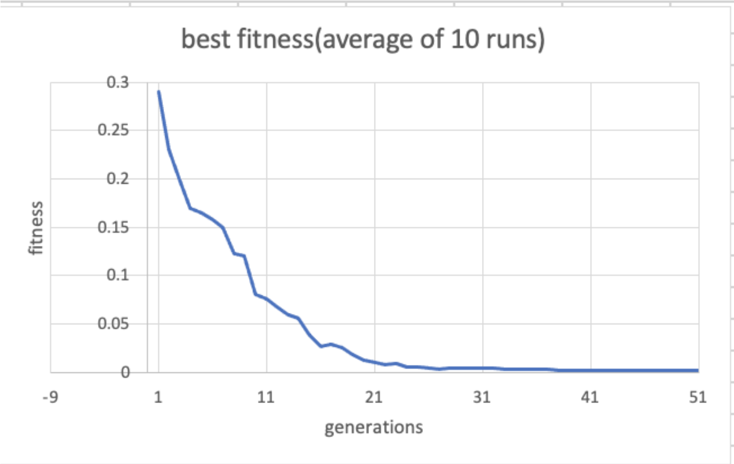
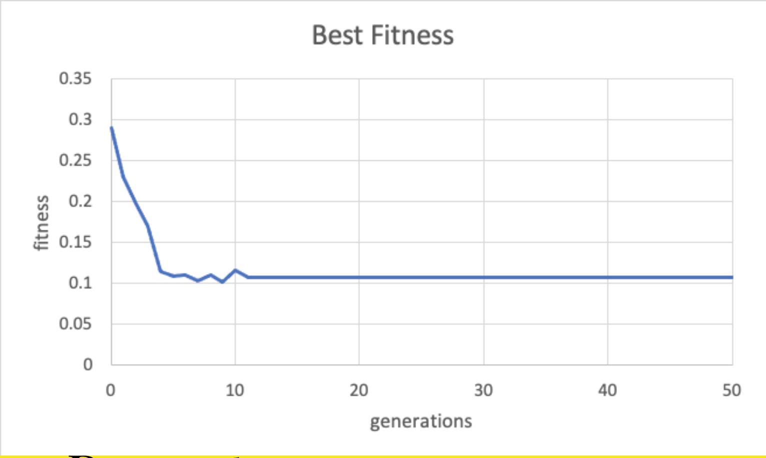
Advanced Training Strategies: Pyramid Search

Genetic Programming

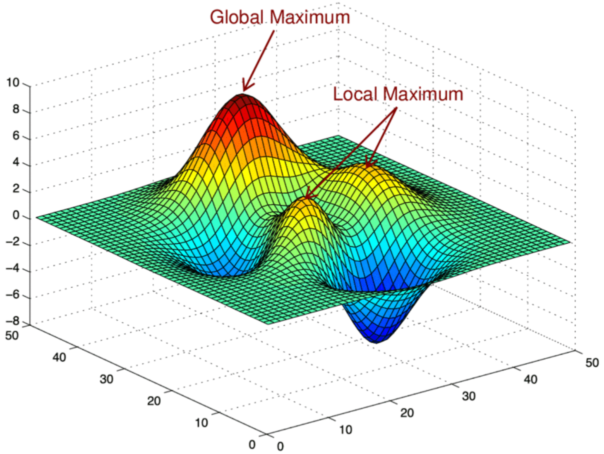
By Thanushan Pirapakaran, Adrian Binu, Brett Terpstra



Normal Graph



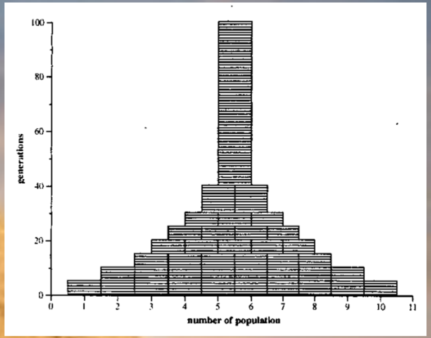
Premature convergence



Idea:

- Run many independent GP runs at the same time (in Parallel)
- Compare fittest individuals between the program's populations
- Prunes least fit populations
- No sharing of individuals between populations

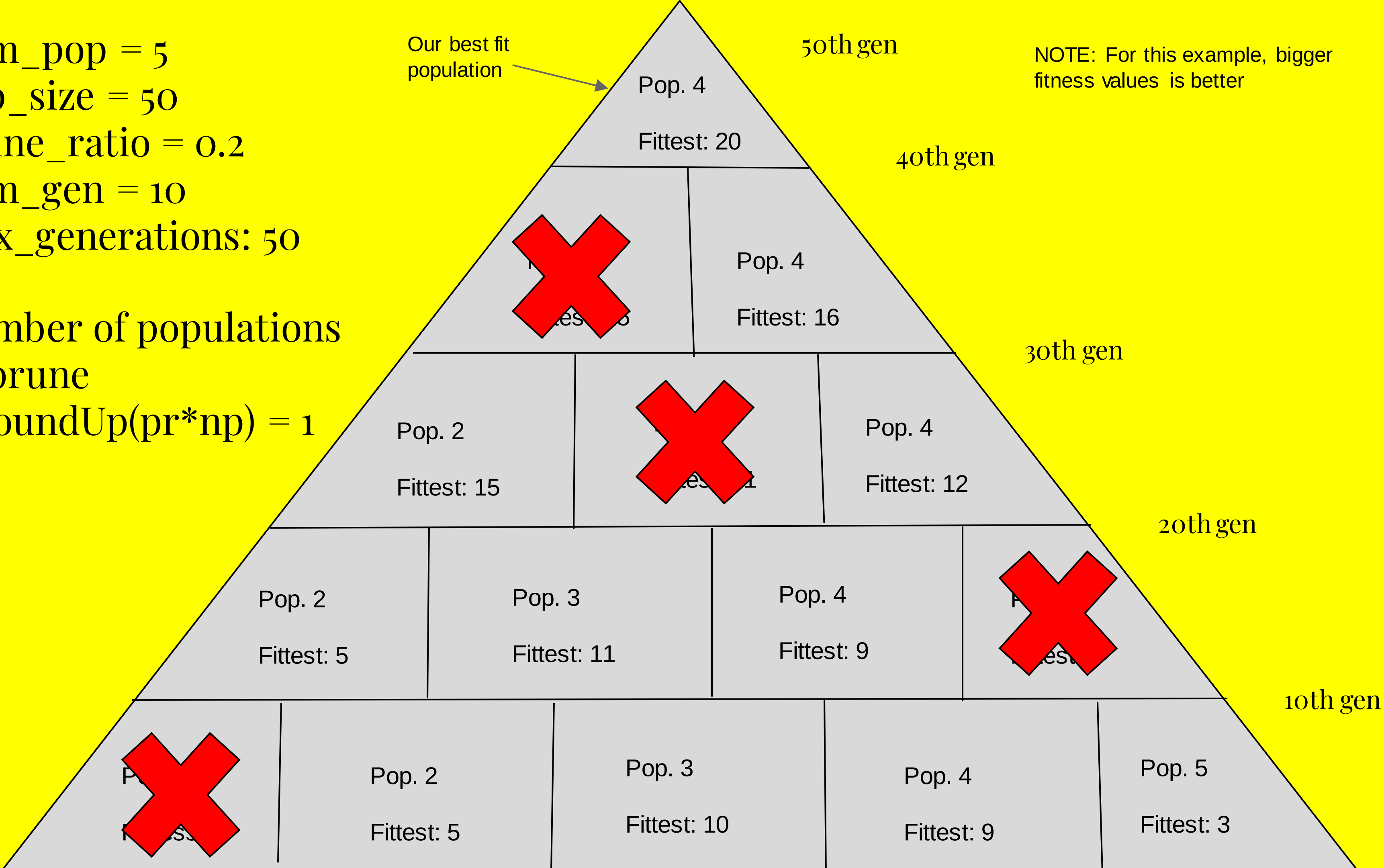
Pseudo Code




```
def pyramid(num_pops, pop_size, prune_ratio, num_gen, max_generations){
  popsleft = num_pops
  while(not solved and popsleft != 1 and generations < max_generations){
    evolve_pops(pop_size, num_gen)
    remove_least_fit_pops(round(popleft * prune_ratio))
    popsleft = popsleft - round(popleft * prune_ratio)
    generations += num_gen
  }
}
```

num_pop = 5
pop_size = 50
prune_ratio = 0.2
num_gen = 10
max_generations: 50

Number of populations to prune
= roundUp(pr*np) = 1



Pros of Pyramid Search

- Great for large scale problems or if the problems have a complex fitness landscapes with multiple local optima

- It is faster than running multiple vanilla GP runs because it prunes out worst fit runs

- It has a higher probability of success with fewer evaluations

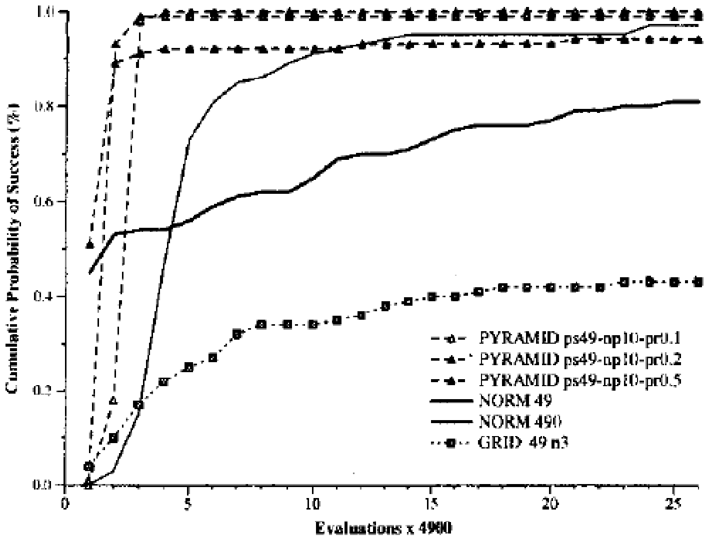
Cons of Pyramid Search

- There are now more parameters to consider and to optimize
- Pyramid Search has to maintain multiple runs so it increases the memory demand
  - Depends on setup and intention, running the same number of subpopulations, pyramid search uses less memory vs running all to completion
  - Pyramid search's increased memory requirements comes from having to run many subpopulations at the same time.

Pyramid Search vs normal paper comparison

Pyramid search has high probability of success with fewer evaluations

"total evaluations" typically refers to the total number of fitness evaluations performed during the execution of the algorithm.



Methods	Num Suc	Mean	SD.
Norm49	81	21740	30838
Norm490	97	24361	16870
Pyr-ps49-np10-pr0.1	100	11253	1675
Pyr-ps49-np10-pr0.2	99	7961	1194
Pyr-ps49-np10-pr0.5	94	6885	11099

Table 1: Mean and Standard Deviation, Number of Evaluations to First Solution, Max Problem

Methods	Runs	Total Evals
Pyr-ps49-np-100-pr0.2	4 for 27x49,000	108x49,000
Pyr-ps49-np-100-pr0.5	7 for 17x49,000	119x49,000
Pyr-ps49-np-10-pr0.1	8 for 15x49,000	120x49,000
Pyr-ps490-np-10-pr0.1	5 for 26x49,000	130x49,000
Pyr-ps490-np-10-pr0.5	8 for 17x49,000	136x49,000
Pyr-ps49-np-10-pr0.5	35 for 4x4,9000	140x4,9000
Pyr-ps49-np-100-pr0.1	3 for 48x49,000	144x49,000
Pyr-ps490-np-10-pr0.2	7 for 23x49,000	161x49,000
Norm4900	7 for 23x49,000	161x49,000
Norm49	32 for 7x49,000	224x49,000
Pyr-ps49-np-10-pr0.2	7 for 45x49,000	315x49,000
Norm490	7 for 46x49,000	322x49,000

WE IMPLEMENTED IT!!!!

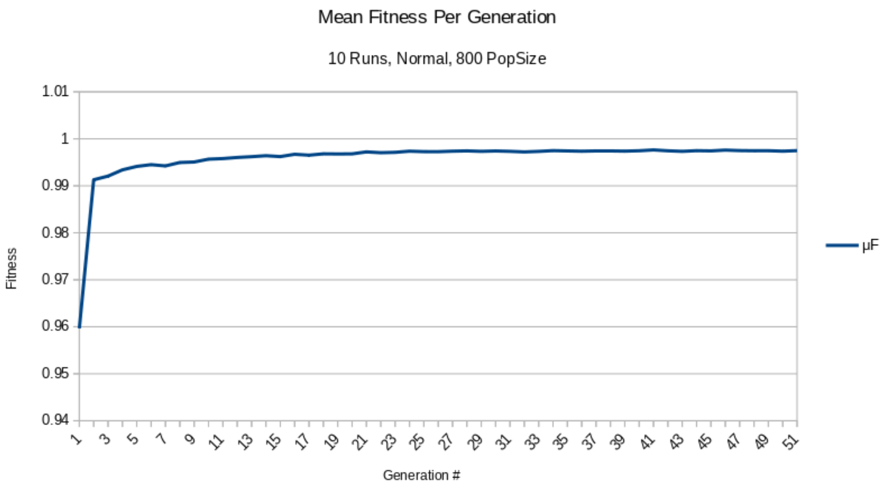
The parameters the paper used

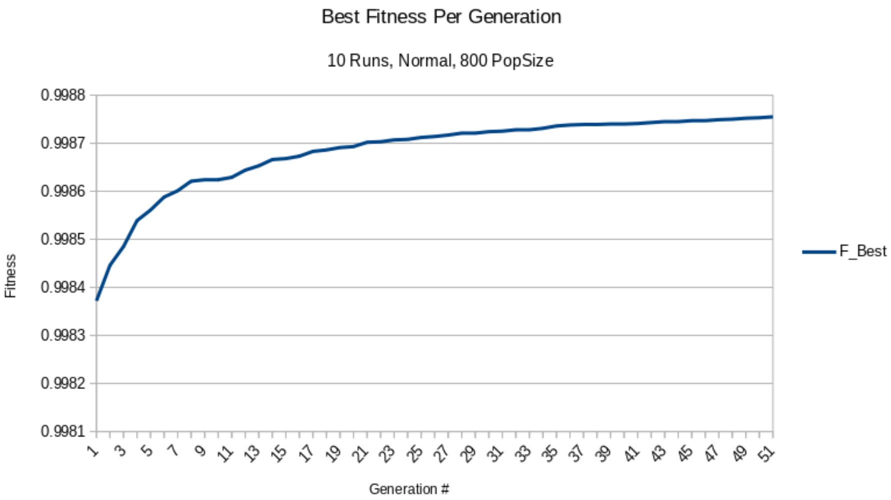
- Variables
- num_pop = 10
- pop_size = 50
- prune_ratio = 0.2
- num_gen = 10
- max_generations: 100

Before we implemented it we were looking for a speedup, an improvement in premature convergence, overall fitness results and memory consumption

Assignment 1 Part B

- Assignment 1 part B also suffered from premature convergence

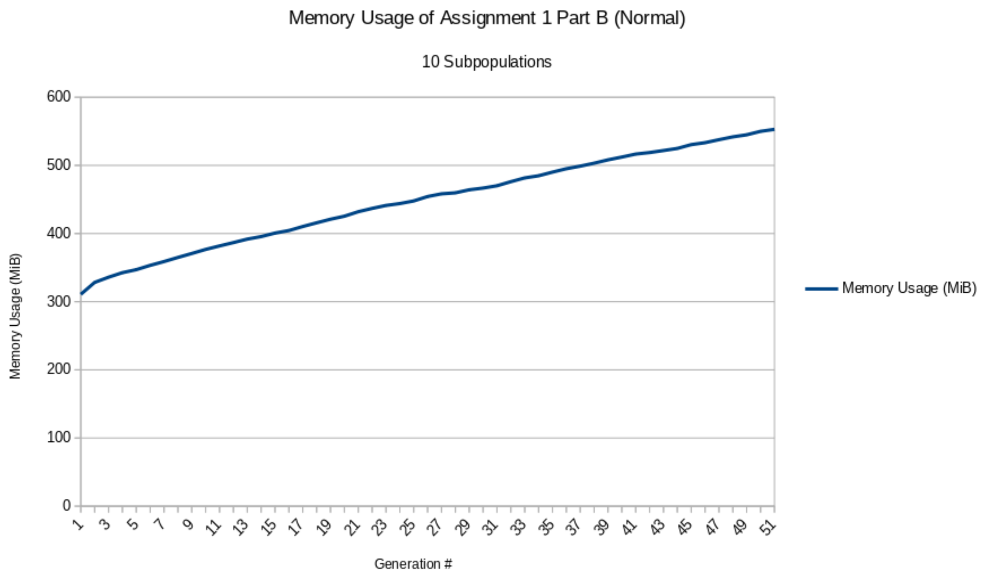




Assignment 1 Part B Cont.

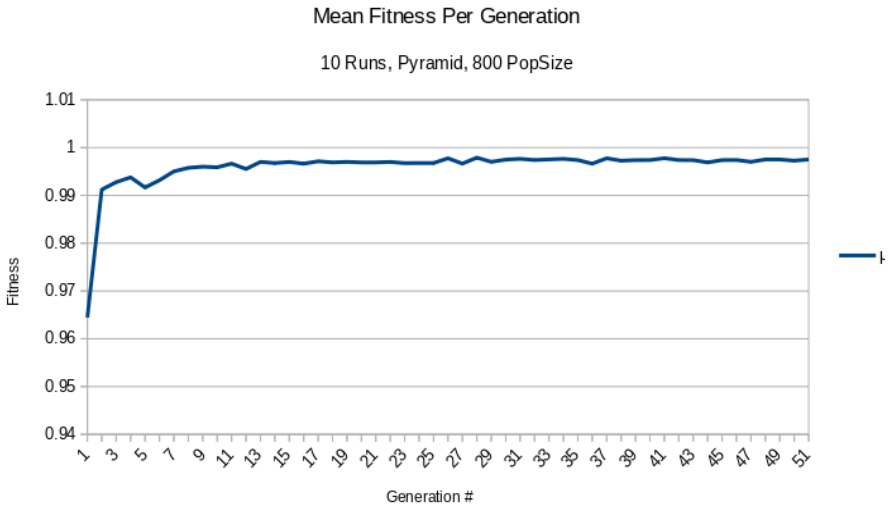
- Average Execution Time 6.786 Seconds

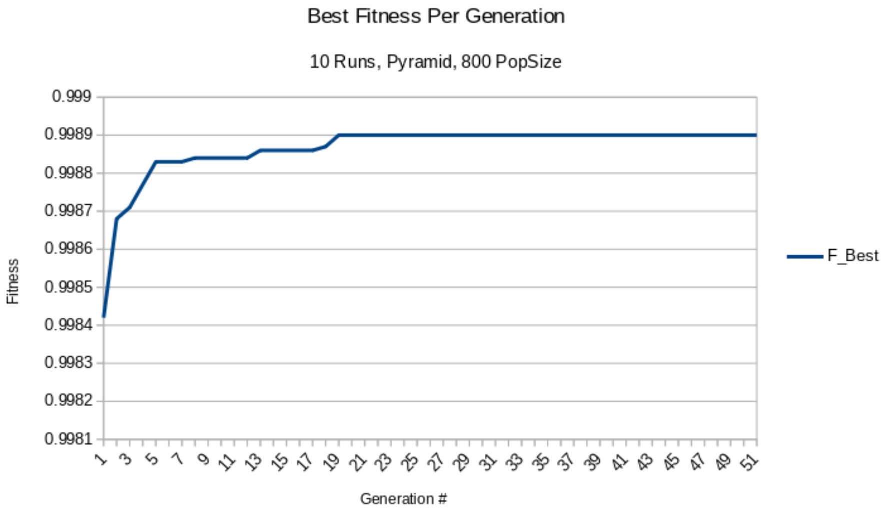
Testing Set			
TARGET / OUTPUT	Cammeo	Osmancik	SUM
Cammeo	945 33.65%	370 13.18%	1315 71.86% 28.14%
Osmancik	183 6.52%	1310 46.65%	1493 87.74% 12.26%
SUM	1128 83.78% 16.22%	1680 77.98% 22.02%	2255 / 2808 80.31% 19.69%



Applying Pyramid Search To Assignment 1

- Using 10 sub-populations, a prune ratio of 0.2 and 10 generations between pruning



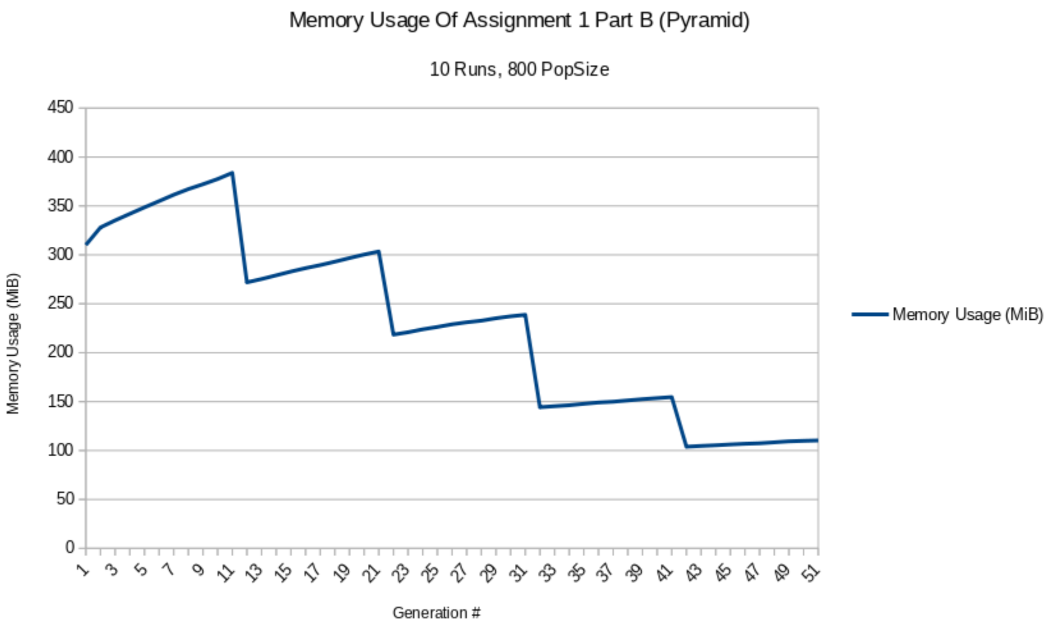


Applying Pyramid Search To Assignment 1 Cont.

- Average Execution Time 4.6055 Seconds

Testing Set			
TARGET / OUTPUT	Cammeo	Osmancik	SUM
Cammeo	1022 36.37%	121 4.31%	1143 89.41% 10.59%
Osmancik	104 3.70%	1563 55.62%	1667 93.76% 6.24%
SUM	1126 90.76% 9.24%	1684 92.81% 7.19%	2585 / 2810 91.99% 8.01%

Before it was 80.31%



Conclusions

- Running pyramid search gave better results on average in less time
- According to our tests we did not reduce premature convergence or increase genetic diversity of our populations
- Peak memory usage was less than running 10 subpopulations without with pyramid search

References

[1] Ciesielski, V., & Li, X. (2003). Pyramid search: finding solutions for deceptive problems quickly in genetic programming. *CEC: 2003 CONGRESS ON EVOLUTIONARY COMPUTATION, VOLS 1-4, PROCEEDINGS*, *2*, 936-943 Vol.2. https://doi.org/10.1109/CEC.2003.1299767

[2] Rice (Cammeo and Osmancik). (2019). UCI Machine Learning Repository. https://doi.org/10.24432/C5MW4Z.

[3] Poli, Riccardo, et al. *A Field Guide to Genetic Programming*. Lulu Press, 2008.

The MAX Problem from the Paper

- Goal is to find maximum value for a given depth using a GP with a function set of {x, +} and terminals of {0.5}
- Desired solution is a full tree with 0.5 on all terminals
- This problem which was presented in the paper is known as a "deceptive problem" that "displays obvious premature convergence behaviour"