Reinforcement Learning for Algorithm Selection

Michail G. Lagoudakis
Department of Computer Science
Duke University
Durham, NC 27708
mgl@cs.duke.edu

Michael L. Littman
AT&T Labs – Research and Duke University
Florham Park, NJ 07932
mlittman@research.att.com

Many computational problems can be solved by multiple algorithms, with different algorithms fastest for different problem sizes, input distributions, and hardware characteristics. We consider the problem of *algorithm selection*: dynamically choose an algorithm to attack an instance or subinstances (due to recursive calls) of a problem with the goal of minimizing the overall execution time. We formulate the problem as a kind of Markov Decision Process (MDP), and use ideas from reinforcement learning (RL) to solve it.

The process' state consists of a set of instance features, such as problem size. Actions are the different algorithms we can choose from. Non-recursive algorithms are terminal in that they solve the problem completely (terminal state). Recursive algorithms create subproblems and therefore cause transitions to other states, making the task a sequential decision task. The immediate cost of a decision is the real time taken for executing the selected algorithm on the current instance, excluding time taken in recursive calls. Thus, the total (undiscounted) cost during an episode is the time taken to solve the problem. The goal is a policy that minimizes the total cost/time. This process differs from a standard MDP as it allows one-to-many state transitions (multiple recursive calls at one level).

Our initial experiments focus on the problem of *order statistic selection*: given an array of n (unordered) numbers and some index i, select the number that would rank i-th if the array were sorted. We picked two algorithms such that neither is best in all cases, otherwise learning would not help. DETERMINISTIC SELECT (D) is an $O(n)$ recursive algorithm and HEAP SELECT (H) is an $O(n \log n)$ algorithm that performs best for indices close to 1 or n. The process' state consists of the size n and the distance d of the index i from the closest end of the array (assuming symmetry). The value of choosing H at some state $s = [n, d]$ is simply the time it takes to solve the corresponding instance, since this is a terminal algorithm. The optimal value of choosing D can be expressed in terms of other state-action values:

$$\begin{aligned} Q\left([n,d],D\right) \quad &= \quad \min_{a=\{H,D\}} \left\{ Q\left([n/5, n/10], a\right) \right\} + \\ &\quad \min_{a=\{H,D\}} \left\{ Q\left([n', d'], a\right) \right\} + R\left([n,d],D\right), \end{aligned}$$

where $R(s,a)$ is the immediate cost of choosing action a in state s, and $n' \le 7n/10 + 6$. The states $[n/5, n/10]$ and $[n', d']$ correspond to the two recursive calls of D. This Bellman equation resembles the recurrence equation for the running time of DETERMINISTIC SELECT (Cormen, Leiserson, and Rivest 1990):

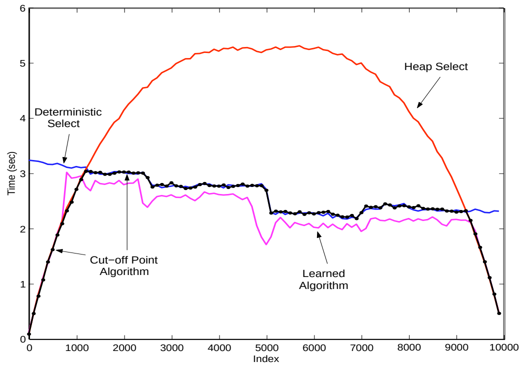
$$T(n) \le T\left(\lceil n/5 \rceil\right) + T(7n/10 + 6) + O(n).$$

Our learning rule is a variation of Q-learning that combines Monte-Carlo (MC) and Temporal Difference (TD) learning:

$$\begin{aligned} Q(s,a) \quad &= \quad (1-\alpha)Q(s,a) + \\ &\quad \alpha \left[\left(R(s,a) + \mathfrak{R}_\pi(s_{n/5}) \right) + \min_{a'} \left\{ Q(s', a') \right\} \right], \end{aligned}$$

where $\mathfrak{R}_\pi(s)$ is the total cost of solving the subproblem s using the current greedy policy (no exploration). Thus, the smaller subproblem is effectively pushed into the immediate cost (MC) and the bigger one is used for bootstrapping (TD).

We trained the system on thousands of randomly generated inputs of size 10000 and various indices, using an $1-\epsilon$ policy and decreasing learning rate. Results are shown below. The "cut-off point algorithm" uses H when the index is within the first 13% or the last 7% of the input (as suggested by the plot), and D otherwise. The learned algorithm performs better with one exception due to the lack of the assumed symmetry. Additional results and extensions are available (Lagoudakis and Littman 2000).



References

Cormen, T.H.; Leiserson, C.E.; and Rivest R.L. 1990. *Introduction to Algorithms*. Cambridge, Mass: MIT Press.

Lagoudakis, M.G., and Littman, M.L. 2000. Algorithm Selection using Reinforcement Learning. In *Proceedings of the Sixteenth International Conference on Machine Learning*. AAAI Press. To appear.