A Multi-GPU Based Approach to the 0/1 Knapsack Problem Using the Discrete Shuffled Frog Leaping Algorithm

Pranav Bhandari, Rahul Chandrashekhar and Peter Yoon

Department of Computer Science, Trinity College, Hartford, CT

Trinity College
HARTFORD CONNECTICUT

The DSFL Algorithm

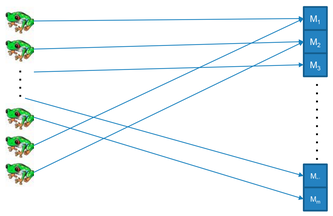
Initialization Step

Generate a random population of frogs, ***P*** and arrange them in decreasing order of *fitness*

Each frog is an *n*-bit binary string where each bit represents an item of the knapsack. Each included item is represented by 1.

Divide

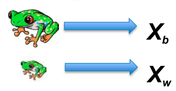
Divide the population of frogs ***P*** into ***M*** memeplexes, so $P = M \cdot X$



Conquer

Perform a local search within each memeplex m_i

Step 1: Determine the best frog X_b and worst frog, X_w of each memeplex



Step 2: Try improving the state of the worst frog and updating it by

$$D_i = Rand() \cdot (X_b - X_w)$$

where D_i is the change in the position of the i^{th} frog and the new position is given by

$$X_w(new) = X_w + D_i$$

Rand() is a random number generated from {0, 1}

Step 3: If the above process produces a better result, then it replaces the worst frog or else the above process if repeated with

$$D_i = Rand() \cdot (X_g - X_w)$$

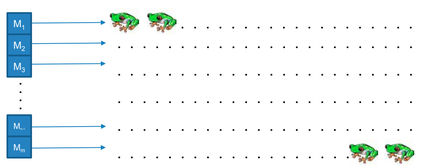
where X_g is the frog with the global best fitness

Every Conquer step is followed by a shuffle operation and the Divide step.

Merge

The algorithm terminates when no further improvement is possible

As it is a meta-heuristic algorithm, the frogs from the memeplexes are merged and an optimal solution set is returned.



The Knapsack Problem is a popular combinatorial optimization problem which is of the type NP-hard. It assumes a case where there is a knapsack which can hold a maximum weight ***W***. There is a set of items ***N*** from which each item has its own weight and value. The task is to pack the knapsack with the maximum possible value while staying under the weight limit of ***W***. The 0/1 Knapsack Problem is a unique case of the classic Knapsack Problem in which each item from the set is either included or excluded in its entirety. A brute force approach can be used which would generate all the subsets of ***N*** and compare them to get the most optimal solution. But as the input size increases, the number of subsets also increases exponentially making this approach computationally impractical. We propose a GPU-based approach to the Discrete Shuffled Frog Leaping Algorithm as a computationally more efficient implementation to solve this problem. It employs the use of dividing the problem into multiple sub-problems making it more suitable for parallel computation. We use multiple GPU threads which simultaneously work on the different sub-problems, hence making the computation much faster and efficient. The comparison of our implementation with the serial implementation gives a speedup of up to 5x for large data sets. Our present approach employs only a single GPU and we further aim to extend it to a multi-GPU approach in the near future.

A GPU-Based Approach

- The DSFLA is embarrassingly parallelizable hence making our approach a viable alternative to the serial approach.
- The multiple parts of the algorithm, namely the Initialization, Divide, Conquer and Merge steps can all be parallelized to increase the performance by as much as 5x.
- The most time consuming and repeated process in the serial approach is the Local Search. Through the parallel approach, the time taken by this process is highly minimized and this effect is more distinct as we move to higher dimensions of the problem.

Applications to Real Estate Data

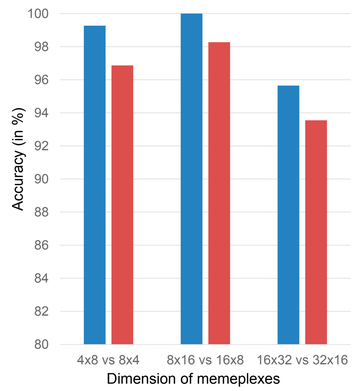
- Real estate data proved to be suitable to test our algorithm on as a relatively fewer number of factors are required to predict the best estate to invest in. Data regarding real estate prices from the Hartford county was gathered.
- Other factors like demand of the estate, change in price, profit/loss from the previous year, change in sales from the previous year and expected sales in the following year are taken into account to get the most optimum result.
- This approach to calculating the most optimum estate to invest in can be extended to other counties and the state as well.

Reference

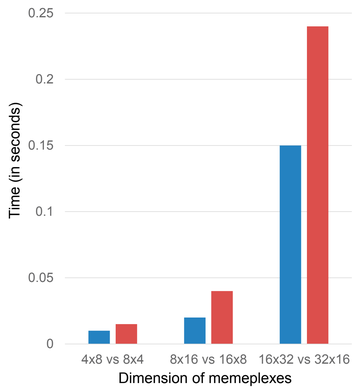
1. K.K. Bhattacharjee and S.P. Sarmah, "*Shuffled Frog Leaping Algorithm and its Application to the 0-1 Knapsack Problem*", 2012
2. D. M. Munoz, C. H. Llanos, L. S. Coelho and M. Ayala-Rincon, "*Accelerating the Shuffled Frog Leaping Algorithm by Parallel Implementations in FPGAs*", 2010

Performance Results

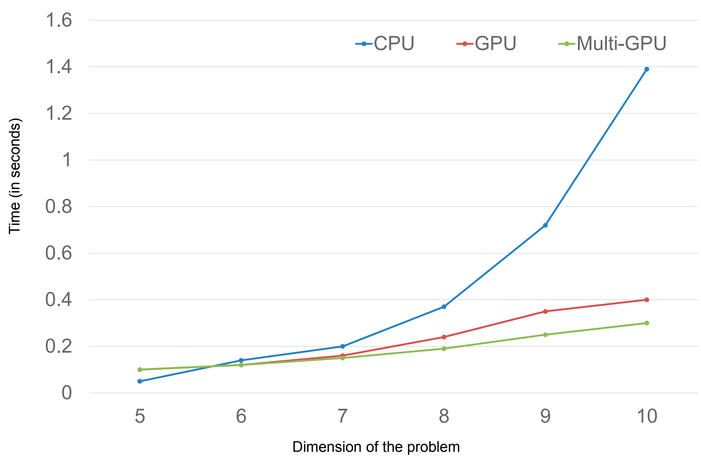
We tested our implementation of the DSFLA under the specifications : Intel® Xeon® E5-2620 CPU + NVIDIA® Tesla® K20c GPU + 64 GB main memory + 5GB GPU memory



Accuracy Result



Timing Result



We compared our GPU-based implementation of the DSFLA with a serial version of the algorithm and observed a speedup of up to 5x for larger data sets. We further extended our approach on a multi-GPU system with 2 GPUs and got a nearly 2x speedup as compared to a single GPU.

Future Work

- Optimization of the algorithm to handle much larger data sets more efficiently
- Extend our application to more extensive data which include other counties and states

Acknowledgment

This research was supported by

- CUDA Teaching Center Program, Nvidia Research
- Interdisciplinary Science Program, Trinity College