The basic idea is to ask: How should we gerrymander precincts 1 through j, for each j? To make this work, though, we have to keep track of a few extra things, by adding some variables. For brevity, we say that the A-votes in a precinct are the votes for party A, and B-votes are the votes for party B. We keep track of the following information about a partial solution.

- How many precincts have been assigned to district 1 so far?
- How many A-votes are in district 1 so far?
- how many A-votes are in district 2 so far?

So let $M[j, p, x, y] = true$ if it is possible to achieve at least x A-votes in district 1 and y A-votes in district 2, while allocating p of the first j precincts to district 1. ($M[j, p, x, y] - false$ otherwise.) Now suppose precinct $j + 1$ has z A-votes. To compute $M[j + 1, p, x, y]$, you either put precinct $j + 1$ in district 1 (in which case you check the results of sub-problem $M[j, p - 1, x - z, y]$) or in precinct 2 (in which case you check the results of sub-problem $M[j, p, x, y - z]$). Now to decide if there's a solution to the whole problem, you scan the entire table at the end, looking for a value of "true" in any entry of the form $M[n, n/2, x, y]$, where each of x and y is greater than $mn/4$. (Since each district gets $mn/2$ votes total.)

We can build this up in order of increasing j, and each sub-problem takes constant time to compute, using the values of smaller sub-problems. Since there are n^2m^2 sub-problems, the running time is $O(n^2m^2)$.

[1] ex706.269.18