HÁSKÓLINN Í REYKJAVÍK
REYKJAVÍK UNIVERSITY

Research Methodology

Review P3: Write Abstract for paper

Review 17. September 2007

Objective of project

- Write an abstract for a short article
- Get exercise in writing abstracts
- Get feedback on different solutions

- When various algorithms exist for the same problem domain there is often not any clear choice of the most efficient one. Efficiency usually depends on the input data and external factors such as platform differences. For different sets of data or different platforms, different algorithms turn out to be the most efficient ones. So how do we select the right one?
- In this paper the authors demonstrate how a Markov Decision Process (MPD) can be utilized to design an efficient hybrid of several algorithms. In particular this approach is well suited for recursive algorithms. A case study of sorting is examined, yielding an efficient hybrid of insertion sort, merge sort and quicksort, that applies the most appropriate algorithm to each subproblem. An empirical study of the hybrid clearly shows its benefits.
- The paper discusses how this approach applies to other problem domains and briefly considers cases where the fixed nature of our MDP premises can have an effect. To this end it is noted that machine learning can dynamically improve our hybrid algorithm at run time.

- When trying to solve a particular problem users usually come across multiple algorithms. The hard part, is that the efficiency of the algorithm can depend on the input given. In this paper we state that by dynamically selecting the algorithm based on what input we are considering at the moment, we can get the best (fastest) algorithm for the job at hand. We show, that by formulating the problem as a Markov Decision Process and estimating the cost based on trial runs of the algorithms we can solve problems more efficiently.

- For many computational problems, there are multiple algorithms to choose from. Typically, one algorithm excels with a certain kind of input and another one shines with some other input. Then they may both do badly in each other's strong zone. Sorting fits this description well, as quicksort does best with large samples, mergesort with medium and insertionsort with small samples. Most sorting implementations stick to using only quicksort, regardless of sample size. It is proposed that better solutions may often be found if the best algorithm is chosen for each step of the way, and that a cost function is needed to achieve that. For the sorting problem, one can use the size of a sample to decide what algorithm to choose. We developed an algorithm that chooses the sorting method recursively for data sets of any size. This implementation is on par with insertionsort and mergesort on small samples, while very quickly proving to be by far the best algorithm for larger samples. We conclude that this hybrid technique may well prove to be a fruitful path for optimizing solutions of various tasks, especially where the existing algorithms use recursion to tackle smaller and smaller subproblems.

- This article discusses the problem of choosing the right algorithm. We developed a hybrid sorting algorithm that is based on the Markov Decission Process. It uses InsertionSort, MergeSort and Quicksort. In the study we proved that this hybrid version can safe between 43 to 69% of runtime compared to the single sorting algorithms.

- When Programmers are looking at existing algorithms to solve a known problem they are usually faced with a wide array of algorithms. Each of them is performing differently under specific circumstances. To keep the programmer from having to decide beforehand which algorithm best fits his needs, this paper proposes a method to let the program itself decide which algorithm to use for a given problem instance. Using a predefined set of rules, the specific problem set is analyzed and the most efficient algorithm is selected. For demonstrational purposes, the method is successfully applied to three different search algorithms, but it is also proposed to use this technique on other domains of algorithms.