CMPEN 431 – Project 2

Design Space Exploration

Report

Savinay Nagendra (sxn265)

1. Description:

Design Space Exploration is defined as the process of evaluating multiple design configurations for a system and choosing the optimal (best) configuration based on some pre-defined metrics. In this project, the goal was to explore the design space of a processor and select the best performing design based on Execution Time and Energy Efficiency. We were provided with 18 dimensions, each having multiple parameter values. The goal was to find best value for each dimension. Since the total number of designs in the space was of the order of $\sim 10^{12}$, searching exhaustively is impossible. A framework was needed to quickly test designs and calculate metrics, based on which new configurations could be picked up. This is achieved with the help of the provided framework. We were provided access to an architectural simulator called SimpleScalar, which enabled a study of how different processor and memory system parameters affect performance (time) and energy efficiency with minimum simulation time. With the help of some base constraints, many design points (configurations) were marked invalid. Since the simulator stores access files for seen configurations, the simulation time further decreased when testing multiple times. Thus, the provided framework was successfully used to explore the given design space and extract the best configurations based on both performance and energy efficiency.

2. Design Points Chosen by DSE

a) Performance (Best Time):

i) bestTimeconfig:

0 1 0 2 0 8 0 3 0 3 0 0 4 3 2 0 0 0

ii) bestEDPconfig:

0 1 0 2 0 8 0 3 2 3 0 0 4 3 2 0 0 0

b) Energy Efficiency

i) bestTimeConfig:

0 0 0 2 0 7 1 0 3 2 0 0 4 3 0 0 0 0

ii) bestEDPConfig:

0 0 0 2 0 7 0 0 3 2 0 0 4 3 1 0 0 0

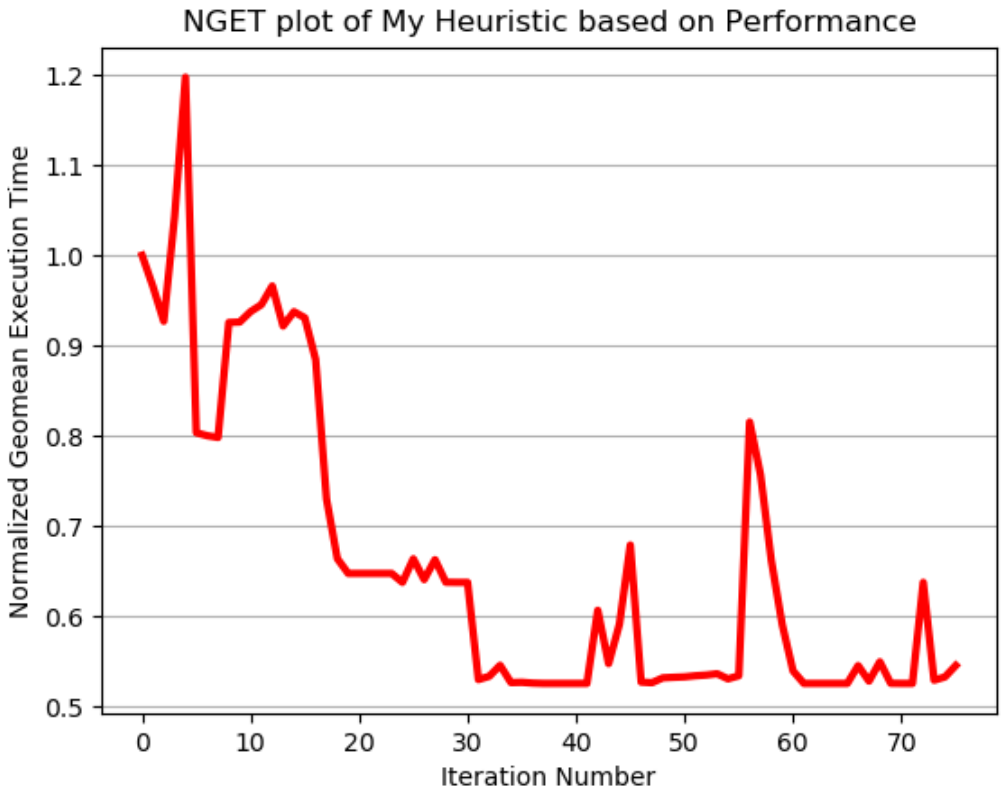
A1. Table

Parameter	Performance	EDP
width	Value = 1 Why = Pipelining leads to higher IPC, thus reducing time. Multi-width pipelines have more complications and a lot more data hazard checks	Value = 1 Why = Pipelining leads to higher IPC, thus reducing time and hence, EDP. Multi-width pipelines have more complications and a lot more data hazard checks
scheduling	Value = Out-of-order Why = OOO executes every instruction greedily, thus leading to a faster execution	Value = Out-of-order Why: Stalling does not happen in OOO, leading to lesser energy
l1block	Value = 8 Why = Higher block size means lower miss rate. Very high block size is also bad. Hence, optimal value, such as the one chosen, will reduce AMAT, hence giving better performance	Value = 8 Why = Higher block size means lower miss rate. Very high block size is also bad. Hence, optimal value, such as the one chosen, will reduce AMAT, hence giving better performance
dl1sets	Value = 128 Why = Optimal number of data sets reduces miss rates	Value = 128 Why = Optimal number of sets reduces miss rates
dl1assoc	Value = 1 Why = Higher data associativity improves Cache miss rates	Value = 1 Why = Higher data associativity improves Cache miss rates
il1sets	Value = 8192 Why = More Instruction Sets increases IPC	Value = 8182 Why = More Instruction Sets increases IPC
il1assoc	Value = 1 Why = Low instruction associativity has been chosen as the number of sets is very high	Value = 1 Why = Low instruction associativity has been chosen as the number of sets is very high
ul2sets	Value = 2048 Why = Ideally L1 will be fast enough to match the speed of the CPU while L2 will be large enough to reduce the penalty of going to main memory. More sets in L2 decreases Global Miss Rate	Value = 2048 Why = Ideally L1 will be fast enough to match the speed of the CPU while L2 will be large enough to reduce the penalty of going to main memory. More sets in L2 decreases Global Miss Rate
ul2block	Value = 16 Why = Optimal block size reduces Global AMAT	Value = 64 Why = Optimal block size reduces Global AMAT. A relatively higher block size is chosen for more energy efficiency.
ul2assoc	Value = 8 Why = Higher associativity leads to lesser Global miss rates	Value = 8 Why = Higher associativity leads to lesser Global miss rates
replacepolicy	Value = LRU Why = Most used policy as multiple data blocks and instructions, once used, would be used again with high probabilities.	Value = LRU Why = Most used policy as multiple data blocks and instructions, once used, would be used again with high probabilities.
fpwidth	Value = 1 Why = More ALUs allow extraction of more ILP and increase performance	Value = 1 Why = More ALUs allow extraction of more ILP and increase performance
branchsettings	Value = -bpred comb -bpred:comb 1024 Why = Branch Prediction takes care of control hazards, thus increasing performance.	Value = -bpred comb -bpred:comb 1024 Why = Branch Prediction takes care of control hazards, thus increasing performance.
ras	Value = 8 Why = Higher the value, lower the miss penalty	Value = 8 Why = Higher the value, lower the miss penalty
btb	Value = (512 – 4) Why = Larger btb has lesser misses	Value = (512-4) Why = Larger btb has lesser misses

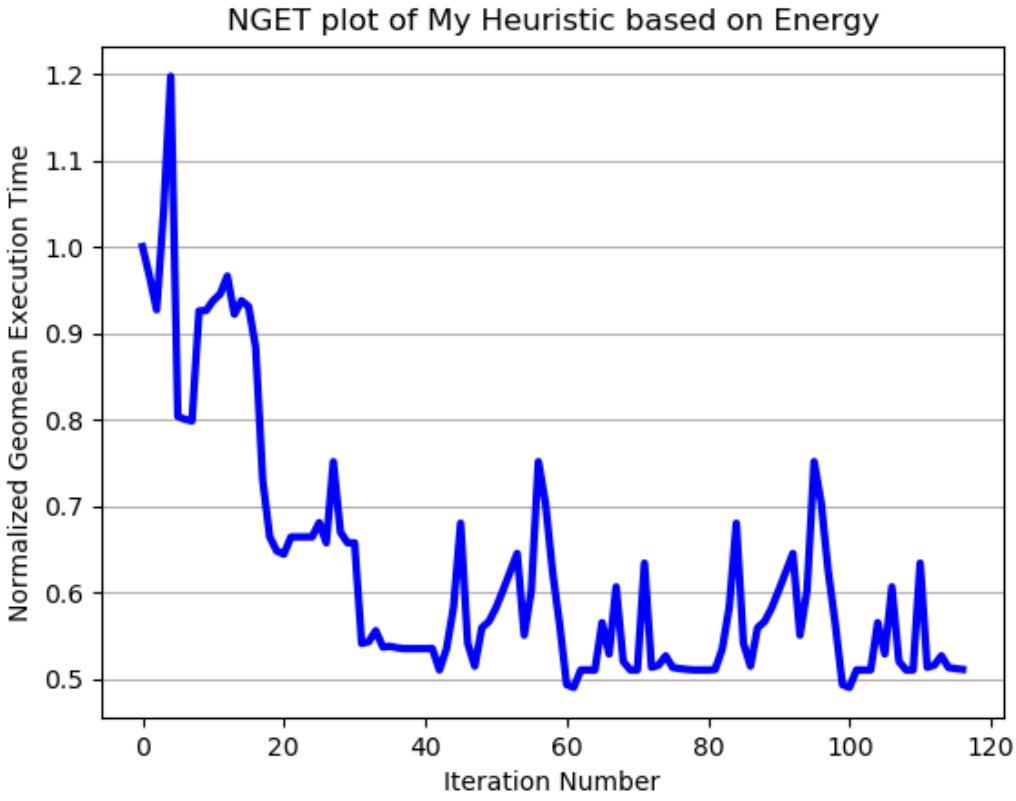
Note: The dependent dimensions are not mentioned.

A2. Plots

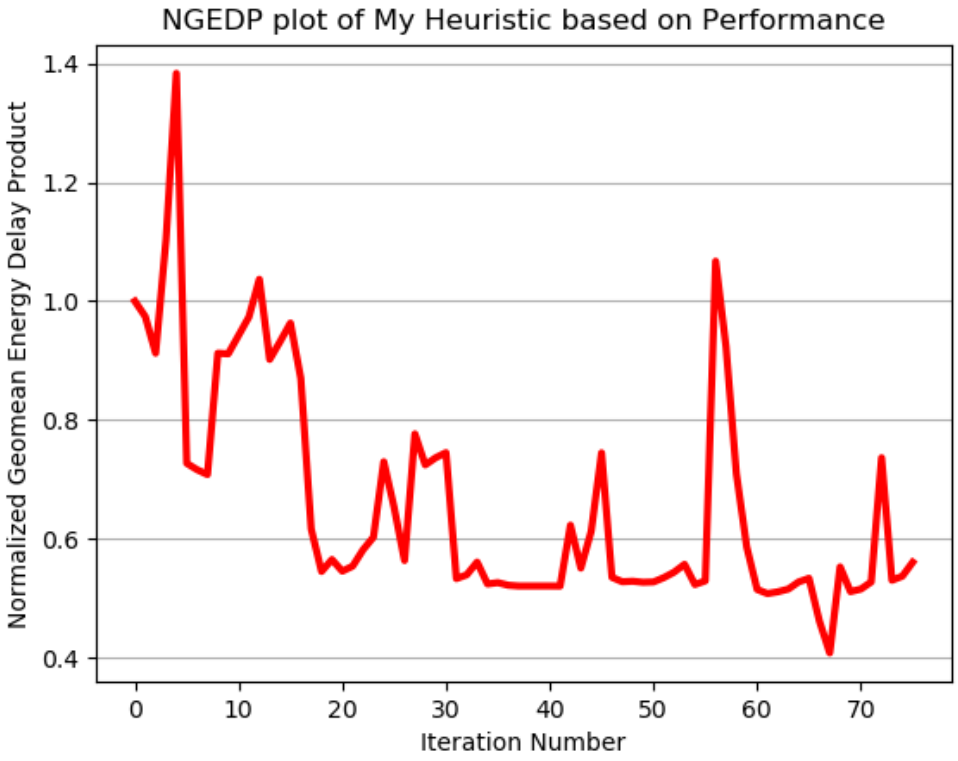
A. NGET: Performance



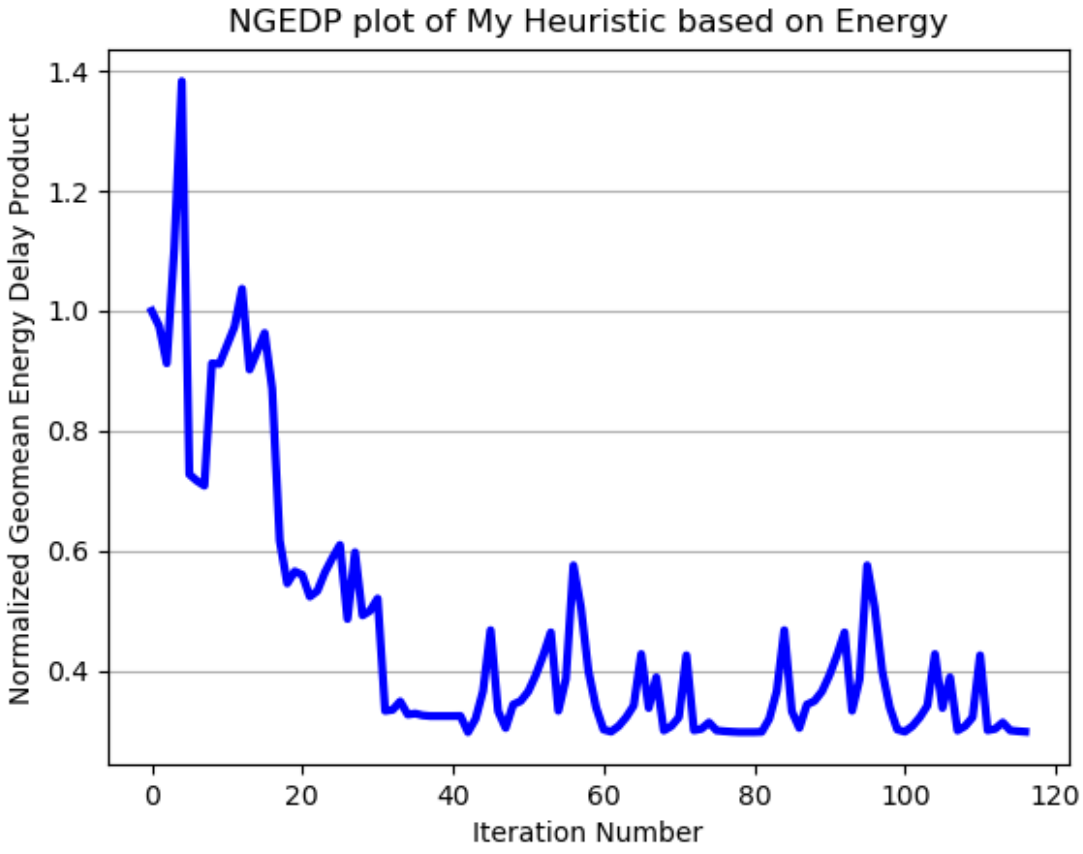
NGET: Energy



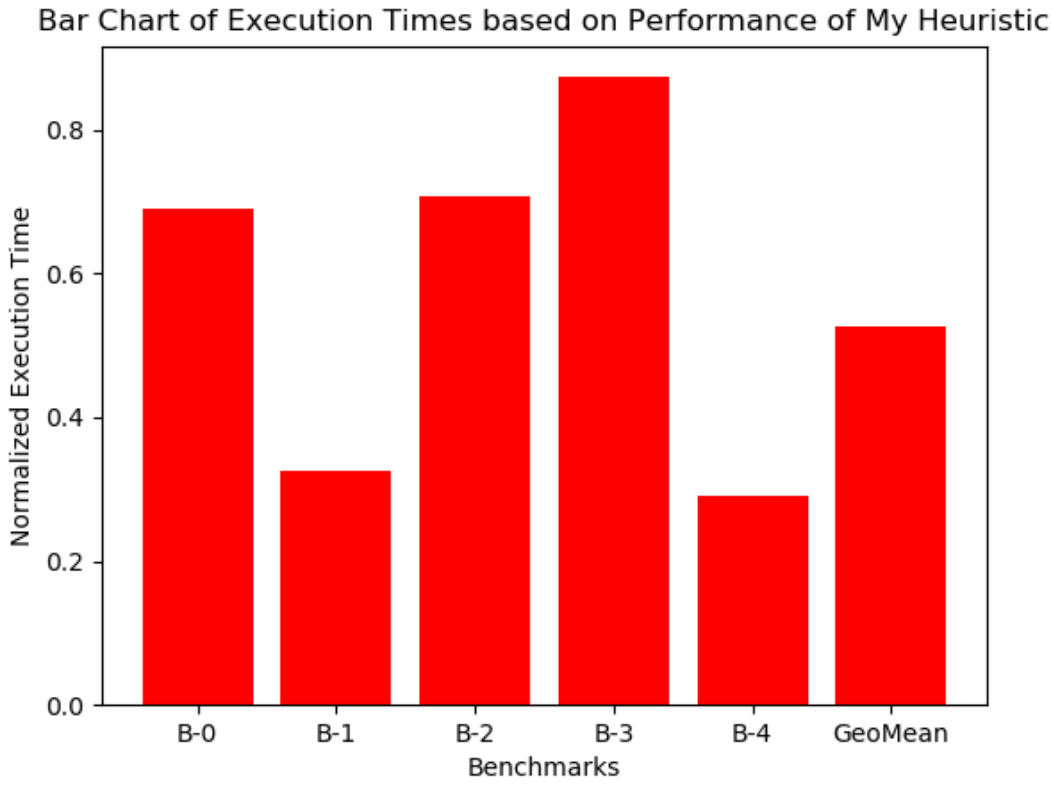
B. NEDP: Performance



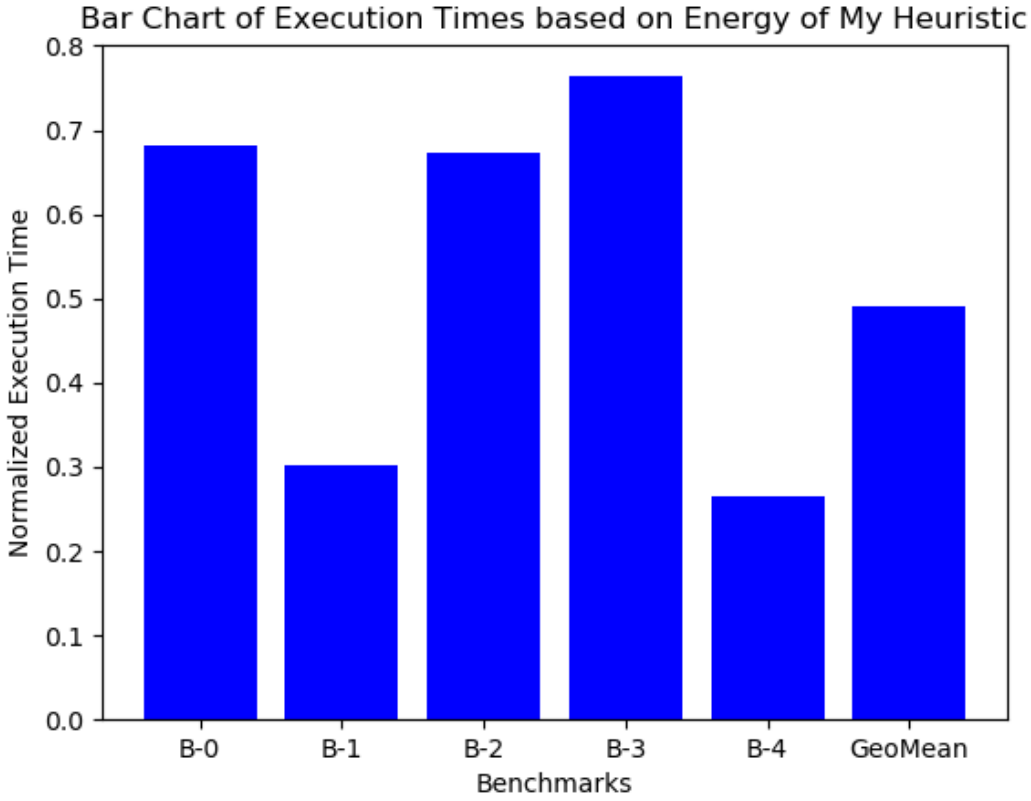
NEDP: Energy



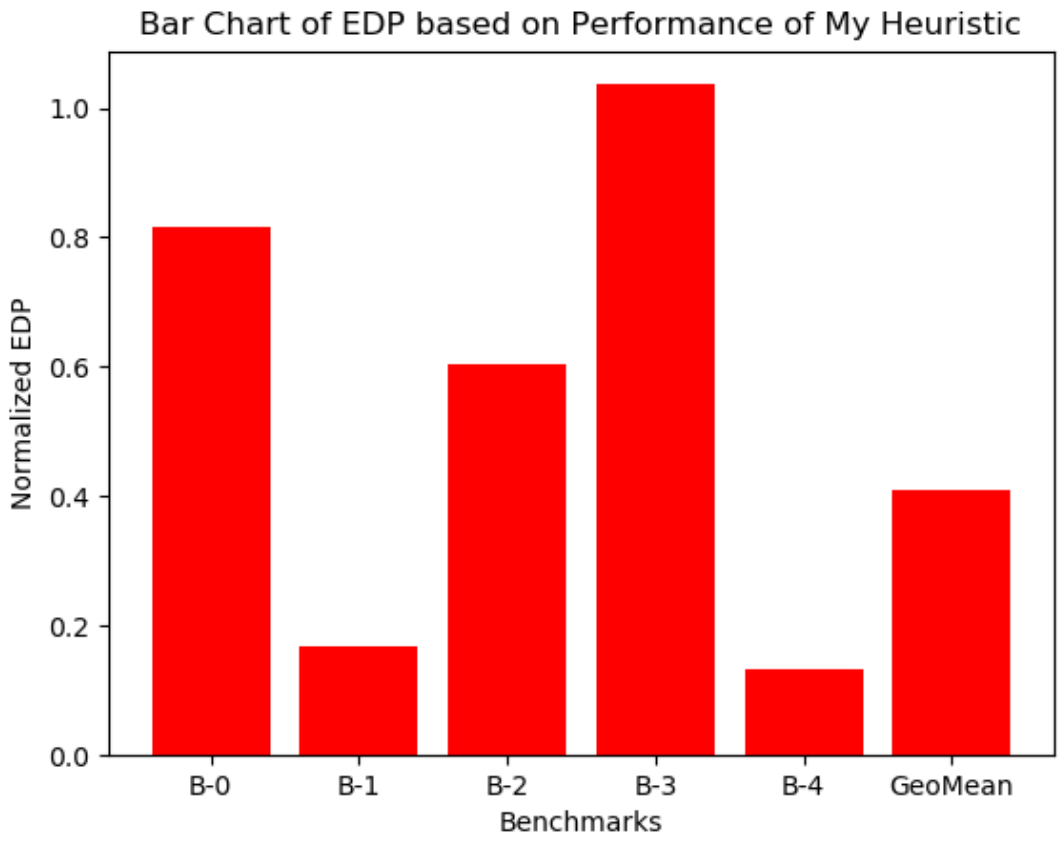
C. ExecTime: Performance



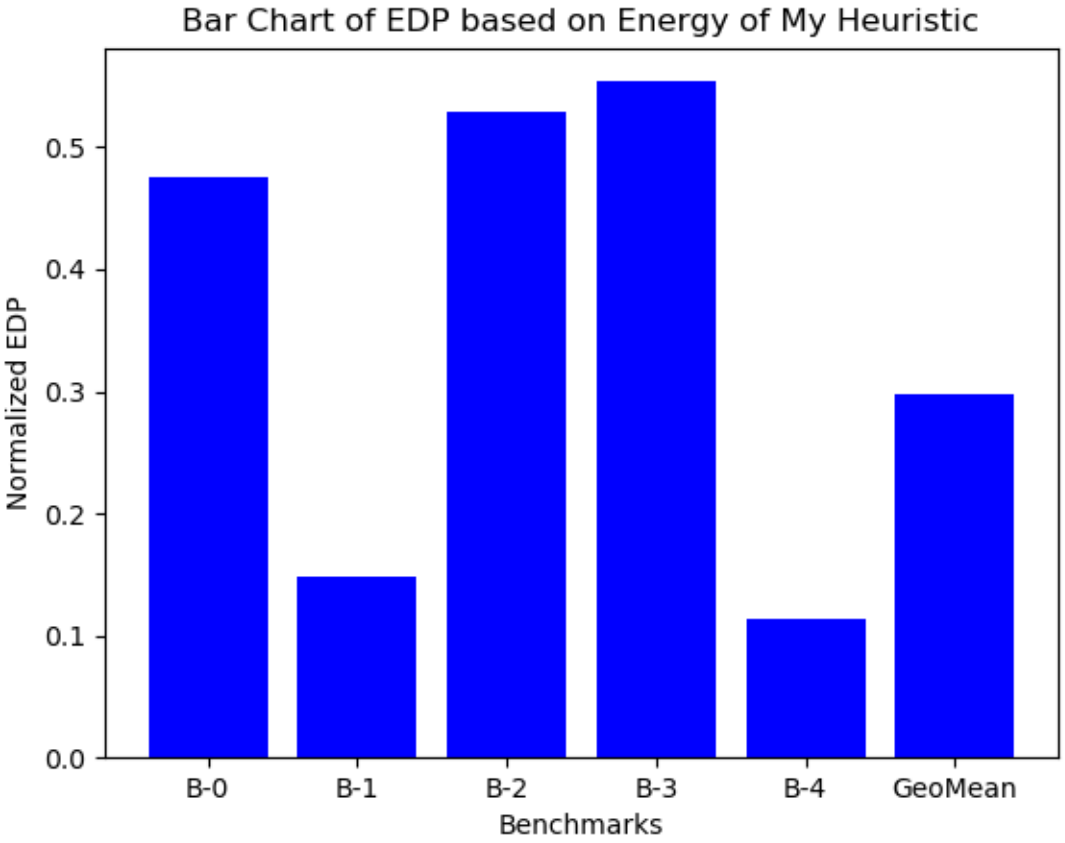
ExecTime: Energy



D. EDP: performance



EDP: Energy



5. More Sophisticated Heuristic

This example is a perfect example of **Reinforcement Learning**, a domain of Machine Learning designed for state space exploration. Every Reinforcement Learning system is designed as a State-Action-Reward model. The state space is analogous to the design space in this problem, and the action is analogous to the configuration to be chosen. For every configuration that decreases the EDP and Execution Time, a positive reward (reinforcement) is provided so that the system confidently explores the neighboring values. Also, reinforcement learning is based on a process called exploration-exploitation, which increases performance and leads to a better search algorithm. Also, with enough sampling, the model might learn to obtain the local minima, or the best configuration possible. It is more sophisticated, but certainly will find the best design point within 1000 iterations.

6. Insights gained

a) As a person with Machine learning background and with a lot of experience in Reinforcement learning, I was able to connect the Reinforcement learning method used for an optimization problem I solved at Schlumberger Doll Research during my internship to this problem. Both problems are design space exploration, but in very different domains, one in architecture and the other in Oil & Gas. But both have analogous applications.

b) I gained insight SimpleScalar simulator, which I had never heard of before. I got a glimpse of how processors are designed on an industrial level with lot of customizations and sophisticated mechanisms.