Parabola Volume 55, Issue 3 (2019)

How not to play Connect Four:
an introduction to Dynamic Programming

Trevor Chi-Yuen Tao[1]

1 Introduction

Dynamic Programming (DP) is a well-known optimization method for solving computational problems, such as the number of paths between two nodes or the edit distance between strings [2]. According to Wikipedia, DP attempts to simplify a complex problem by breaking it down into subproblems and then recursively solving these subproblems. However, this notion is rather vague. The essential idea behind Dynamic Programming is that we have a number of states in a graph or table. For each state we compute a desired quantity, such as the number of paths from A to B or the cost of the cheapest path. The quantity need only be computed once and the results can be used as many times as necessary to facilitate a similar computation for "future" states. When all quantities are computed, the final state will have the correct answer to our computational problem. If done correctly, we can compute the desired output much more efficiently than a brute force algorithm.

A simple example of dynamic programming is to compute Fibonacci numbers, defined by $F(n) = F(n-1)+F(n-2)$ and $F(1) = F(2) = 1$. If we blindly applied the definition of, say, $F(20)$ then we get lots of repeated calls for $F(19), F(18), \ldots$. But if we compute from the "bottom up", starting with $F(1), F(2)$ etc., then we only compute each $F(n)$ once which is much more efficient. Another example would be counting paths on a graph from A to B [4]. Again, we need only compute a single value for each node, starting from the bottom up.

[1]Trevor Tao has a PhD in applied mathematics. He is a research scientist currently working for the Australian Department of Defence. Trevor is a keen chess and scrabble player and his other hobbies include mathematics and music. Trevor is the brother of world-renowned mathematician Terence Tao.

2 Connect Four

Connect Four is a well-known two-player game by Milton Bradley. The objective is to line up four pieces of your colour horizontally vertically or diagonally before your opponent does. The game is solved, and is proven to be a win for the first player [1]. The decision is very close: The first player must start in the middle column to avoid a draw (or even a loss). Moreover, she requires all 21 discs of her colour to achieve victory if her opponent puts up maximum resistance.

The analysis of expert-level play in [1] is extremely tedious, so instead we consider a slightly different version of Connect Four where each of the seven columns is assigned one of the days of the week and individual cells have numbers between 1 and 31. Cells without numbers are not used and are only included to make the board resemble the standard version of Connect Four with six rows and seven columns. Each day represents a Bernoulli trial with a red or yellow disc indicating one of two possible outcomes. Discs are always placed in order, starting with the cell numbered 1 and counting upwards. For instance Trevor can play one game of four-suit Spider Solitaire per day and record the result (win or loss). Four yellow discs in a row wins the game for Trevor, or four reds is a win for Spider Solitaire. If neither side achieves connect 4 before the end of the month the game is a draw. A possible game state is shown in Figure 1. After a balanced start in the first week, Trevor has gained the upper hand and only needs a win on 12th of September to achieve Connect Four. If Trevor cannot obtain Connect Four on the 12th, then the game will last at least seven more days since the earliest possible decision can occur with either side declaring Connect Four on the third week (15-16-17-18). Of course, if the third week is still not enough for either side to achieve victory, then the endgame becomes much more exciting since diagonal and vertical threats of Connect Four suddenly come into play!

The revised version of this game is much easier to analyse since we don't have to deal with arcane concepts such as which player controls the Zugzwang in a given game state [1]. We will give it the rather unimaginative name "How Not to Play Connect Four" (HNTPC4). We wish to investigate the following question: How many ways can HNTPC4 end in a draw? Obviously this means filling up the whole board without either side declaring Connect Four.

3 Analysis of HNTPC4

By changing the rules of the game, the set of possible game states changes drastically. In the traditional version of Connect Four, the discs must be "balanced", in the sense that the number of yellow and red discs must be equal or differ by 1, depending on whose move it is. Furthermore if a cell c is empty, then all cells above c must also be empty. In HNTPC4, (i) the number of red and yellow discs need not be balanced but (ii) if a cell is empty then all cells to the right as well as above must also be empty.

Figure 1: How Not To Play Connect Four

For instance, cell 22 cannot contain a red or yellow disc unless all cells below 22 also contain discs of either colour.

3.1 Trivial case

Clearly, if we ignored the possibility of connect 4 for either side, then the number of possible end states is 2^{30} since all 30 days can have a red or yellow disc, depending on how well Trevor plays on a given day (or whether the Spider software is rigged [3]). Every possible end state corresponds to a path in Figure 2. For instance if we alternated red and yellow discs, then the path would be S (for start) $\rightarrow$ Red 1 $\rightarrow$ Yellow 2 $\rightarrow$ Red 3 etc. For any state labelled with number N and colour C, there are 2^N paths from the start node to that state, where N is an integer and C is either red or yellow.

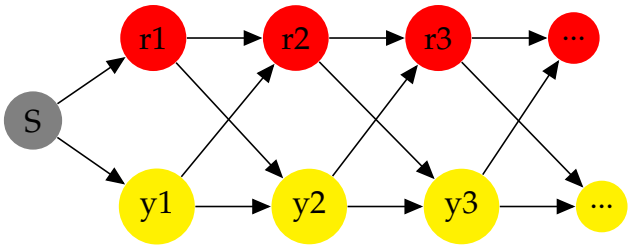

Figure 2: The trivial case

The important point is that *once we reach a particular node such as "Red 3", then it matters not how we get there*. The number of ways to proceed after reaching "Red 3" is independent of the path required to get there from the starting state. This is the fundamental reason why the dynamic programming algorithm can work.

3.2 One-dimensional grid

Now let us add a stipulation that four discs of the same colour on consecutive days is a win (for one of the players), even if it "splits" two rows such as 6-7-8-9 or 14-15-16-17. Note that this is equivalent to flattening the calendar month into a one-dimensional line with 30 cells. In this case, dynamic programming fails. For instance, suppose we reach the node "Red 15". If the previous two days (13,14) were also red, then the next day (16) must be yellow to avoid Red declaring Connect 4. But if the previous two days were yellow, then there would be no such restriction.

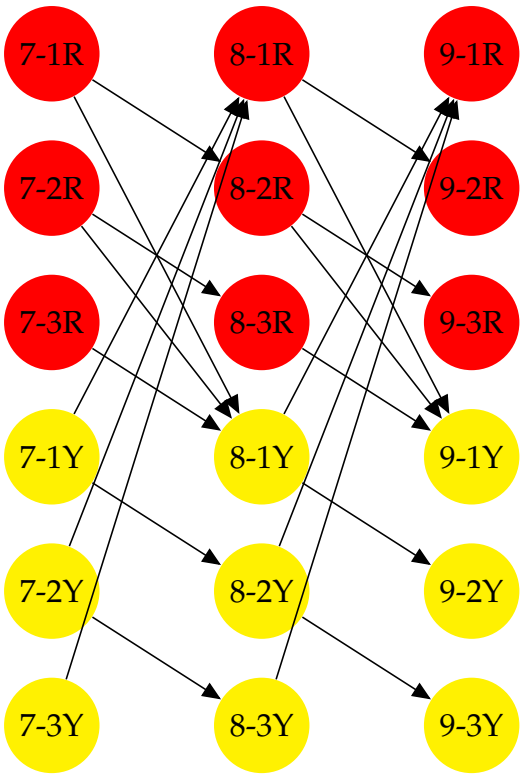

Figure 3: Graph of states for one-dimensional HNTPC4

However, let us change the definition of state so that we record not only the colour of the last day but also the streak. For instance "15, RR" means day 15 and the last two results (but not last three) are red. Therefore after day 16 the state will either be "16, RRR" or "16, Y". This gives us enough information to determine if a connect 4 by either side is looming. In this case, dynamic programming works again. The price we pay is

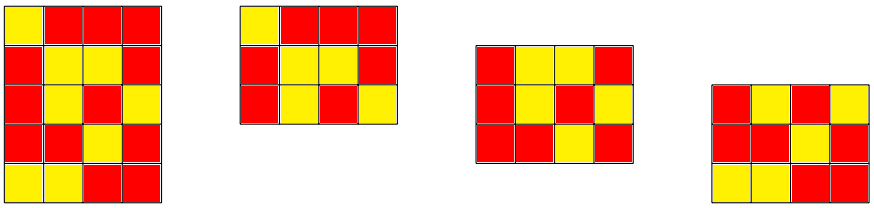

Figure 4: Building a path by chaining compatible blocks.

obviously the fact that we have many more states corresponding to a single day. The graph showing all paths will look something like Figure 3 (restricted to days 7,8,9). With some computation, one can show there are exactly $197,900,192$ ways for a 30-day game of one-dimensional HNTPC4 to end in a draw. This an order of magnitude less than $2^{30} = 1,073,741,824$ ways of filling a 30-day grid ignoring Connect Fours by either side.

Note that one can further simplify the graph by observing the symmetry between red and yellow discs. For instance a streak of two reds is equivalent to a streak of two yellows etc. However, this has not been done in Figure 3.

3.3 Two-dimensional grid

Now let us consider the 2-dimensional case. To simplify the argument, assume there are only 4 columns but any number of rows. To get the dynamic programming principle to work, we need to consider each row of four squares as a "unit". By stacking three units we obtain a "block" of three rows and four columns. Note that once we reach row N, the previous three rows are the only information required to determine which threats of connect 4 are looming for either side. Let us say that two blocks A, B are "compatible" if it is possible to merge them into a 4×4 subgrid without either side declaring Connect Four. Note that compatibility is not commutative, so if A, B can be merged, then that does not imply B, A can also be merged. There are $2^{12} = 4096$ possible blocks. We can eliminate some blocks by noting that if A contains a horizontal Connect Four of either colour, then it won't be compatible with any other block B (or by using the red-yellow symmetry), but this number is still large.

Figure 4 shows how one can build a 5×4 grid by chaining together a number of 3×4 blocks as described above. With dynamic programming one can show that there are 124318 ways to complete a 5×4 grid without either side declaring Connect Four. The network of paths will have three layers (since three blocks are needed to merge into a 5×4 grid) and 4096 nodes per layer, if we neglect the simplications described above. For any pair of adjacent layers a link between two nodes is required for every pair of compatible blocks. With an 8×4 grid, one gets $51,309,598$ different solutions. The number of solutions (and computation required) grows rapidly as we add more layers!

4 Conclusion

In dynamic programming, the basic idea is to set up a network of states so that some quantity (such as number of paths) need only be computed once for each state. With the ideas thus presented, the reader should be able to implement an algorithm to count the number of possible draws in HNTPC4. Further variations on these themes are possible and are left as an exercise for the interested reader. For instance:

- What happens on larger grids, e.g. size 8×20? Does the dynamic programming algorithm scale up easily to large grids?
- What happens if the grid is not a rectangle? For instance September 2019 will have 5 days missing from a 7×5 grid.
- What happens if neutral discs are allowed? For instance if Trevor plays Spider Solitaire at work (but not home), then neutral discs will appear in the Saturday and Sunday columns, plus any day(s) that Trevor has to call in sick.
- What happens if the start-position is non-empty? For example, how many ways can Trevor defeat Spider Solitaire in HNTPC4 in the month December 2019 given the first three results are Yellow-Yellow-Red?

References

[1] V. Allis, *A knowledge-based approach of Connect-Four. The game is solved: White wins*, Masters Thesis, Department of Mathematics and Computer Science, Vrije Universiteit Amsterdam, 1988.

[2] I. Setiadi, Damerau-Levenshtein Algorithm and Bayes Theorem for spell checker optimization, *Makalah IF2211 Strategi Algoritma – Sem. I Tahun* (2013/2014).

[3] T. Tao, Random Walks: an application for detecting bias in Spider Solitaire programs, *Parabola* **55**, no. 1, (2019).

[4] A. Wang, Counting paths on grids with obstructions, *Parabola* **55**, no. 1, (2019).