Cedar Wingate

MUMT 621

Professor Ichiro Fujinaga

15 October 2009

Dynamic Programming Summary

In 1950, looking for a way to describe the work he was doing at the RAND corporation, but not wanting to fuel the ire of the sitting secretary of defense, who had an irrational aversion and hatred of words like "research," Richard Bellman termed the phrase "dynamic programming:" dynamic meaning multistage, time-varying, and dynamic in the classical sense; and programming referring to the intent for decision making, thinking and planning (Dreyfus 2002, p. 48). Simply put, dynamic programming "refers to simplifying a complicated problem by breaking it down into simpler subproblems in a recursive manner" (Wikipedia contributors 2009b).

Dynamic programming is both a mathematical optimization method and a method of computer programming. In mathematics, “it refers to the simplification of a decision by breaking it down into a sequence of decision steps over time (Wikipedia contributors 2009b). In computer programming there are two requirements, optimal substructure and overlapping subproblems. Optimal substructure refers to the problem being able to be solved through optimal solutions to its subproblems usually by recursion, which is a method where a function is used within its own definition (Wikipedia contributors 2009b

and Wikipedia contributors 2009c). Overlapping subproblems refers to the solving the same subproblems over and over instead of generating new subproblesm (Wikipedia contributors 2009b). There two basic approaches to solving the subproblems, top-down and bottom-up.

Using the example of the Fibonacci series ($F_i = F_{i-1} + F_{i-2}$) (ibid.), a top-down approach starting on the 5th Fibonacci number would solve : F_4 (F_3 ($F_2 + F_1$)+ F_2 ($F_1 + F_0$))+ F_3 (F_2 ($F_1 + F_0$) + F_1) + F_2 ($F_1 + F_0$) + F_1) + F_1. For the bottom-up approach, it would solve: F_5: $F_1 + F_2$ ($F_1 + F_0$)+ F_3 ($F_2 + F_1$) + F_4 ($F_3 + F_2$)+ F_5 ($F_4 + F_3$). Assuming that the solutions to subproblems that have already been solved are being stored in memory, they both could take the same amount of time, but the first would take up more space in memory (Wikipedia contributors 2009b).

Another example taken from David Smith's introduction to dynamic programming is called the potential partner problem (1997). In this problem, a scale is set up of 1-1000 millihelens, referring to the old myth that Helen of Troy had a face that launched 1000 ships; therefore a millihelen is a face that would launch one ship. Faced with a number of potential partners, the obvious process is to start with your first choice and build up a comparison between all the other choices. Once can't make a decision, however, until all possible decisions have been contemplated. In dynamic programming, one would start with the final choice and knowing only the score for that option and the potential average for the whole set (500 for a scale of 1-1000), one would decide based on whether the choice in front of him is higher than 500. If it is, that one is chosen; if it isn't, then one moves onto the next one (Smith 1997).

The main application of dynamic programming in the Music Information Retrieval field is for beat tracking. Daniel Ellis created an algorithm that used dynamic programming for beat tracking (2007). He basically compared the onset of different rhythmic emphasis in tracks to predict the next onset and to determine the beats per minute tempo of a track (Ellis 2007). McKinney et. al. compared a number of different algorithms including a dynamic programming algorithm (2007). Three of the eight used dynamic programming inside their algorithms. The general findings were that automatic beat tracking does not perform as well as humans, but that mixing all the algorithms could produce something that would perform better than a human (McKinney et. al. 2007). Wright et. al. explored the use of dynamic programming in recognizing clave rhythms in clave music. They added an extra layer to the process by giving a template of the even-odd separation of the clave rhythm (2008).

Dixon, Simon. 2007. “Evaluation of the Audio Beat Tracking System BeatRoot.” *Journal of New Music Research* 36, no. 1: 39-50.

Dreyfus, Stuart. 2002. "Richard Bellman on the birth of Dynamic Programming." *Operations Research* 50, no. 1: 48-51, http://www.wu.ac.at/usr/h99c/h9951826/bellman_dynprog.pdf (accessed 7 October 2009).

Ellis, Daniel P. W. 2007. “Beat Tracking by Dynamic Programming.” *Journal of New Music Research* 36, no. 1: 51-60.

McKinney, M. F., D. Moelants, M. E. P. Davies, and A. Klapuri. 2007. “Evaluation of Audio Beat Tracking and Music Tempo Extraction Algorithms.” *Journal of New Music Research* 36, no. 1: 1-16.

Smith, David K., and PASS Maths. 1997. “Dynamic programming: an introduction.” *+plus magazine*. http://plus.maths.org/issue3/dynamic/ (accessed 7 October 2009).

Wikipedia contributors. 2009a. "Bellman equation." *Wikipedia, The Free Encyclopedia,* http://en.wikipedia.org/wiki/Bellman_equation (accessed 7 October 2009).

Wikipedia contributors. 2009b. "Dynamic Programming." *Wikipedia, The Free Encyclopedia,* http://en.wikipedia.org/wiki/Dynamic_programming (accessed 7 October 2009).

Wikipedia contributors. 2009c. "Recursion." *Wikipedia, The Free Encyclopedia,* http://en.wikipedia.org/wiki/Recursion (accessed 7 October 2009).

Wright, Matthew, W. Andrew Schloss, and George Tzanetakis. 2008. “Analyzing Afro-Cuban Rhythm Using Rotation-Aware Clave Template Matching With Dynamic Programming.” *Proceedings of the International Conference on Music Information Retrieval (ISMIR) 2008.* Philadelphia, Pennsylvania.