Star discrepancies for generalized Halton points sets

T. Espitau & O. Marty | 2016

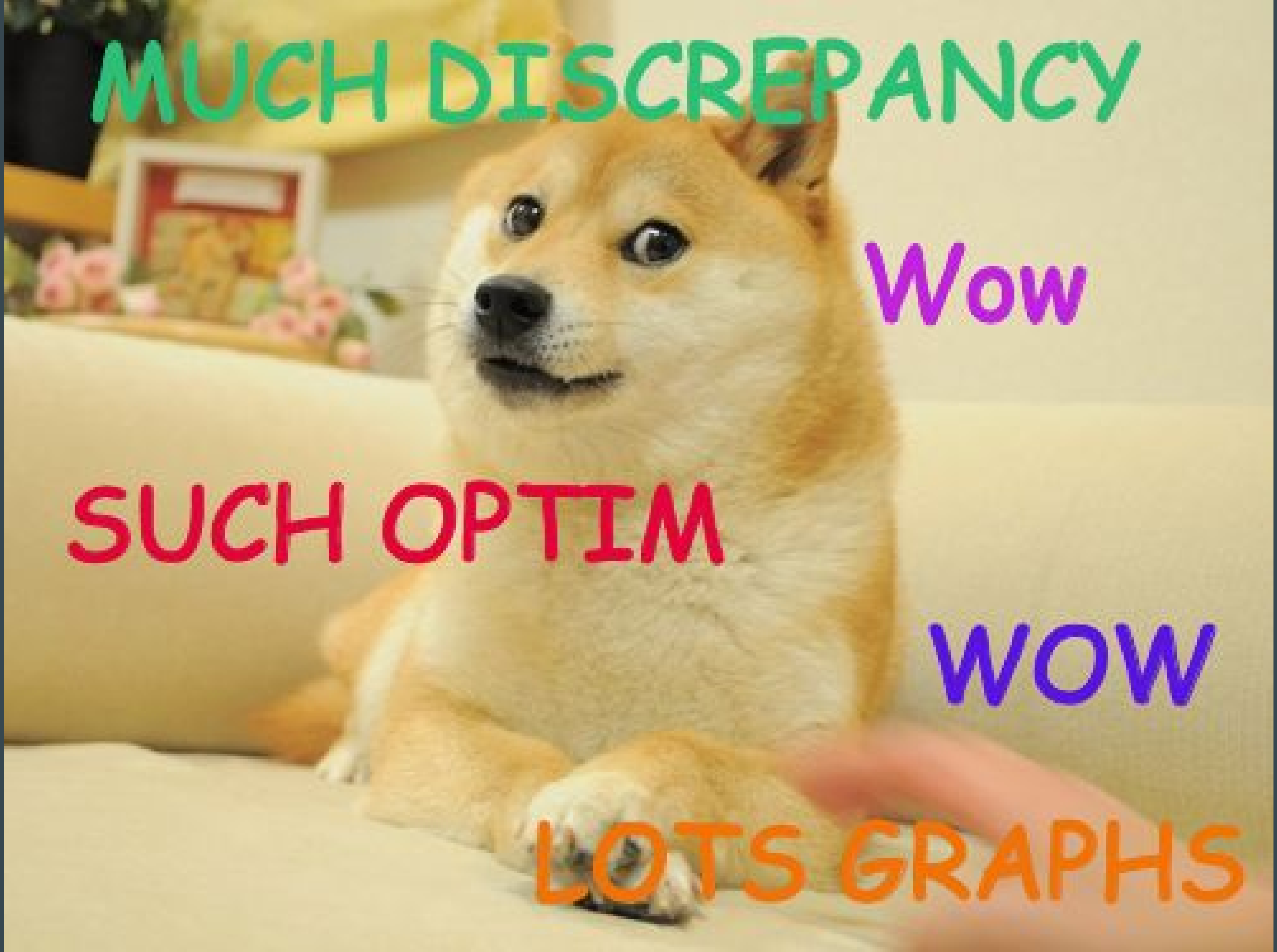
Overview

Languages used

- C | Discrepancy calculation
- C++ | Unit simulations
- Python | Wrappers

Heuristics implemented

- Fully random search
- Simulated annealing for local rand search
- $(\mu+\lambda)$ Genetic search


MUCH DISCREPANCY
Wow
SUCH OPTIM
WOW
LOTS GRAPHS

Random Search

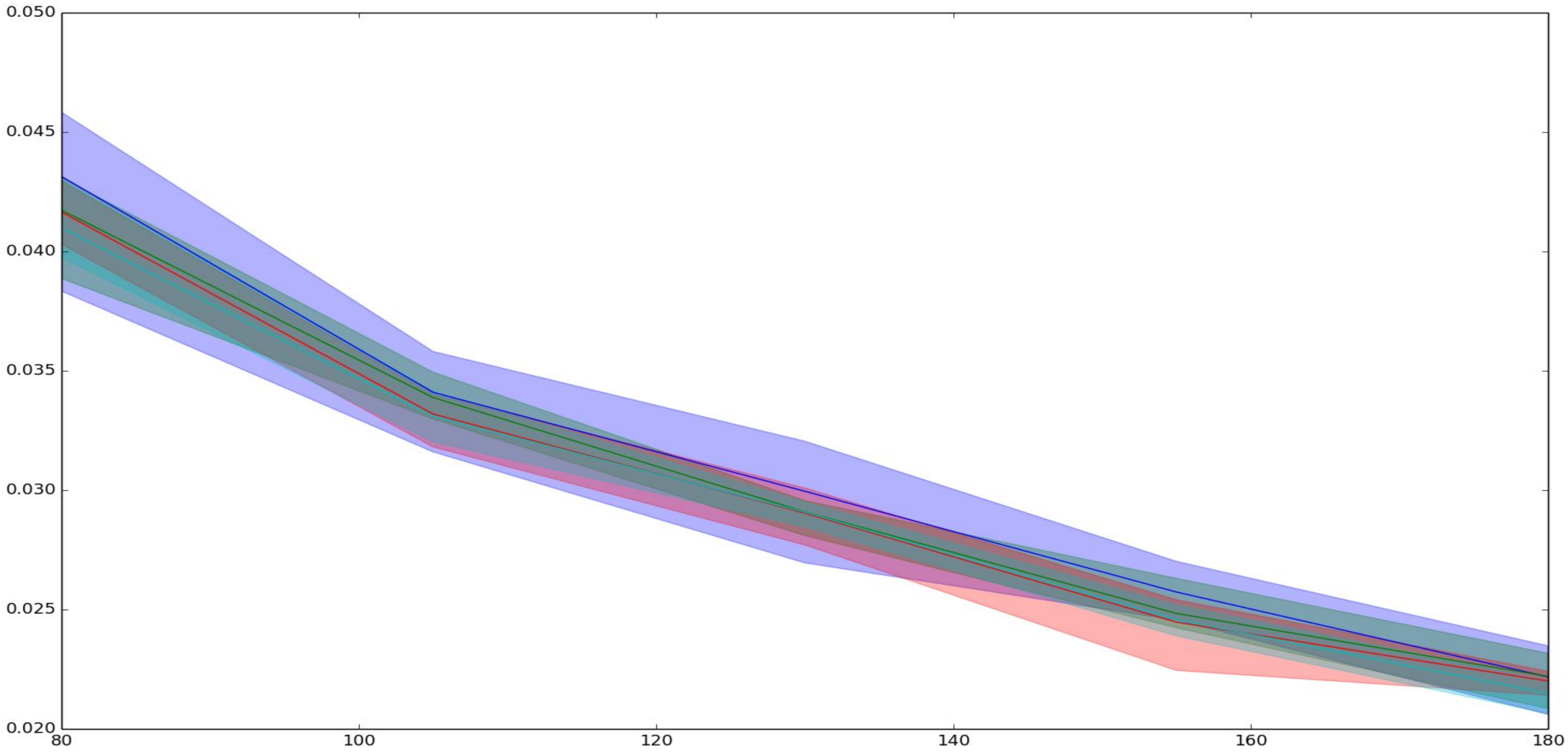
Overview

- Simplest search: regenerate the whole permutation at each iteration.
- The prime based is fixed `{7, 13, 29, 3}`
- Invariant `pi[0] = 0` maintained + *KFY shuffle* called on `pi[1,..., n]`.

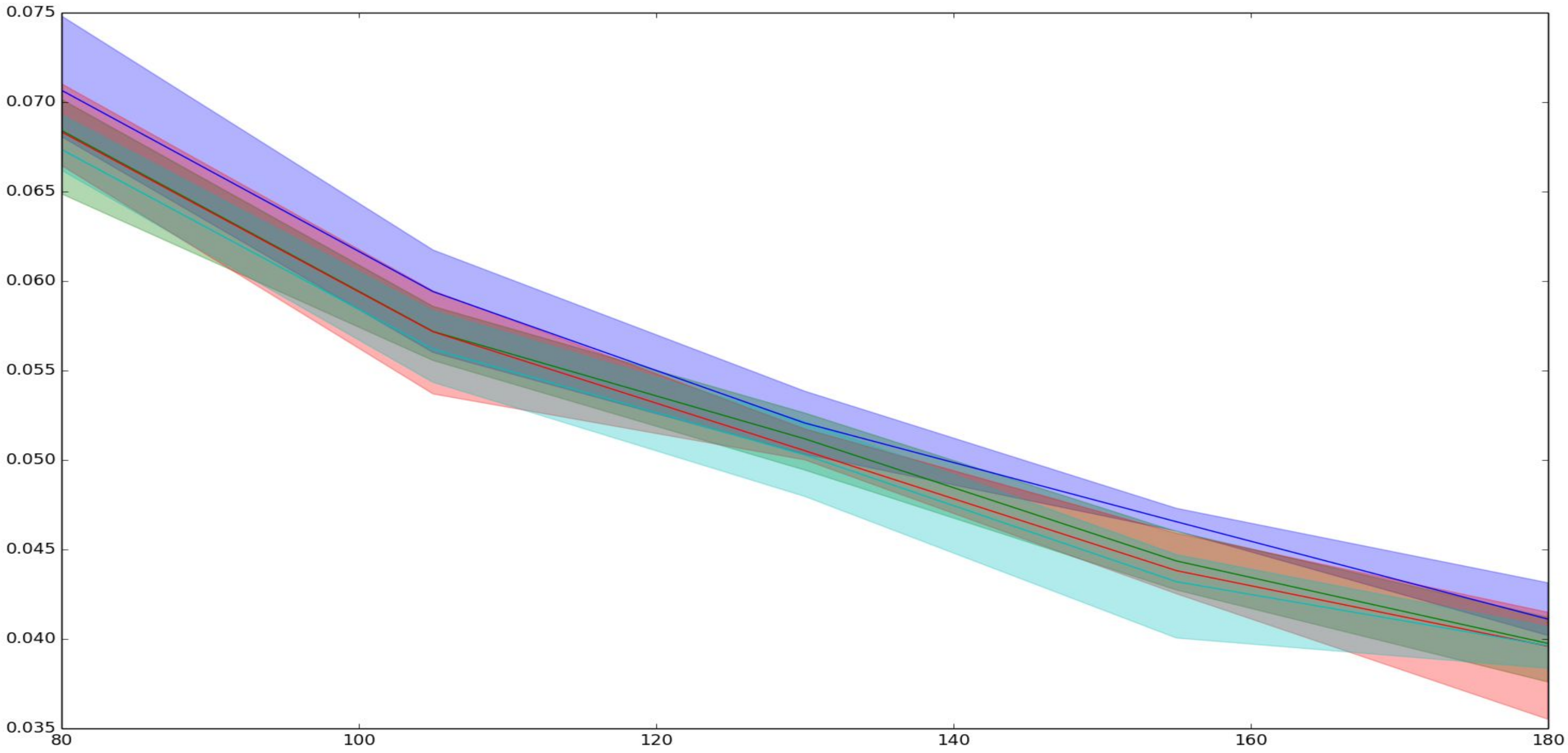
Knuth-Fisher-Yates Shuffle

```
-- To shuffle an array a of n elements
A[0..n-1];
for i from n-1 downto 1 do
     j ← random integer in [0,...,i];
     swap( A[j] , A[i] );
done
```

Random Search | Stability (D=2)



Random Search | Stability (D=3)



Random Search Results

Large errors bands

Number of iterations > 1000 doesn’t give much better results

Simulated Annealing

What's that?

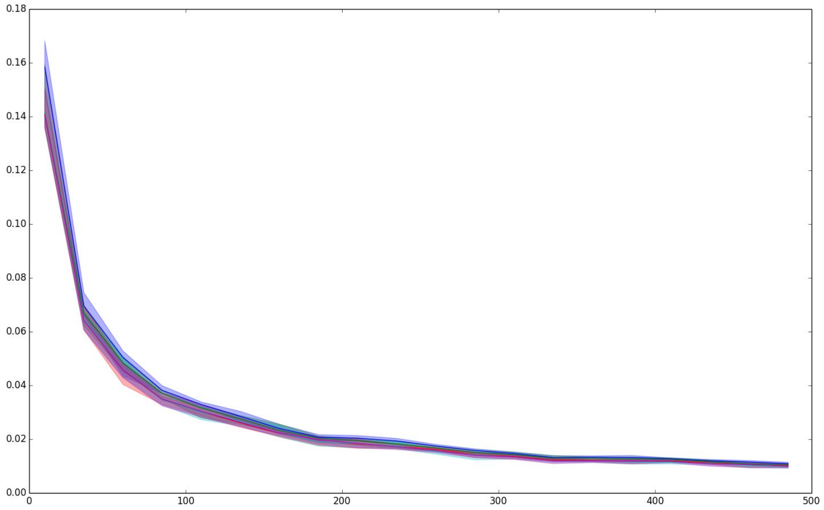
- Interprets slow cooling as a slow decrease in the probability of accepting worse solutions
- Accepting worse solutions is a fundamental property to avoid local minima.

Pseudo-Code

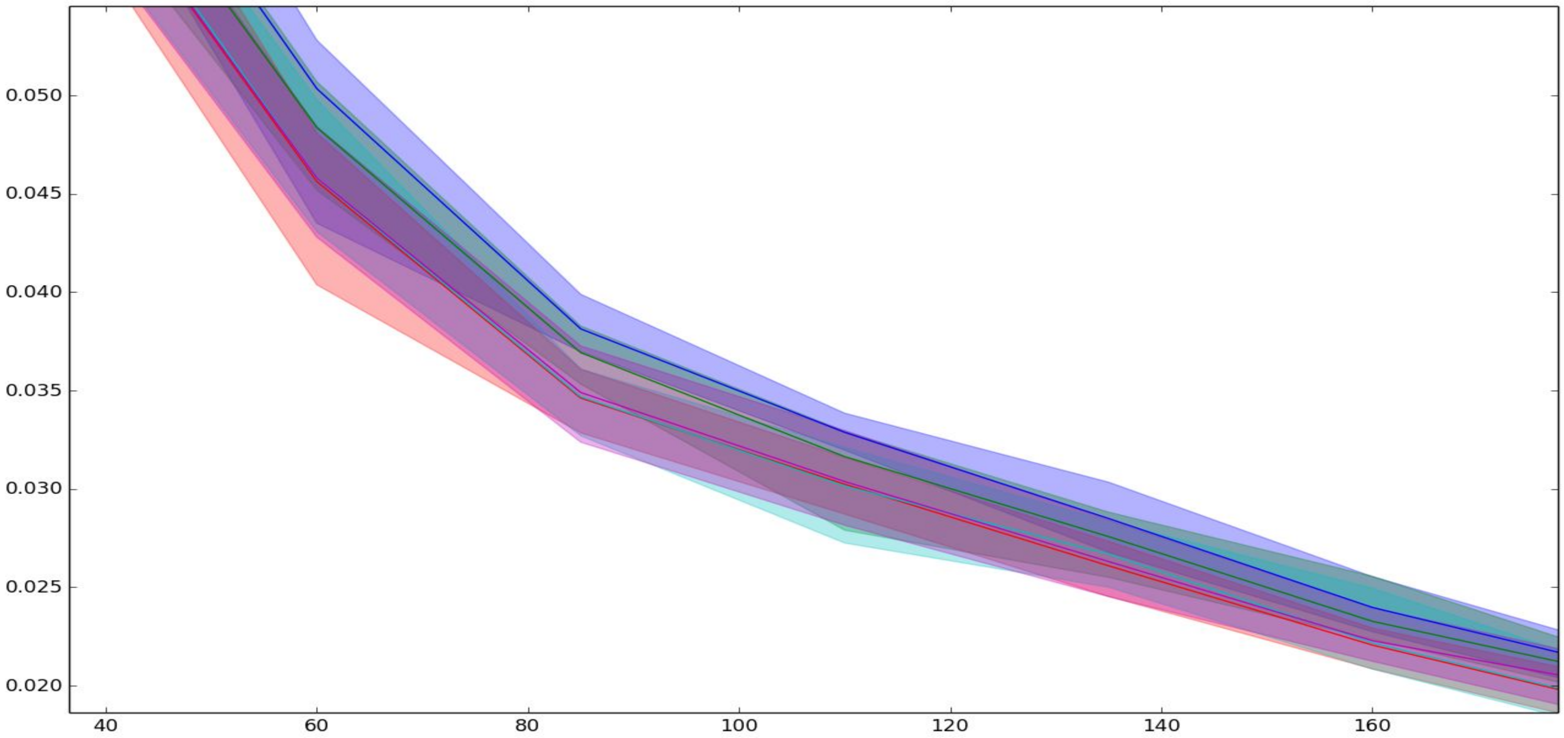
```
s ← s_0;
For k = 0 to k_max do
    s_new ← neighbour(s);
    If e^((E(s)-E(snew))/T) ≥ random(0, 1) then
        s ← s_new;
        T ← T*λ; // λ = 0.992
done
return s;
```

Sim. Annealing Results

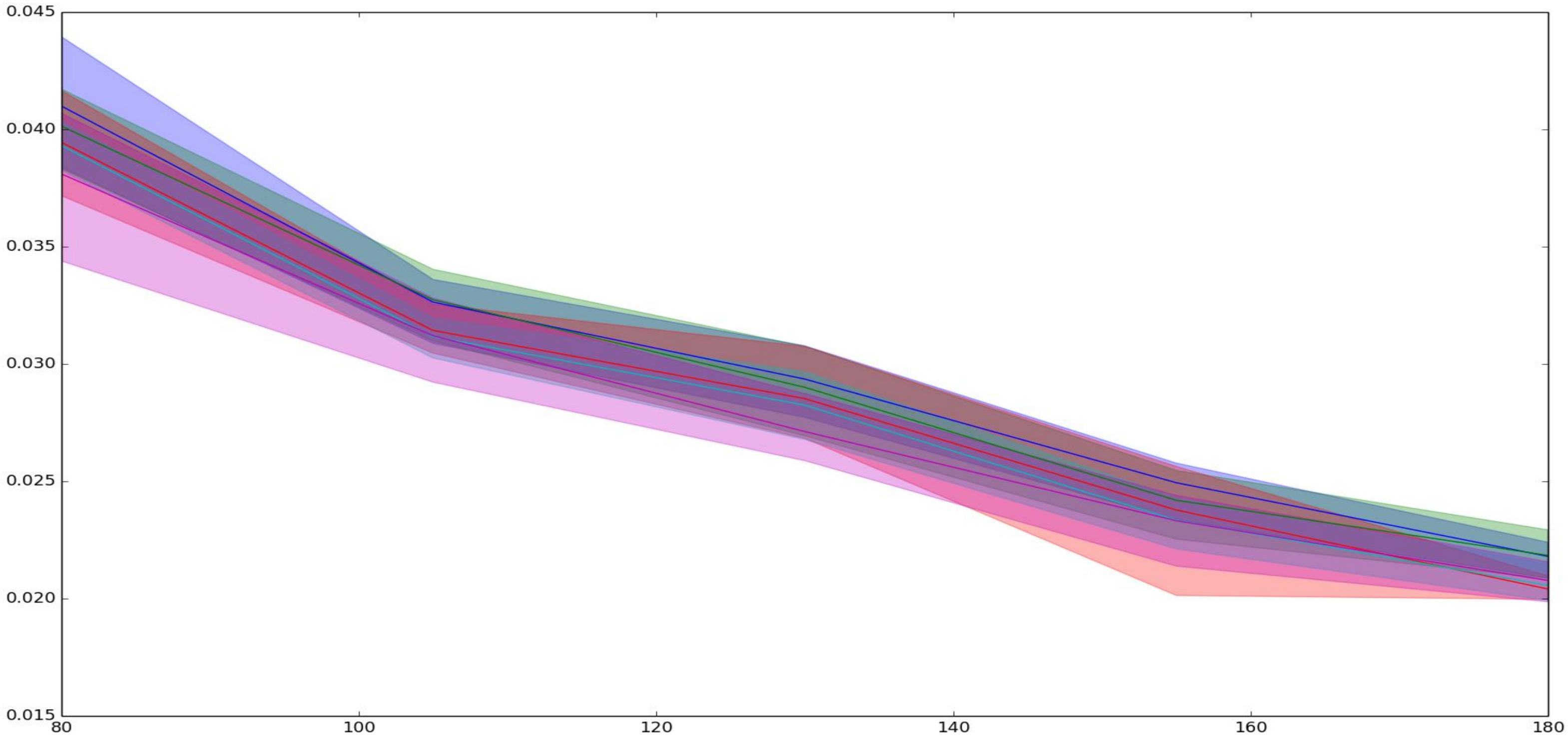
Dependance on initial temperature



Sim Annealing | Temperature (D=3)



Sim Annealing | Stability (D=2)



Sim. Annealing Results

Lowest temperature gives best results (saturates if < 0.001)

Number of iterations > 1000 doesn’t give much better results

Smaller error bands.

(μ+λ) Genetic Search

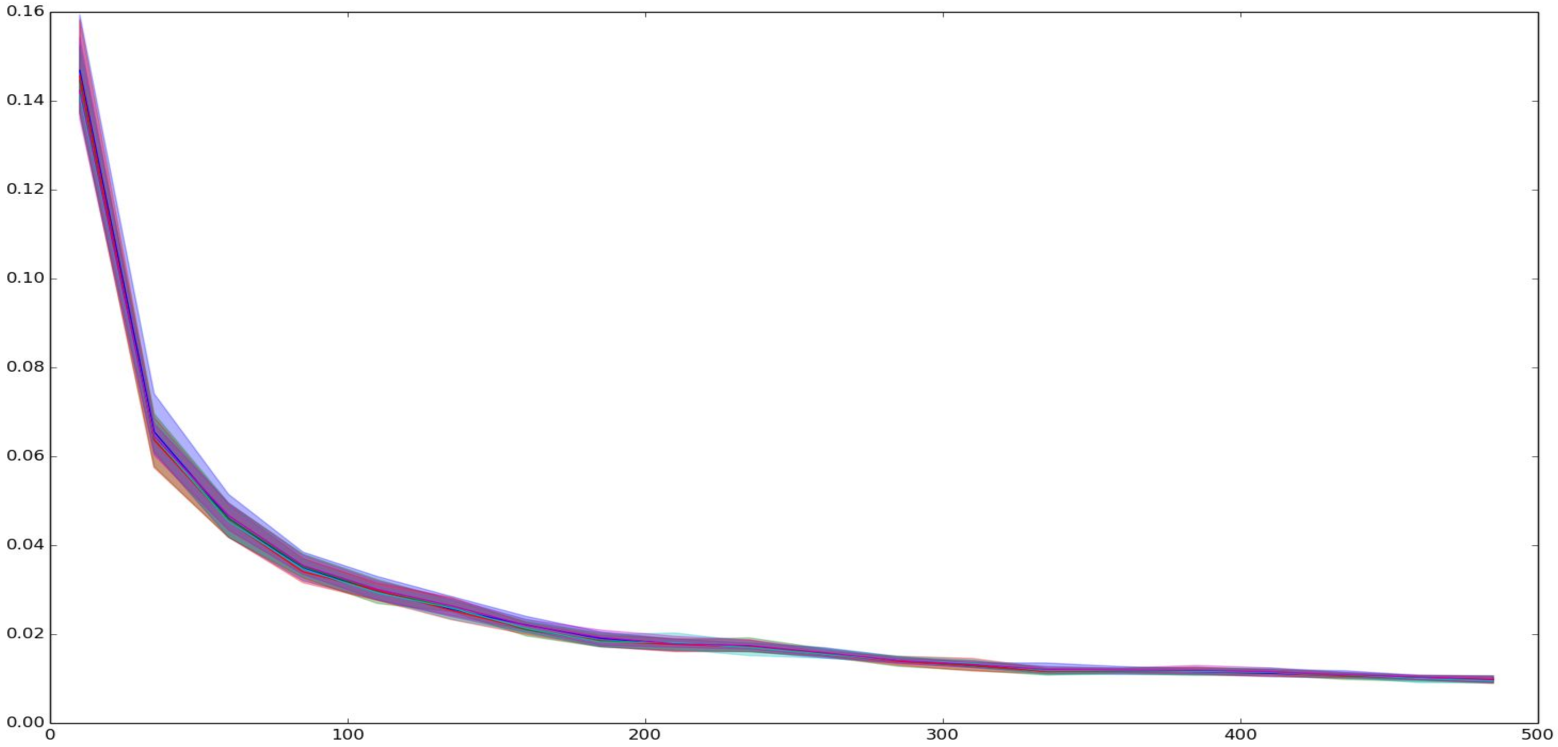
Overview

- (5,5) search implemented.
- Keep common values.
- Close to A or B.
- Don’t modify more the end of the permutation than its start.

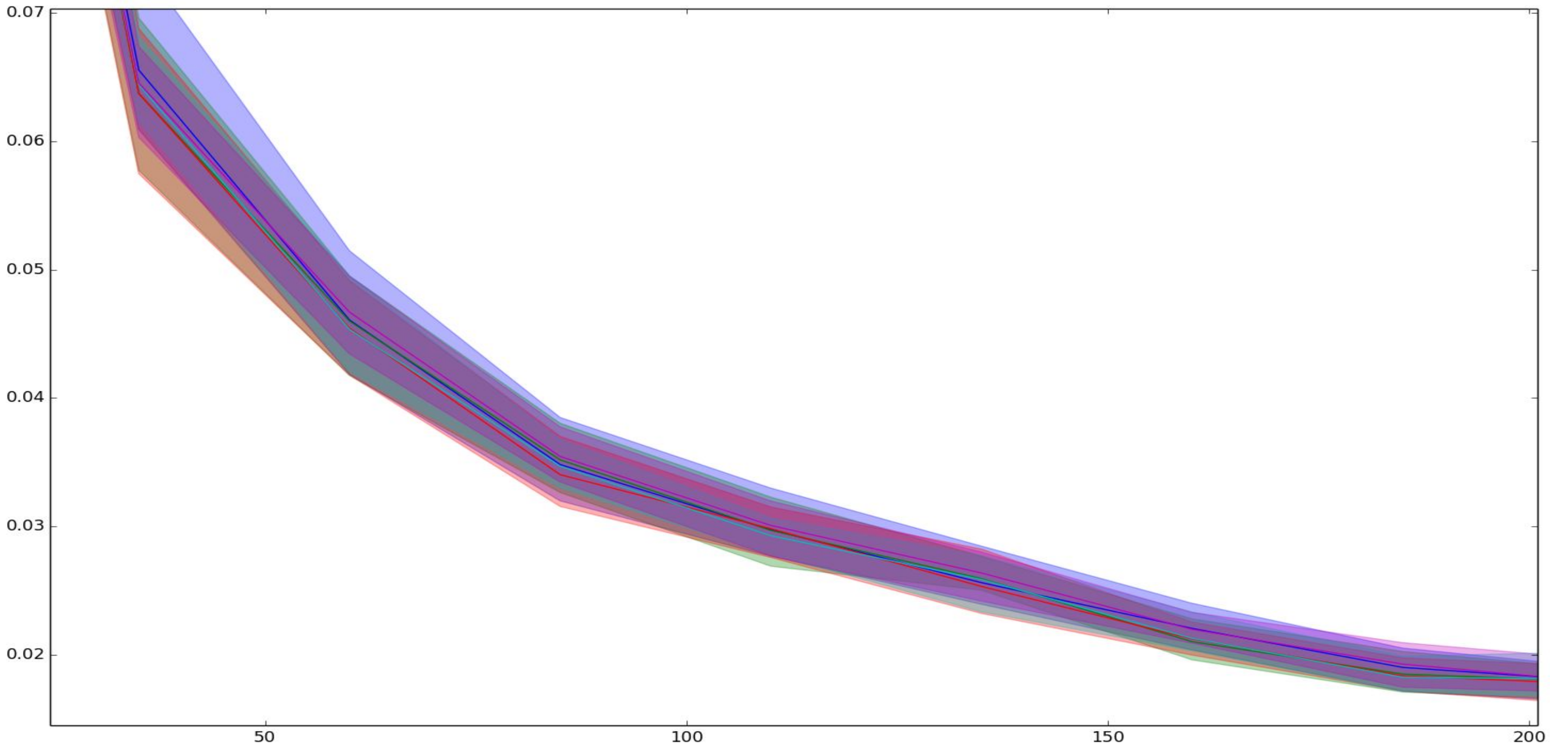
Crossover algorithm

```
A[0..n-1]; B[0..n-1];
pi ← KFY shuffle [1..n];
availables ← ∅;
for i from 1 downto n do
    if A[pi[i]] and B[pi[i]] already chosen then
        C[pi[i]] ← Random in availables;
    elif A[pi[i]] already chosen then
        C[i] ← B[pi[i]];
    elif B[pi[i]] already chosen then
        C[i] ← A[pi[i]];
    else
        swap A and B w.p. 1/2
        C[i] ← A[pi[i]];
        available ← available U {B[pi[i]]};
    done
    availables ← availables \ {C[pi[i]]}
done
```

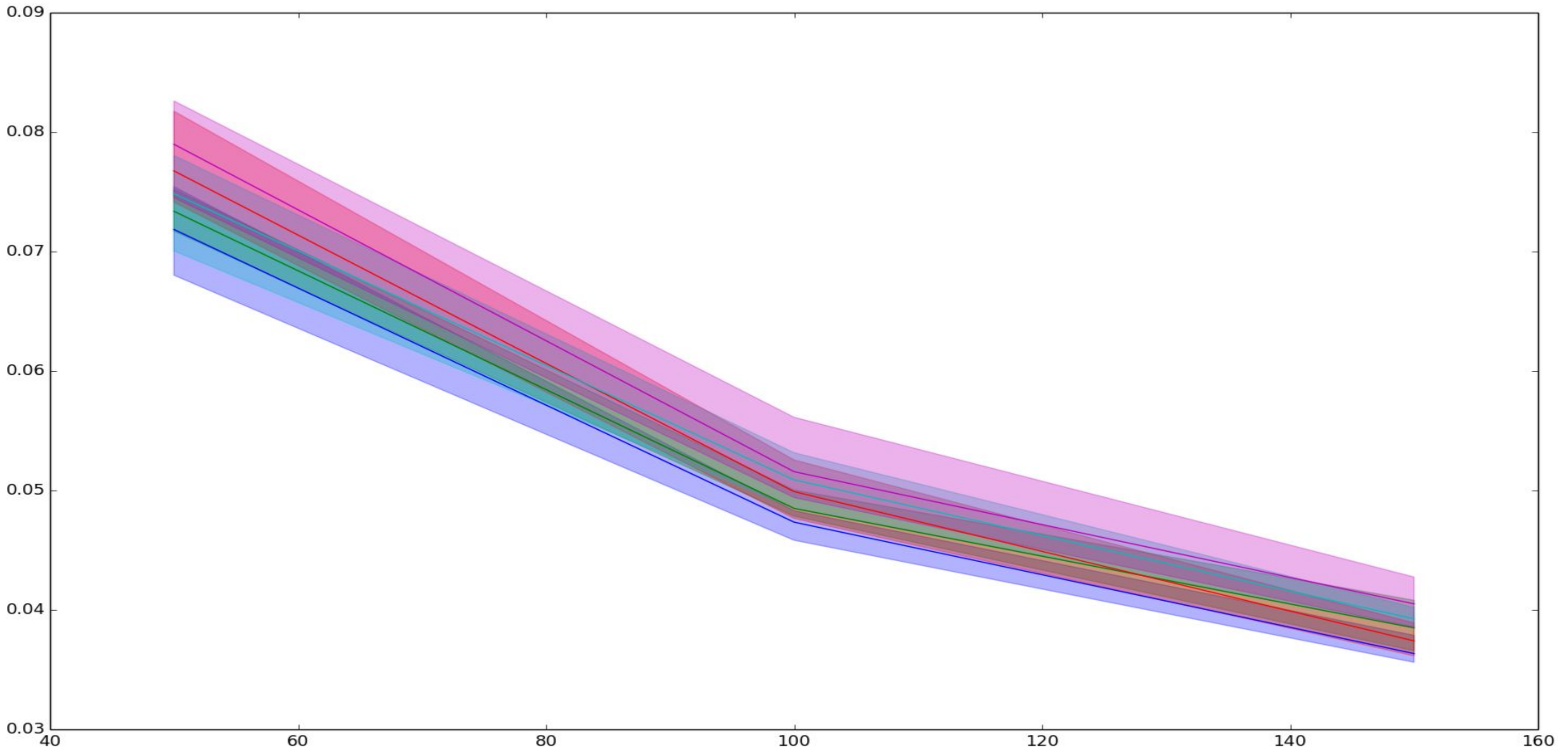
Genetic Search | p dependence (D=2)



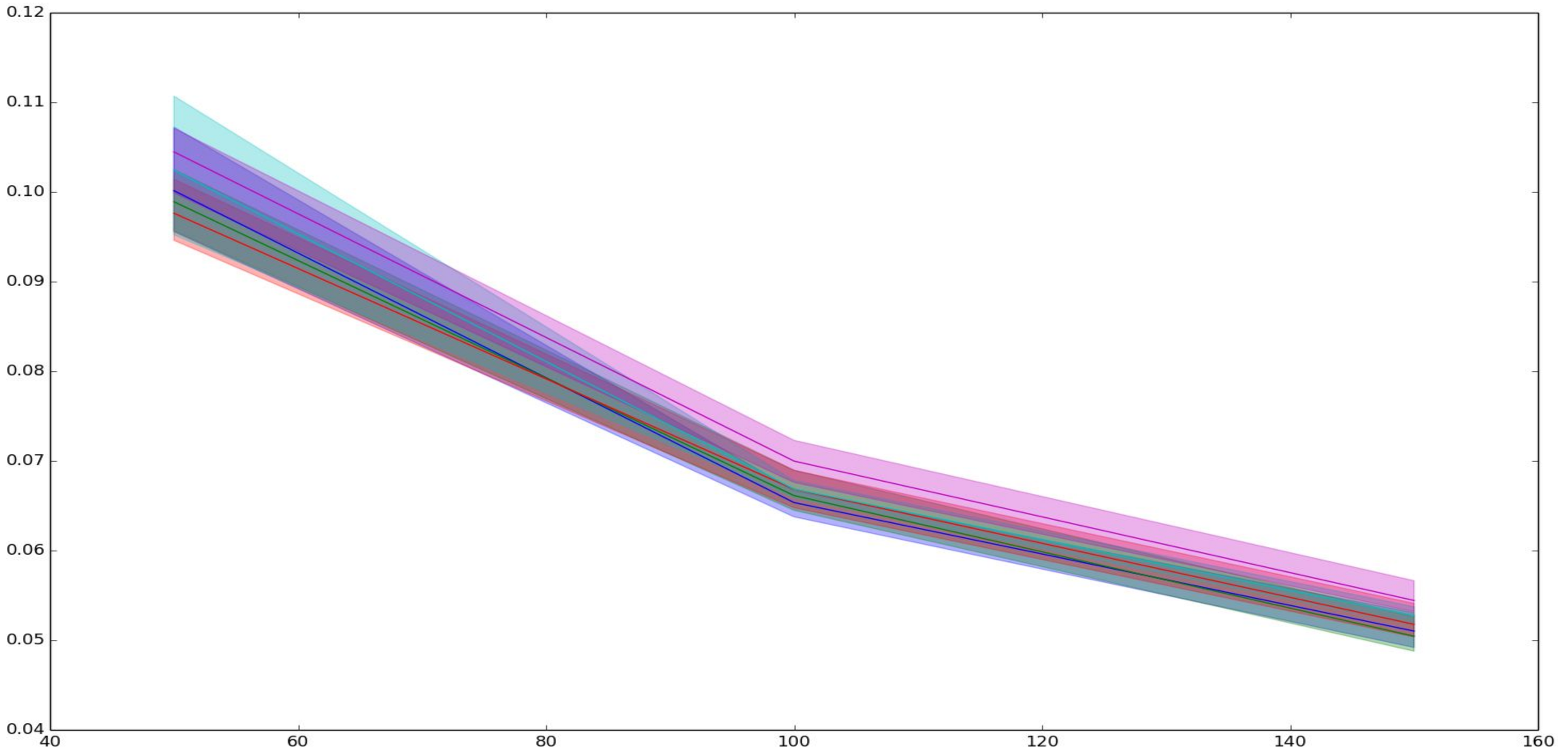
Genetic Search | p dependence (D=2)



Genetic Search | p dependence (D=3)



Genetic Search | p depedence (D=4)



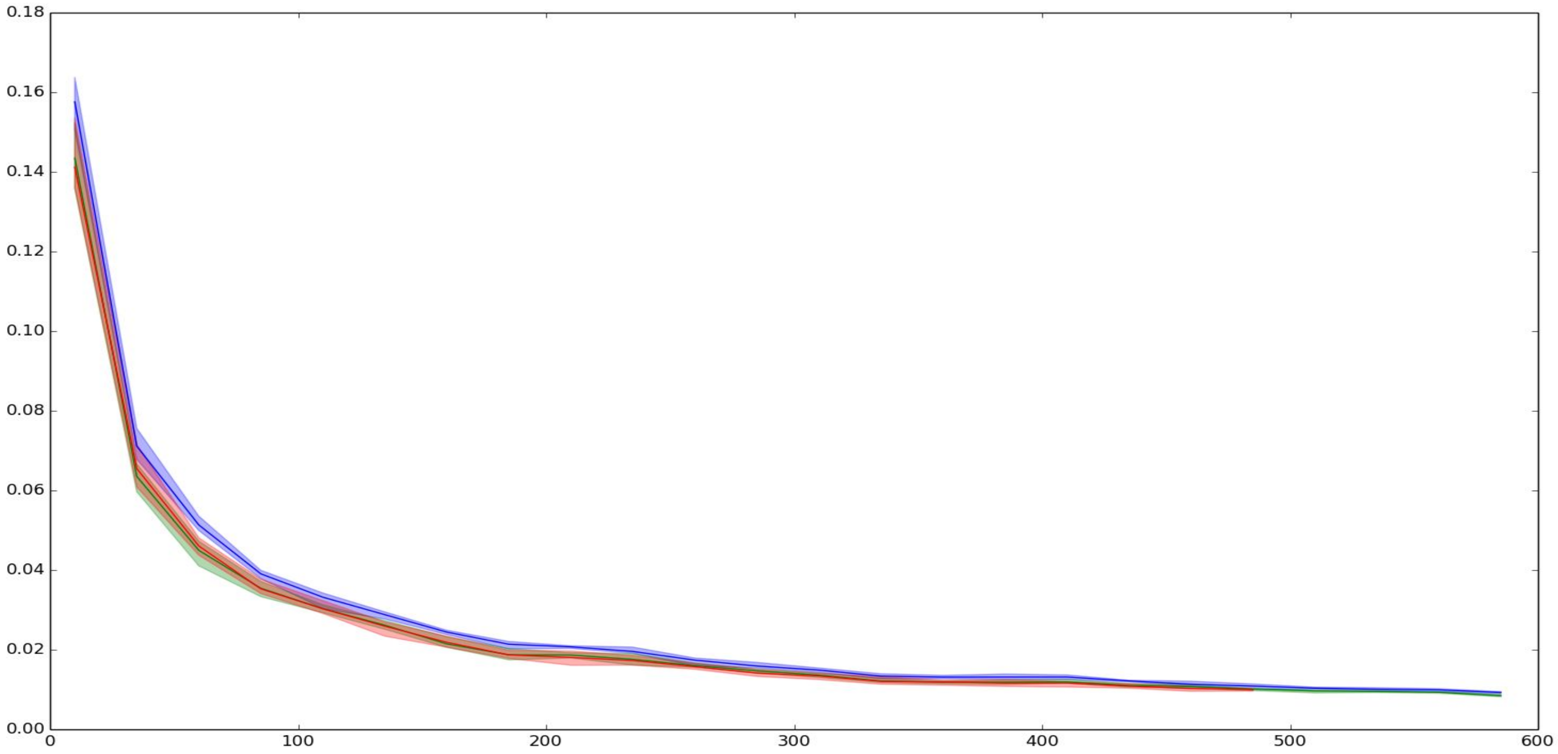
Genetic Search Results

Best results on average for
c=0.5/0.6

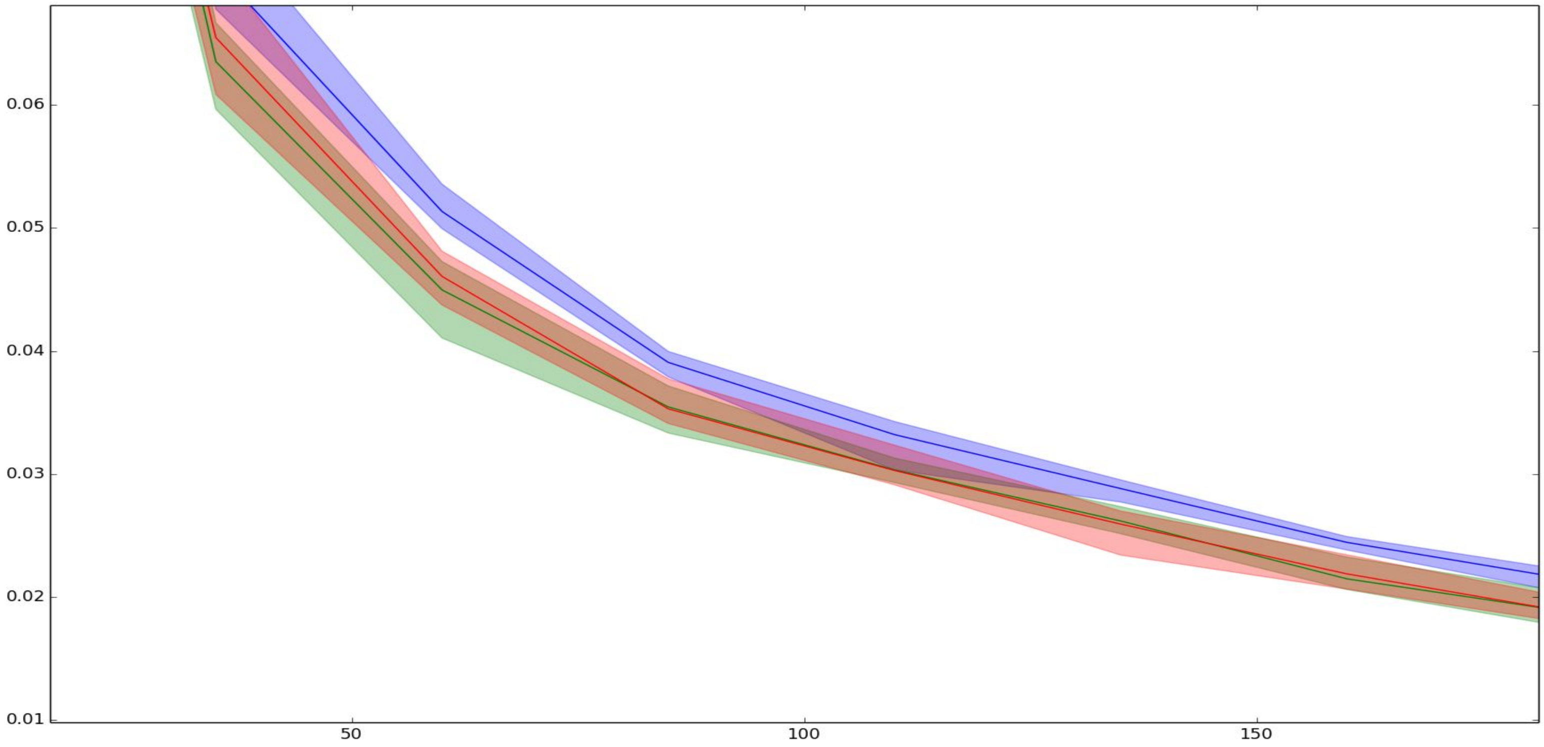
Number of iterations > 1000 doesn't
give much better results

Big errors bands on
low number of points.

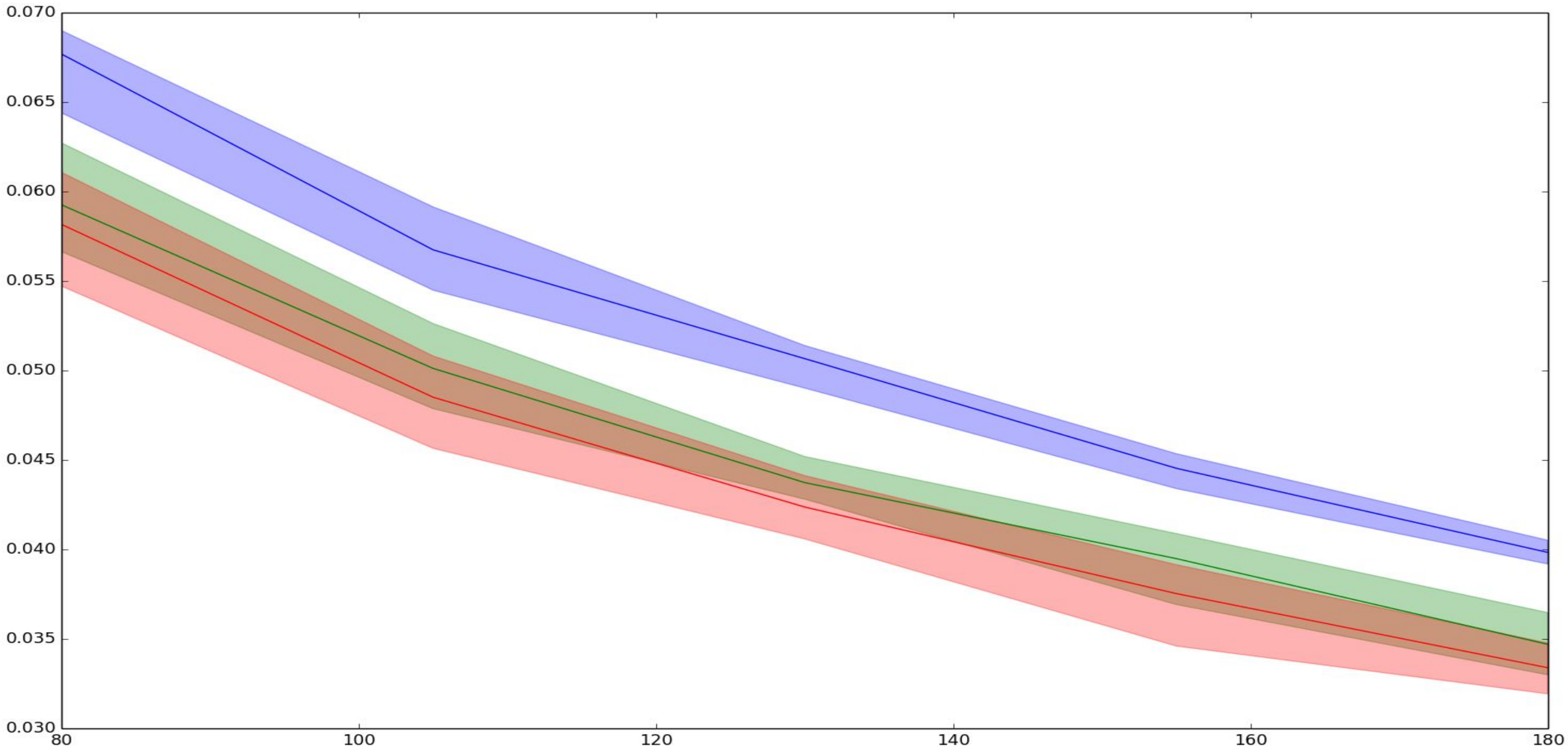
Wrap up all heuristics (D=2)



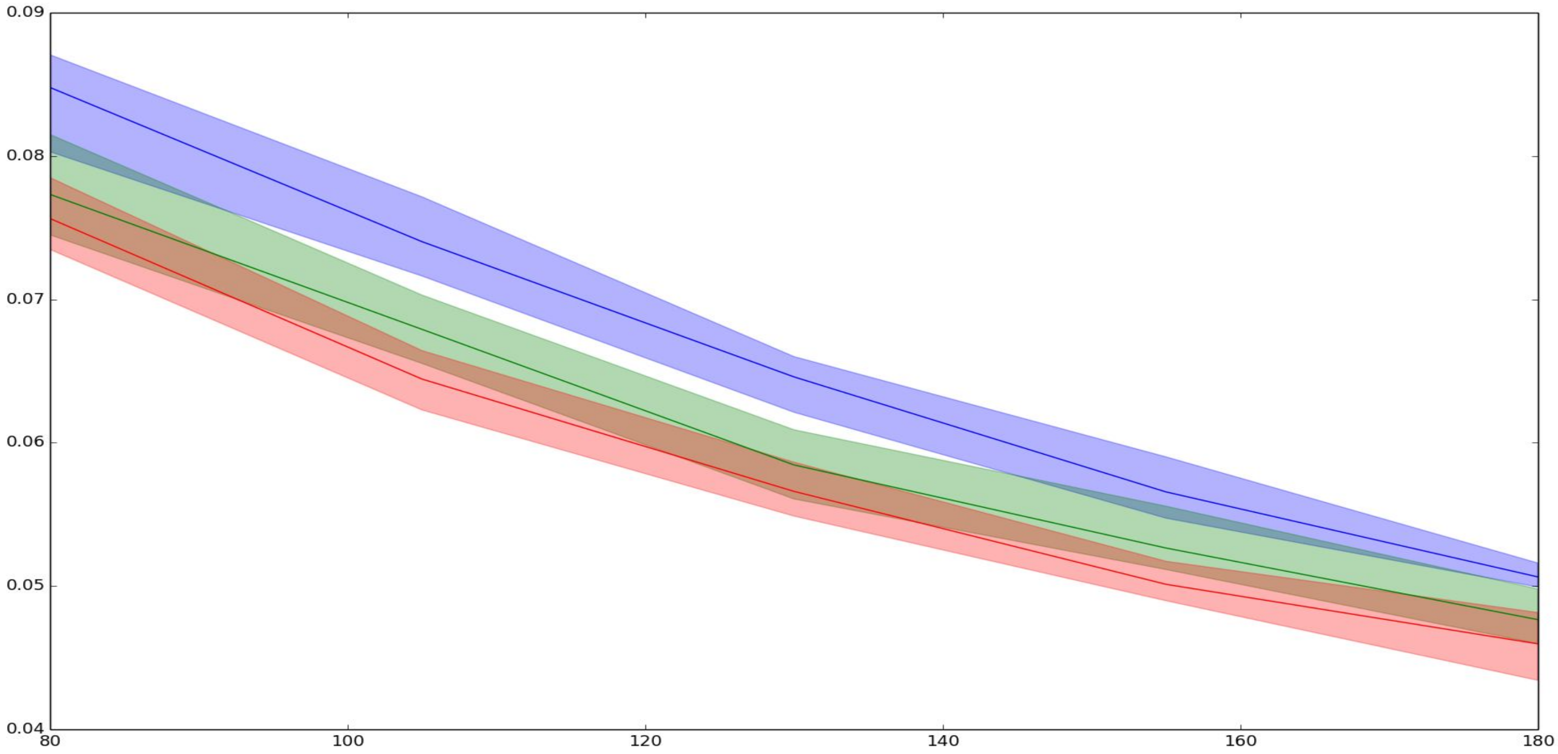
Wrap up all heuristics (D=2)



Wrap up all heuristics (D=3)



Wrap up all heuristics (D=4)



Wrap up

“Intelligent” heuristics
are better than
fully random search

Genetic or S.A. is better
depending on the dimension.