Evolutionary Algorithms II

Ender Özcan

Lecture 6

Computational Optimisation & Learning Lab

The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Questions

- When does a genetic algorithm perform better than a memetic algorithm?
- Is the choice of **meme** (hill climbing/local search algorithm) important?
- Which meme does have a better performance in a memetic algorithm?
- Can we somehow combine several memes to obtain a synergy leading to improved performance?

Benchmark Functions

Why to use benchmark (test) functions for optimisation

- Benchmark functions serves as a testbed for performance comparison of (meta/hyper)heuristic optimisation algorithms
  - Their **global minimum** are **known**
  - They can be **easily computed**
  - Each function is recognised to have certain characteristics potentially representing a different real-world problem
    - E.g. ,separable vs non-separable
- COmparing Continuous Optimisers (COCO) provide benchmark function testbeds: http://coco.gforge.inria.fr/

Classification of benchmark (test) functions

- A common classification is:
  - **Continuity (Differentiability)**
    - **Discontinuous** vs **continuous**
  - **Dimensionality**
    - **Scalability**
  - **Separability**

$$\underset{x_1,\ldots,x_p}{\arg\operatorname{minimize}} f(x_1,\ldots,x_p) = \left(\underset{x_1}{\arg\operatorname{minimize}} f(x_1,\ldots),\ldots,\ \underset{x_p}{\arg\operatorname{minimize}} f(\ldots,x_p)\right)$$

  - **Modality**
    - **Unimodal**
    - **Multimodal** with few local minima
    - **Multimodal** with exponential number of local minima

A collection of **unconstrained** benchmark functions: https://arxiv.org/pdf/1308.4008.pdf

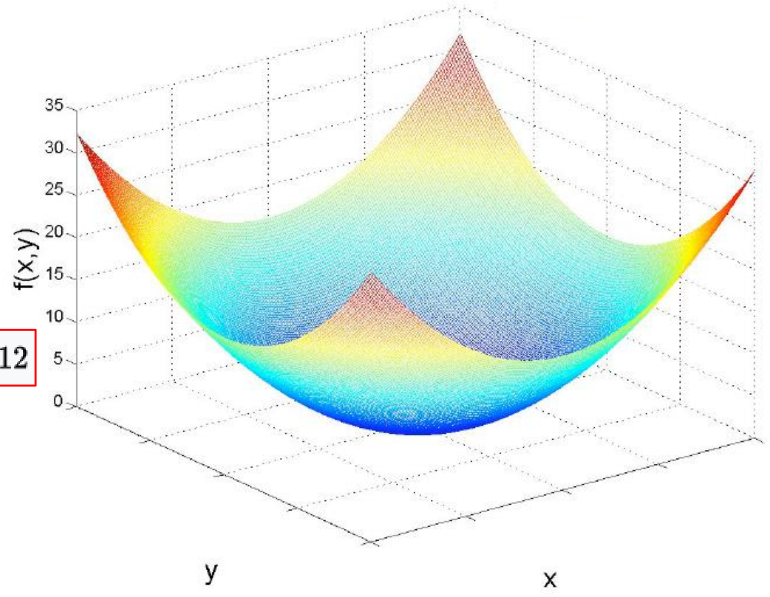
Example – unimodal function: Sphere function

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$$

$$f(x_1, \dots, x_n) = f(0, \dots, 0) = 0$$

$n = 2$

$$-5.12 \leq x, y \leq 5.12$$



- continuous
- differentiable
- separable
- scalable

Delta Evaluation in Function Optimisation

- Separable functions allows delta evaluation
- Example, sphere function

$$\boxed{f(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2}$$

$$\underbrace{x_1 \quad \ldots\, \overset{x_{43}}{6} \,\ldots \quad x_n}_{\underline{s}} \;\Rightarrow\; \underbrace{x_1 \quad \ldots\, \overset{x_{43}}{5} \,\ldots \quad x_n}_{\underline{s}'}$$

$$f(s) = \Sigma \quad \Rightarrow \quad f(s') = \Sigma + \Delta$$

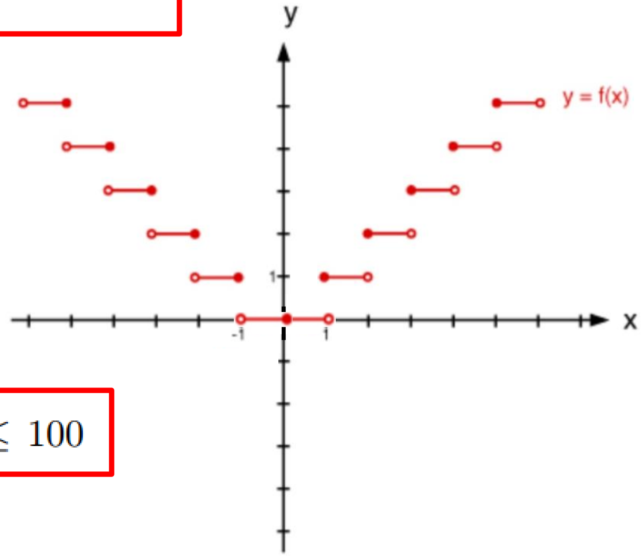
$$(5^2 - 6^2)$$

Example – unimodal function: Step function

$$f(\mathrm{x}) = \sum_{i=1}^{n} (\lfloor |x_i| \rfloor)$$

$$f(x_1, \ldots, x_n) = f(0, \ldots, 0) = 0$$

$n = 1$



$$-100 \leq x_i \leq 100$$

- discontinuous
- non-differentiable
- separable
- scalable

Example – multimodal function Rastrigin's function

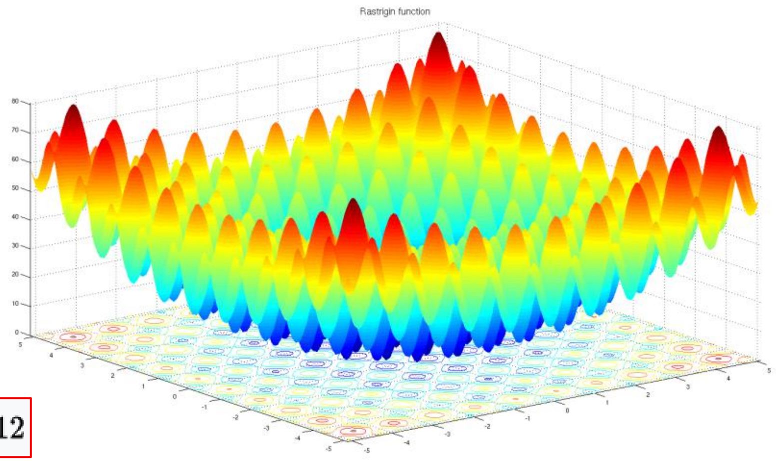
$$f(\mathbf{x}) = An + \sum_{i=1}^{n} \left[x_i^2 - A\cos(2\pi x_i)\right]$$

where: $A = 10$

$n = 2$

$f(0,0) = 0$

$-5.12 \leq x, y \leq 5.12$



- continuous
- differentiable
- separable
- scalable

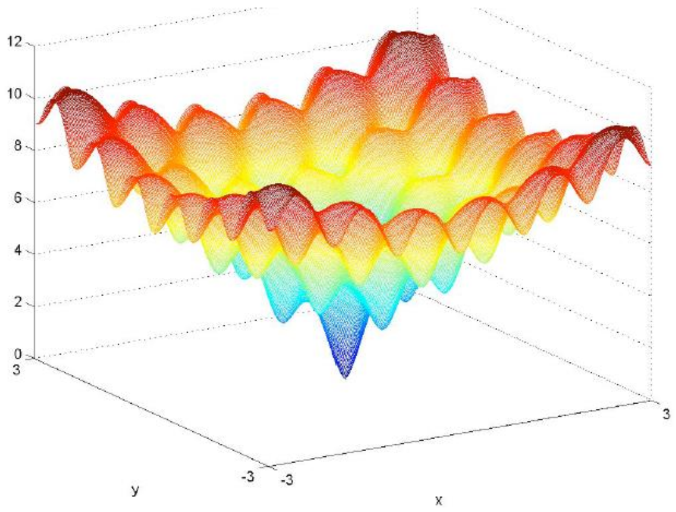
Example – multimodal function Ackley's function

$$f(x,y) = -20\exp\left(-0.2\sqrt{0.5\left(x^2+y^2\right)}\right) - \exp(0.5\left(\cos(2\pi x)+\cos(2\pi y)\right)) + e + 20$$

$n = 2$

$$f(0,0) = 0$$

$$-5 \le x, y \le 5$$



- continuous
- differentiable
- non-separable
- non-scalable

Example –
Sphere Function Optimisation

- **<u>Objective:</u>**
- Find the set of integers $\boldsymbol{x_i}$ which maximize the function $\boldsymbol{f}$ given below.

$$\boxed{f(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2} \Rightarrow \text{for } n=3,\ f(x) = x_1^2 + x_2^2 + x_3^2$$

$$-512 < x_i \leq 512$$

Fitness function

Sphere Function Optimisation – Representation I

- There are 1024 integers in the given interval

  $-512 < x_i \leq 512$

- In binary representation, 1024 different integers can be represented using 10 bits

<u>Encoding of</u> x_i

0 : 0000000000 (-511)
1 : 0000000001 (-510)
... ...
1023 : 1111111111 (512)

<u>Decoding of</u> x_i

- Convert binary # into decimal, e.g., x_i = (0000000011) = 3
- Subtract 511 from that value, e.g., 3-511 = -509

Sphere Function Optimisation – Representation II

- Function has 3 parameters: $x_1,\ \ x_2,\ \ x_3$
- Each parameter can be represented by 10 bits
- Chromosome consists of 30 bits
- Example

1100101010110000000000000000110

x_1 $\qquad\qquad$ x_2 $\qquad\qquad$ x_3

Sphere Function Optimisation – Fitness Evaluation (Decoding)

110010101011000000000000000110

$\boldsymbol{x_1}$ $\boldsymbol{x_2}$ $\boldsymbol{x_3}$

$$f(x) = x_1^2 + x_2^2 + x_3^2$$

$x_1 = (\mathbf{810} - 511) = 299$

$x_2 = (\mathbf{768} - 511) = 257$

$x_3 = (\mathbf{6} - 511) = -505$

Fitness: $f(x) = (299)^2 + (257)^2 + (-505)^2$

$= 410425$

More on Function Optimization

- What if x_i were real numbers in the interval
  $-5.12 < x_i \leq 5.12$
- Solution:
  - Use binary representation
    - with precision of 2 digits after decimal point
    - use 1024 different numbers
    - divide number by 100
  - Use other representation (e.g. real)
- What if the representation has redundancy?

  e.g. $-5.40 < x_i \leq 5.40$

Case Studies:

Benchmark Function Optimisation

Travelling Salesman Problem

Ender Özcan, Burak Bilgin, Emin Erkan Korkmaz, A Comprehensive Analysis of Hyper-heuristics, Intelligent Data Analysis, 12:1, pp. 3-23, 2008. [PDF]

BENCHMARK FUNCTION OPTIMISATION

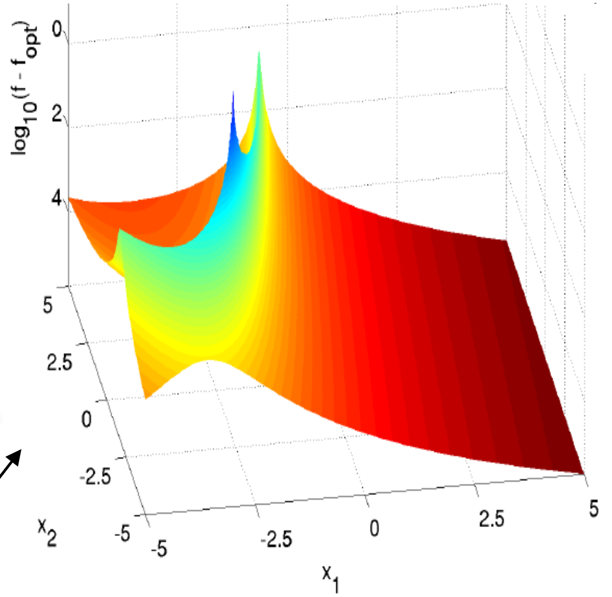
Benchmark Function Optimi

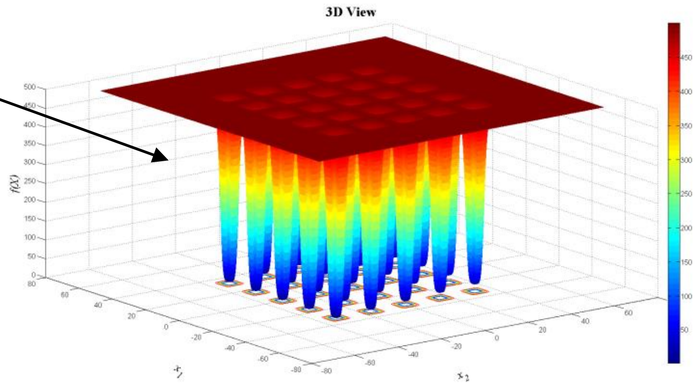
Label	Function name	lb	ub	opt	*isContinuous*	*isSeparable*	*isMultimodal*
F1	Sphere	−5.12	5.12	0	yes	yes	no
F2	Rosenbrock	−2.048	2.048	0	yes	yes	no
F3	Step	−5.12	5.12	0	yes	yes	no
F4	Quartic *with noise*	−1.28	1.28	1	yes	yes	yes
F5	Foxhole	−65.536	65.536	1	yes	no	yes
F6	Rastrigin	−5.12	5.12	0	yes	yes	yes
F7	Schwefel	−500	500	0	yes	yes	yes
F8	Griewangk	−600	600	0	yes	no	yes
F9	Ackley	−32.768	32.768	0	yes	no	yes
F10	Easom	−100	100	−1	yes	no	no
F11	Schwefel's Double Sum	−65.536	65.536	0	yes	no	no
F12	Royal Road	–	–	0	no	yes	n/a
F13	Goldberg	–	–	0	no	yes	n/a
F14	Whitley	–	–	0	no	yes	n/a

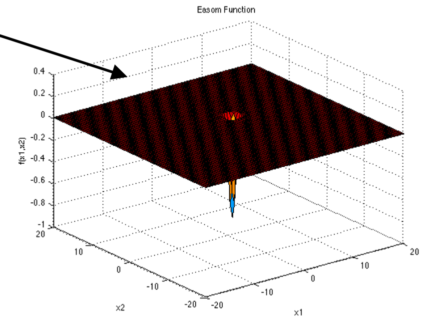
http://www.cs.nott.ac.uk/~pszeo/docs/publications/IDA08.pdf

Benchmark Function Optimisation

Label	Function name	lb	ub
F1	Sphere	−5.12	5.12
F2	Rosenbrock	−2.048	2.048
F3	Step	−5.12	5.12
F4	Quartic *with noise*	−1.28	1.28
F5	Foxhole	−65.536	65.536
F6	Rastrigin	−5.12	5.12
F7	Schwefel	−500	500
F8	Griewangk	−600	600
F9	Ackley	−32.768	32.768
F10	Easom	−100	100
F11	Schwefel's Double Sum	−65.536	65.536
F12	Royal Road	s_1 = 11111111**; s_2 = ********11111111**; s_3 = ****************11111111**; s_4 = ************************11111111********************************;	—
F13	Goldberg	s_5 = ********************************11111111************************; s_6 = **11111111****************;	—
F14	Whitley	s_7 = **11111111********; s_8 = **11111111;	—







Binary vs Gray Encoding

- Gray encoding ensures a Hamming distance of 1 for the adjacent numbers
- Shown to be useful in GAs empowering the algorithm to mutate a solution in the right direction

Dec	Binary	Gray
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

Components of Evolutionary Algorithms and Settings

Label	*dim*	*bits*	*Chrom. Len.*	*Pop. size*	*Max. hcteps*
F1	10	30	300	60	600
F2	10	30	300	60	600
F3	10	30	300	60	600
F4	10	30	300	60	600
F5	2	30	60	20	120
F6	10	30	300	60	600
F7	10	30	300	60	600
F8	10	30	300	60	600
F9	10	30	300	60	600
F10	6	30	180	36	360
F11	10	30	300	60	600
F12	8	8	64	20	128
F13	30	3	90	20	180
F14	6	4	24	20	48

- Representation: Gray encoding
- Initialisation: random
- Mate selection: Tournament with tour size of 2
- Crossover: 1PTX (p_c=1.0)
- Traditional mutation based on bit-flip with mutation rate 1/(2 x chromosome_length)
- A trans-generational EA with a replacement method which keeps only two best individuals from the previous generation.

Components of Evolutionary Algorithms and Settings II

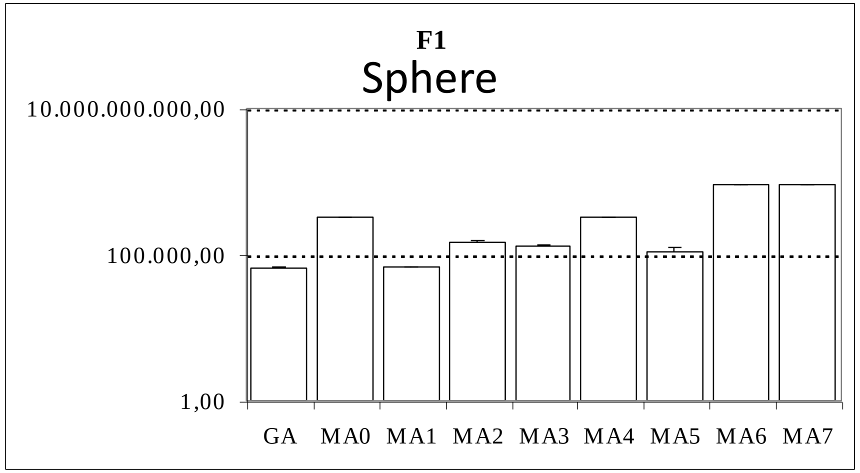
Label	dim	bits	Chrom. Len.	Pop. size	Max. hcteps
F1	10	30	300	60	600
F2	10	30	300	60	600
F3	10	30	300	60	600
F4	10	30	300	60	600
F5	2	30	60	20	120
F6	10	30	300	60	600
F7	10	30	300	60	600
F8	10	30	300	60	600
F9	10	30	300	60	600
F10	6	30	180	36	360
F11	10	30	300	60	600
F12	8	8	64	20	128
F13	30	3	90	20	180
F14	6	4	24	20	48

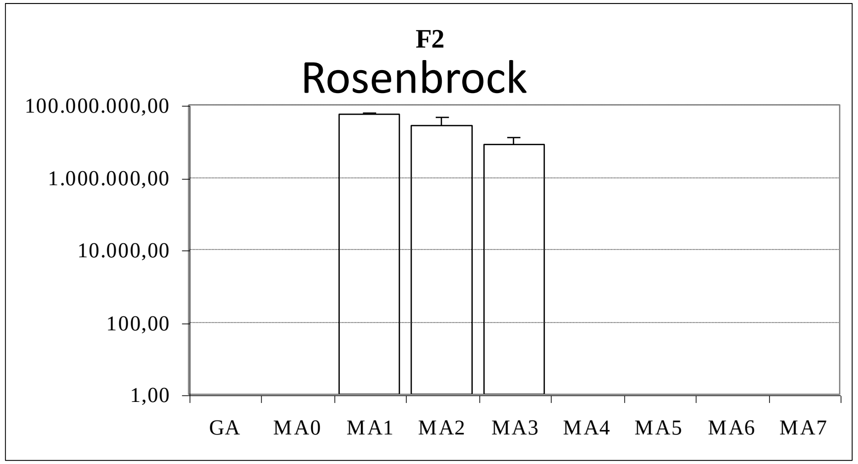
- Memes (using bit-flip):
  - GA: Genetic Algorithm (none)
  - MA0: MA with SDHC (best imp.)
  - MA1: MA with NDHC (first imp.)
  - MA2: MA with RMHC
  - MA3: MA with DBHC
- Expectedly, poor performing memes (using biased moves) are designed:
  - MA4: SDHC $\Rightarrow$ neighborhood move: AND with 0
  - MA5: SDHC $\Rightarrow$ neighborhood move: OR with 1
  - MA6: NDHC $\Rightarrow$ neighborhood move: AND with 0
  - MA7: NDHC $\Rightarrow$ neighborhood move: OR with 1

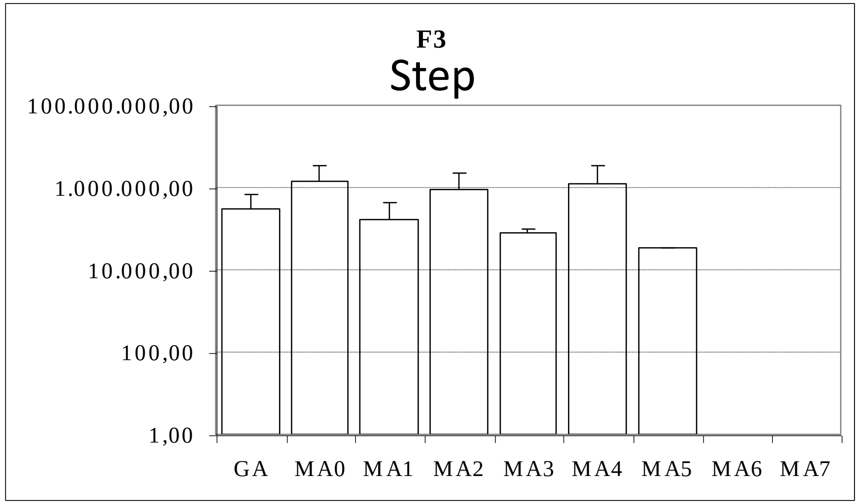
Termination and Performance Comparison Criteria

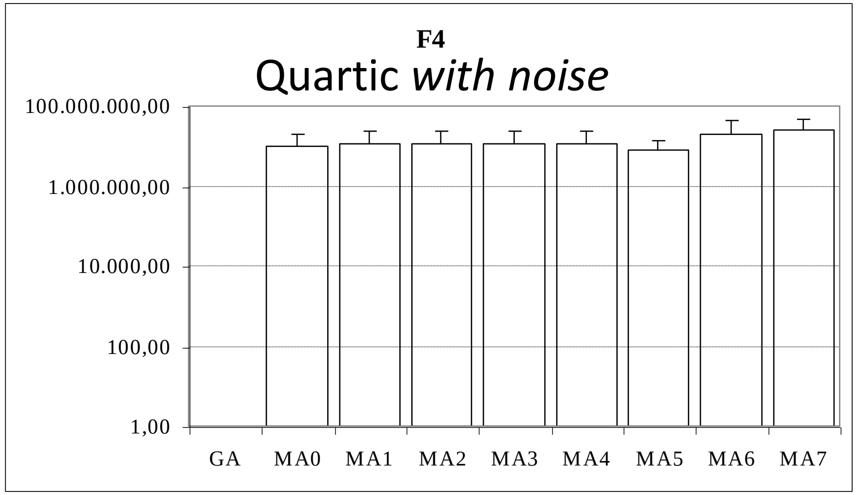
- Termination: Runs are terminated whenever the overall CPU time exceeds 600 sec., or an expected fitness (optimum) is achieved.
  - Pentium IV 2 GHz. machines with 256 MB RAM are used
- All runs are repeated 50 times.
- Performance indicators:
  - Success rate (effectiveness): The ratio of the number of runs returning the expected optimal solution to the total number of runs (50)
  - Average number of evaluations/configurations (efficiency)
  - Bar chart plots showing the average no. of evaluations in log scale for each algorithm, if the success rate is 100% (all 50 runs yield expected optimum), otherwise no bar is drawn.

Results

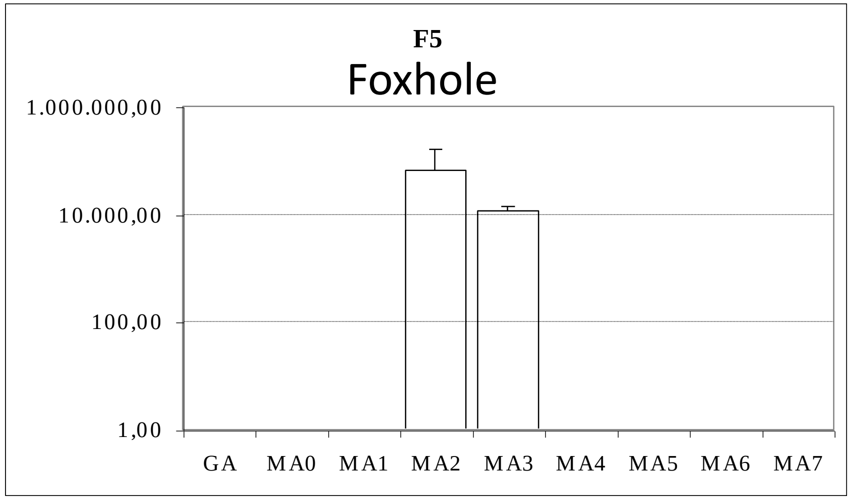

F1
Sphere
10.000.000.000,00
100.000,00
1,00
GA
MA0
MA1
MA2
MA3
MA4
MA5
MA6
MA7

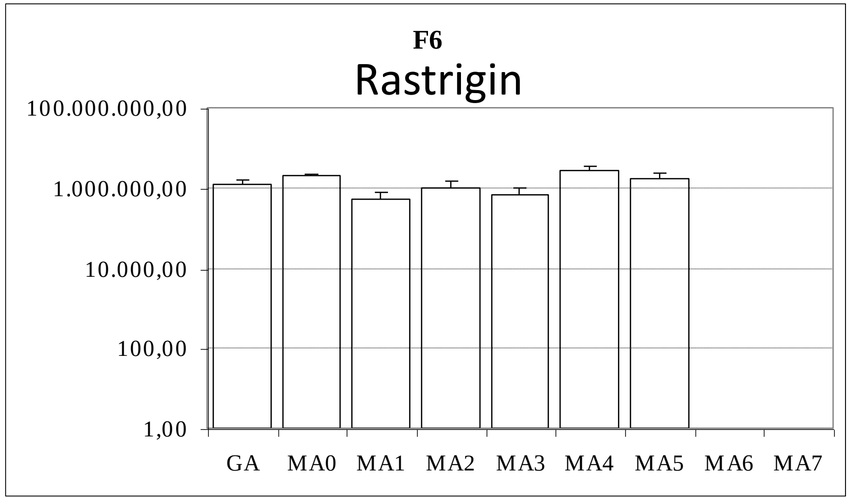

F2
Rosenbrock
100.000.000,00
1.000.000,00
10.000,00
100,00
1,00
GA
MA0
MA1
MA2
MA3
MA4
MA5
MA6
MA7

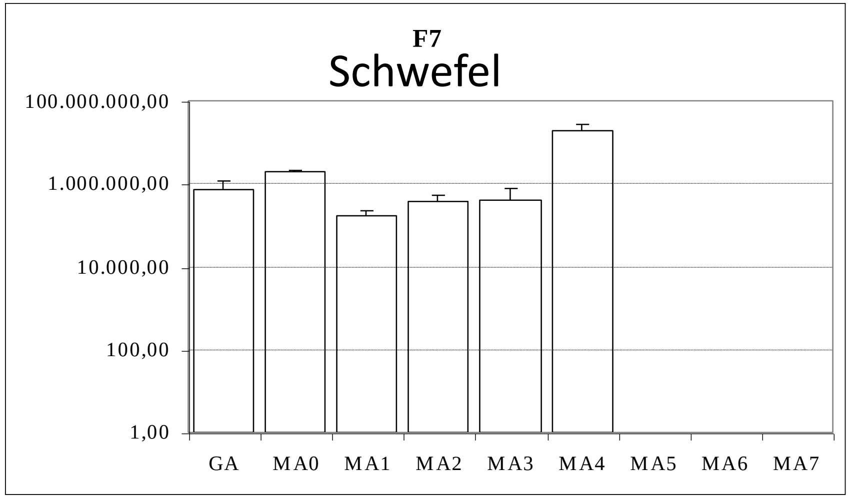

F3
Step
100.000.000,00
1.000.000,00
10.000,00
100,00
1,00
GA
MA0
MA1
MA2
MA3
MA4
MA5
MA6
MA7

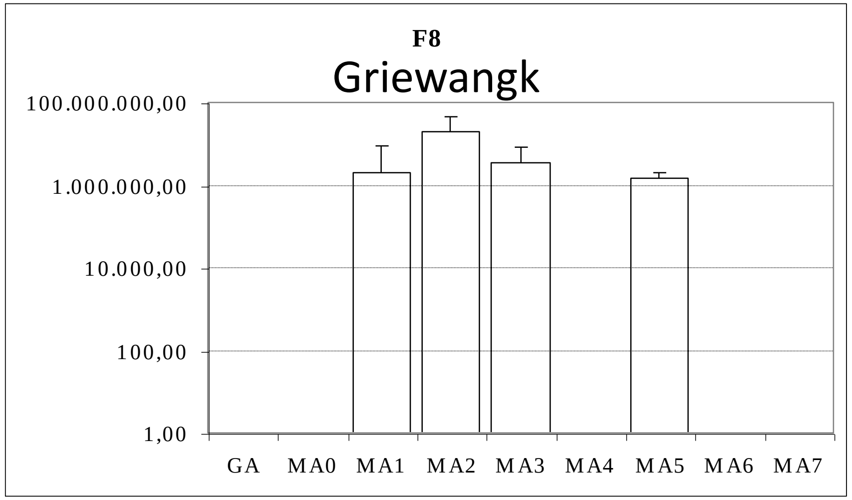

F4
Quartic *with noise*
100.000.000,00
1.000.000,00
10.000,00
100,00
1,00
GA
MA0
MA1
MA2
MA3
MA4
MA5
MA6
MA7

Results II

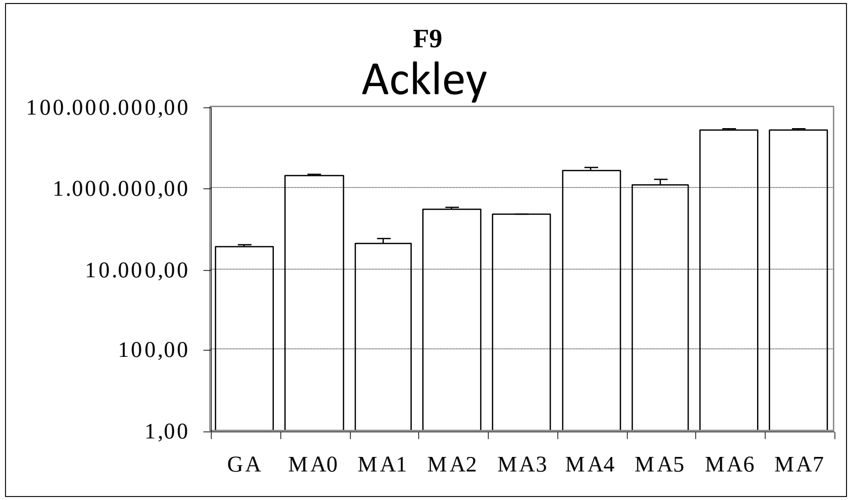

F5
Foxhole
1.000.000,00
10.000,00
100,00
1,00
GA
MA0
MA1
MA2
MA3
MA4
MA5
MA6
MA7

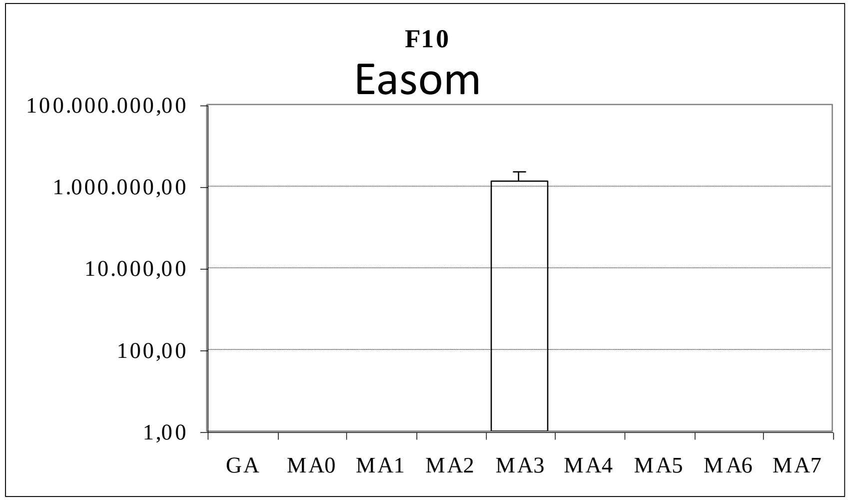

F6
Rastrigin
100.000.000,00
1.000.000,00
10.000,00
100,00
1,00
GA
MA0
MA1
MA2
MA3
MA4
MA5
MA6
MA7

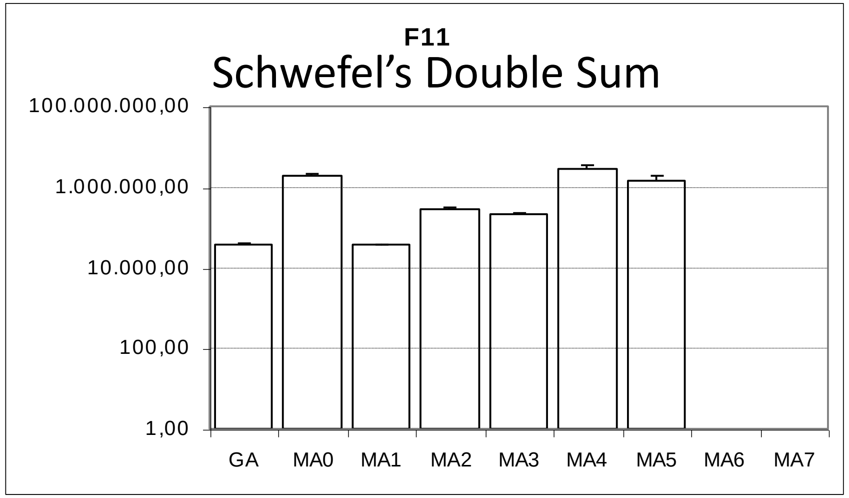

F7
Schwefel
100.000.000,00
1.000.000,00
10.000,00
100,00
1,00
GA
MA0
MA1
MA2
MA3
MA4
MA5
MA6
MA7

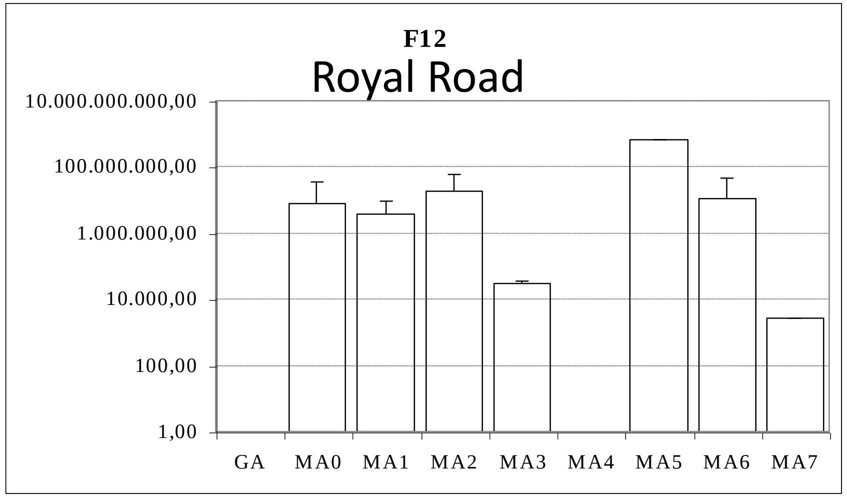

F8
Griewangk
100.000.000,00
1.000.000,00
10.000,00
100,00
1,00
GA
MA0
MA1
MA2
MA3
MA4
MA5
MA6
MA7

Results III

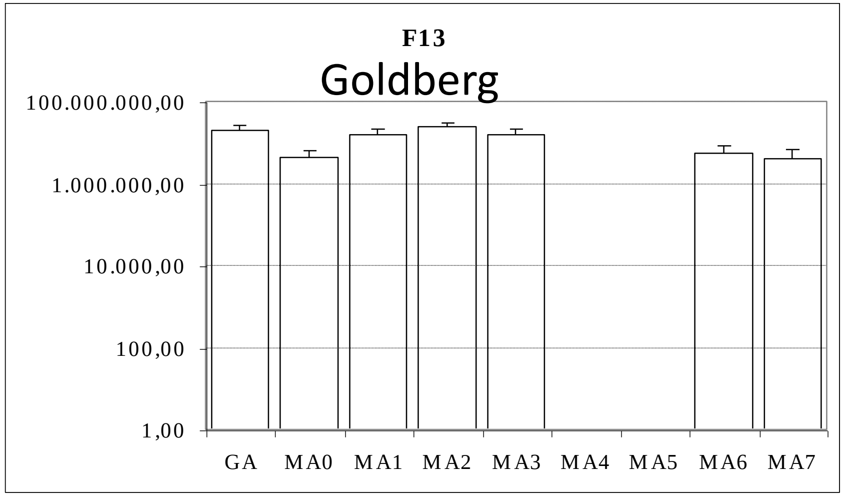

F9
Ackley
100.000.000,00
1.000.000,00
10.000,00
100,00
1,00
GA
MA0
MA1
MA2
MA3
MA4
MA5
MA6
MA7

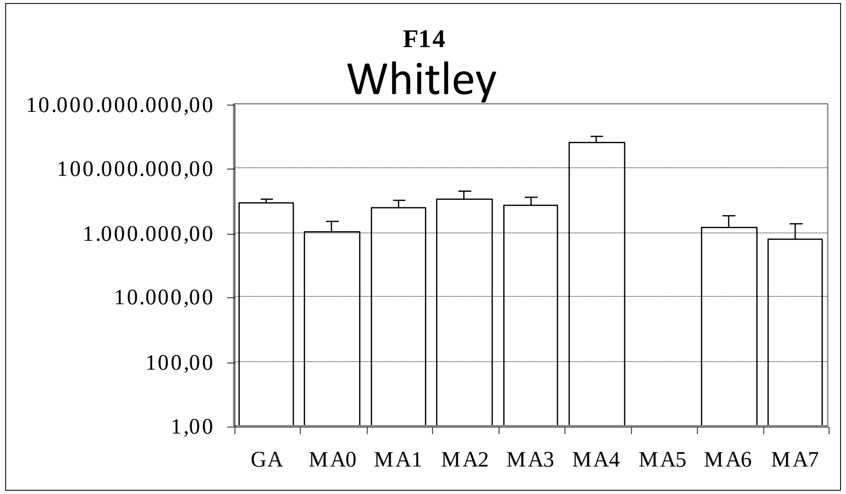

F10
Easom
100.000.000,00
1.000.000,00
10.000,00
100,00
1,00
GA
MA0
MA1
MA2
MA3
MA4
MA5
MA6
MA7


F11
Schwefel's Double Sum
100.000.000,00
1.000.000,00
10.000,00
100,00
1,00
GA
MA0
MA1
MA2
MA3
MA4
MA5
MA6
MA7


F12
Royal Road
10.000.000.000,00
100.000.000,00
1.000.000,00
10.000,00
100,00
1,00
GA
MA0
MA1
MA2
MA3
MA4
MA5
MA6
MA7

Results IV


F13
Goldberg
100.000.000,00
1.000.000,00
10.000,00
100,00
1,00
GA
MA0
MA1
MA2
MA3
MA4
MA5
MA6
MA7


F14
Whitley
10.000.000.000,00
100.000.000,00
1.000.000,00
10.000,00
100,00
1,00
GA
MA0
MA1
MA2
MA3
MA4
MA5
MA6
MA7

Summary

- MA0 (SDHC) is the best meme choice for F4, F13 and F14. (noisy + deceptive)
- MA1 (NDHC) is the best meme choice for F6-F8. (multimodal)
- MA3 is (DBHC) the best meme choice for F2, F3, F5, F10, F12. (functions with plateaus)
- For functions F1, F11 (unimodal) and F9, GA performs slightly better than the memetic algorithm with the meme MA1 (NDHC). The performance difference is insignificant.
- In the overall, MA2 (RMHC) and MA3 (DBHC) turn out to be the worst and the best meme, respectively, among MA0-MA3, considering the success rates and average number of evaluations.
- Different memes yield different performances, designing the right meme for the problem in hand is important.

E. Özcan, and M. Erenturk, A Brief Review of Memetic Algorithms for Solving Euclidean 2D Traveling Salesrep Problem, Proc. of the 13th Turkish Symposium on Artificial Intelligence and Neural Networks, pp. 99-108, June 2004. [PDF]

TRAVELLING SALESMAN PROBLEM

Binary Representation for Encoding Permutation

- Binary representation and classical random initialisation, cross-over and mutation operators are not suitable for encoding permutation
  - E.g., for TSP, could cause illegal tours to form
    - not all cities visited
    - undefined city codes
    - cities visited more than once
    - loops formed within the tour
  - needs repair algorithms

Example

Given N=5 cities $\Rightarrow$ 3 bits per city,

2 5 3 1 4

individual 1: 0|10101011001100

5 2 1 4 3

individual 2: 1|01010001100011

2 | 5 3 1 4

5 | 2 1 4 3

After applying 1-point crossover:

1 2 1 4 3

individual 1: 001010001100011

6 5 3 1 2

individual 2: 110101011001100

2 2 1 4 3

5 5 3 1 4

Generic Permutation based Genetic Operators

Partially Mapped Crossover (PMX)

- Builds offspring by
  - choosing a subsequence of a tour from one parent
    - choose two random cut points to serve as swapping boundaries
    - swap segments between cut points
  - preserving the order and position of as many cities as possible from other parent
- Exploits important similarities in the value and ordering simultaneously

PMX

Example:

p1: (1 2 3 | 4 5 6 7 | 8 9)

p2: (4 5 2 | 1 8 7 6 | 9 3)

step 1: swap segments

o1: (x x x | 1 8 7 6 | x x)

o2: (x x x | 4 5 6 7 | x x)

this also defines mappings:

1⇔4, 8⇔5, 7⇔6, 6⇔7

step 2: fill in cities from other parents if no conflict

o1: (x 2 3 | 1 8 7 6 | x 9)

o2: (x x 2 | 4 5 6 7 | 9 3)

step 3: use mappings for conflicted positions

o1: (**4** 2 3 | 1 8 7 6 | **5** 9)

o2: (**1** **8** 2 | 4 5 6 7 | 9 3)

Order Crossover (OX)

- Builds offspring by
  - choosing a subsequence of a tour from one parent
  - preserving relative order of cities from other parent
- Exploits the property that ordering of cities important, not positions

  9-3-4-5-2-1-8-7-6 and

  4-5-2-1-8-7-6-9-3 are identical

OX

Example:

p1: (1 2 3 | 4 5 6 7 | 8 9)

p2: (4 5 2 | 1 8 7 6 | 9 3)

step 1: copy segments into offspring

o1: (x x x | 4 5 6 7 | x x)

o2: (x x x | 1 8 7 6 | x x)

step 2: starting from 2nd cut point of one parent, cities from other parent copied in same order, omitting symbols already present; if end of string reached, continue from beginning of string

sequence of cities in 2nd parent (from 2nd cut point) is

9-3-4-5-2-1-8-7-6 (remove 4,5,6,7 which are in 1st offspring)

9-3-2-1-8

place into first offspring   o1: (2 1 8 | 4 5 6 7 | 9 3)

similarly the 2nd offspring o2: (3 4 5 | 1 8 7 6 | 9 2)

Cycle Crossover (CX)

- Builds offspring by
  - choosing each city and its position from one of the parents
  - and when a cycle is completed, the remaining cities filled in from the other parent
- Preserves absolute positions of the elements in the parent sequence

Cycle Crossover (CX)

p1: (1 2 3 4 5 6 7 8 9)

p2: (**4 1 2 8 7 6 9 3 5**)

o1: (1 x x x x x x x x)

o1: (1 x x 4 x x x x x)

o1: (1 x x 4 x x x 8 x)

o1: (1 x 3 4 x x x 8 x)

o1: (1 2 3 4 x x x 8 x)

Randomly select a starting point(city) in p1: 1

Copy cities from p1 until a cycle is obtained while mapping from p1 to each corresponding city in p2

2 requires selection of 1 causing a cycle, so rest filled from other parent:

o1: (1 2 3 4 **7 6 9** 8 **5**)

similarly

o2: (**4 1 2 8** 5 6 7 **3** 9)

More Genetic Operators – Crossover for TSP

Operator Name	Date	Authors
Alternating Position Crossover (AP)	(1999)	Larranaga, Kuijpers, Poza, Murga
Cycle Crossover (CX)	(1987)	Oliver, Smith and Holland
Distance Preserving Crossover (DPX)	(1996)	Freisbein and Merz
Edge Assembly Crossover (EAX)	(1997)	Nagata and Kobayashi
Edge Recombination Crossover (ER)	(1989)	Whitley, Timothy, Fuquay
Heuristic Crossover (HEU)	(1987)	Grefenstette
Inver-over Operator (IOO)	(1998)	Tao and Michalewicz
Maximal Preservative Crossover (MPX)	(1988)	Mühlenbein, Schleuter and Krämer
Position Based Crossover (POS)	(1991)	Syswerda
Order Crossover (OX1)	**(1985)**	**Davis**
Order Based Crossover (OX2)	(1991)	Syswerda
Partially mapped Crossover (PMX)	**(1985)**	**Goldberg and Lingle**
Voting Recombination Crossover (VR)	(1989)	Mühlenbein
Natural Crossover (NX)	(2000)	Jung and Moon
Voronoi Quantized Crossover (VQX)	(2002)	Seo and Moon

⋮

More Genetic Operators – Mutation for TSP

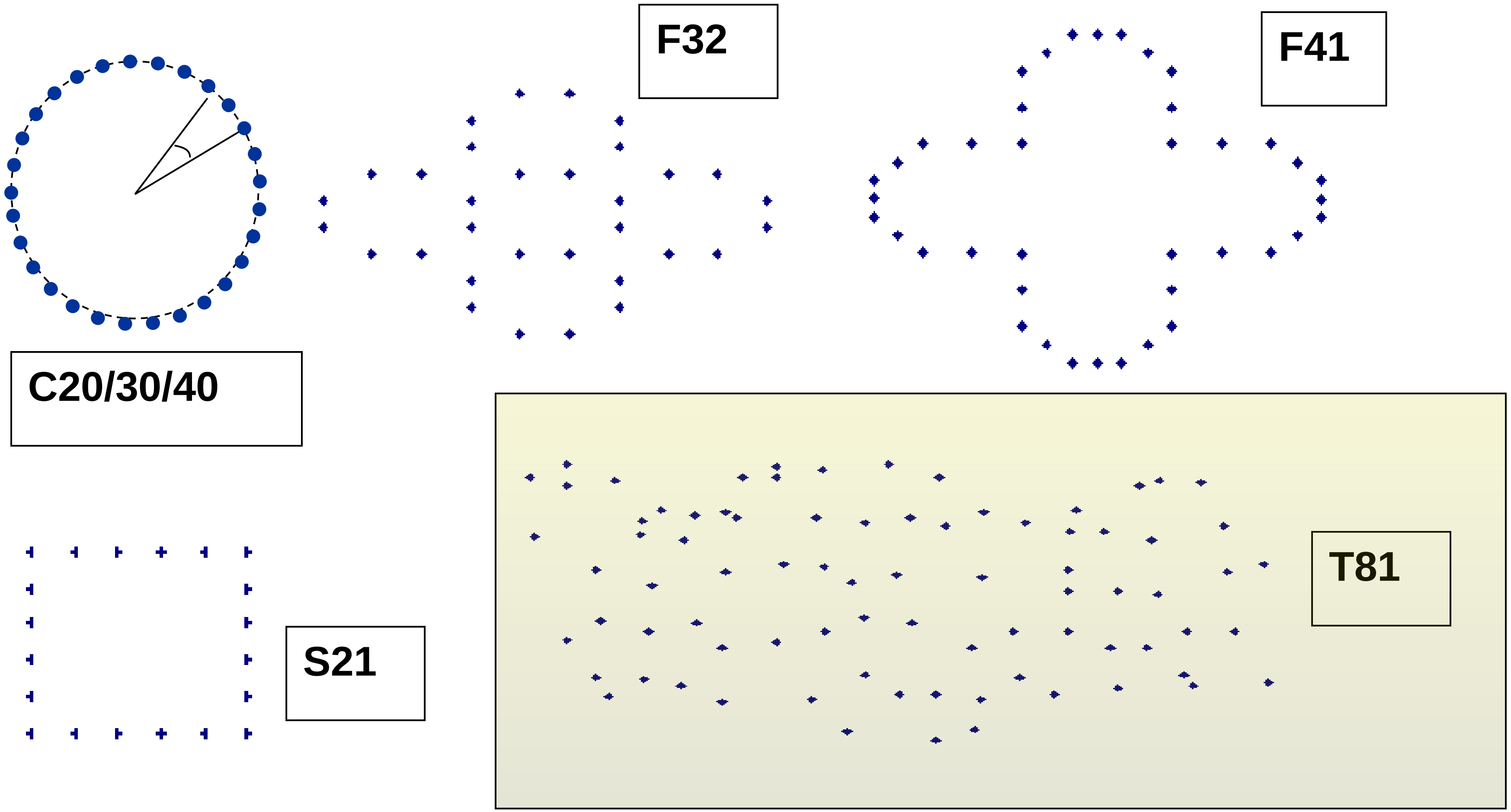
Operator Name	Date	Authors
Displacement Mutation (DM)	(1992)	Michalewicz
Exchange Mutation (EM)	(1990)	Banzhaf
Insertion Mutation (ISM)	(1988)	Fogel
Inversion Mutation (IVM)	(1990)	Fogel
Scramble Mutation (SM)	(1991)	Syswerda
Simple Inversion Mutation (SIM)	(1975)	Holland

GA vs MA for Solving TSP Experiments – Components

- Representation: *path representation* (permutation)
  - For example, (1 3 5 2 6 4) represents a tour starting from city 1, visiting 3, 5, 2, 6, 4 in that order and returning back to 1.
- Fitness function: path length
- Crossover: PMX, 2PTX (with repair/patch-up), OX1
- Mutation Operators: ISM, EM,
- Mate Selection: RANK, TOUR
- Replacement: TG, SS
- Hill Climbing: RMHC using ISM, RMHC using EM

  A hill climbing step is applied, as long as, max. number of iterations is not exceeded and the indiv. is improved.

Experimental Data



Results

GA TYPE	D.L.	ISM α	ISM β	EM α	EM β
SSGA	C20	145.655	**62.575**	**110.786**	71.431
	C30	147.590	**62.716**	**142.798**	**62.716**
	C40	207.804	**62.768**	**157.104**	121.454
	S21	116.598	**60.000**	**98.714**	**60.000**
	F32	159.191	**91.094**	**118.792**	108.694
	F41	207.856	**77.609**	**151.658**	120.573
SSGA HC	C20	149.799	**62.575**	**110.654**	**62.575**
	C30	186.314	**62.716**	**131.118**	88.975
	C40	243.077	**62.768**	**144.962**	114.914
	S21	129.523	**60.000**	**99.411**	**60.000**
	F32	157.291	**89.288**	**114.841**	106.003
	F41	205.571	**68.168**	**134.152**	92.426
TGGA	C20	173.683	**62.575**	**167.233**	**62.575**
	C30	236.148	62.716	**213.842**	**60.000**
	C40	344.740	**62.768**	**278.614**	**62.768**
	S21	169.553	**60.000**	**151.198**	**60.000**
	F32	201.821	92.945	**182.890**	**84.180**
	F41	287.101	**68.168**	**241.825**	**68.168**
TGGA HC	C20	153.128	**62.575**	**147.378**	**62.575**
	C30	198.448	**62.716**	**187.580**	**62.716**
	C40	309.523	**62.768**	**282.683**	**62.768**
	S21	149.920	**60.000**	**133.237**	**60.000**
	F32	172.591	**84.180**	**159.893**	86.734
	F41	**239.901**	**68.168**	241.260	**68.168**

α: avr. fitness per generation at each run
β: best fitness across 100 runs

From the best to worst, considering the average fitness per generation at each run (100 runs)

- Crossover: **OX1**, 2PTX, PMX
- Mutation: **EM**, ISM (ISM is better in finding the best), 1/chromosome-lenght
- Mate Selection: **TOUR** (tour-size:2), RANK
- Replacement: **TG** (keep top 2 and replace the rest with offspring, *g*=popSize-2), SS (replace the worst pair of individuals with the best pair among the offspring and parents)
- Approach: **MA** (Hill Climbing: **EM**, ISM), GA

A Classification of Memetic Algorithms

Adaptive Type		Adaptive Level		
		External	*Local*	*Global*
Static		Basic meta-Lamarckian learning / Simplerandom		
Adaptive	*Qualitative Adaptation*		Randomdescent / Randompermdescent	Tabu-search
	Quantitative Adaptation		Sub-Problem Decomposition/ Greedy	Straightchoice/ Rankedchoice/ Roulettechoice/ Decompchoice/ Biased Roulette Wheel
Self-Adaptive			Multi-memes/ Co-evolution MA	

Yew-Soon Ong; Meng-Hiot Lim; Ning Zhu; Kok-Wai Wong, "Classification of adaptive memetic algorithms: a comparative study," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* , vol.36, no.1, pp.141,152, Feb. 2006 [PDF]

Multimeme Memetic Algorithms

Self Adaptation for Genetic Operators

- <u>Self Adaptation</u>: Deciding which operators and settings to use on the fly whenever needed receiving feedback during the evolutionary search process
- Davis used varying probabilities of applying different operators
  - Uses the performance of the operator within the last few generations to update probabilities
- Grefenstette, Kakuza, Friesleben, Hartfelder used subpopulations, each having different set of parameters and operators
- Spears used an additional bit to decide whether apply 2-PTX or UX to an individual (co-evolution)

Multimeme Memetic Algorithms

- Introduce memetic material for each individual
  - **Co-evolve** genetic and memetic material
- Krasnogor formalised in his PhD thesis*.
- A meme encodes how to apply an operator (which one to apply, max. no. of iterations, acceptance criteria), when to apply, where to apply, and how frequent to apply
- Meme of each operator can be combined under a *memeplex*

*Krasnogor N (2002) Studies on the theory and design space of memetic algorithms, PhD Thesis, University of the West of England, Bristol, UK

Grammar for a Memeplex– Compound of Memes

- Memeplex = <*Meme*>;<*Meme*>:<*Memeplex*>
- Meme = (< *Where* >, < *When* >, < *How* >, < *Frequency* >)
- Where = < *Location* > *Crossover* ; < *Location* > *Mutation*;
- Location = *After* ; *Before*
- When = < BooleanCondition >
- How = *GeneralHillClimber* < *Move* >< *Strategy* >< *Iterations* >; *BoltzmanAdaptiveHillClimber* < *Move* >< *Strategy* >< *Iterations* > …
- Frequency = *Always*; *For* < *Iterations* >; *Never* …
- Move = *2-exchange*, *3-exchange*
- Strategy = *NextAscent*; *SteepestAscent*; *MiniMax* < *Size* >; *MaxiMin* < *Size* >
- Size = < *Num* >
- Iterations = < *Num* >

⋮

Multimeme Memetic Algorithms – Features

- Memes represent instructions for self-improvement
  - Specify set of rules, programs, heuristics, strategies, behaviors, etc.
- Interaction between memes and genes are not direct
  - Mediated by individual (genetic and memetic material)
- Memes can evolve, change by means of nontraditional genetic transformations and metrics

Examples – Implementation

genetic material memetic material

- MAX-SAT: (1 0 0 0 1 + 2)

  memeplex: Hc2 (Hc: meme and 2:meme option)

  - Hc (hill climbing operator)→ 0: RMHC, 1: SDHC, 2: NDHC

  Mp2Mo1Cp1Co2W0Hc1Hs1I2

- TSP:(4 5 2 1 8 7 6 9 3 + 2 1 1 2 0 1 1 2)

  #1
  - Mp (mutation probability) → 0: $2/n$, 1: $1/n$, 2: $1/(2xn)$, 3: 0.1,…
  - Mo (mutation operator) → 0: Exchange, 1: Insertion,…

  #2
  - Cp (crossover probability) → 0: 1.0, 1: 0.95, 2: 0.90,…
  - Co (crossover operator) → 0: 1PTX+repair, 1: OX, 2: CX,…

  #3
  - W (where to apply HC) → 0: After Mutation, 1: After Crossover,…
  - Hc (strategy/how to apply HC) → 0: first impr., 1: best improvement,…
  - Hs (move/HC step)→ 0: Exchange, 1: Adjacent interchange,…
  - I (frequency/number of HC iterations) → 0: n, 1: $2n$, 2: $4n$, 3: $8n$,…

memes *meme options*

Inheriting Memetic Material – SIM

(1 0 0 0 1 + Hc1Hs1I2) *f* =2

(**1 1** **0 1 1** + **Hc0Hs4I1**) *f* =1

(**1 1** 0 0 1 + **Hc0Hs4I1**)

(1 0 **0 1 1** + **Hc0Hs4I1**)

```
Individual_Level_Crossover(parent1, parent2)
BEGIN
 IF(both parents carrie the same meme}
  Cross parents genetic material.
  Inherit common meme to offspring.
 ELSE-IF (parent1.fitness()==parent2.fitness())
  /* the two parents have different memes          */
  /* but their fitness are comparable hence        */
  /* a random choice is made                       */
  Cross parents genetic material.
  Choose a meme randomly from any of the two parents.
  Inherit selected meme to offspring.
 ELSE
  /* parents don't share memes nor fitness values  */
  /* hence the fittest individual                  */
  /* imposes  its meme preference                  */
  Cross parents genetic material.
  Choose meme from fittest parent.
  Inherit the chosen meme to offspring.
END
```

Krasnogor N, Smith JE (2001) Emergence of profitable search strategies based on a simple inheritance mechanism. In: Proc of the Genetic and Evolutionary Comp. Conf., pp 432–439
http://eprints.uwe.ac.uk/11088/1/11088.pdf

Mutating Memes during Evolution

- Inovation rate (IR) $\in$[0,1], is the probability of mutating the memes
- Mutation randomly sets the meme option to one of the other options
  - MAX-SAT: (1 0 0 0 1 + Hc2) $\rightarrow$ 0: (1 0 0 0 1 + Hc0)
  - TSP: (4 5 2 1 8 7 6 9 3 + Mp2Mo1Cp1Co2W0Hc1Hs1I2) $\rightarrow$
    (4 5 2 1 8 7 6 9 3 + Mp0Mo2Cp0Co1W1Hc3Hs0I4)
- IR=0; no innovation, if a meme option is not introduced in the initial generation, it will not be reintroduced again
- IR=1; All different strategies implied by the available M memes might be equally used.

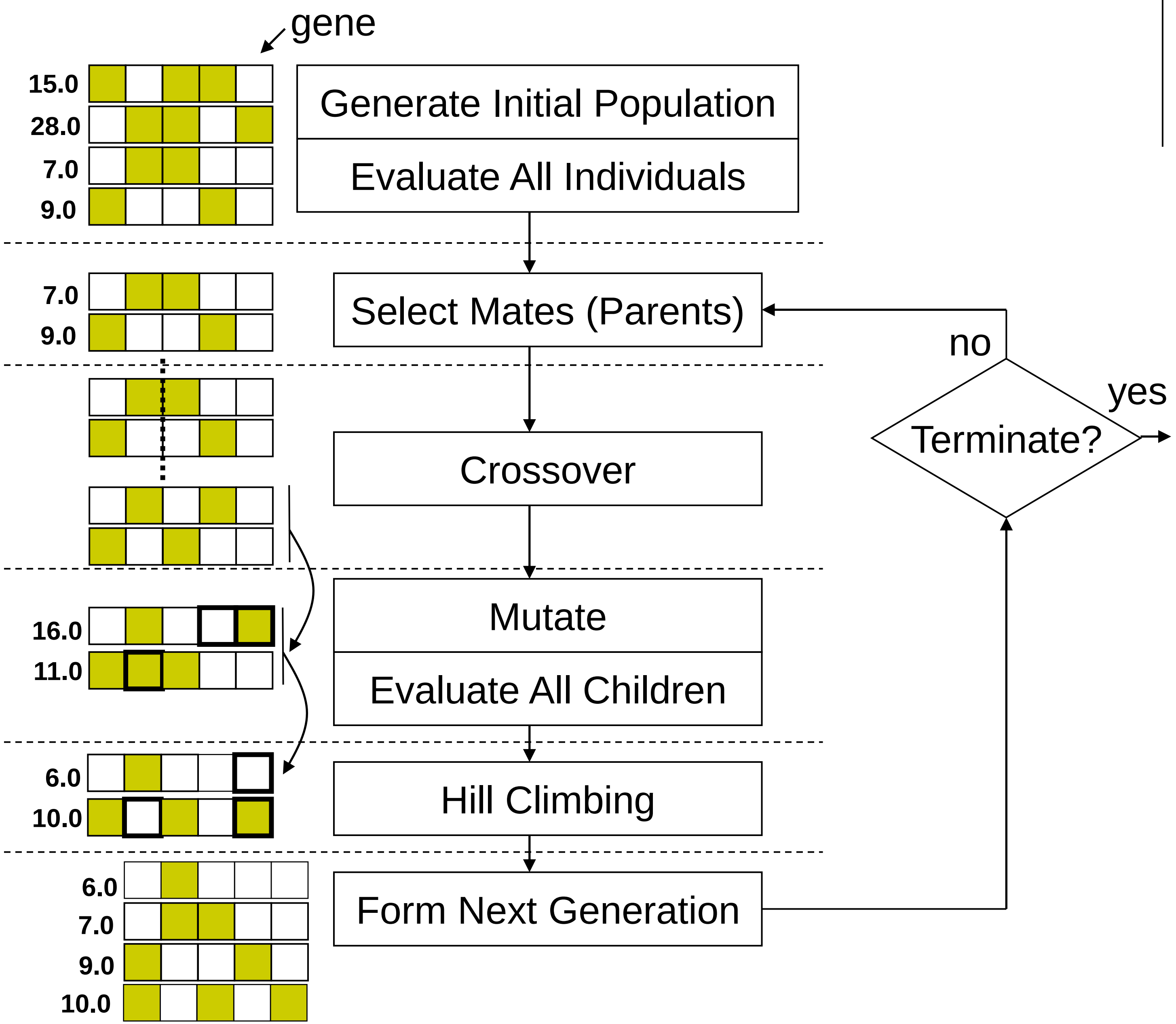
Measure for Evaluating Meme Performance

- *Concentration of a meme* ($c_i(t)$) is the total number of individuals that carry the meme *i* at a given generation *t*,
  - Crude measure of a meme success; gives no information about continual usage of a meme
- *Evolutionary activity of a meme* ($a_i(t)$) is the accumulation of meme concentration until a given generation,

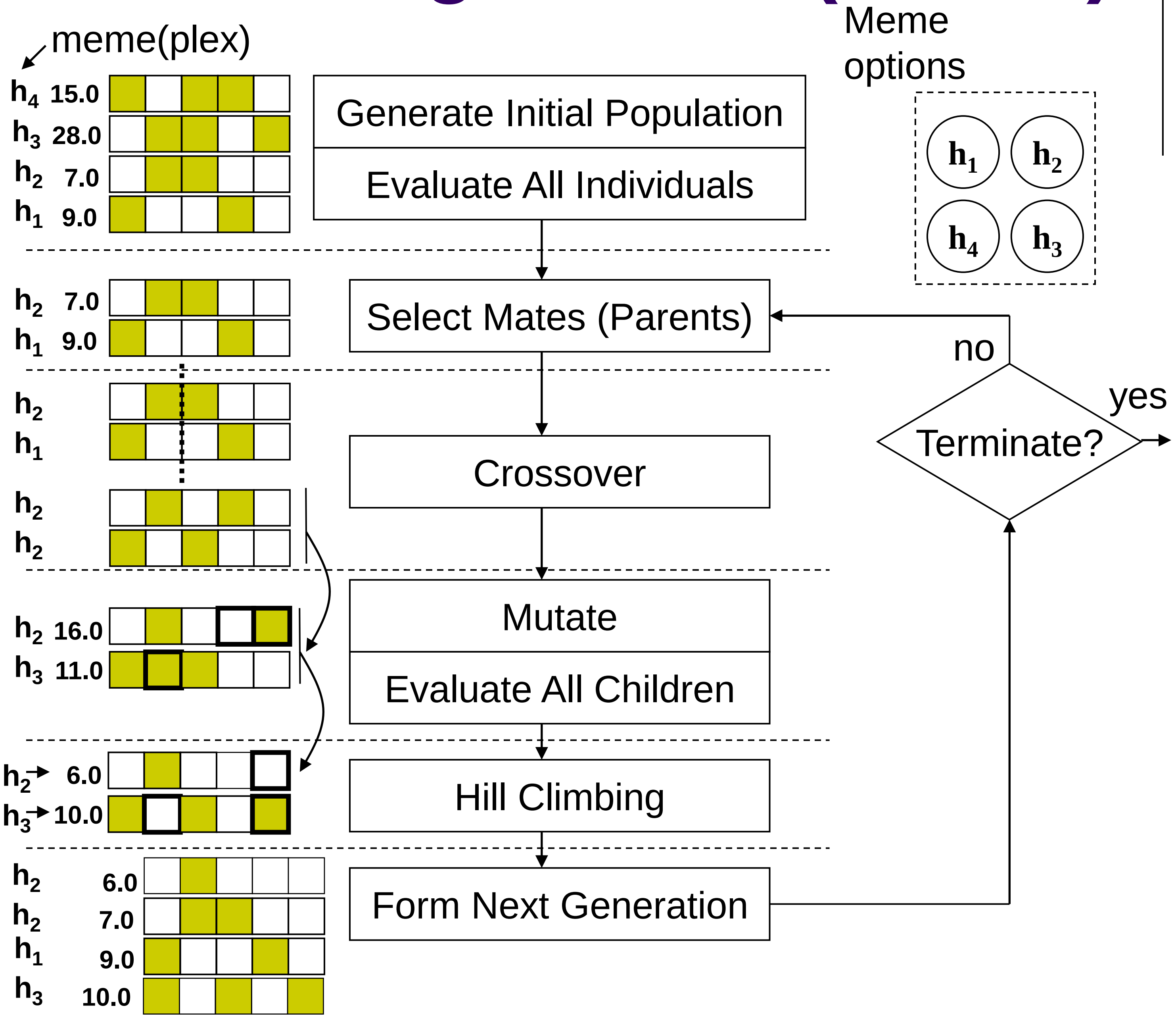
$$a_i(t) = \begin{cases} \int_0^t c_i(t)dt, & \text{if } c_i(t) > 0 \\ 0, & \text{otherwise} \end{cases}$$

  - Slope in a plot represents the rate of increase of a meme concentration

Memetic Algorithms (MAs)



Multimeme Memetic Algorithms (MMAs)



Ender Özcan, Burak Bilgin, Emin Erkan Korkmaz, A Comprehensive Analysis of Hyper-heuristics, Intelligent Data Analysis, 12:1, pp. 3-23, 2008. [PDF]

BENCHMARK FUNCTION OPTIMISATION - REVISITED

MMA Experiments for Benchmark Function Optimisation – Additional Settings

- IR rate is fixed as 0.20
- Two sets of experiments are performed

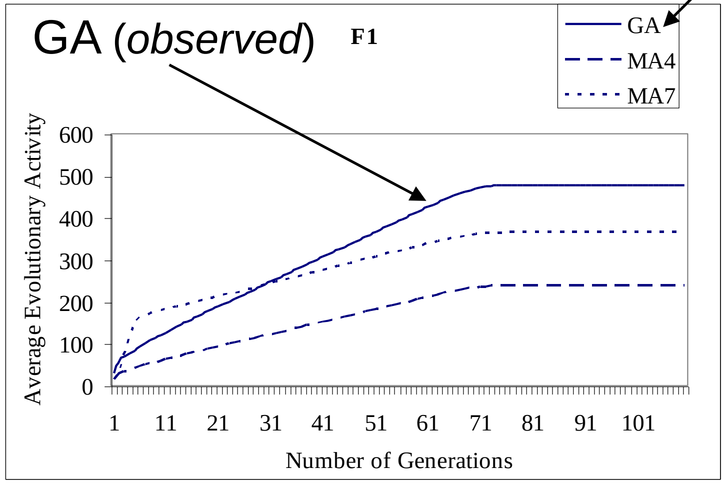
MMA1. A single *good* meme and two *poor performing* memes are used to test the power of MMAs in identifying the *good* ones.

MMA2. Memes GA, MA0, MA1, MA2 and MA3 are used to observe whether MMA will provide some type of synergy between memes.

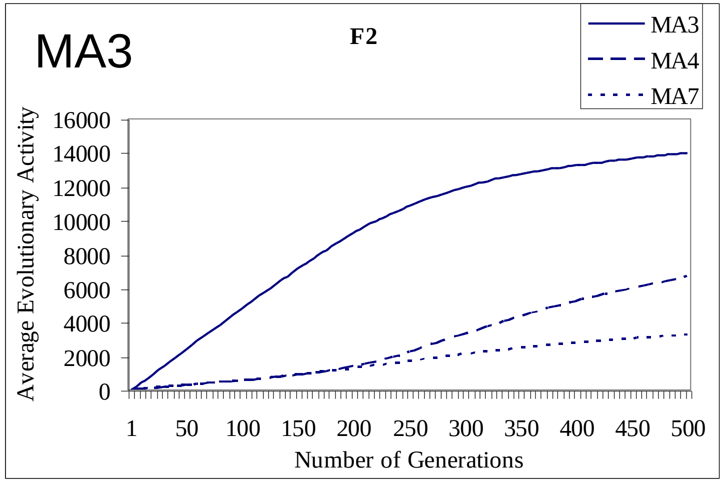
MMA 1.a

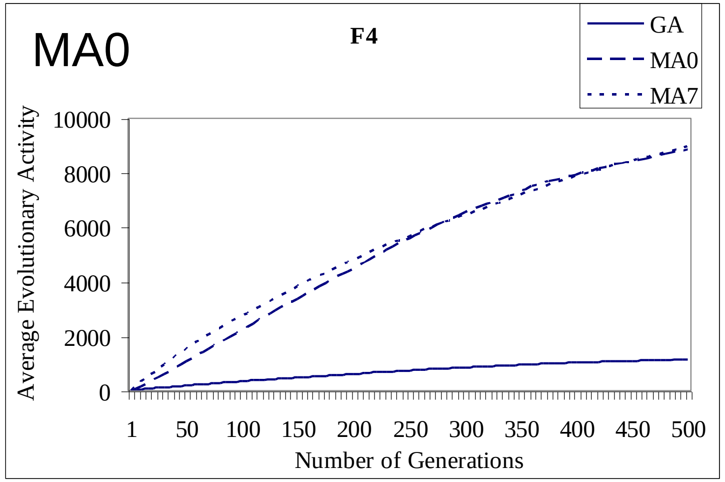
Good meme (GA) followed by two poor performing memes

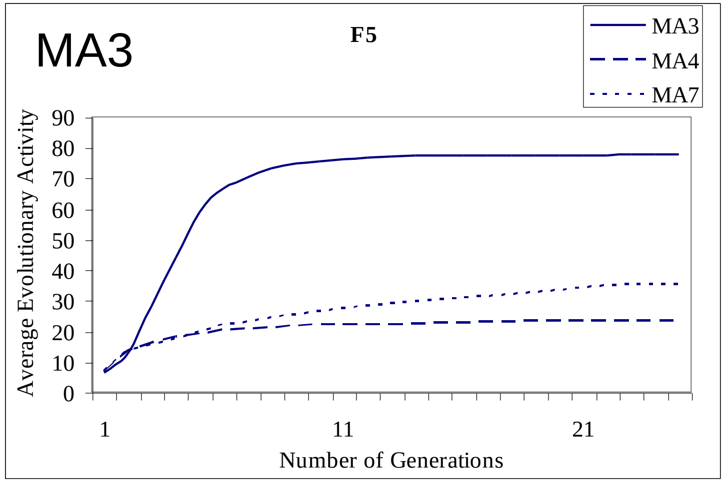
GA (*expected – good meme*)



MA3





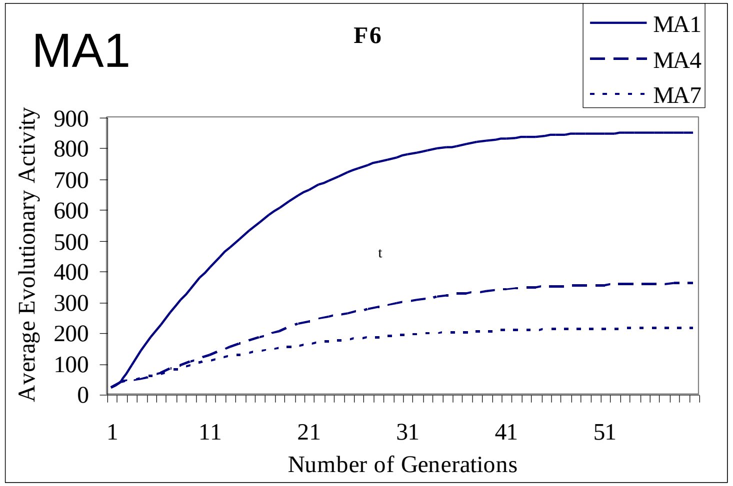


MA0

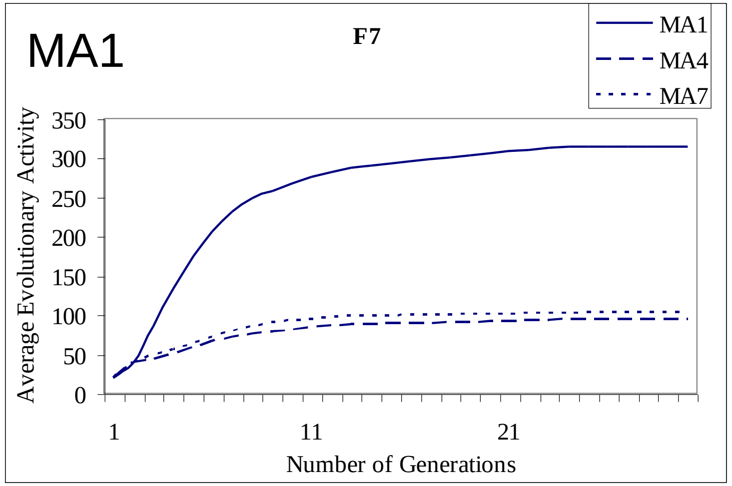
MA3

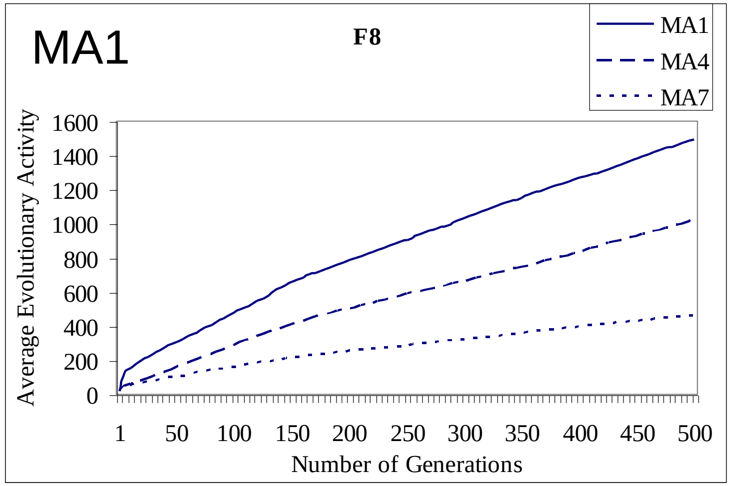
MMA 1.b

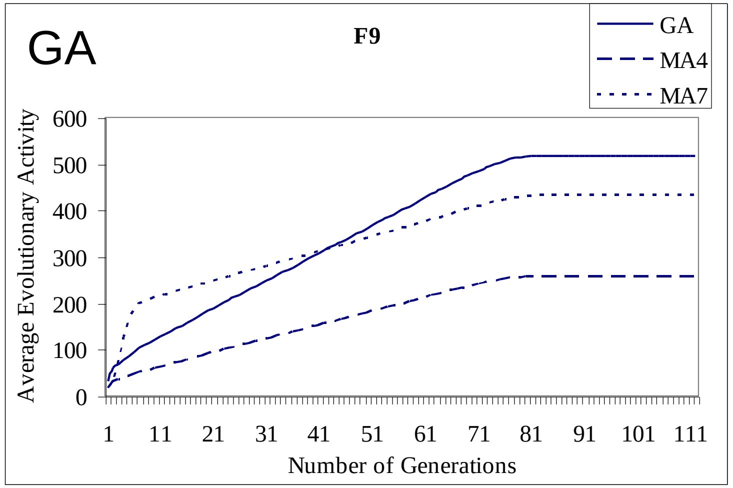
MA1



MA1





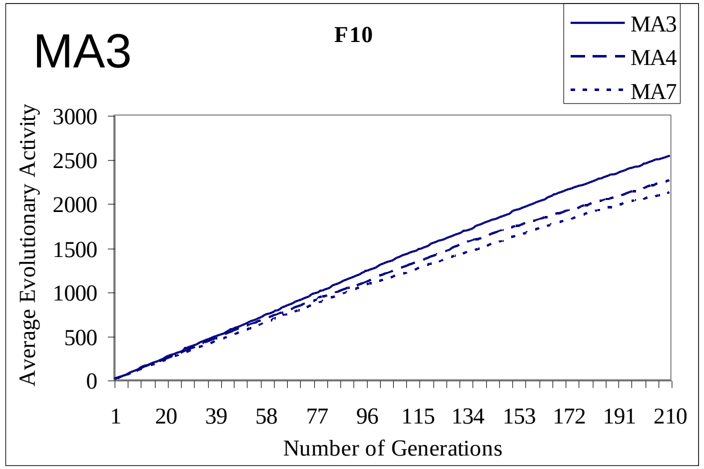


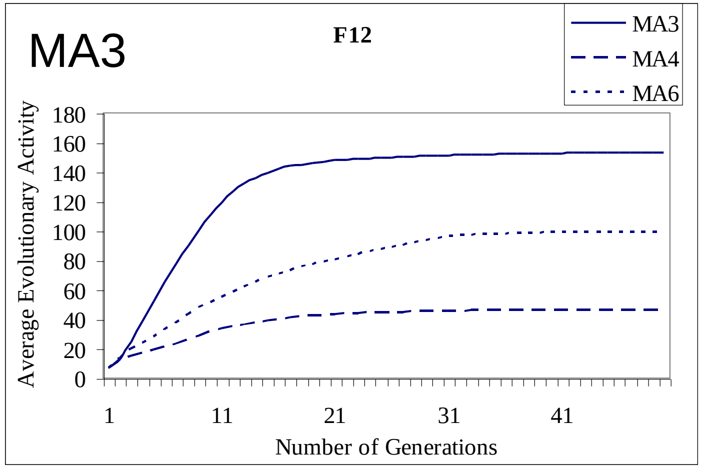
MA1

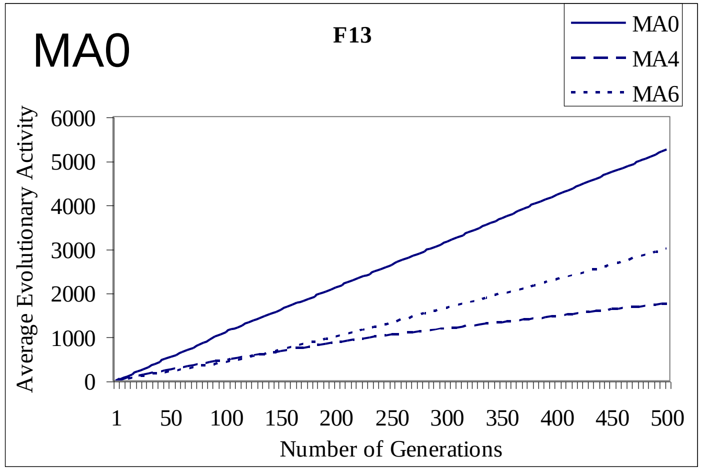
GA

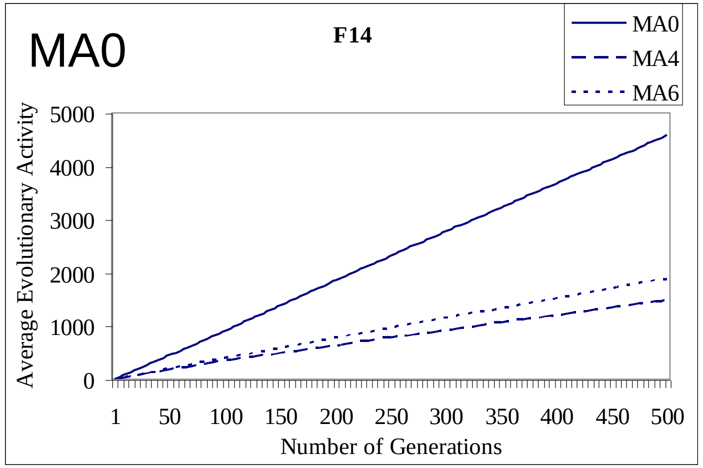
MMA 1.c

MA3 *MA3*









MA0 *MA0*

MMA 2 Experiments – Synergy Between Memes

label	type	avr. no. of evals.	st.dev.
F1	MMA	17,580	2,226
	MA1	92,256	0
F2	MMA	23,605,004	24,364,979
	MA3	8,455,507	3,803,504
F3	MMA	72,252	11,772
	MA3	82,769	16,512
F4	MMA	12,926,879	11,435,876
	MA0	9,494,844	10,332,574
F5	MMA	46,975	79,394
	MA3	11,619	2,293
F6	MMA	553,306	231,124
	MA1	525,398	262,055
F7	MMA	349,250	324,544
	MA1	167,799	60,577

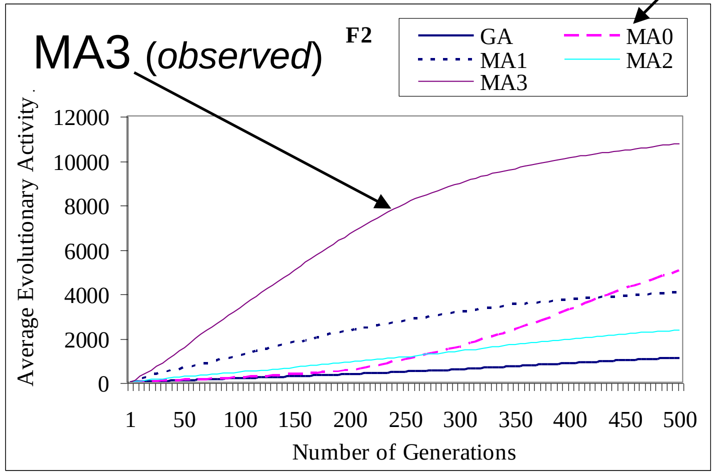
label	type	avr. no. of evals.	st.dev.
F8	MMA	5,215,787	9,658,230
	MA1	1,906,134	6,646,991
F9	MMA	43,871	12,193
	MA1	180,783	12,647
F10	MMA	3,100,515	4,565,736
	MA3	1,340,811	988,971
F11	MMA	17,580	2,226
	MA1	36,060	0
F12	MMA	31,297	14,961
	MA3	29,246	4,936
F13	MMA	7,667,352	2,832,376
	MA0	4,348,896	1,617,951
F14	MMA	3,674,932	2,623,300
	MA0	1,072,117	1,111,825

Second entry for each function indicates the best performing hill climbing under the generic MA framework where HC is applied after mutation

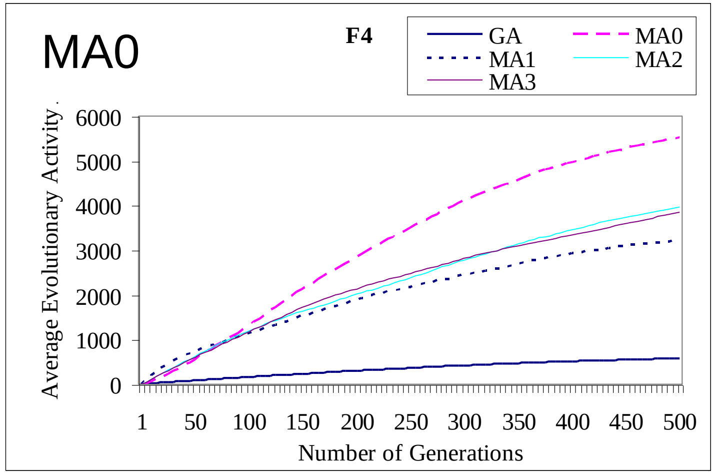
MMA 2.a

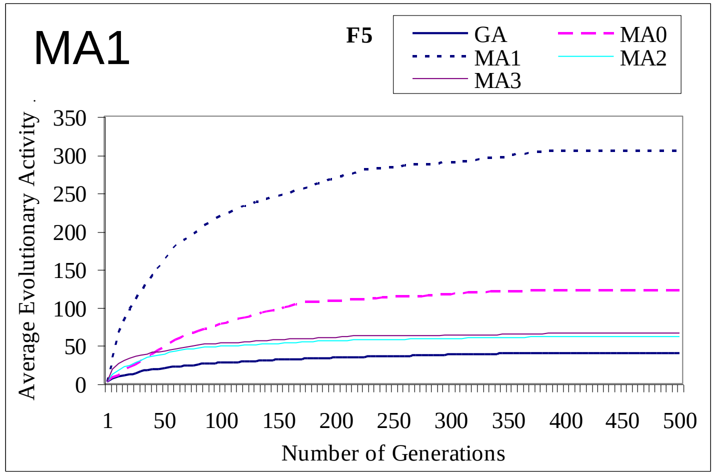
Memes are fixed

MA3 (expected)

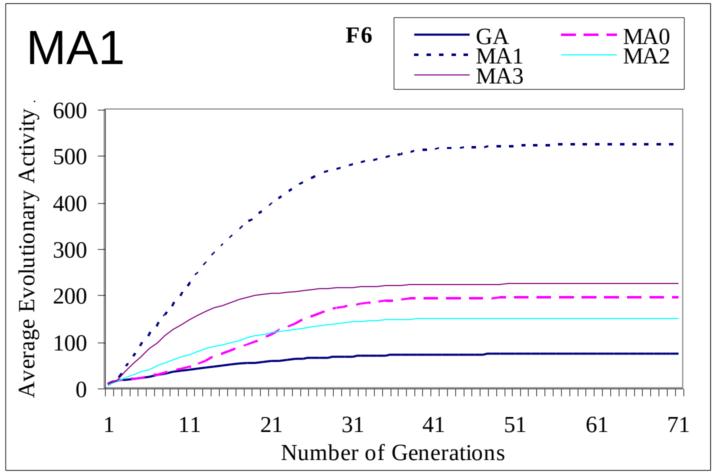


MA0



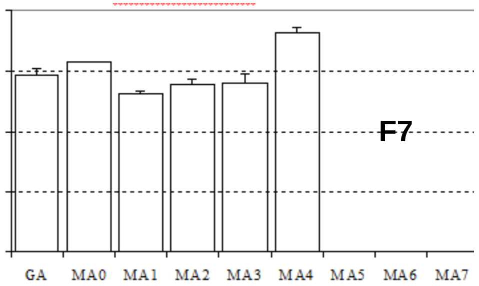


MA3

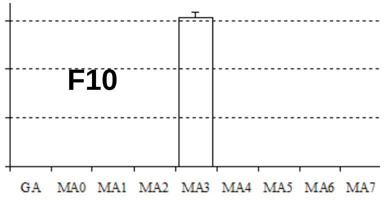


MA1

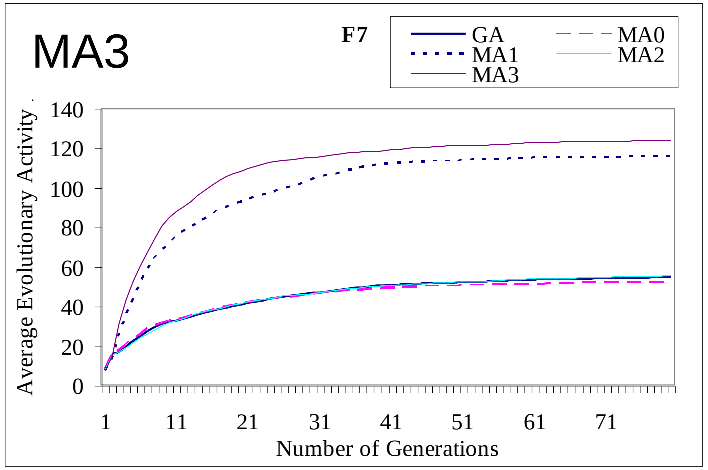
MMA 2.b

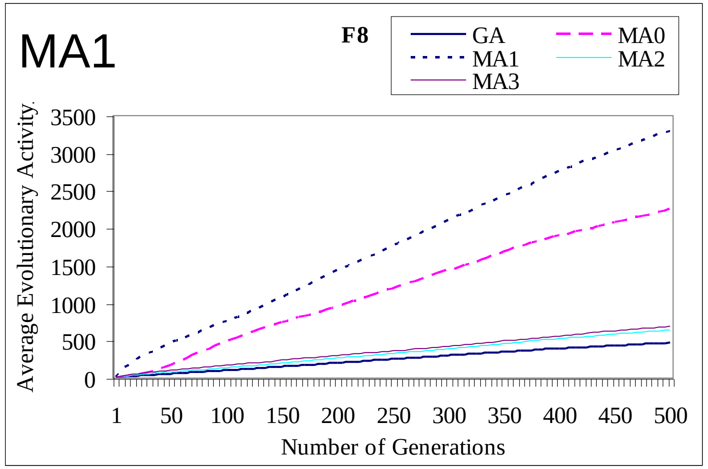


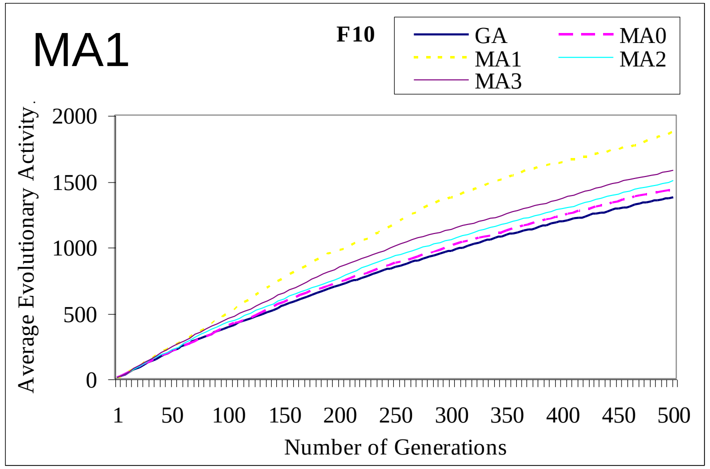
MA1

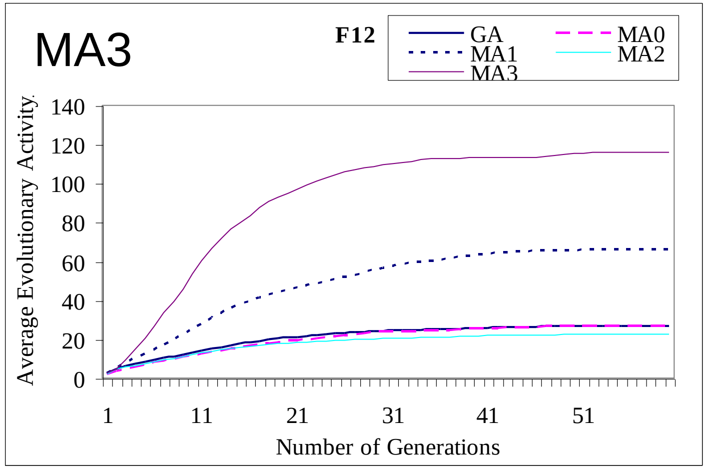


MA1







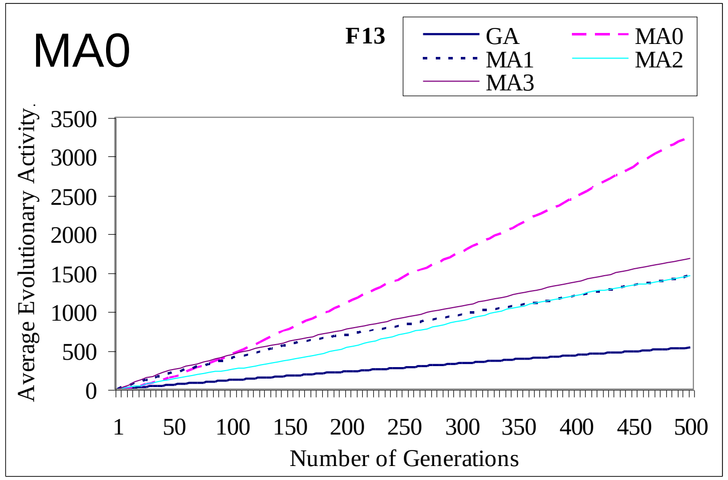


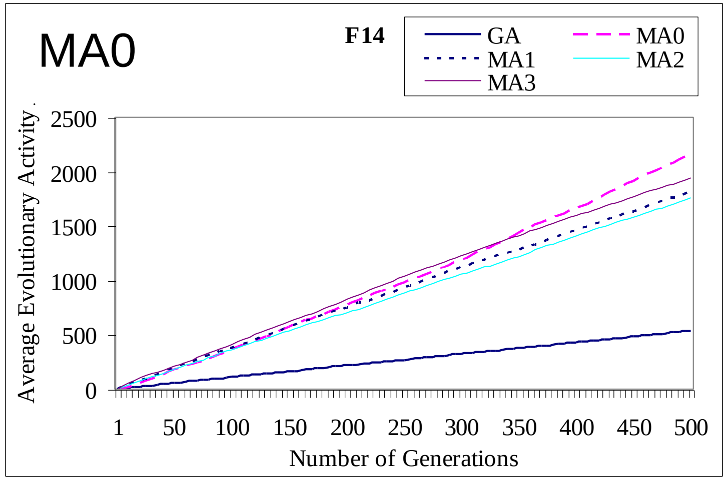
MA3 *MA3*

MMA 2.c

MA0

MA0





Results

- Average evolutionary activity plots show that the multimeme approach successfully identifies useful memes.
  - Even if there are useless operators in the system
- Any hill climber seems to attain the optimum fast for F1, F3 and F11 under any setting
- In all runs full success is achieved for all cases.

Results II

- MMA can identify the best meme or a meme that does not perform significantly better than the best meme for almost each benchmark function
- Comparing the experimental results obtained using MMA and MA with the best meme for each benchmark function indicate that MA with the best meme is superior based on the average number of evaluations, except for F1, F3, F9 and F11
  - Learning is time consuming hence there is a trade-off

Weaknesses of Evolutionary Algorithms

- Limited theoretical and mathematical analyses – this is a growing field of study
  - Schema theorem and building block hypothesis
  - Markov chain analysis
  - Chaos theory
  - Martingale theory (for convergence analysis)
  - Runtime analysis with respect various parameter settings (e.g., expected runtime analysis based on fitness levels/partitions, drift analysis)

  Per Kristian Lehre and Pietro S. Oliveto, GECCO 2021, [Runtime Analysis of Evolutionary Algorithms: Basic Introduction]
- Considered slow for online applications and for some large offline problems
  - Speedy hardware
  - Parallel/distributed processing

Overall Summary

- Genetic Algorithms represent a subclass of nature inspired Evolutionary Computation Techniques
  - GAs creates an initial population of individuals and performs the search by applying selection, crossover and mutation on individuals at each evolutionary cycle
  - GA components including the parameter settings require careful tailoring to the problem in hand
- Memetic Algorithms hybridise GAs with Hill Climbing
- There is a variety of benchmark functions used for performance comparison of search algorithms
- In general, a memetic algorithm performs better than a genetic algorithm
- Choice of meme (operators and their settings) along with the encoding influence the performance of a memetic algorithm

Overall Summary (cont.)

- Multimeme Algorithm can indeed learn how to choose an operator and relevant settings through the evolutionary process (co-evolution)
  - There is a trade-off as learning requires time and a memetic algorithm with a single setting could perform better
- If there is limited number of operators and settings MMA can be used. If the number of operators and settings are large then additional analyses and methods could be needed to reduce the number of options leading to an MMA with improved performance
- There is no single recipe for even choosing the set of memes while designing a memetic algorithm for solving a given problem.
- Synergy between memes/meme components could be possible

Q&A

Thank you.

Ender Özcan ender.ozcan@nottingham.ac.uk

University of Nottingham, School of Computer Science
Jubilee Campus, Wollaton Road, Nottingham
NG8 1BB, UK
http://www.nottingham.ac.uk/~pszeo/