Evolutionary Techniques Applied to Hashing: An efficient data retrieval method

Daniar Hussain
danhussain@insanemath.com
Richland Senior High School
Johnstown, PA 15904

Steven Malliaris
mallia@attglobal.net
New Trier Township High School
Winnetka, IL 60091

Abstract

Hashing is an efficient method for storage and retrieval of large amounts of data. Presented here is an evolutionary algorithm to locate efficient hashing functions for specific data sets by sampling and evolving from the set of polynomials. Functions derived in this way show consistently better performance than other common hashing methods, and indicate the power of evolutionary algorithms in search and retrieval.

1 BACKGROUND

Hashing is a non-sequential ordering and retrieval of data that operates by assigning to each data set a unique memory address location. The relationship that maps items in the data set to the range of storage addresses is called a hashing function. Finding efficient hashing functions is a hard problem in this field, and is essential to the success of this method. Efficiency is measured by the total number of memory address probes required to locate or place one data item. Many different attempts using classical deterministic algorithms to generate hashing functions have met with limited success. (See for example Tenenbaum, et. al., 1990.) The enormous size of the search-space of functions makes it an ideal candidate for evolutionary computing.

2 THE ALGORITHM

In this context, the polynomials make up the set of evolving organisms that are thought to populate the data set, which plays a role analogous to that of the environment in biological evolution. The selection criteria is determined by the total number of collisions that a particular polynomial encounters when hashing a particular data set. A bare outline of the algorithm is as follows: (1) A random set of polynomials is generated. (2) The number of collisions when hashing with each polynomial is determined. (3) The polynomials are ranked based on their collision rating, and their relative ranking determines whether they survive to the next generation. A random mutation is also applied to all polynomials, and polynomials may be paired to simulate multiple chromosomes. These steps are repeated until a polynomial is found that achieves a desired minimum collision rating.

3 RESULTS

Table 1 reports the results comparing the evolved polynomials with two other common hashing techniques, random number generator and modulo-n. The evolved polynomials consistently have a lower number of collisions compared to the other two hashing methods, and the difference increases with larger data density. A similar trial was performed on both random data and structured data, as well as with both successful and unsuccessful search. In all four scenarios, the performance of the evolved polynomial surpassed the performance of all other hashing methods.

Table 1: Average number of probes per data item

Density	Rand	Mod n	Polynomial
0.25	1.128	1.412	1.436
0.50	1.462	1.696	1.244
0.75	2.430	2.68	1.70
0.90	4.77	6.48	2.47
0.95	7.68	10.8	3.07

References

A. Tenenbaum, Y. Langsam, and M. J. Augestein (1990). Hashing. *Data structures using C*, 454-502.