Lecture L4: Real-Parameter EAs

- Difficulties with binary EAs
- Real-parameter GAs
  - Simulated binary crossover (SBX) and others
- Differential evolution (DE)
- Particle swarm optimization (PSO)
- Evolution Strategy and CMA-ES

Binary GAs in Continuous Search Space

- Binary GAs make the search space discrete
- Hamming cliffs: (10000)'s neighbor (01111)
  - Gray coding isn't the solution
- Arbitrary precision impossible due to fixed-length coding
- Search primarily along variable directions
- Search restricted with variable boundaries
- Not all Holland's schemata are important
  - (1****) more important than (****1)
- Redesign crossover which gives more importance to meaningful schemata in real space

Real-Parameter Genetic Algorithms

- Decision variables are coded directly, instead of using binary strings
- Recombination and mutation need structural changes
- Selection operator remains the same

Recombination

$(x_1 x_2 \ldots\ldots\ldots\ldots x_n)$

$(y_1 y_2 \ldots\ldots\ldots\ldots y_n)$

$\Rightarrow ?$

Mutation

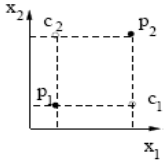
$(x_1 x_2 \ldots\ldots\ldots\ldots x_n) \Rightarrow ?$

- Simple exchanges are not adequate

Naive Recombination

- Crossing at boundaries do not constitute adequate search
  - Least significant digits are taken too seriously

15.345	3.569	$\rightarrow$	11.142	3.587
21.142	5.687		25.345	5.669



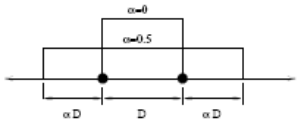
- Two Remedies:
  - Parent values (variable-wise) need to be blended to each other
  - Vector-wise recombination

Variable-wise Blending of Parents

- Use a probability distribution to create child
- Different implementations since 1991:
  - Blend crossover (BLX-α), 1991
  - Simulated binary crossover (SBX-β), 1995
  - Fuzzy recombination (FR-d), 1995
- Main feature: Difference between parents used to create children
  - Provides self-adaptive property

Blend Crossover (BLX-α)

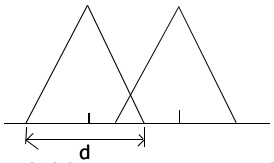
- Uniform probability distribution within a bound controlled by α



- Diversity in children proportional to that in parents
- Too wide a search, if parents are distant

Fuzzy Recombination (FR-d)

- Triangular probability distribution controlled by *d*



- Diversity in children proportional to that in parents

☐ All past operators seem to be ad-hoc

Making of the Simulated Binary Crossover (SBX)

- What are schemata in real space?
  - Goldberg (1990) virtual alphabets and Eshelman and Schaffer (1992) interval schemata
- Interval schema (a,b) represents all values in the range
- Contiguous regions are interval schemata (virtual alphabets)
- Radcliffe (1991) suggested a number of criterion for a good crossover design
- Ergodicity criterion: Ability to create any point in the search space
- Preservation: All common schemata among parents are preserved

Properties of a Binary Single-point Crossover Operator

- What are common interval schemata?
  - A thorough study is in order
  - We assume only integers here
- Eshelman and Schaffer (1992) counted (p+1)(b-a+1-p) interval schemata representing a point p in [a,b]
- Number of common interval schemata (C) between two points x_l and x_h is $(x_l+1)(N-x_h)$ (N is total number of points)

Properties of a Binary Single-point Crossover Operator (cont.)

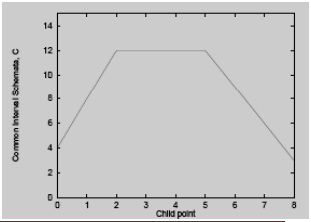
- An example: x in [0,8], p_1=2, p_2=5
- 12 common interval schemata

Number of interval schemata						
Parent point at 2			Parent point at 5			
(2,2)			(5,5)			
(1,2)	(2,3)		(4,5)	(5,6)		
(0,2)	(1,3)	(2,4)	(3,5)	(4,6)	(5,7)	
(0,3)	(1,4)	(2,5)	(2,5)	(3,6)	(4,7)	(5,8)
(0,4)	(1,5)	(2,6)	(1,5)	(2,6)	(3,7)	(4,8)
(0,5)	(1,6)	(2,7)	(0,5)	(1,6)	(2,7)	(3,8)
(0,6)	(1,7)	(2,8)	(0,6)	(1,7)	(2,8)	
(0,7)	(1,8)		(0,7)	(1,8)		
(0,8)			(0,8)			

Properties of a Binary Single-point Crossover Operator (cont.)

- Number of common interval schemata as a function of child (x)
- Ergodicity criterion not satisfied
- Allow probability of creating a child proportion to C it shares with parents.
- Probability away from [p_1,p_2] should reduce
- BLX-α does not do this

$$C = \begin{cases} (x+1)(9-5) = 4(x+1) & \text{if } 0 \le x < 2 \\ (2+1)(9-5) = 12 & \text{if } 2 \le x \le 5 \\ (2+1)(9-x) = 3(9-x) & \text{if } 5 < x \le 8 \end{cases}$$



Motivation for the SBX Operator

- Simulate processing in a binary crossover, say single-point crossover
- *Gedanken* experiment:
  - Two parent solutions in real number
  - Code them in l-bit strings
  - Use single-point crossover in all l-1 places and find children strings
  - In each case map strings back to real numbers as children
  - Observe the relationship between parents and children
  - Use this relationship to directly recombine parents to form children

Motivation for the SBX Operator

- Simulate processing in a binary crossover, say single-point crossover
- *Gedanken* experiment:
  - Two parent solutions in real number
  - Code them in l-bit strings
  - Use single-point crossover in all l-1 places and find children strings
  - In each case map strings back to real numbers as children
  - Observe the relationship between parents and children
  - Use this relationship to directly recombine parents to form children

Properties of Binary Crossover

- To make crossovers independent of parents, define a spread factor, β
- Define probability distribution as a function of β
- Two observations:
  - Mean decoded value of parents is same as that of children
  - Child strings crossed at the same place produce the same parent strings

$$\beta = \left|\frac{c_1 - c_2}{p_1 - p_2}\right|$$

B_1				A_1			DV
1	0	1	0	1	0	1	85
0	1	1	0	0	1	1	51
B_2				A_2			68
B_1				A_2			DV
1	0	1	0	0	1	1	83
0	1	1	0	1	0	1	53
B_2				A_1			68

$\Longrightarrow$

Lessons Learned

- Three cases:
  - All crossovers with β<1 are contracting
  - All crossovers with β=1 are stationary
  - All crossovers with β<1 are expanding
- For each contracting crossover with a $\beta = \beta_1$ there exists a expanding crossover with $\beta = 1/\beta_1$

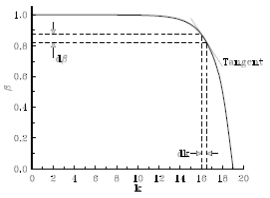
Analyze Single-Point Crossover

- Two parent strings (a_i, b_i in {0,1})
- Two child strings with a cross-site k
- Calculate spread factor
- $a_i = 1$, $b_i = 0$

$$x_1 = \sum_{i=0}^{\ell-1} a_i 2^i \quad x_2 = \sum_{i=0}^{\ell-1} b_i 2^i$$

$$y_1 = \sum_{i=0}^{k-1} b_i 2^i + \sum_{i=k}^{\ell-1} a_i 2^i \quad y_2 = \sum_{i=0}^{k-1} a_i 2^i + \sum_{i=k}^{\ell-1} b_i 2^i$$

$$\beta(k) = \frac{1 - \zeta(k)}{1 + \zeta(k)}$$



Analyzing Single-Point Crossover (cont.)

- Compute desired probability for dβ->0 $\mathcal{C}(\beta) = \frac{1/\left|\frac{d\beta}{dk}\right|_\beta}{\int_0^1 \left(1/\left|\frac{d\beta}{dk}\right|_\beta\right) d\beta}$
- Probability of creating points near to the parents is more
- Keeping probabilities identical in contracting and expanding cases

$$\mathcal{E}(\beta)\,[(\beta + d\beta) - \beta] = \mathcal{C}(\frac{1}{\beta})\left[\frac{1}{\beta} - \frac{1}{\beta + d\beta}\right]$$

- Relationship between expanding and contracting crossovers:

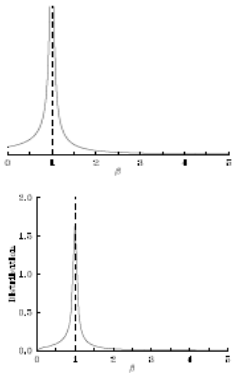
$$\mathcal{E}(\beta) = \frac{1}{\beta^2}\mathcal{C}(\frac{1}{\beta})$$

Analyzing Single-Point Crossover (cont.)

- For the special case

$\mathcal{C}(\beta) = \frac{\kappa}{1-\beta}$ $\mathcal{E}(\beta) = \frac{\kappa}{\beta(\beta-1)}$

- Once again, probability of creating points closer to parents is more
- Experiments: l=15, all string pairs and all 14 crosses (about 7.5 billion)

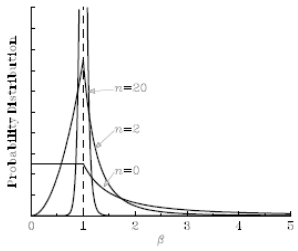


Simulated Binary Crossover (SBX-β)

- Probability is large near the parents
- Extent is controlled by β
- Diversity in children proportional to that in parents
- Obeys *interval-schema* processing

$\mathcal{C}(\beta) = 0.5(n+1)\beta^\eta$

$\mathcal{E}(\beta) = 0.5(n+1)\frac{1}{\beta^{n+2}}$



SBX Procedure

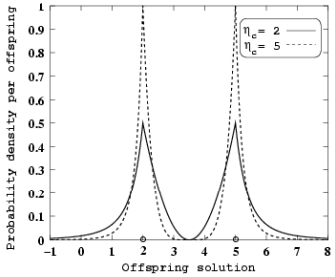
- Step 1: Choose a random number $u \in [0,1]$.
- Step 2: Calculate β_q:

$$\beta_q = \begin{cases} (2u)^{\frac{1}{\eta_c+1}}, & \text{if } u \le 0.5 \\ \left(\frac{1}{2(1-u)}\right)^{\frac{1}{\eta_c+1}}, & \text{otherwise} \end{cases}$$

- Step 3: Compute two offspring:

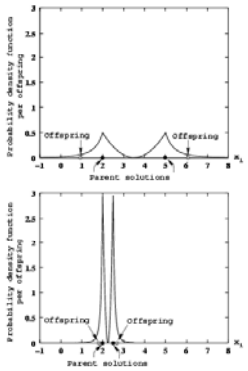
$$c_1 = 0.5\left(\left(1+\beta_q\right)p_1 + \left(1-\beta_q\right)p_2\right)$$

$$c_2 = 0.5\left(\left(1-\beta_q\right)p_1 + \left(1+\beta_q\right)p_2\right)$$



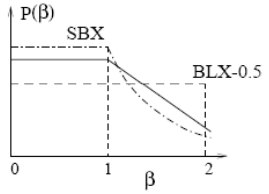
Properties of SBX Operator

- If parents are distant, distant offspring are likely
- If parents are close, offspring are close to parents
- Self-adaptive property (discuss more in L8)



Interval Schema Processing in SBX

- Consider SBX with η=0 and BLX-0.5
  - SBX respects interval schema-processing better than BLX-0.5
- SBX with non-zero η respects similar schema that represents contiguous points with varying importance
- For multiple variables, each variable is SBX-ed with probability 0.5



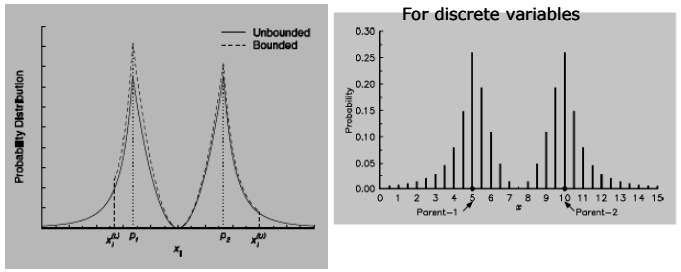
Simulation Results with SBX

- V-function: Minimize |x-0.5| in 0≤x≤1
- Hamming cliff is a part of the problem
- 2,147 in $(0.5\text{-}10^{-6}, 0.5\text{+}10^{-6})$ with l=30

Operator	Range	N	η/ℓ	Perf. of 100 runs Suc.	Fail	Trials
1-pt	(0,1)	50	30	23	77	834.8
SBX	(0,1)	50	0	100	0	929.5
			2	100	0	748.5
			5	100	0	818.5
		100	0	100	0	1739.0
			2	100	0	1396.0
			5	100	0	1321.0
BLX-0.5	(0,1)	50		100	0	746.0
		100		100	0	1368.0
SBX	(0.9,1)	50	0	100	0	1279.5
			2	100	0	2018.5
BLX-0.5	(0.9,1)	50		24	76	2604.2
SBX	(0.9999,1)	50	0	100	0	1790.0
			2	100	0	4318.0
BLX-0.5	(0.9999,1)	50		0	61(39)	

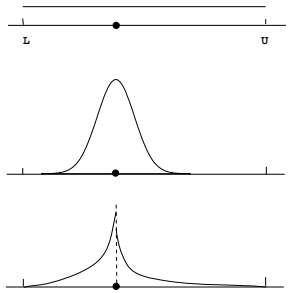
Variations of SBX

- For bounded and discrete variables



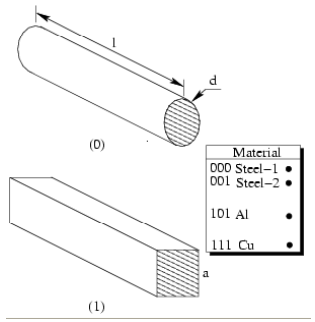
Real-Parameter Mutation Operators

- Idea: Create a neighboring solution
- Random mutation
- Normally distributed mutation
- Non-uniform mutation
- Extensions to bounded and discrete cases exist



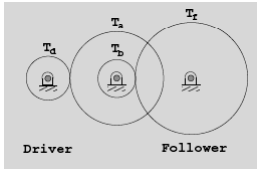
Mixed-Integer Programming

- Level-wise application in classical methods
  - Multiple simulations
- No need for level-wise optimization with EAs
- A mixed representation possible:
  (1) 14 23.457 (101)
- Recombination and mutation can be used variable-wise
- GeneAS (Deb and Goel, 1996)



Gear-Train Design

- Objective: Obtain a gear ratio 6.931:1
- Gears must have 12 to 60 teeth (integers)
- GeneAS1: Direct integer-valued (12,60)
- GeneAS2: Binary-coded (12-75), add T_i ≤60



GeneAS1	(19,16,49,43)	$2.7(10^{-12})$
GeneAS2	(17,14,33,50)	$1.4(10^{-09})$
AL	(13,15,33,41)	$2.1(10^{-08})$

Spring Design

- Minimize weight (Wt) of spring
- 8 nonlinear constraints
- d discrete, D real, N integer
- A typical solution:
  [0.148 1.563 (01101)]

g1 ≥ 0

x	GA	AL	BB
N	9	7	10
d	0.283	0.283	0.283
D	1.226	1.329	1.180
g1	713.5	-10172	5500
Wt	2.665	2.364	2.798

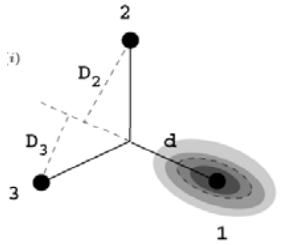
Vector-Wise Recombination Operators

- Variable-wise recombination cannot capture nonlinear interactions
- Recombine parents as vectors
  - Parent-centric recombination (PCX)
  - Unimodal normally-distributed crossover (UNDX)
  - Simplex crossover (SPX)
- Difference between parents is used to create offspring solutions
- DE, PSO, CMA-ES, and others

Parent Centric Recombination (PCX)

- μ parents create a offspring
- Each parent has its turn
- $e^{(i)}$ orthonormal bases spanning the subspace perpendicular to $d^{(p)}$
- w_ζ and w_η are user-defined parameters, controlling extent of search

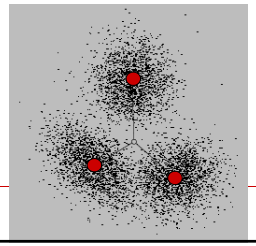
$$\vec{y} = \vec{x}_p + w_\zeta |\vec{d}^{(p)}| + \sum_{i=1,\ i\neq p}^{\mu} w_\eta \bar{D} \bar{e}^{(i)}$$

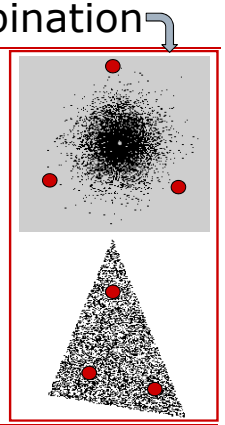


Mean-Centric Recombination

- UNDX (Ono and Kobayashi, 1997): $O(\mu^2)$ per offspring
- SPX (Higuchi et al., 2000): $O(\mu)$ per offspring

Compare with PCX (parent-centric)





Generalized Generation Gap (G3) Model (Steady-state approach)

1. Select the best parent and μ-1 other parents randomly
2. Generate λ offspring using a recombination scheme
3. Choose two parents at random from the population
4. Form a combination of two parents and λ offspring, choose best two solutions and replace the chosen two parents

- Parametric studies with λ and N

Three Test Problems

- 20-variable problems, x_i = *[-10,-5]* for all *i*
- 50 runs performed
- $F^t_{elp}=0.5(10^6)$, $F^t_{sch}=1(10^6)$, $F^t_{ros}=1(10^6)$

$$F_{\text{elp}} = \sum_{i=1}^{n} i x_i^2 \quad \text{(Ellipsoidal)}$$

$$F_{\text{sch}} = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2 \quad \text{(Schwefel)}$$

$$F_{\text{ros}} = \sum_{i=1}^{n-1} \left(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2 \right) \quad \text{(Rosenbrock)}$$

Quasi-Newton Method

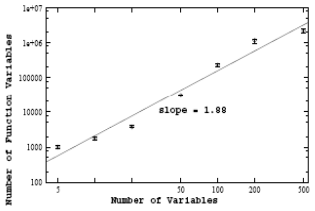
- Accuracy obtained by G3+PCX is 10^{-20}

Func.	FE	Best	Median	Worst
F_{elp}	6,000	$8.819(10^{-24})$	$9.718(10^{-24})$	$2.226(10^{-23})$
F_{sch}	15,000	$4.118(10^{-12})$	$1.021(10^{-10})$	$7.422(10^{-9})$
F_{ros}	15,000	$6.077(10^{-17})$	$4.046(10^{-10})$	3.987
F_{elp}	8,000	$5.994(10^{-24})$	$1.038(10^{-23})$	$2.226(10^{-23})$
F_{sch}	18,000	$4.118(10^{-12})$	$4.132(10^{-11})$	$7.422(10^{-9})$
F_{ros}	26,000	$6.077(10^{-17})$	$4.046(10^{-10})$	3.987

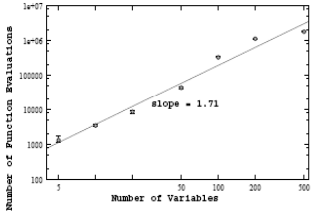
Scalability Study

- Accuracy 10^{-10} is set

Ellipsoidal

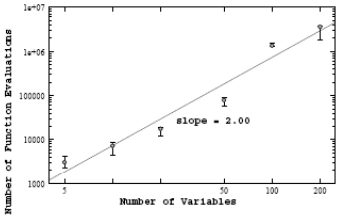


Schwefel



Scalability Study (cont.)

Rosenbrock



- All polynomial complexity $O(n^{1.7})$ to $O(n^2)$ similar to those reported by CMA-ES approach (Hansen and Ostermeier, 2001)

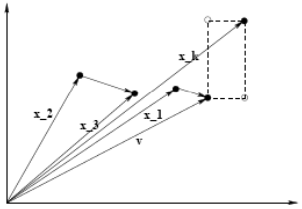
Differential Evolution (DE)

1. Start with a pool of random solutions
2. Create a child v
3. x_k and v are recombined with *p*

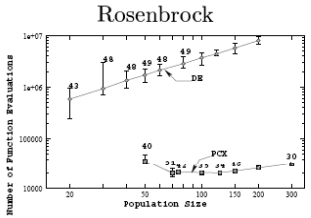
- Difference of parents in creating a child is important
- A number of modifications exist

$$\mathbf{v} = \mathbf{x}^{(1)} + \lambda(\mathbf{x}^{(2)} - \mathbf{x}^{(3)})$$

$$y_i = \begin{cases} v_i, & \text{with a prob. } p \\ \mathbf{x}_i^{(k)}, & \text{else} \end{cases}$$



DE Results



	Best	Median	Worst
	F_{elp}		
DE	9,660	12,033	20,881
G3	5,826	6,800	7,728
	F_{sch}		
DE	102,000	119,170	185,590
G3	13,988	15,602	17,188
	F_{ros}		
DE	243,800	587,920	942,040
G3	16,508	21,452	25,520

Particle Swarm Optimization (PSO)

- Kennedy and Eberhart, 1995
- Particles fly through the search space
- Velocity dynamically adjusted
- $x_i = x_i + v_i$
- $v_i = v_i + c_1 rnd()(p_{i,best} - x_i) + c_2 rnd()(p_g - x_i)$
- p_i: best position of i-th particle
- p_g: position of best particle so far
  - 1st term: momentum part (history)
  - 2nd term: cognitive part (private thinking)
  - 3rd term: social part (collaboration)

PSO Algorithm

1. Initialize a random population
2. For each particle
   - Evaluate and update p_{best}
3. Identify p_g
4. Compute velocity and position for each particle
5. Loop till termination

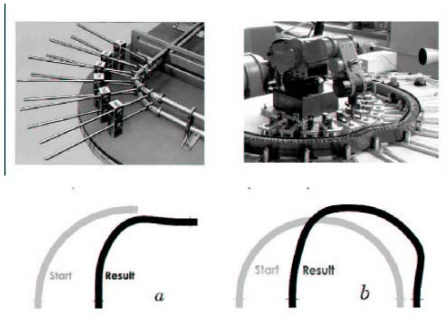
- Modifications exist
- Constrained PSO, Hybrid PSO, multi-objective PSO, etc.

Evolution Strategy (ES)

- Rechenberg and Schwefel, 1975 at TUB, Germany
- (1+1)-ES, one parent, one offspring
- Offspring is created using normal distribution around parent
- Better of offspring and parent is retained
- Optimum mutation strength (variance of normal distribution)
- Successful in many experimental problems

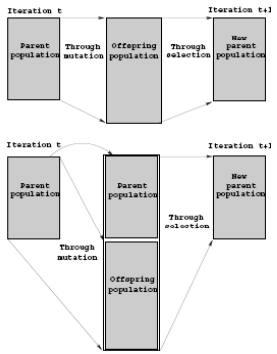
Pipe-Bending Experiments

Rechenberg, 1965



Multi-Parent ESs

- (μ, λ)-ES and $(\mu + \lambda)$-ES
  - μ parents created at random
  - λ (usually $> \mu$) using normal distr. around parents
  - Best μ of (λ or $\mu+\lambda$) are selected



Important Results on ESs

- Optimal ES parameter setting for faster growth rate
  - Sphere model
  - Corridor model
- 1/5th success rule
  - Increase mutation strength, if more than 1/5th of offspring are successful
  - Decrease, otherwise
  - Theory based on sphere model, but used in other problems
- Early studies lacked recombination

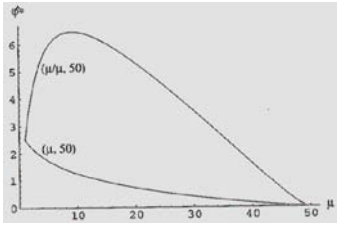
Recombinative ESs

- Intermediate recombination
  - Mean of ρ parents are mutated
- Discrete recombination
  - Uniform crossover of ρ parents

Parent 1:	$x_1^{(1)}$	$x_2^{(1)}$	$x_3^{(1)}$	$x_4^{(1)}$	$x_5^{(1)}$	$x_6^{(1)}$
Parent 2:	$x_1^{(2)}$	$x_2^{(2)}$	$x_3^{(2)}$	$x_4^{(2)}$	$x_5^{(2)}$	$x_6^{(2)}$
Parent 3:	$x_1^{(3)}$	$x_2^{(3)}$	$x_3^{(3)}$	$x_4^{(3)}$	$x_5^{(3)}$	$x_6^{(3)}$
Parent 4:	$x_1^{(4)}$	$x_2^{(4)}$	$x_3^{(4)}$	$x_4^{(4)}$	$x_5^{(4)}$	$x_6^{(4)}$
Recombinant:	$x_1^{(2)}$	$x_2^{(3)}$	$x_3^{(4)}$	$x_4^{(2)}$	$x_5^{(4)}$	$x_6^{(3)}$

Recombinative ESs (cont.)

- Beyer (2002) shows improved performance
- Recombination helps improve performance
- Benefit of recombination lies in genetic repair of genes caused by mutation

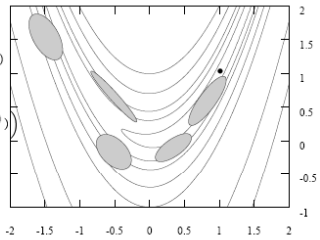


Correlated ES

- n mutation strengths σ and $\binom{n}{2}$ covariances (rotation angles α) are evolved

$$\sigma_i^{(t+1)} = \sigma_i^{(t)} \exp(\tau' N(0,1) + \tau N_i(0,1))$$
$$\alpha_j^{(t+1)} = \alpha_j^{(t)} + \beta_\alpha N_j(0,1)$$
$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \mathbf{N}\left(\mathbf{0}, C(\sigma^{(t+1)}, \alpha^{(t+1)})\right)$$

- Update is done with good parents
- ES learns the beneficial directions towards optimum



Correlated ES on Three Earlier Test Problems

EA	F_{elp} Best	F_{elp} Median	F_{elp} Worst	F_{sch} Best	F_{sch} Median	F_{sch} Worst
$(1,10)$-ES	28,030	40,850	87,070	72,330	105,630	212,870
$(15,100)$-ES	83,200	108,400	135,900	173,100	217,200	269,500
G3+PCX	5,826	6,800	7,728	13,988	15,602	17,188
	F_{ros}					
$(1,10)$-ES	591,400	803,800	997,500			
$(15,100)$-ES	663,760	837,840	936,930			
G3+PCX	16,508	21,452	25,520			

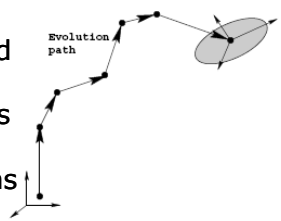
Performance not as good as G3+PCX

CMA-ES
(Hansen & Ostermeier, 1996)

- Mix of evolution history (P) and currently best pop. (Z)

$C^{t+1}=(1-b)C^t+b(aP^{t+1}+(1-a)Z^{t+1})$

- Sucessful steps are recorded
- They are analyzed to find uncorrelated basis directions and strengths
- Required $O(n^3)$ computations to solve an eigenvalue problem
- Rotation invariant



CMA-ES On Three Test Problems

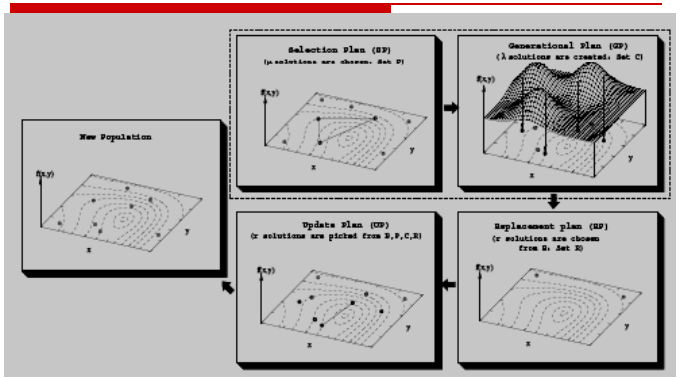
EA	F_{elp} Best	F_{elp} Median	F_{elp} Worst	F_{sch} Best	F_{sch} Median	F_{sch} Worst
CMA-ES	8,064	8,472	8,868	15,096	15,672	16,464
DE	9,660	12,033	20,881	102,000	119,170	185,590
G3+PCX	5,826	6,800	7,728	13,988	15,602	17,188
	F_{ros}					
CMA-ES	29,208	33,048	41,076			
DE	243,800	587,920	942,040			
G3+PCX	16,508	21,452	25,520			

- Accuracy $1X10^{-20}$
- Comparable to G3+PCX

Population-Based Optimization Algorithm-Generator

- Four functionally different Plans
  - Selection plan (SP): choose μ solutions from B to create P
  - Generation plan (GP): create λ solutions (C) using P
  - Replacement plan (RP): choose r solutions (R) from B
  - Update plan (UP): update B by replacing R (r solutions) from (P,C,R)

A Sketch of an Iteration



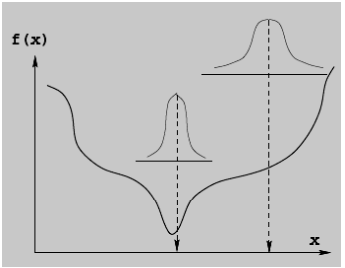
Designing an Optimization Procedure

1. Choose SP, GP, RP and UP
2. Perform a parametric study on a set of test problems
   - N, μ, λ, r

- Most classical and non-classical optimization algorithms explained
- Interestingly, PSO, DE, ES, EP can also be explained
- Possibility for a unified optimization approach

Self-Adaptive Evolutionary Algorithms for Real-Parameter Opt.

- Why Self-adaptation is important?
- Biological model: Repair enzymes, mutator genes
- No deterministic (a priori) control
- Strategy parameters evolve with fitness information
- Applications: Dynamic environment, arbitrary precision, etc.



Why Self-Adaptation is Important in Optimization?

- Knowledge of lower or upper bounds are usually not known
- Coded EAs (Binary GAs) need such information
- Optimum may be desired with arbitrary precision
  - Fixed-length coding schemes cannot achieve this
  - Variable-length GAs (ARGOT or Dynamic parameter mapping GAs) suggested
  - But, subjective to decisions of switch
- Objective function and constraints may change with time
  - Diversity is usually lost for the new problem

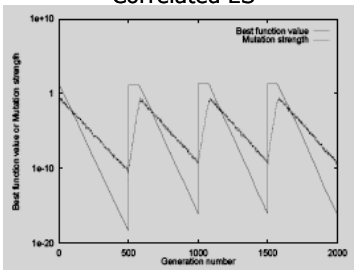
Self-Adaptive Evolution Strategy

- Meta-ES (Herdy, 1992)
  - Top-level optimizes for strategy parameters
  - Bottom-level optimizes for object parameters
- Adaptation of Covariance matrix (CMA) (Hansen and Ostermeier, 1996):
  - Population history to calculate variance and covariance among object parameters
- Explicit Strategy Parameters (Rechenberg, 1973; Schwefel, 1987):
  - Strategy parameters are evolved with object parameters

Time-varying Sphere Model

- Sphere model:
  $f(x) = |x - x^*|^2$
- Optimum x^* is shifted after every 500 generations
- (15, 100)-ES, no recombination
- Simple model of a dynamic environment

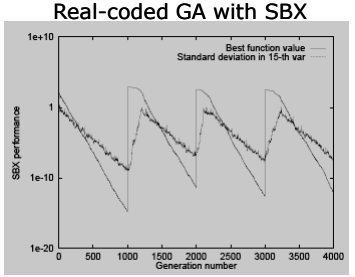
Correlated ES



Time-varying Sphere Model (cont.)

- Real-coded GAs with SBX applied on time-varying sphere model
- Optimum shifted every 1,000 generations for a stiffer test
- Similar performance to correlated SA-ES

Real-coded GA with SBX



Conclusions

- For continuous variables, use real-coded EAs
- Recombination and mutation are probability based
- Mixed-integer programming made possible and effective
- Self-adaptive EAs are possible and more robust

Demonstrating GA Codes

- http://www.iitk.ac.in/kangal/soft.htm
- sga.c: Binary genetic algorithms
- rga.c: Real-parameter genetic algorithms