Paper presentation 1
TaejoonYang 20190379

Relative Building-Block fitness and the Building-Block Hypothesis

Introduction

Relative Building-Block fitness and the

Building-Block Hypothesis

Written by Stephanie Forrest, Melanie Mitchell

Written on 1992

Introduction-from class

Schema Theorem

Short,

Low-order,

Highly-fit schemas(building blocks)

receive exponentially increasing trials in subsequent generations

Introduction

On what condition does GA perform well?

Identify features of fitness landscapes that are particularly relevant to the GA's performance-

building blocks

Design simplified landscapes containing different configurations of such features-distribution, frequency, size

Study in detail the effects of these features on the GA's behavior-the way in which schemas are processed and building blocks are combined

Stepping-stones in the crossover landscape

Two landscape features of building-block hypothesis

presence of short, low-order, highly fit schemas

the presence of intermediate “stepping stone”-intermediate-order higher-fitness schemas that result from combinations of the lower-order schemas

=>how much higher in fitness do the intermediate stepping stones have to be for the GA to work well?

Stepping-stones in the crossover landscape

Royal Road functions

Select an optimum string and break it up into a number of small building blocks

Assign values to each low-order schema and each possible intermediate combination of low-order schemas-

use the values to compute the fitness of a bit string

Stepping-stones in the crossover landscape

Royal Road functions-R1

s_1 = 11111111**; $c_1 = 8$
s_2 = ********11111111**; $c_2 = 8$
s_3 = ****************11111111**; $c_3 = 8$
s_4 = ************************11111111********************************; $c_4 = 8$
s_5 = ********************************11111111************************; $c_5 = 8$
s_6 = **11111111****************; $c_6 = 8$
s_7 = **11111111********; $c_7 = 8$
s_8 = **11111111; $c_8 = 8$
s_{opt}=1111111111111111111111111111111111111111111111111111111111111111

R1(x) is computed by summing the coefficients c(order)

of corresponding schema of which x is an instance

R1(1111111100….0)=8

R1(1111111100……011111111)=16

Can know how lower-level blocks are combined into higher-level blocks

Stepping-stones in the crossover landscape

Royal Road functions-R2- consider more on crossover

s_1 = 11111111**; $c_1 = 8$
s_2 = ********11111111**; $c_2 = 8$
s_3 = ****************11111111**; $c_3 = 8$
s_4 = ************************11111111********************************; $c_4 = 8$
s_5 = ********************************11111111************************; $c_5 = 8$
s_6 = **11111111****************; $c_6 = 8$
s_7 = **11111111********; $c_7 = 8$
s_8 = **11111111; $c_8 = 8$
s_9 = 1111111111111111**; $c_9 = 16$
s_{10} = ****************1111111111111111********************************; $c_{10} = 16$
s_{11} = ********************************1111111111111111****************; $c_{11} = 16$
s_{12} = **1111111111111111; $c_{12} = 16$
s_{13} = 11111111111111111111111111111111********************************; $c_{13} = 32$
s_{14} = ********************************11111111111111111111111111111111; $c_{14} = 32$
s_{opt} = 1111111111111111111111111111111111111111111111111111111111111111

R2(1111111100….0)=8

R2(1111111100……011111111)=16

R2(11111111111111110………0)=32

R2(opt)=8*8+16*4+32*2=192

Royal Road experiments

Initial Variables

Length of string: 64

GA population size:128

Initial population generated randomly

GA continue until find optimum discovered-check total number of function evaluations performed

Single crossover: rate 0.7

Mutation : rate 0.005

Royal Road experiments

Initial Variables

Reproduction: expected number of offspring

$$1 + \frac{F_i - \bar{F}}{2\sigma}$$

Maximum expected offspring of any string was 1.5-if the formula gives higher value, it was reset to 1.5

-most individuals will reproduce only 0, l, 2 times.

-to slow down convergence

Royal Road experiments

Initial Variables

Randomly generated sting on bottom-level

Probability of having order-8 Schemas

$=8 * \frac{1}{2^8} = \frac{1}{32}$

Initial number of order-8 schemas :128/32=4

Royal Road experiments

Experiments on R1 and R2

R2 has more clear path by crossover(high fitness value for highest order schemas)

Has stronger path to optimum

Hypothesis: R2 will perform well than R1 finding optimum

Royal Road experiments

Experiments on R1 and R2

ORIGINAL EXPERIMENT		
	Function Evaluations to Optimum	
500 runs	$R1$	$R2$
Mean	62099 (std err: 1390)	73563 (std err: 1794)
Median	56576	66304

Table 1: Summary of results of running the GA on $R1$ and $R2$. The table gives the mean and median function evaluations taken to find the optimum over 500 runs on each function. The numbers in parentheses are the standard errors.

R1 is better than R2!

Royal Road experiments

ORIGINAL EXPERIMENT		
	Function Evaluations to Optimum	
500 runs	$R1$	$R2$
Mean	62099 (std err: 1390)	73563 (std err: 1794)
Median	56576	66304

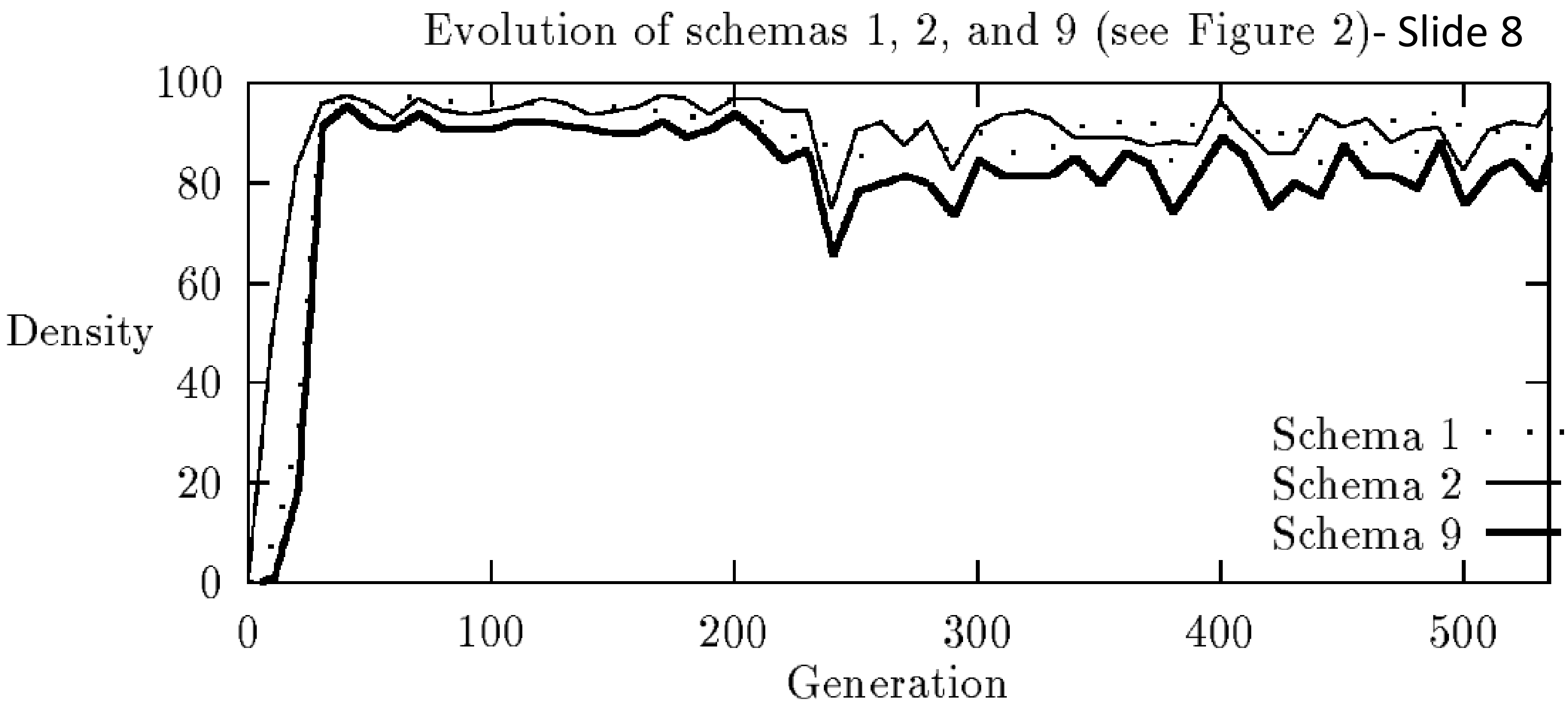
Table 1: Summary of results of running the GA on $R1$ and $R2$. The table gives the mean and median function evaluations taken to find the optimum over 500 runs on each function. The numbers in parentheses are the standard errors.

Potential bottleneck?

Deception?=> R2 is non deceptive function

=>trace schemas through run

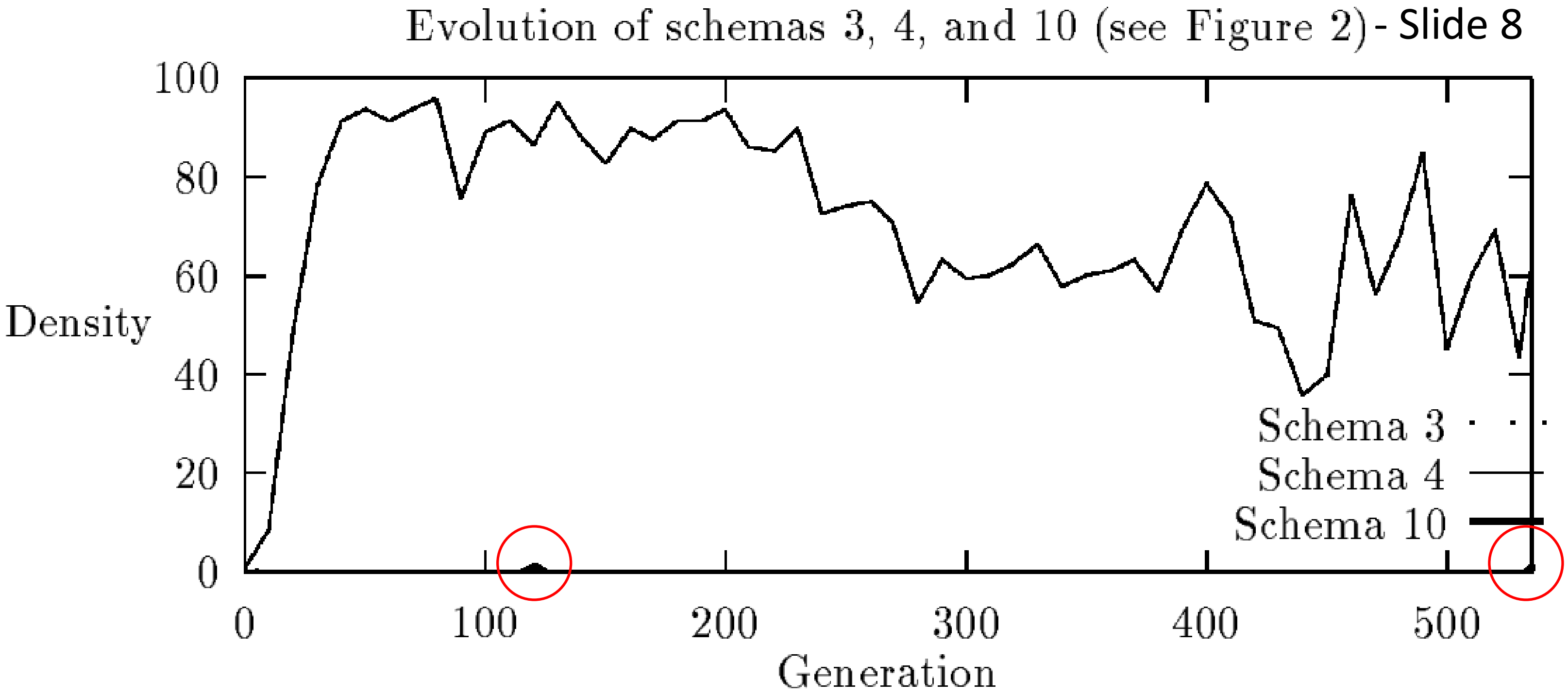
Royal Road experiments



s1 and s2 combine quickly and has high density of the population

Remember small dip on 220

Royal Road experiments

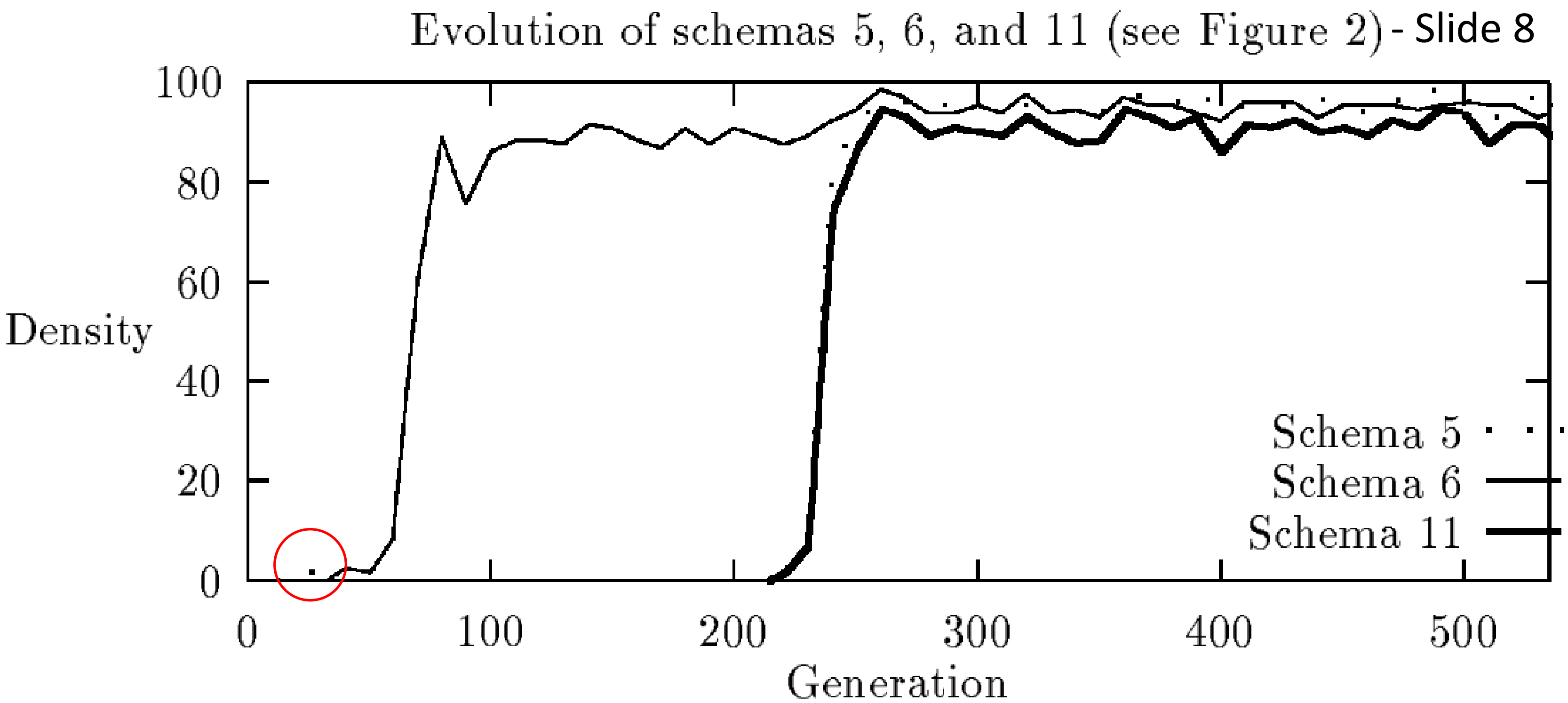


s4 rises fast, but s3 and s10 does not have population

on generation 120 and 535 it gets generation but it dies out

s4 has a dip on time 220

Royal Road experiments



s6 appears around generation 30 and rise quickly.

s5 appears around generation 20 and disappears, and appears again on generation 220, and grows quickly with s11

This rise coincide with the minor dip on s1,s2,s9, major dip on s4

Royal Road experiments

s9 has high fitness value of 32, it causes very quickly rise compared to s4 which has fitness value of 8

It tends out to push out existing instances of s4 in the population

"Hitchhiking"

0's in other positions in the string hitchhike along with the highly fit s11

Most likely positions for hitchhikers are those close to the highly fit schema's defined positions-cause by crossover

Power of crossover to combine lower-level building blocks was hampered, because it is get suppressed partially or totally by the quick rise of disjoint building blocks

Royal Road experiments

For R1, which laks the extra fitness given to some intermediate-level schemas, hitchhiking problem does not occur to such a devasting degree-s11 fitness value is only 16

Contrary to hypothesis, extra reinforcement from some intermediate-level stepping-stones actually harms the GA

Royal Road experiments

Change on variables

Hypothesis: The result is from lack of population-sampling error

Experiment1: GA with population size 1024

Experiment2: GA with lowest-order schemas length 4 instead of 8

Royal Road experiments

POPULATION SIZE 1024		
	Function Evaluations to Optimum	
200 runs	$R1$	$R2$
Mean	37453 (std err: 868)	43213 (std err: 1275)
Median	34816	36864

Table 2: Summary of results of 200 runs of the GA with population size 1024 on $R1$ and $R2$.

LOWEST-ORDER SCHEMAS LENGTH 4		
	Function Evaluations to Optimum	
200 runs	$R1$	$R2$
Mean	6568 (std err: 198)	11202 (std err: 394)
Median	5760	9600

Table 3: Summary of results of 200 runs of the GA on modified versions of $R1$ and $R2$, in which the lowest-order building blocks are of length 4.

Do not change the qualitative difference between R1 and R2

Royal Road experiments

Conclusion

We observe premature convergence even in very simple setting

The population loses useful schemas once one of the disjoint good schemas is found suggests one reason that the rate of effective implicit parallelism of GA may need to be reconsidered

Royal Road experiments

Using introns

Hitchhiking occurred in the loci that were spatially adjacent to the high-fitness schemas

Construct new function R2introns by introducing blocks of 8 “introns” between each of the 8-bit blocks of 1’s

s1=11111111********……

s2=****************11111111********…..

s9=11111111********111111111*********….*

Blocks are each separated so has least damage by hitchhiking

Royal Road experiments

Weaker nonlinear reinforcement

R1 has linear reinforcement, the fitness of an instance of an intermediate-order schema is always the sum of the fitness of instances of the component blocks

R2 has nonlinear reinforcement, the fitness goes much higher

=> weaker nonlinear reinforcement function : R2flat

c1-c14 is set to 1, so s9 has fitness value 3, s1,s2=1 it is still nonlinear, but more flatten then R2

Royal Road experiments

ORIGINAL EXPERIMENT		
	Function Evaluations to Optimum	
500 runs	$R1$	$R2$
Mean	62099 (std err: 1390)	73563 (std err: 1794)
Median	56576	66304

Table 1: Summary of results of running the GA on $R1$ and $R2$.

VARIANTS OF R2		
	Function Evaluations to Optimum	
200 runs	$R2_{introns}$	$R2_{flat}$
Mean	75599 (std err: 2697)	62692 (std err: 2391)
Median	70400	56448

Table 4: Summary of results of 200 runs of the GA on two variants of $R2$.

R2introns vs R2 / R2flat vs R1

Royal Road experiments

R2introns vs R2

-similar with R2 (no advance)

-convergence is so fast that hitchhikers are possible even in loci that are relatively distant from the schema’s defined position

R2flat vs R1

-average time approximately same with R1

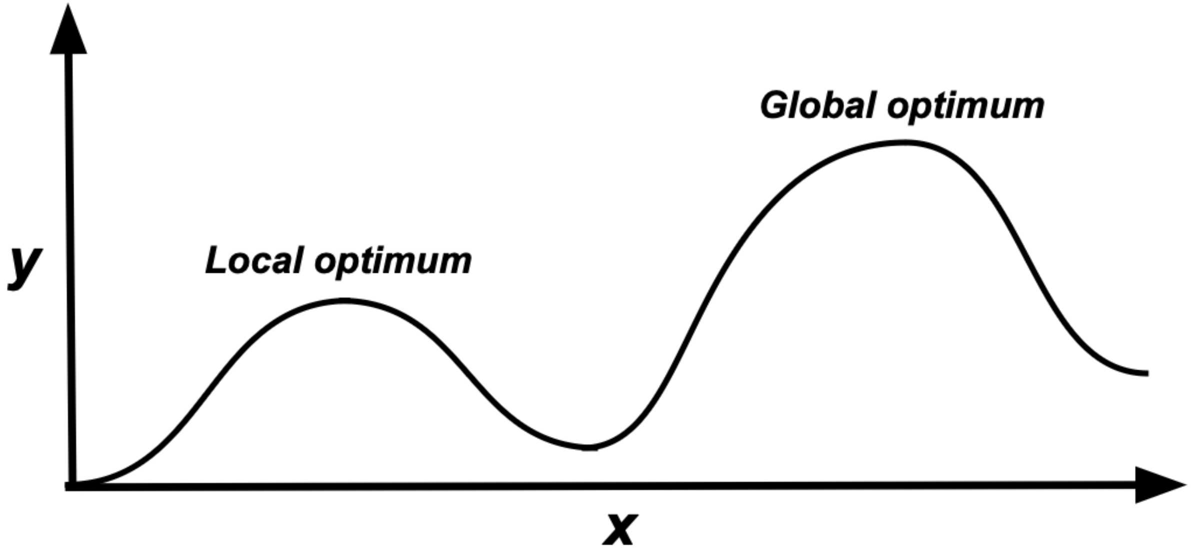
-no advance, but no hurt on performance

Experiments with hill-climbing

Steepest-ascent hill-climbing

Next-ascent hill-climbing

Random-mutation hill-climbing



Experiments with hill-climbing

- **Steepest-ascent hill-climbing (SAHC):**
  1. Choose a string at random. Call this string *current-hilltop*.
  2. Systematically mutate each bit in the string from left to right, recording the fitnesses of the resulting strings.
  3. If any of the resulting strings give a fitness increase, then set *current-hilltop* to the resulting string giving the highest fitness increase.
  4. If there is no fitness increase, then save *current-hilltop* and go to step 1. Otherwise, go to step 2 with the new *current-hilltop*.
  5. When a set number of function evaluations has been performed, return the highest hilltop that was found.

Experiments with hill-climbing

- **Next-ascent hill-climbing (NAHC):**
  1. Choose a string at random. Call this string *current-hilltop*.
  2. Mutate single bits in the string from left to right, recording the fitnesses of the resulting strings. If any increase in fitness is found, then set *current-hilltop* to that increased-fitness string, without evaluating any more single-bit mutations of the original string. Go to step 2 with the new *current-hilltop*, but continue mutating the new string starting after the bit position at which the previous fitness increase was found.
  3. If no increases in fitness were found, save *current-hilltop* and go to step 1.
  4. When a set number of function evaluations has been performed, return the highest hilltop that was found.

Experiments with hill-climbing

- **Random-mutation hill-climbing (RMHC):**
  1. Choose a string at random. Call this string *best-evaluated*.
  2. Choose a locus at random to mutate. If the mutation leads to an equal or higher fitness, then set *best-evaluated* to the resulting string.
  3. Go to step 2.
  4. When a set number of function evaluations has been performed, return the current value of *best-evaluated*.

Experiments with hill-climbing

HILL-CLIMBING ON R2			
	Function Evaluations to Optimum		
200 runs	SAHC	NAHC	RMHC
Mean	$> 256{,}000$ (std err: 0)	$> 256{,}000$ (std err: 0)	6551 (std err: 212)
Median	$> 256{,}000$	$> 256{,}000$	5925

Table 5: Summary of results of 200 runs of various hill-climbing algorithms on $R2$.

GA performs better than SAHC, NAHC-they didn't get even optimum

RMHC has average 10 times faster than GA on population size 128, 6 times faster than size 1024

-it is ideal for the Royal Road functions, but will have trouble on function with local minima

Experiments with hill-climbing

Online performance

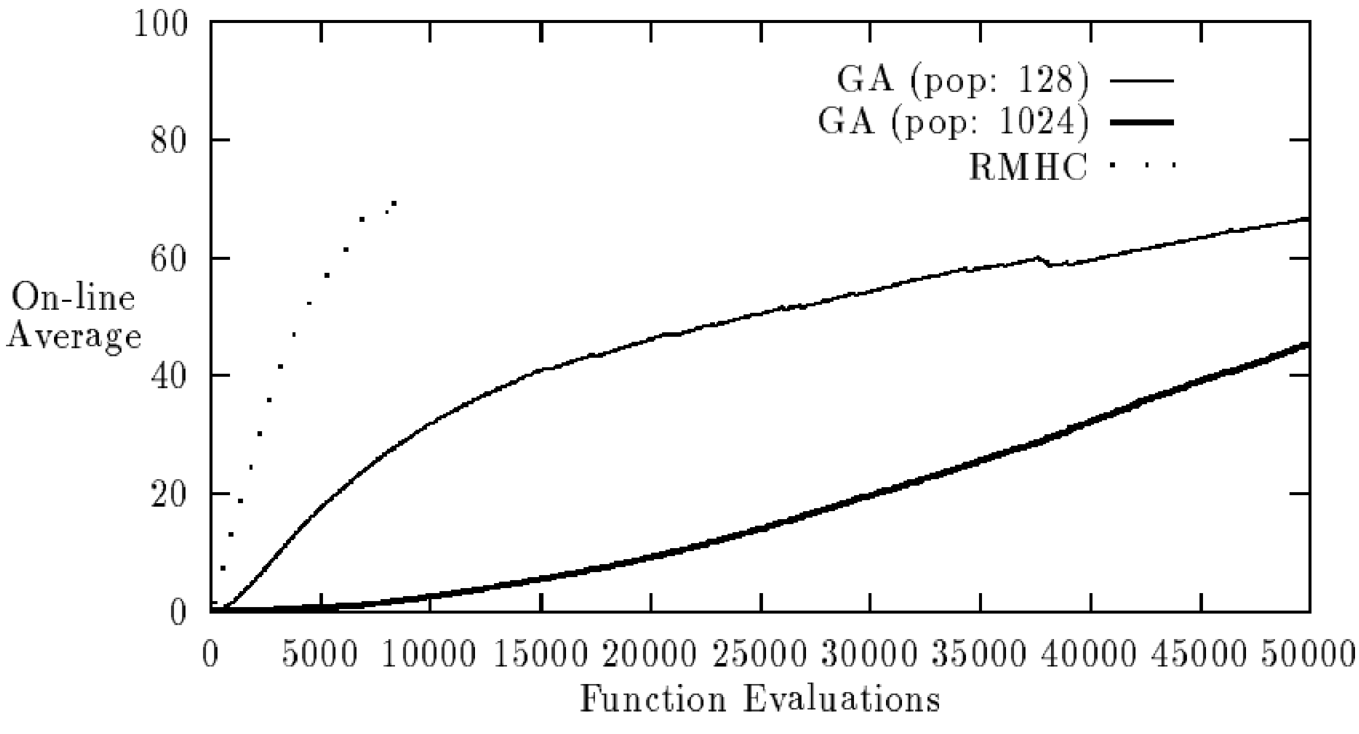


Figure 4: Plots of the average on-line performance of the GA (population sizes 128 and 1024) and of random-mutation hill-climbing (RMHC), over 100 runs. The plot for RMHC stops at around 6000 function evaluations because RMHC had almost always found the function optimum by that time.

Conclusions

Can understand more precisely how schemas are processed under crossover.

Hitchhiking is evidently one bottleneck for GA

Can improve this by adding noise, including all combinations of lower-order schemes in the explicit list of schemas, allowing schemas to overlap

+GA compared with Hill climbing on TSP

Comparison of Genetic Algorithm and Hill Climbing for Shortest Path Optimization Mapping- [Mona Fronita, Rahmat Gernowo , and Vincencius Gunawan]

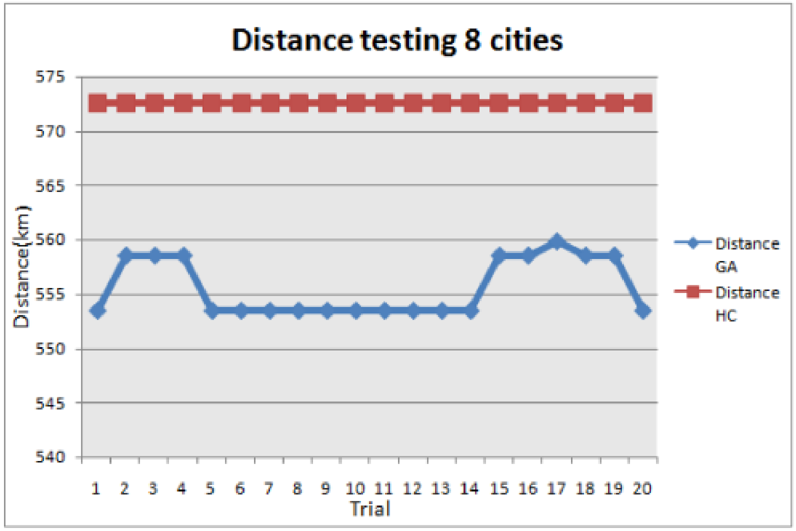


Fig. 4 Distance testing 8 cities

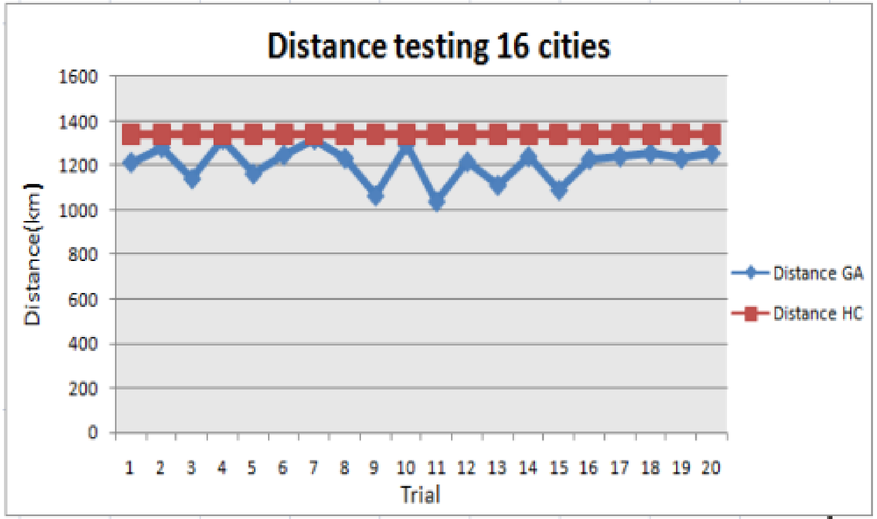


Fig. 5 Distance testing 16 cities

+GA compared with Hill climbing on TSP

Comparison of Genetic Algorithm and Hill Climbing for Shortest Path Optimization Mapping- [Mona Fronita, Rahmat Gernowo , and Vincencius Gunawan]

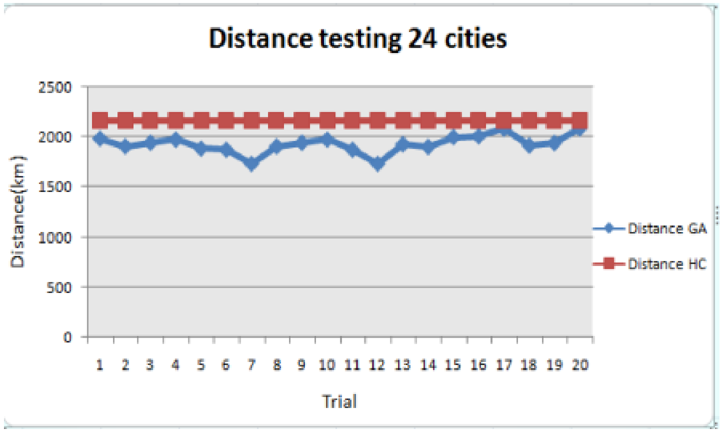


Fig. 6 Distance testing 24 cities

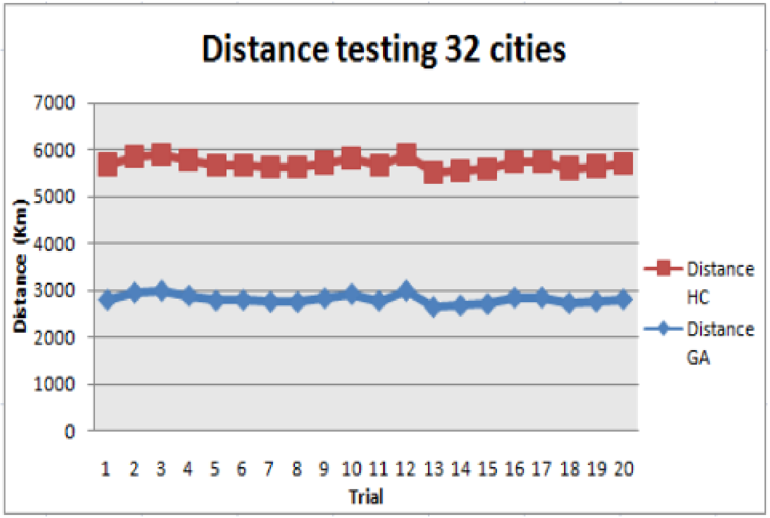


Fig. 7 Distance testing 32 cities

Thank you for listening!