



© 1997–2004, Millennium Mathematics Project, University of Cambridge.

Permission is granted to print and copy this page on paper for non-commercial use. For other uses, including electronic redistribution, please contact us.

September 1997

Features

Dynamic programming: an introduction

by David K. Smith and PASS Maths



In this issue's article "[Mathematics, marriage and finding somewhere to eat](#)", David Smith investigated the problem of finding the best potential partner from a fixed number of potential partners using a technique known as "optimal stopping". Inevitably, mathematicians and mathematical psychologists have constructed other models of the problem...

Measuring potential partners

An alternative way of looking at the problem assumes that you know that each potential partner has a score. Helen of Troy is fabled to have had "a face that could launch a thousand ships". There's an old joke that beauty can therefore be measured: one *helen* is the beauty needed to launch a thousand ships, one *millihelen* is therefore the beauty required to launch just one ship!



Helen: her affair with Paris, Prince of Troy, started the Trojan wars.

Suppose we use this scale to measure each potential partner's score from *Omilihelens* up to a maximum of *1000millihelens* with all values equally likely. We must now search for a rule which will make sure that the average score of the partner we choose is as large as possible.

Dynamic programming: an introduction

The obvious way to look at this would be to think about what you would do when you met the first potential partner. Then you would compare this partner's score with what you might expect to get later on if you rejected them. Unfortunately, working out what you might expect later on is a complicated mixture of all the possible decisions you could make that becomes too much to work out.

Dynamic programming

Instead, a mathematical way of thinking about it is to look at what you should do at the end, if you get to that stage. So you think about the best decision with the last potential partner (which you must choose) and then the last but one and so on. This way of tackling the problem backwards is *Dynamic programming*.

The word *Programming* in the name has nothing to do with writing computer programs. Mathematicians use the word to describe a set of rules which anyone can follow to solve a problem. They do not have to be written in a computer language.

Dynamic programming was the brainchild of an American Mathematician, Richard Bellman, who described the way of solving problems where you need to find the best decisions one after another. In the forty-odd years since this development, the number of uses and applications of dynamic programming has increased enormously.

For example, in 1982 David Kohler used dynamic programming to analyse the best way to play the game of darts. In his paper, published in the *Journal of the Operational Research Society*, he acknowledges "The Wheatsheaf at Writtle and the Norfolk Hotel in Nairobi for making research facilities available to me".



In darts, each player must score *exactly* 301, starting and finishing on a double.

Solving the potential partner problem

The dynamic programming approach to the potential partner problem starts by thinking about what happens when faced with the last partner.

If you need to make a decision about potential partner number N then you must accept their score (which we'll call $X[N]$) and live happily ever after!

When you encounter potential partner number $N-1$ all that you know are the values $X[N-1]$ and what you expect to get if you wait. You expect to get the average value of $X[N]$, and that will be 500 because $X[N]$ varies from 0-1000. Common sense says that you take the better of these, so your rule will be:

Dynamic programming: an introduction

If $X[N-1]$ is more than 500, accept that potential partner.

If not, go on to potential partner number N

When you encounter potential partner number $N-2$, you know $X[N-2]$ and the average value of the score you will get by waiting. Half the time, waiting will mean that you accept potential partner number $N-1$, whose score will be between 500 and 1000, averaging 750; the other half of the time, you will pass over that potential partner and you will expect a score of 500. So, waiting will give you an average score of 625, and taking the better will give the rule:

If $X[N-2]$ is more than 625, accept that potential partner

If not, go on to potential partner number $N-1$

And so on. It is much simpler than starting with potential partner number 1 and trying to think of all the possible sequences of decisions, and working forwards.

For each potential partner that you meet, the best set of decisions afterwards will give a critical value for comparison; if the potential partner does better than it, choose that partner. If not, go on, even though the future is not certain.

The critical values when $N=10$ are:

Potential partner:	1	2	3	4	5	6	7	8	9	10
Critical value:	861	850	836	820	800	775	742	695	625	500

One of the characteristics of dynamic programming is that the solution to smaller problems is built into that of larger ones. Thus, if you wanted to know the critical values when there are only 6 potential partners, all you need to do is look at the last 6 values in the table, 800, 775 and so on.

Acknowledgements

This article was based on material written by Dr David K. Smith, Mathematical Statistics and Operational Research Department, University of Exeter.



Plus is part of the family of activities in the Millennium Mathematics Project, which also includes the NRICH and MOTIVATE sites.