of memory. Splitting a set of equally sized sequences means that some opportunities to join sequences after the next extension will be missed, but this seems to be a price one has to pay for not running out of memory.

The exact algorithm for counting connected sequences then proceeds as follows:

1. Set $count = 0$.
2. Generate all partial, connected sequences of size $s = 3$ and, for each of them sequence e, store the number of feasible constructions f_e.
3. For $s \leq n - 1$, where n is the number of jigsaw pieces, do the following:
   (a) Check if there is enough memory available to do the extension and joining step to $s = s + 1$.
   (b) If no, split the set of partial sequences in two halves, run the algorithm from step 3 with the first half, then with the second half.
   (c) If yes, do the following for all each sequence e:
      i. Set *allNeighbors* = *false*.
      ii. If $s > n/3$ Check if all $m = n - s$ remaining pieces are neighbors. If yes, set *allNeighbors* = *true*, delete the sequence e and increment *count* by $m! * f_e$.
      iii. If *allNeighbors* = *false*: Extend the sequence e in all feasible ways to size $s + 1$ by adding a new piece that is a neighbor, but not already a member and delete e. For each feasible extension e^*, check if a sequence e' of size $s + 1$ containing the same states as e^* is already present. If yes, set $f_e = f_{e'} + f_{e^*}$ and delete e^*. If no, add e^* to the set of sequences.
4. When the algorithm stops, *count* holds the number of connected sequences.

2.2 Estimating the number of connected sequences

The speedup techniques described in the previous subsection can be used to solve slightly larger instances in reasonable time, compared to the basic version of the algorithm. Like in most situations with exponential algorithms, however, this does not help much in terms of solving really large instances. It still takes a long time, probably several centuries, to count the number of connected sequences for the 48 lower US states, so it would be useful to have an efficient and reliable algorithm for estimating the number of connected sequences.

Given the proportion p of connected sequences and the total number of sequences for an instance of the CSP, the number of connected sequences X can be found, and vice versa. For small instances, where X can be computed exactly, this means p is known with certainty. For larger instances, an estimate of p can be used to estimate X. From statistics, see, e.g., [6], it is known that a sample proportion $\hat{p}$ can be used as an estimator of p under certain conditions. Given a sample size n, the sample proportion $\hat{p}$ and the number of successes $\hat{X}$ (the number of connected sequences found) have an approximate normal distribution if n $n\hat{p}(1 - \hat{p}) \geq 9$, giving the confidence interval $\hat{p} \pm z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$. The proportion p is unknown, and for large instances of the CSP it is believed to be extremely small. If p is equal to, say, 10^{-9}, this corresponds to

one success every one billion trials, the sample size should then be 10 billion according to the formula above. This means that using a sample size corresponding to 10 successes seems reasonable in order to get a reasonably good estimate of p. Because p, and thus the required sample size, is unknown in advance, one can draw random sequences until 10 successes have been found to get an estimate with the desired properties. It should be noted that stopping exactly when the 10^{th} success is found, and using the number of samples up to this point as the sample size, will mean that one overestimates p. The reason is that, on average, the number of samples needed to find the first success is $k/2$, where k is the average number of samples between two successes. This means that one can expect to find success number 10 after $9.5k$; samples, and thus the number of samples needed to reach success number 10 should be divided by 0.95 to find the correct sample size.

3. Computational experiments

In order to test the performance of the algorithms described in the previous section, computational experiments have been conducted. The exact algorithm was implemented in Java, the sampling procedure to estimate $\hat{X}$ and $\hat{p}$ was coded in C++. All tests are run on a 2.99 GHz Intel(R) Pentium(R) 4 CPU PC with 1.5 GB of RAM, running Microsoft Windows XP Profes-sional version 2002, Service Pack 3. The exact algorithm was first coded in C++, but we switched to Java because the BigInteger class in Java was very convenient to use in this case.

3.1 Problem instances

A geographical jigsaw initiated the work described in this paper, and only geographical jigsaw puzzles are used as test instances. Table 1 gives an overview of instances used.

Problem instances Table 1

Name	Size
South America	13
Norway	19
US 19	19
US 25	25
US 37	37
US 48	48

US 48 contains the 48 lower US states and is the largest instance, the main reason to use it is to try to estimate the number of connected sequences in the "original" problem. This turned out to be quite difficult, see 3.3 and thus some smaller instances are also used. In US 37 and US 25, states are removed from US 48 starting from the east. Norway and US 19 has the same size, but while the 19 Norwegian counties are "poorly connected", meaning that many counties have only one or two neighbors, US 19 contains most of the states west of the Mississippi which are much more closely connected, and the number of connected sequences is almost 100 times higher. South America with its 13 countries was used for speedup testing for the first and relatively slow versions of the exact algorithm. Especially the idea of checking if all remaining pieces are neighbors to the part of the jigsaw already completed turned out to make a huge difference to the solution time for South America. The main reason is probably the special structure of this instance, where Brazil is a neighbor to all other countries except two. This means that if Brazil appears early in the sequence, the probability is high that all pieces left are neighbors. Each instance of the CSP has its own special structure that affects both the number of connected sequences and how much speedup can be gained from the speedup techniques described in Section 2.1.

3.2 Exact solutions

Table 2 shows results for the exact algorithm with the speedup techniques from Section 2.1 implemented. The basic version described in Section 2.1, without the speedup procedures, was able to solve South America in one hour and to find 584 million, or about 0.013 %, of the sequences for Norway in one hour. At the time of this writing, US37 has been running for a few weeks, and it seems it will keep running for years without even being close to completion. Even though the results clearly show the significant speedup achieved, they also illustrate the need of estimates if one wants to know approximate results for larger instances.

Test results for the exact algorithm Table 2

Instance	X	p	Solution time
South America	620 534 320	0.0997	$<$ 1 sec
Norway	4 312 436 579 064	$3.5*10^{-5}$	2 sec
US 19	228 521 921 047 808	$1.88*10^{-3}$	75 sec
US 25	1 278 165 933 715 723 695 360	$8.2*10^{-5}$	40 hours
US 37	unknown	unknown	centuries?

3.3 Estimating the proportion of connected sequences

Table 3 summarizes the results of the estimations. For Norway, US 19 and US 25, which have been solved exactly, the proportion p and the number X of connected sequences are known with certainty, so these instances are estimated only to be able to compare estimates and known values. For all three instances, the confidence interval contains p and thus the estimate is quite good.

For the two larger instances of the CSP where exact solutions cannot be found in reasonable time, only estimates are available.

Estimated and exact proportions and numbers of connected sequences for CSP instances Table 3

Instance	p	X	$\hat{p}$	90% Conf int for $\hat{p}$	n	$\hat{X}$
Norway	$3.5 * 10^{-5}$	$4.3 * 10^{12}$	$4.7 * 10^{-5}$	$2.3 * 10^{-5} - 7.1 * 10^{-5}$	$2.1 * 10^{6}$	$5.7 * 10^{12}$
US 19	$1.88 * 10^{-3}$	$2.3 * 10^{14}$	$2.6 * 10^{-3}$	$1.3 * 10^{-3} - 4.0 * 10^{-3}$	$3.8 * 10^{3}$	$3.2 * 10^{14}$
US 25	$8.2 * 10^{-5}$	$1.3 * 10^{21}$	$1.1 * 10^{-4}$	$5.5 * 10^{-5} - 1.7 * 10^{-4}$	$8.7 * 10^{4}$	$1.8 * 10^{21}$
US 37	Unknown	Unknown	$3.8 * 10^{-7}$	$1.8 * 10^{-7} - 5.7 * 10^{-7}$	$2.7 * 10^{7}$	$5.2 * 10^{36}$
US 48	Unknown	Unknown	$7.5 * 10^{-11}$	$5.1 * 10^{-11} - 9.9 * 10^{-11}$	$1.3 * 10^{11}$	$9.3 * 10^{50}$

Sample sizes (n) needed to draw 10 connected sequences, as explained in Section 2.2, are also given. It can be seen that the sample size needed for US 48 is quite large, it took 56 hours to do the sampling.

4. Conclusions

This paper presents the Connected Sequence Problem, where a geographical area has to be built piece by piece and kept connected throughout the building process. The number of such connected sequences can be found by exact methods for small instances, or by estimation for larger instances.

References

[1] gamesl.org. http://www.gamesl.org/games/place-states.swf.

[2] NYGREEN, B.: European Assembly Constituencies for Wales - Comparing of Methods for Solving a Political Districting Problem. *Mathematical Programming,* 42:159–169, 1988.

[3] APPEL, K., HAKEN, W.: Every Planar Graph is Four Colorable. *Bulletin of the American Mathematical Society,* 82(5):711–712, September 1976.

[4] JENSEN, T. R., TOFT, B.: *Graph Coloring Problems.* Wiley : New York, 1994.

[5] LEUNG, J. Y-T. editor: *Handbook of Scheduling.* Chapman & Hall, 2004.

[6] CARLSON, W. L., THORNE, B.: *Applied Statistical Methods.* Prentice Hall, Upper Saddle River, New Jersey, 1997.