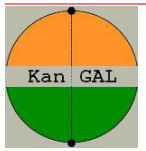
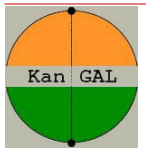
Lectures 10-12:
GP, DE, PSO, Self-Adaptation & Others

Kalyanmoy Deb
Kanpur Genetic Algorithm Laboratory (KanGAL)
Department of Mechanical Engineering
Indian Institute of Technology Kanpur
Kanpur, PIN 208016, INDIA
deb@iitk.ac.in
http://www.iitk.ac.in/kangal/deb.htm



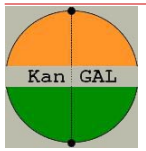
Overview of Lecture 10

- Interactive EMO by Shamik Chaudhuri
- Genetic programming (GP)
  - Procedure
  - Applications
- Movies by John Koza



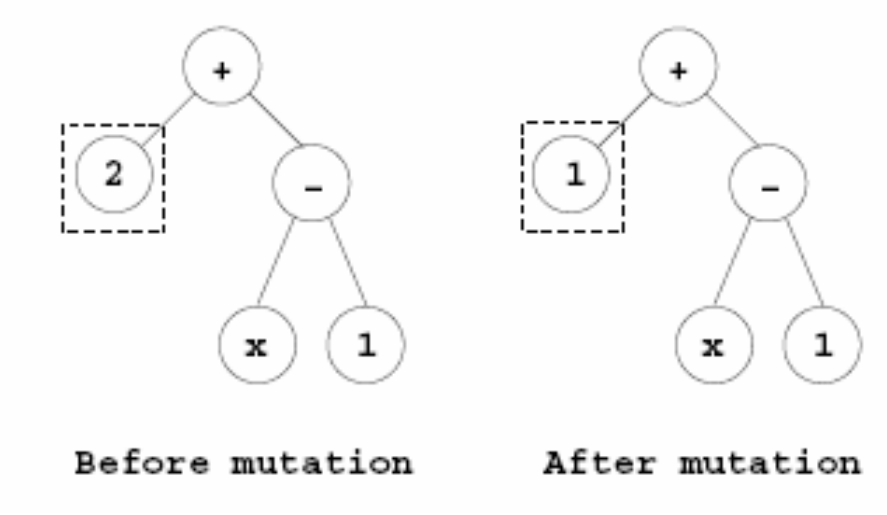
Genetic Programming (GP)

- Programs or Parse-Tress are evolved
- Applicable to numerous learning tasks, automated program generation etc.
- Koza and his collaborators, 1990 onwards
- Initial random population of programs
- Function Set: (+, -, *, /, sin, exp, OR, If, etc.)
- Terminal Set: (2.3, D0, Food-Ahead, etc.)
- Evaluate according to success of executing a task in m/n cases



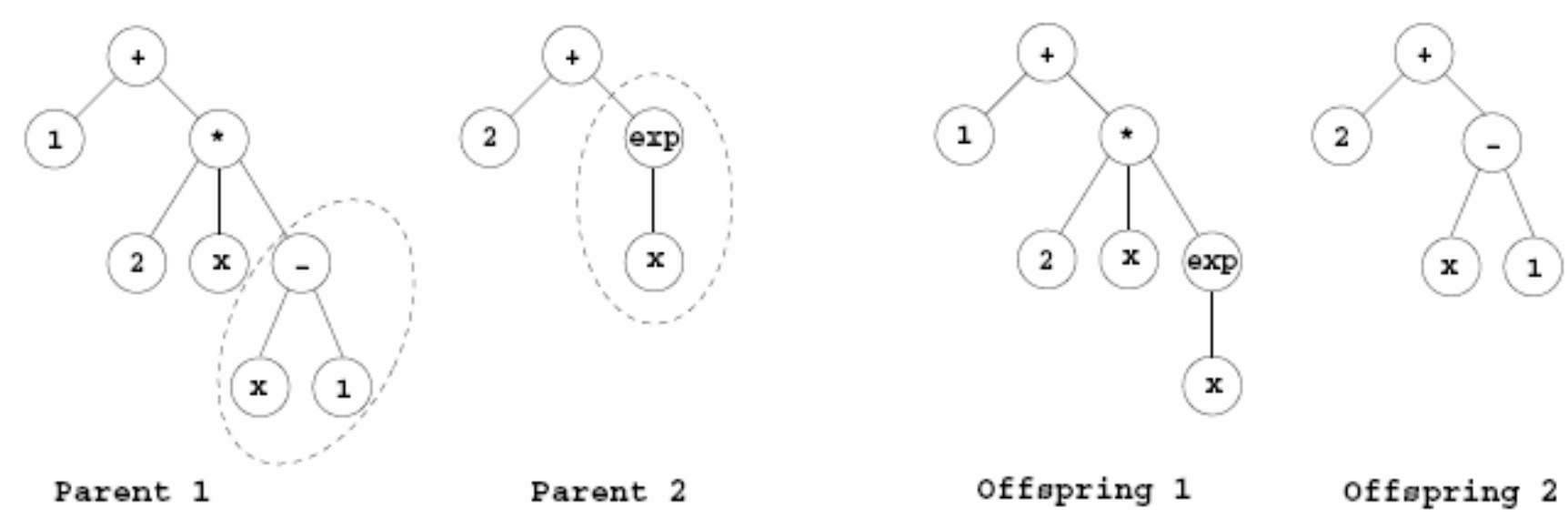
GP Operators

- Random initialization of programs (C++, LISP)
- Recombination and mutation of programs
- Editing, encapsulation, permutation, etc



GP Operators (cont.)

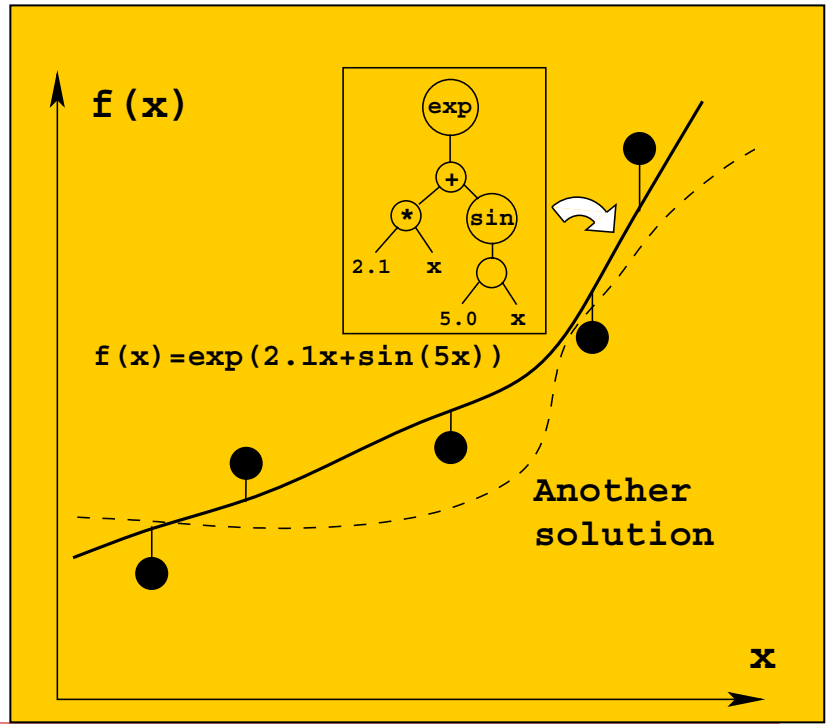
- Recombination: exchange of partial trees



- Rest is identical to a GA
- Automatically defined functions (ADFs)

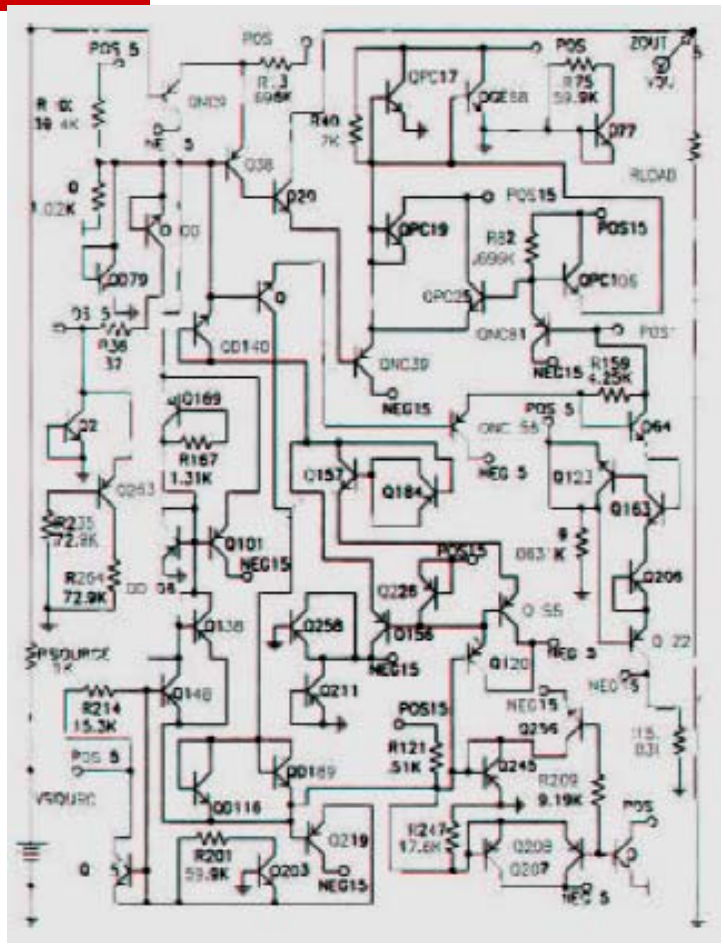
GP on Regression

- No functional structure is known
- Assume F=(exp, x, x^2,sin,log,+,-,*,/)
- Assume $T=(n_1,n_2,\ldots,n_k)$
- GP Goal:
  - Minimize Error in supplied points



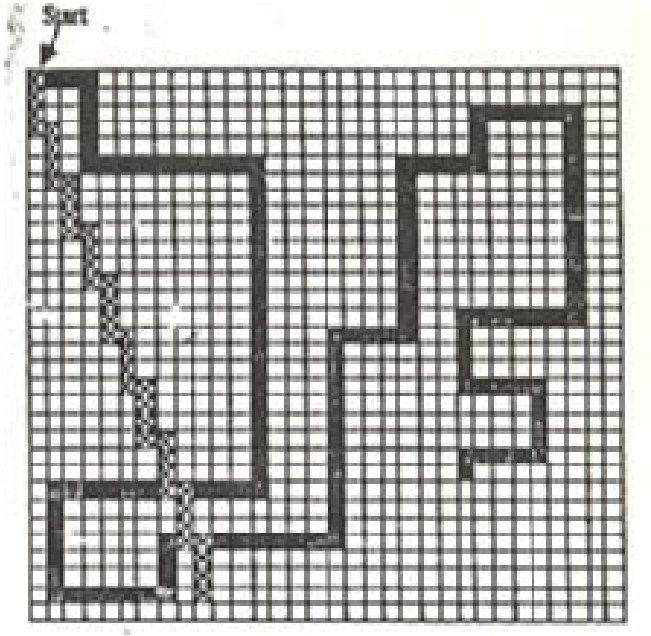
GP for Electronic Circuit Design

- Square-root of a number
- Many human-competitive designs are patented
- Wide variety of other applications



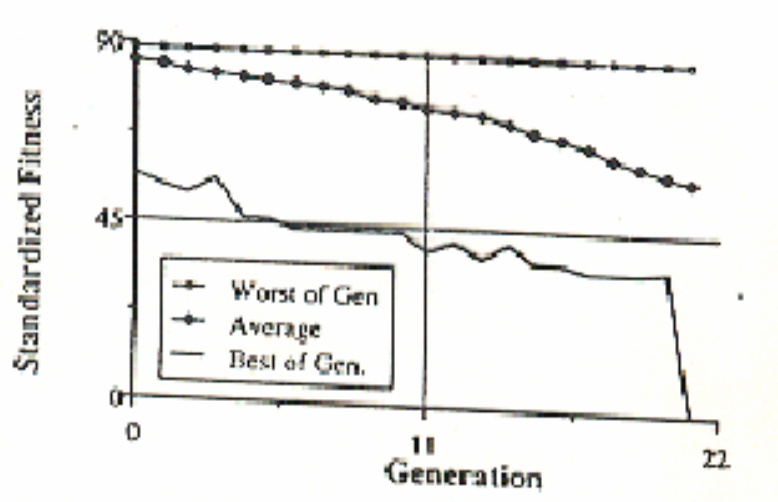
GP for Artificial Ant

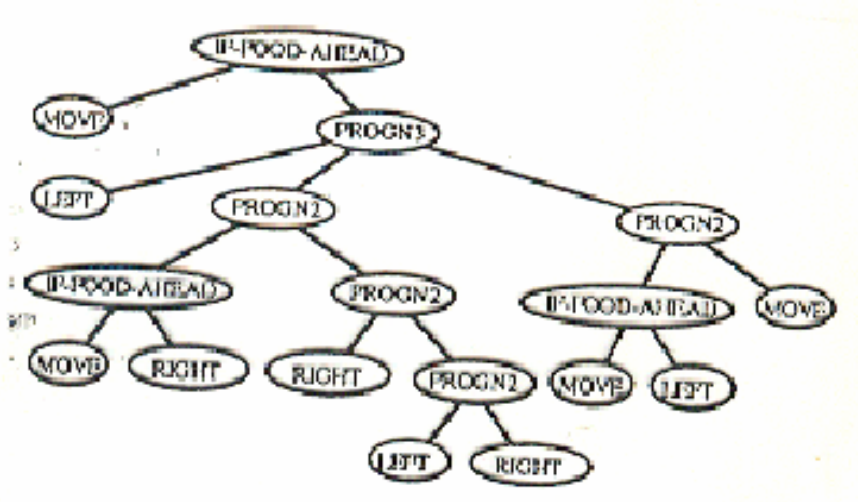
- Control an ant so that it finds all 89 pieces of food on a trail
- Terminal: (LEFT, RIGHT, MOVE)
- Function: (IF-FOOD-AHEAD, PROGN2, PROGN3)
- Fitness: # of good pieces picked within 400 operations
- 500 population size
- Initial best solution (5 hits)



Artificial Ant (cont.)

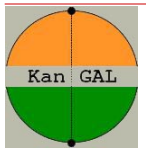
- Generation 21: 89 hits
- Finds a successful solution





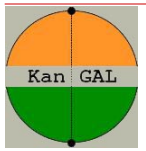
Conclusions

- Genetic programming is an application of GAs on programs
- Useful in learning problems
  - Regression without function structure
  - PDE solvers, differentiation
  - AI problems
  - Discovery of designs, knowledge, strategies
  - Control problems
- Theory of GP started (Poli and Langdon, 2003)
- Movie show



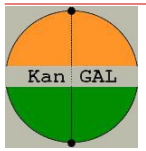
Overview of Lecture 11

- Vector-wise recombination
- Evolution Strategy (ES)
- Differential Evolution (DE)
- Particle Swarm Optimization (PSO)
- Self-adaptive EAs
- Scheduling EAs



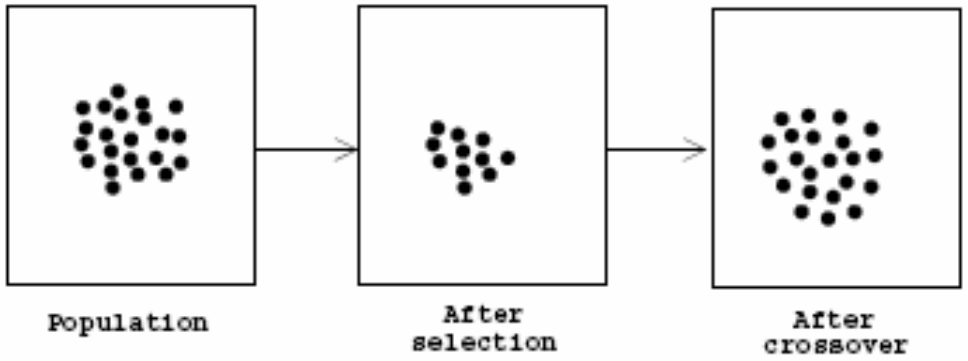
Vector-Wise Recombination Operators

- Variable-wise recombination cannot capture nonlinear interactions
- Recombine parents as vectors
  - Parent-centric recombination (PCX)
  - Unimodal normally-distributed crossover (UNDX)
  - Simplex crossover (SPX)
- Difference in parents is used to create offspring solutions



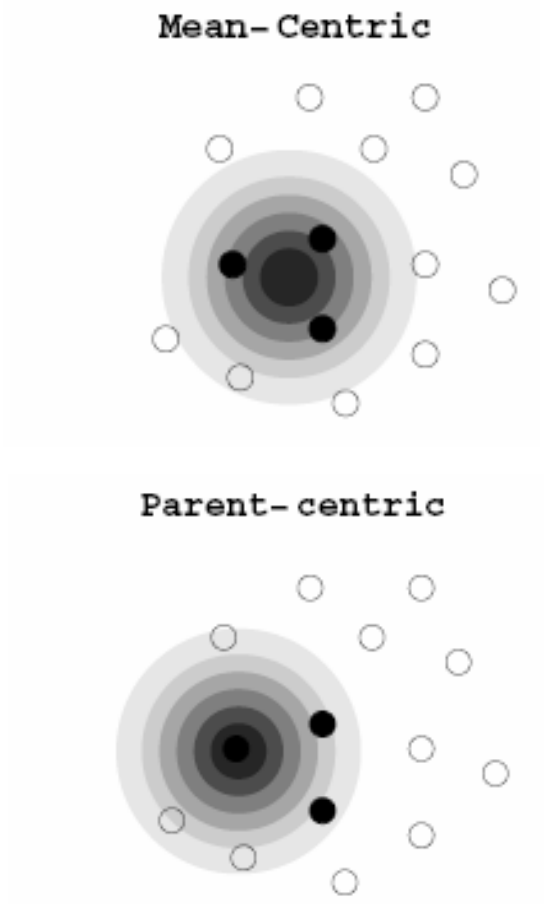
Essence in Real-Parameter EAs

- A standard zero-mean Gaussian-like mutation operator
- Main task is to design a recombination (blend) operator
- Beyer and Deb (2001) suggested two properties
  - Population mean is invariant before and after recombination
  - Population variance may increase after recombination



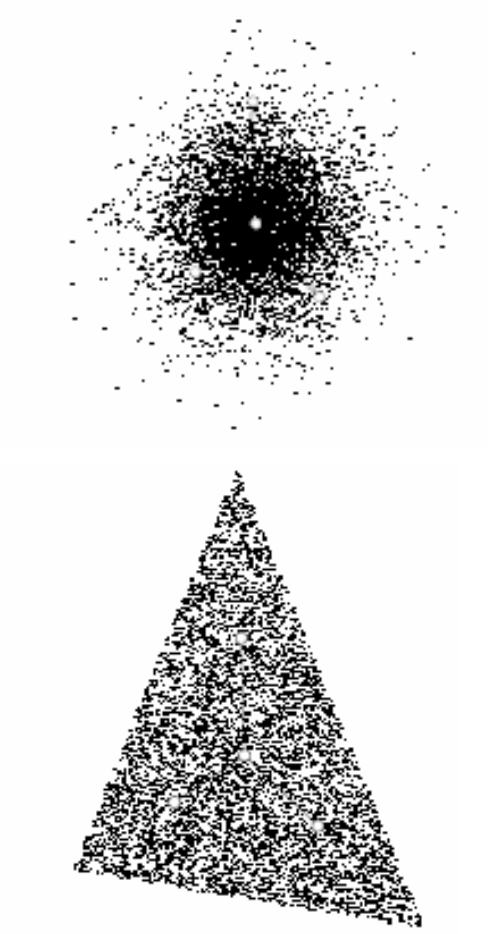
Mean-Centric Versus Parent-Centric Recombination

- Population mean can preserved in two ways:
  - Individual recombination preserves the mean of parents (mean-centric)
  - Individual recombination biases offspring towards a particular parent, but with equal probability to each parent (parent-centric)



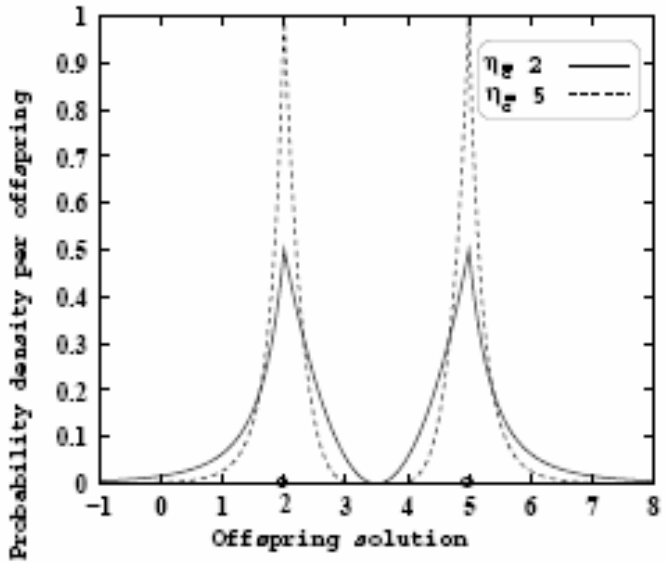
Mean-Centric Recombination

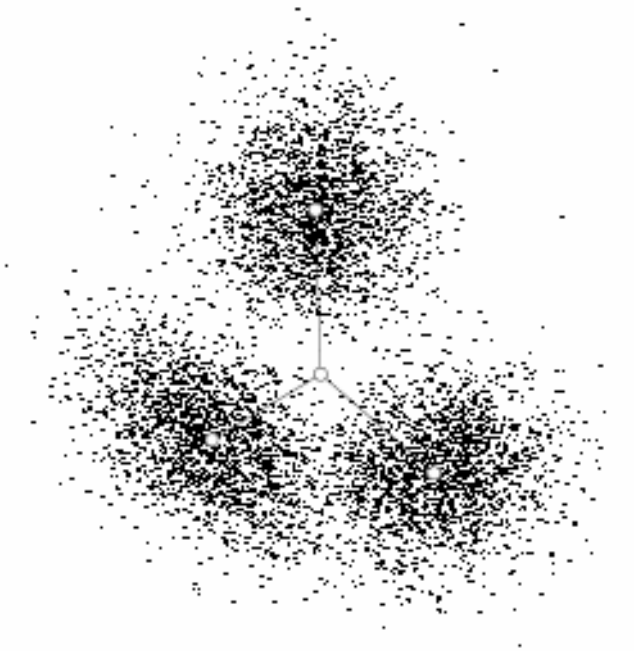
- UNDX (Ono and Kobayashi, 1997): $O(\mu^2)$ per offspring
- SPX (Higuchi et al., 2000): $O(\mu)$ per offspring
- BLX (Eshelman and Schaffer, 1991): discussed earlier



Parent-Centric Recombination

- SBX (Deb and Agrawal, 1995): variable-wise
- Vector-wise parent-centric recombination (PCX): O(μ) per offspring

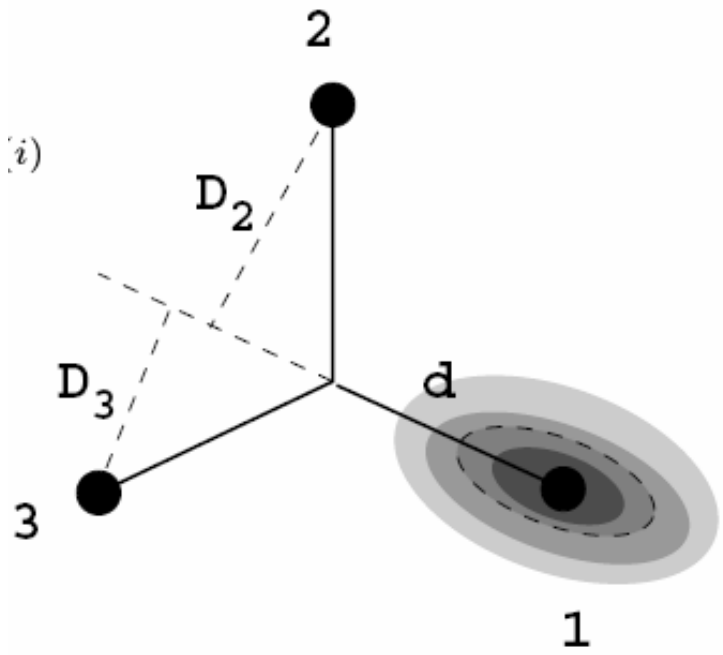




PCX Operator

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

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



Minimal Generation Gap (MGG) Model

1. Select μ parents randomly
2. Generate λ offspring using a recombination scheme
3. Choose two parents at random from the population
4. One is replaced with the best of λ offspring and other with a solution by roulette-wheel selection from a combination of λ offspring and two parents

- Steady-state and elitist GA
- Typical values used: N = 300, λ=200,
- μ=3 (UNDX) and μ=n+1 (SPX)
- Higuchi et al (2000) found SPX is better than UNDX

Three Test Problems

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

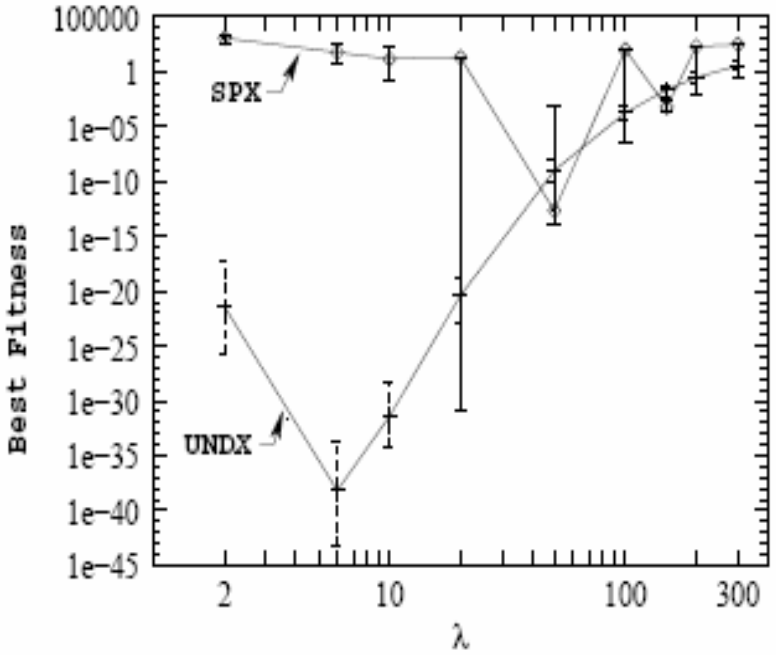
$$F_{\text{elp}} = \sum_{i=1}^{n} i x_i^2 \quad \text{(Ellipsoidal)}$$

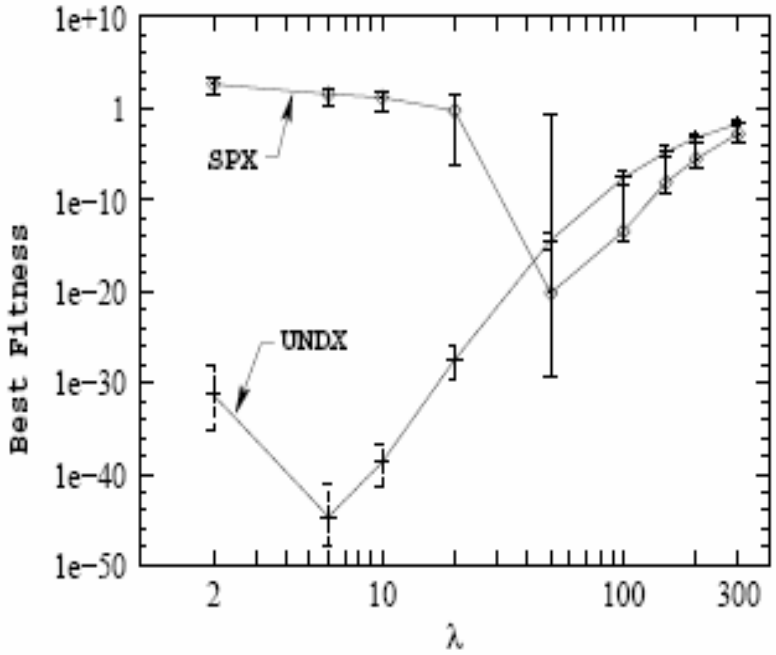
$$F_{\text{sch}} = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2 \quad \text{(Schwefel)}$$

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

Ellipsoidal and Schwefel Functions

- SPX and UNDX works better with small λ
- UNDX is better than SPX

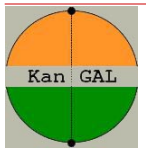




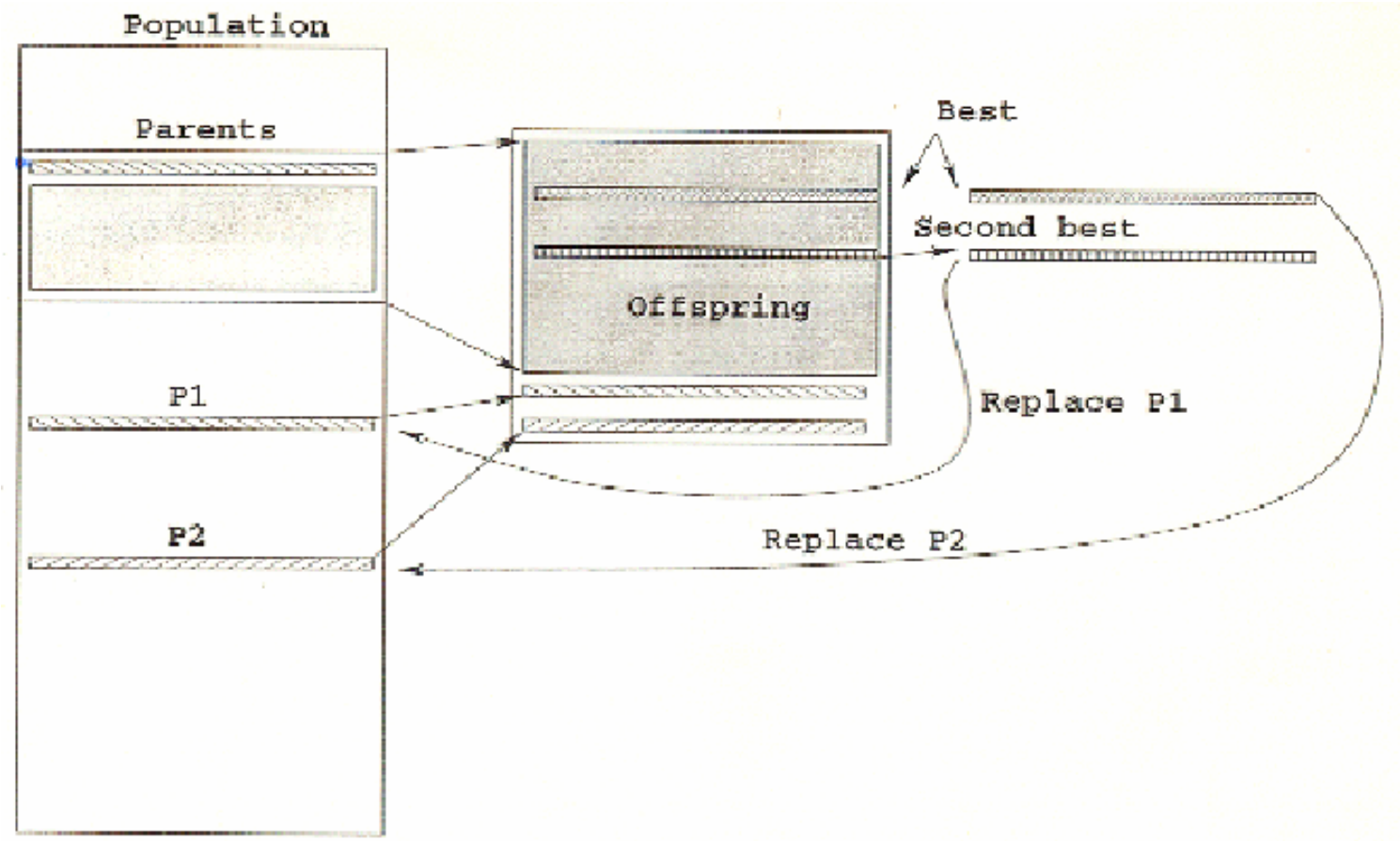
Generalized Generation Gap (G3) Model

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
- Desired accuracy in F is 10^{-20}

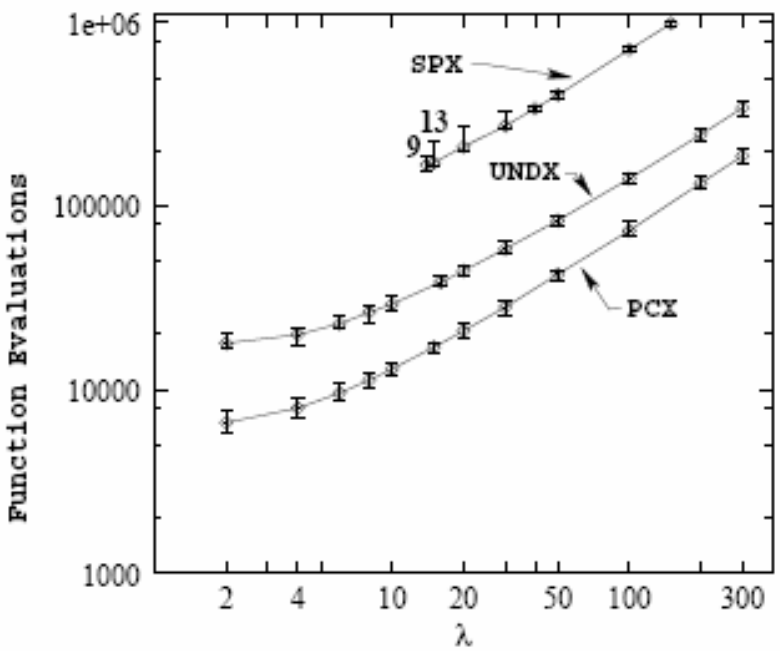


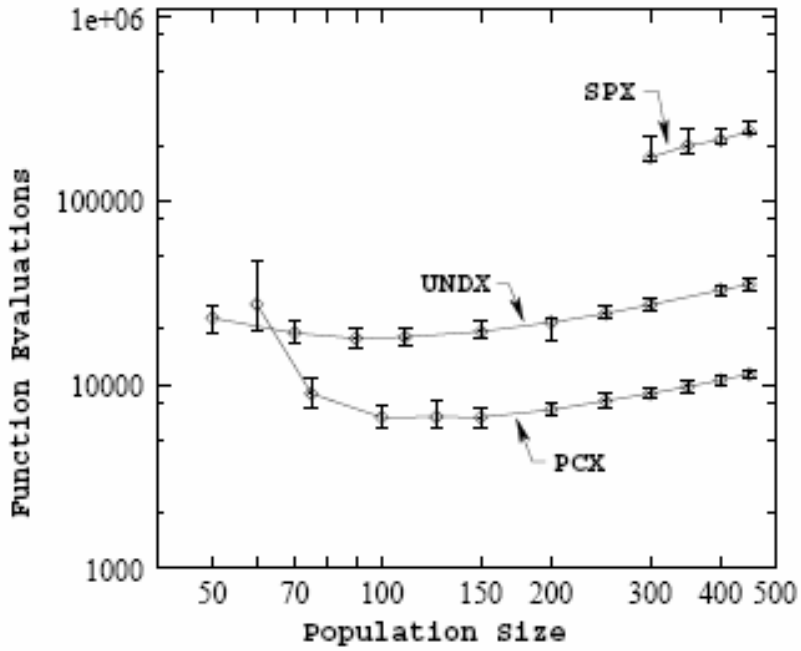
A Sketch of G3 Model



Ellipsoidal Function

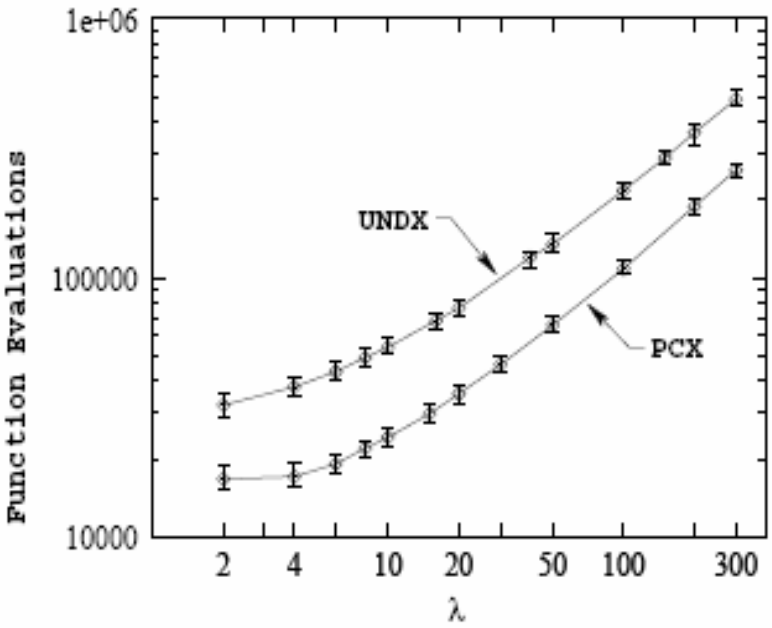
- PCX performs better

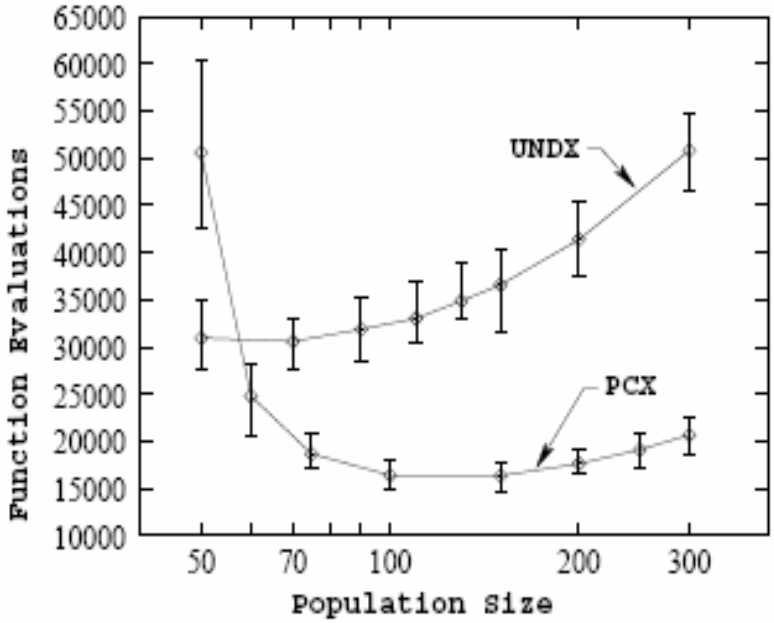




Schwefel's Function

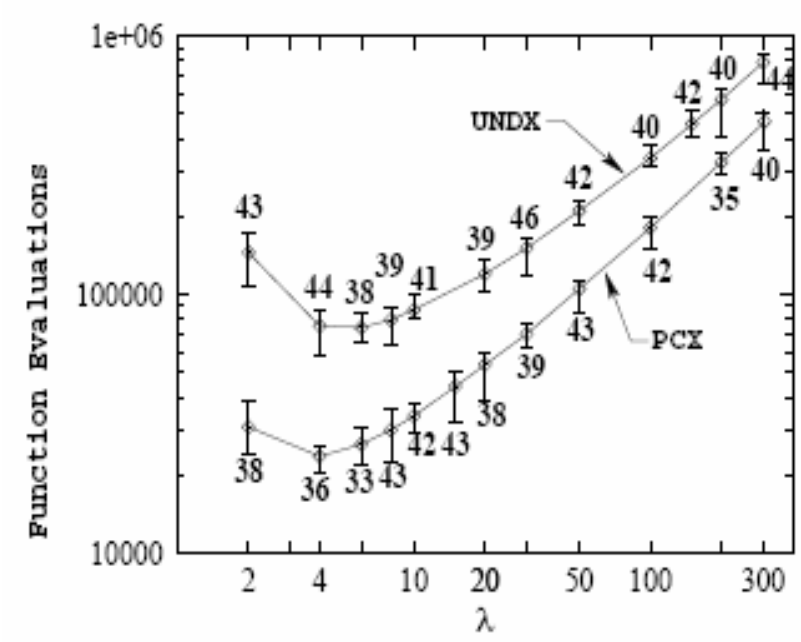
- PCX peforms better

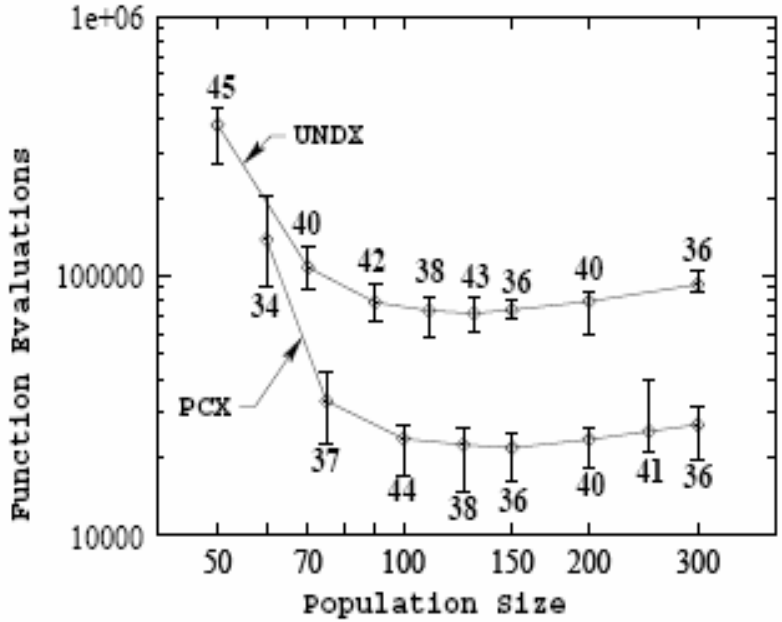




Rosenbrock's Function

- PCX performs better



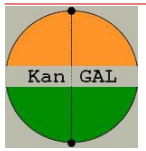


MGG Versus G3

- Performance of UNDX on MGG and on G3
- UNDX on G3 is better
- Modified G3:
  - One parent is replaced, instead of two

Function	MGG	G3
F_{elp}	2,97,546	18,120
F_{sch}	5,03,838	30,568
F_{ros}	9,38,544	73,380

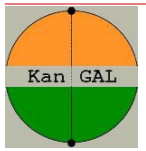
G3	F_{elp}			F_{sch}		
model	Best	Med	Worst	Best	Med	Worst
Original	5,744	6,624	7,372	14,643	16,326	17,712
Modified	5,826	6,800	7,728	13,988	15,602	17,188
	F_{ros}					
Original	14,847	22,368	25,797			
Modified	16,508	21,452	25,520			



Quasi-Newton Method

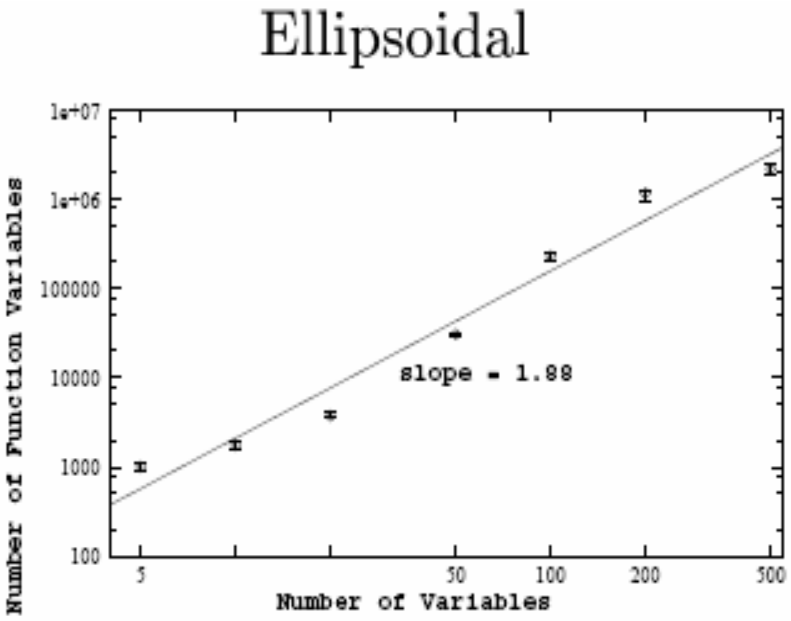
- Accuracy obtained by G3+PCX is 10^{-20}

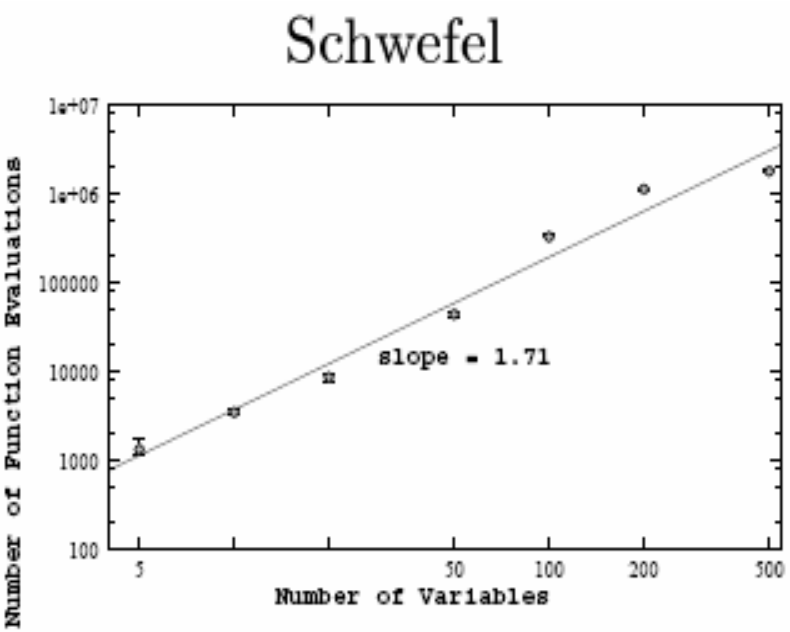
Func.	FE	Best	Median	Worst
F_{elp}	6,000	$8.819(10^{-24})$	$9.718(10^{-24})$	$2.226(10^{-23})$
F_{sch}	15,000	$4.118(10^{-12})$	$1.021(10^{-10})$	$7.422(10^{-9})$
F_{ros}	15,000	$6.077(10^{-17})$	$4.046(10^{-10})$	3.987
F_{elp}	8,000	$5.994(10^{-24})$	$1.038(10^{-23})$	$2.226(10^{-23})$
F_{sch}	18,000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



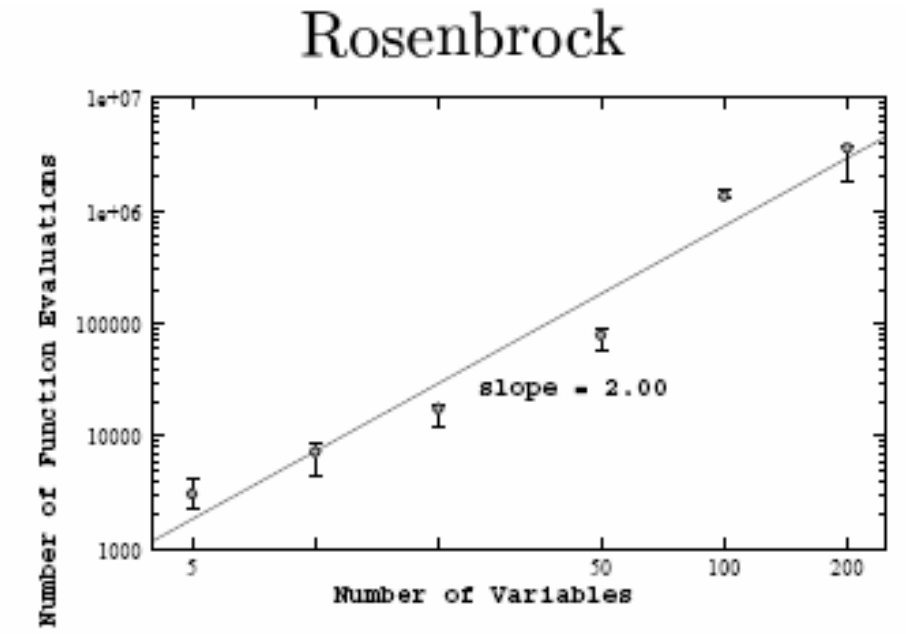
Scalability Study

- Accuracy 10^{-10} is set





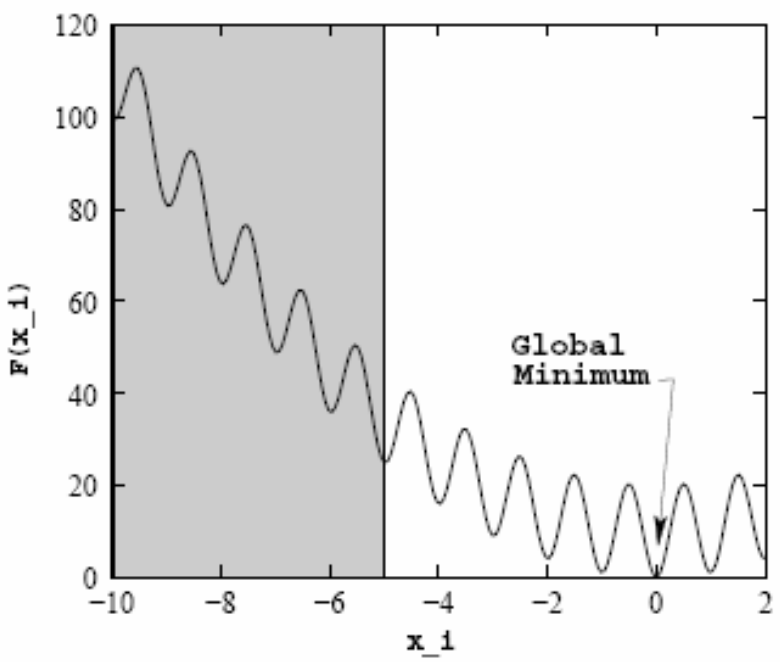
Scalability Study (cont.)



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

Rastrigin's Function Found Difficult

- Population initialized at x_i=[-10,-5]
- On each variable, overcome 4 local minima to reach global minimum
- Using G3+PCX best function value in [15.936,19.899] in one million FE
- Eiben and Back (1997) obtained F>10 starting from x_i=[4,5] in 200,000 FE

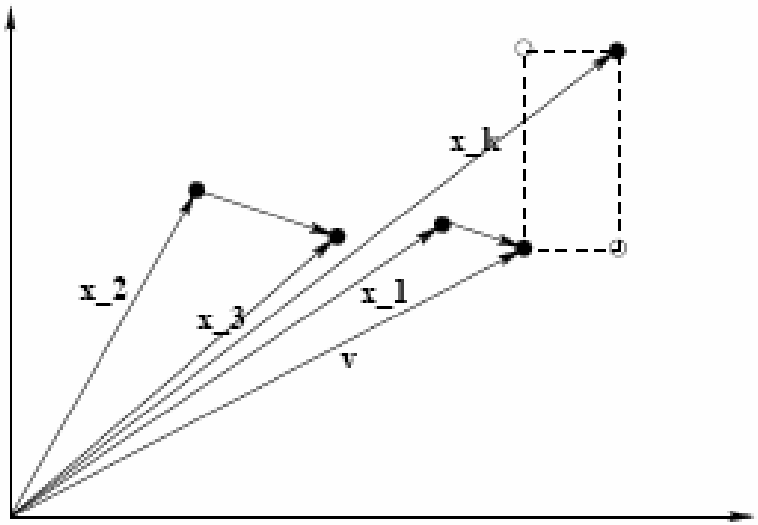


Differential Evolution (DE)

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

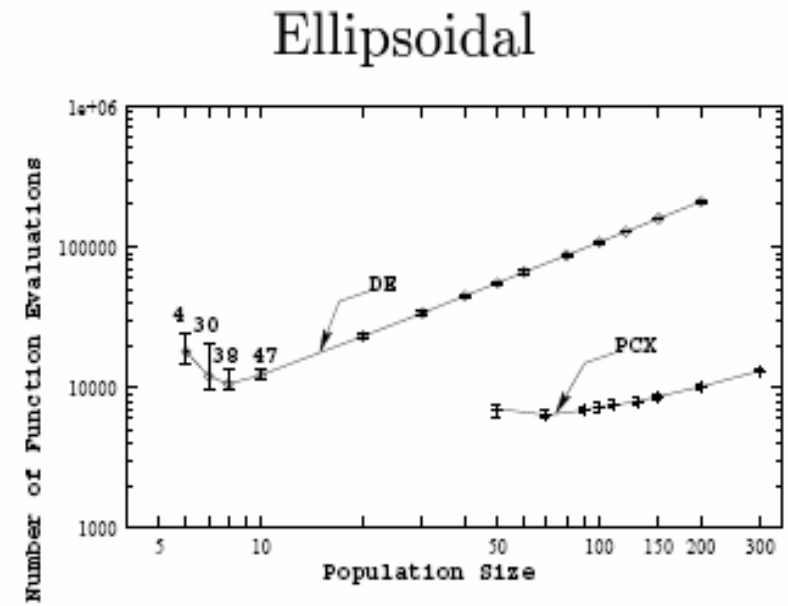
- Difference of parents in creating a child is important
- A number of modifications exist

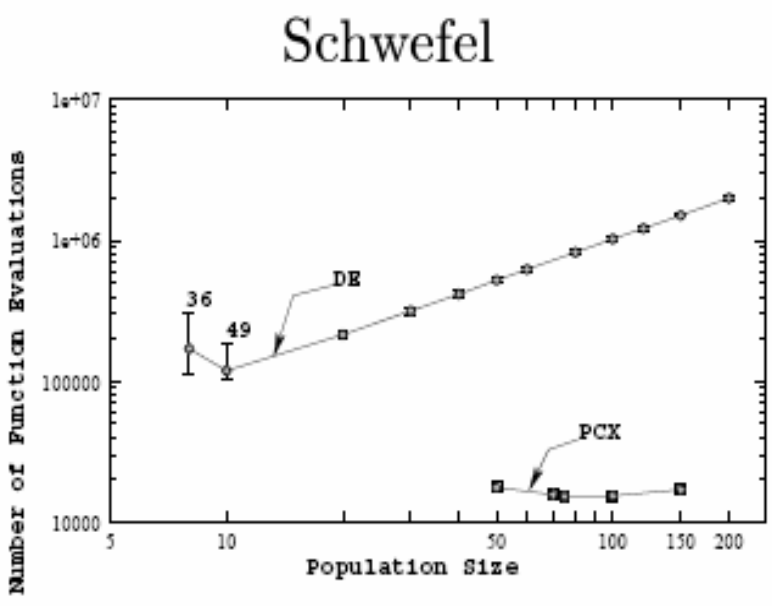
- $\mathbf{v} = \mathbf{x}^{(1)} + \lambda(\mathbf{x}^{(2)} - \mathbf{x}^{(3)})$
- $y_i = \begin{cases} v_i, & \text{with a prob. } p \\ \mathbf{x}_i^{(k)}, & \text{else} \end{cases}$



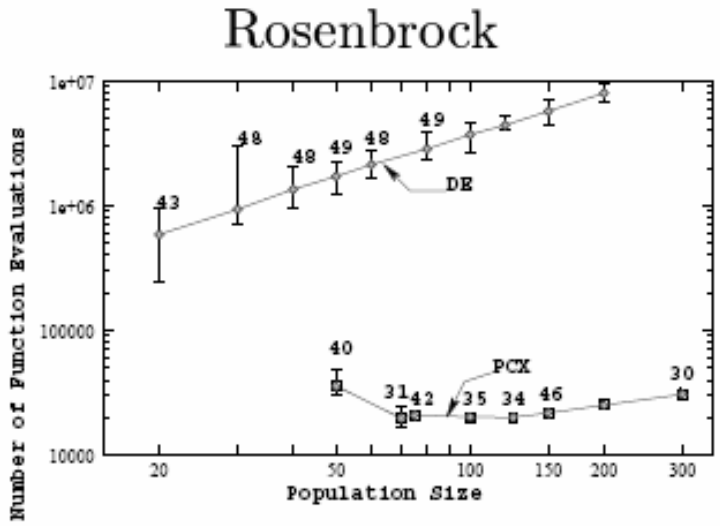
DE Results on Three Test Problems

- Exists an optimal population size
- G3+PCX is better





DE Results (cont.)



	F_{elp}		
	Best	Median	Worst
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

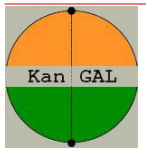
Particle Swarm Optimization (PSO)

- 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

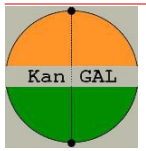
1. Initialize a random population
2. For each particle
   - Evaluate and update p_{best}
3. Identify p_g
4. Compute velocity and position for each particle
5. Loop till termination

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

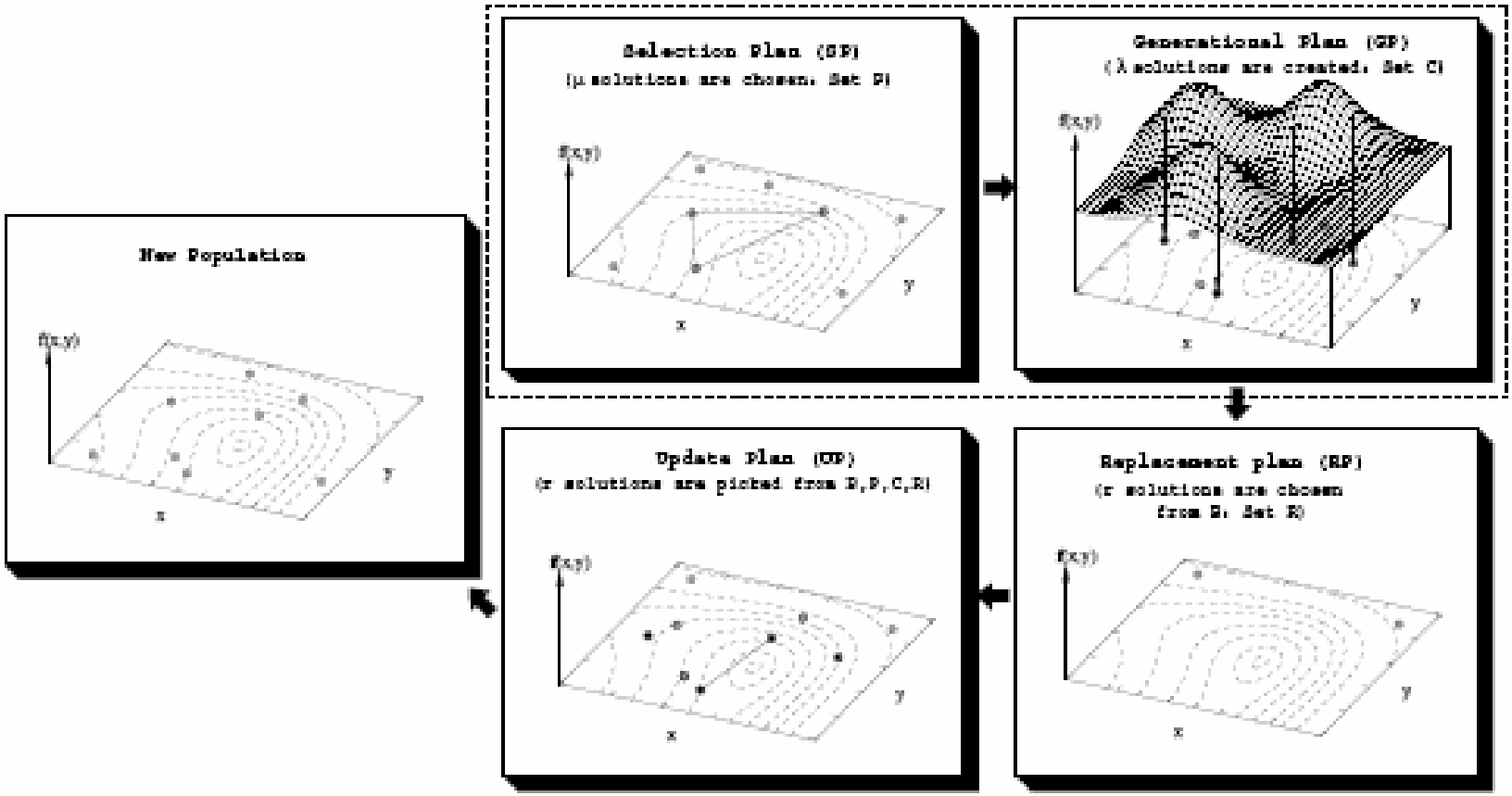


Population-Based Optimization Algorithm-Generator

- 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)



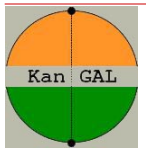
A Sketch of an Iteration



Designing an Optimization Procedure

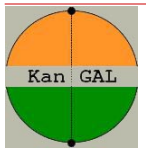
1. Choose SP, GP, RP and UP
2. Perform a parametric study on a set of test problems
   - N, μ, λ, r

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



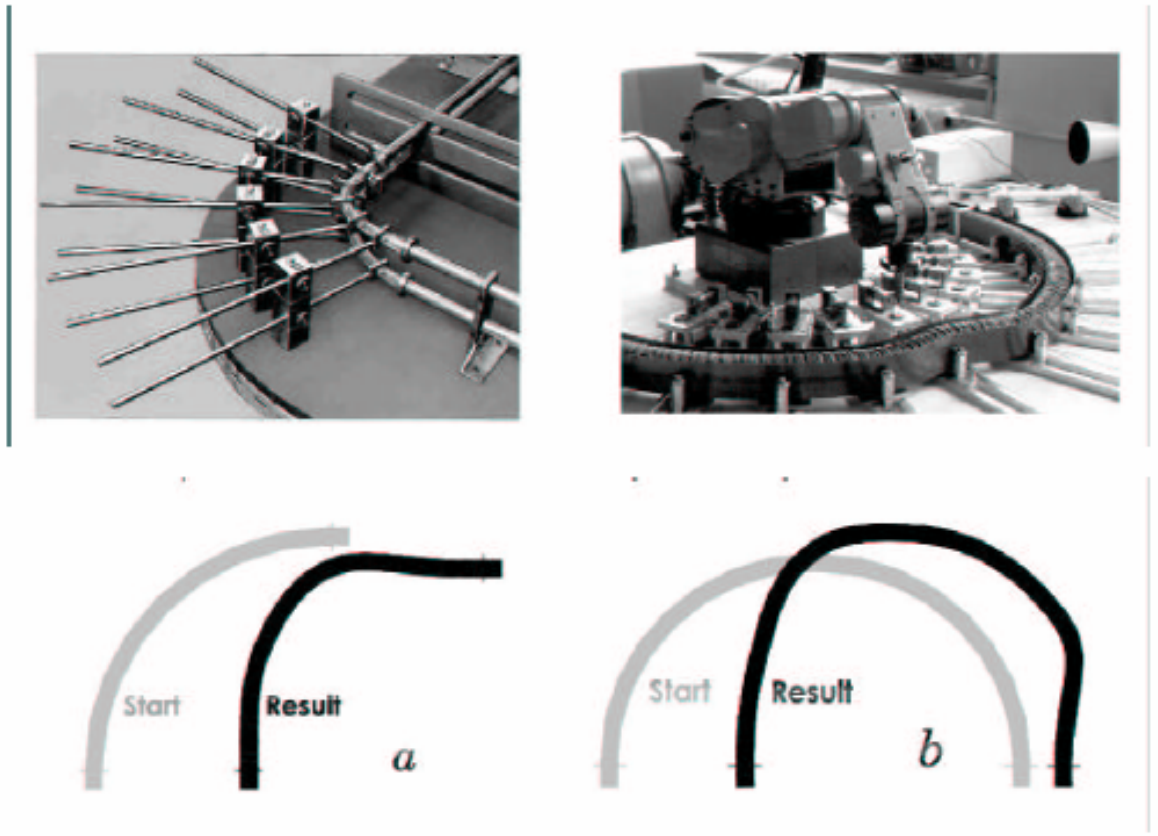
Evolution Strategy (ES)

- 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



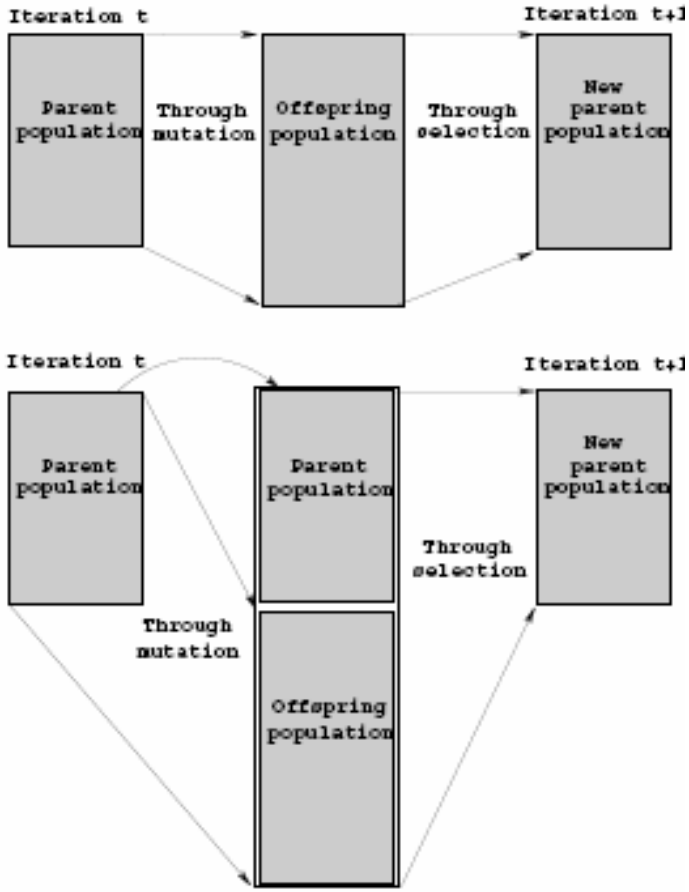
Pipe-Bending Experiments

Rechenberg, 1965



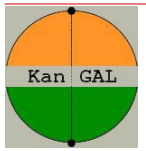
Multi-Parent ESs

- (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
- $1/5^{th}$ success rule
  - Increase mutation strength, if more than $1/5^{th}$ of offsprings are successful
  - Decrease, otherwise
- Early studies lack recombination



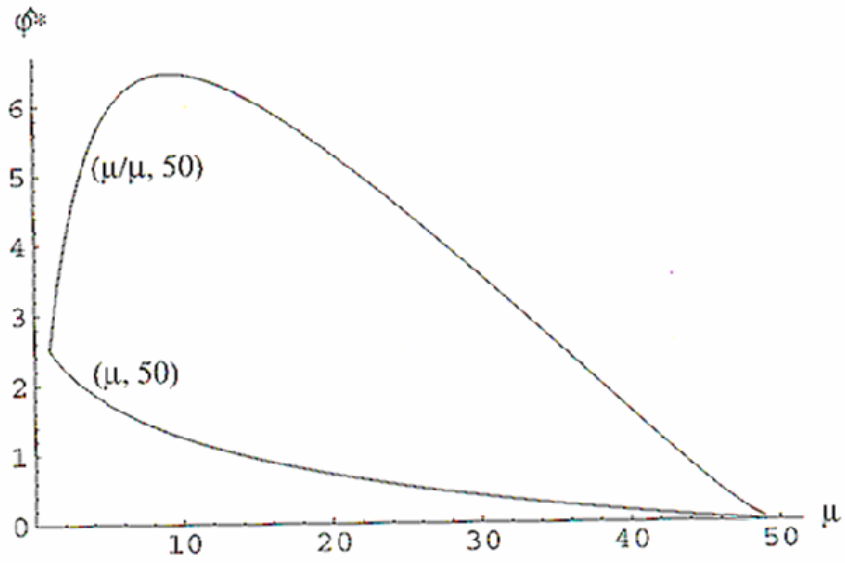
Recombinative ESs

- 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

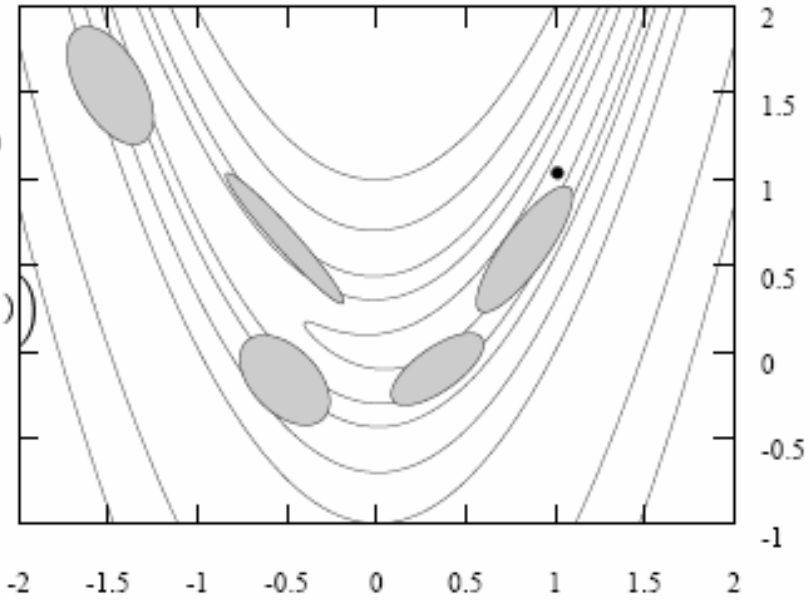


Correlated ES

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

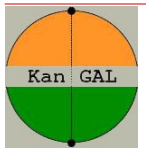
$$\sigma_i^{(t+1)} = \sigma_i^{(t)} \exp(\tau' N(0,1) + \tau N_i(0,1))$$
$$\alpha_j^{(t+1)} = \alpha_j^{(t)} + \beta_\alpha N_j(0,1)$$
$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \mathbf{N}\left(\mathbf{0}, C(\sigma^{(t+1)}, \alpha^{(t+1)})\right)$$

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



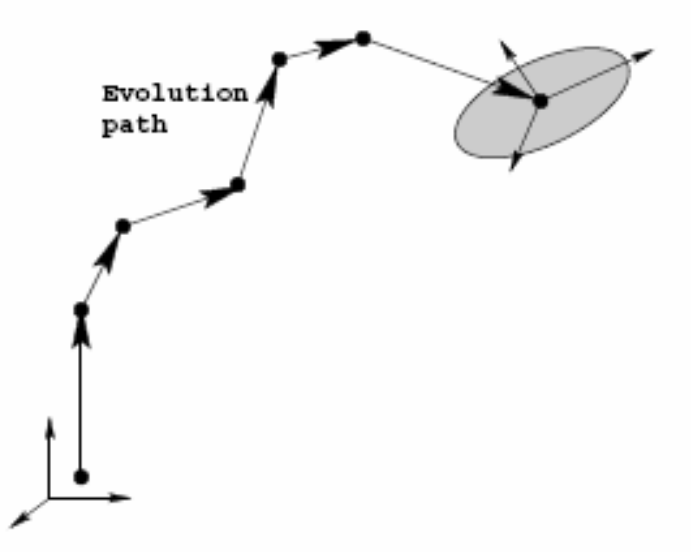
Correlated ES on Three Test Problems

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			



CMA-ES (Hansen & Ostermeier, 1996)

- Selecto-mutation ES is run for n iterations
- 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

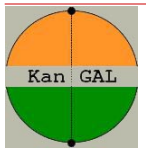


CMA-ES On Three Test Problems

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

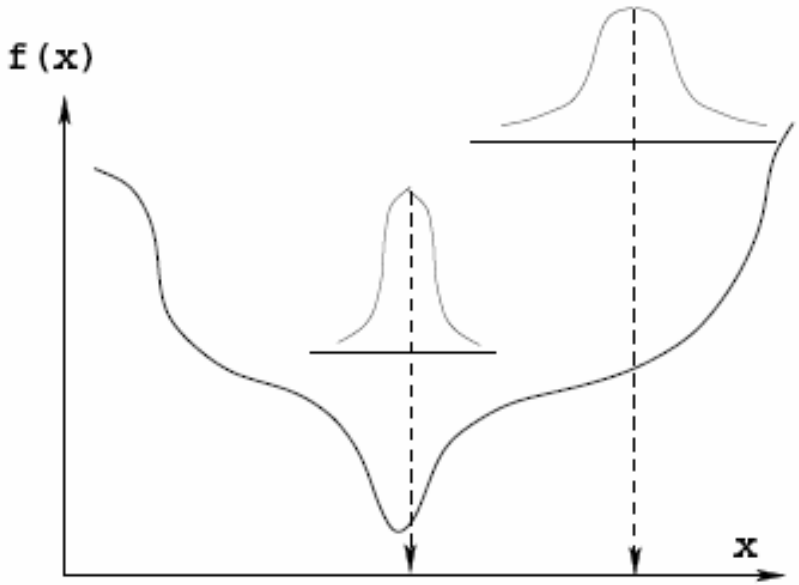
EA	F_{ros} Best	F_{ros} Median	F_{ros} Worst
CMA-ES	29,208	33,048	41,076
DE	243,800	587,920	942,040
G3+PCX	16,508	21,452	25,520

Accuracy $1X10^{-20}$



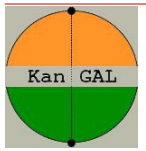
Self-Adaptive Evolutionary Algorithms

- 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.



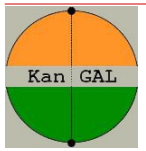
Why Self-Adaptation is Important in Optimization?

- 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



Log-normal Update of Strategy Parameters

- Isotropic Self-Adaptation:

$$\begin{aligned} \sigma^{(t+1)} &= \sigma^{(t)} \exp(\tau_0 \mathrm{N}(0,1)), \\ x_i^{(t+1)} &= x_i^{(t)} + \sigma^{(t+1)} \mathrm{N}_i(0,1), \end{aligned}$$

Learning parameter $\tau_0 = c_{1,\lambda}/\sqrt{N}$

- Correlated Self-Adaptation:

$$\begin{aligned} \sigma_i^{(t+1)} &= \sigma_i^{(t)} \exp\left(\tau' \mathrm{N}(0,1) + \tau \mathrm{N}_i(0,1)\right), \\ \alpha_i^{(t+1)} &= \alpha_i^{(t)} + \beta \mathrm{N}_i(0,1), \\ \vec{x}^{(t+1)} &= \vec{x}^{(t)} + \vec{\mathrm{N}}\left(\vec{0}, \mathrm{C}(\vec{\sigma}^{(t+1)}, \vec{\alpha}^{(t+1)})\right), \end{aligned}$$

where suggested $\beta = 5$ deg

- Non-Isotropic Self-Adaptation:

$$\begin{aligned} \sigma_i^{(t+1)} &= \sigma_i^{(t)} \exp\left(\tau' \mathrm{N}(0,1) + \tau \mathrm{N}_i(0,1)\right), \\ x_i^{(t+1)} &= x_i^{(t)} + \sigma_i^{(t+1)} \mathrm{N}_i(0,1), \end{aligned}$$

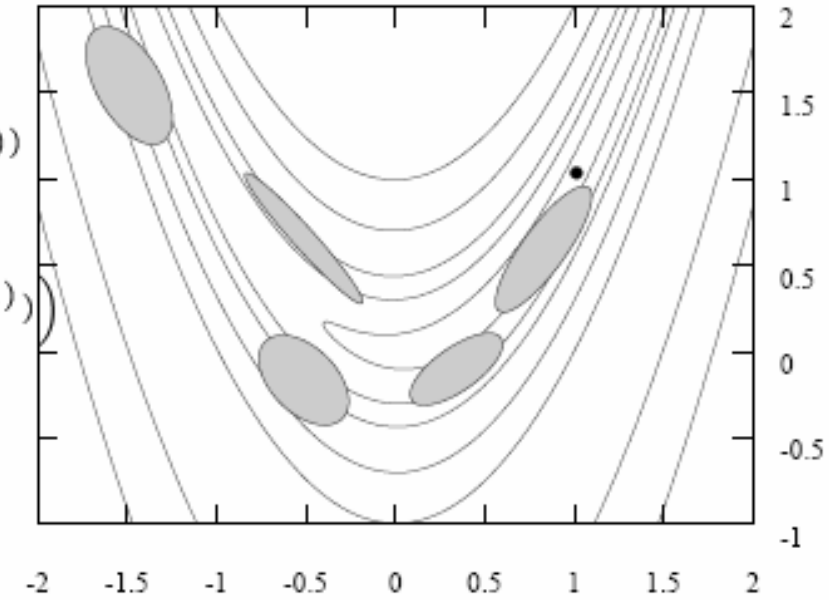
$\tau' \propto (2n)^{-1/2}$ and $\tau \propto (2n^{1/2})^{-1/2}$

Self-Adaptive ESs

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

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

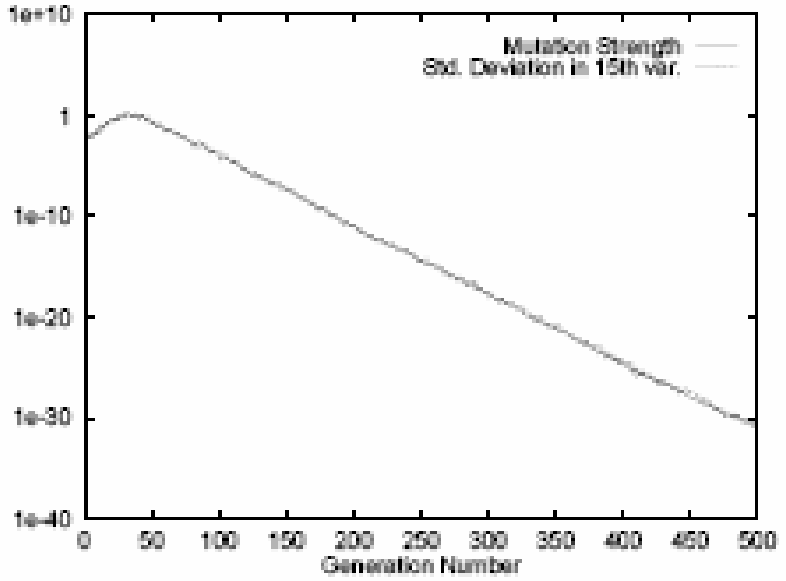
- Update is done with good parents
- ES learns the beneficial directions towards optimum



Properties of Self-Adaptive ES

- Difference in a child and a parent for isometric case
- An instantiation of Δ is a normal distribution with mode at zero
- Standard deviation proportional to $\sigma^{(t)}$, which signifies population diversity
- Mutation strength and std. deviation in x_1 for sphere model -> Similar

$$\Delta = \left(\sigma^{(t)} \exp(\tau_0 N(0,1))\right) N(0,1)$$

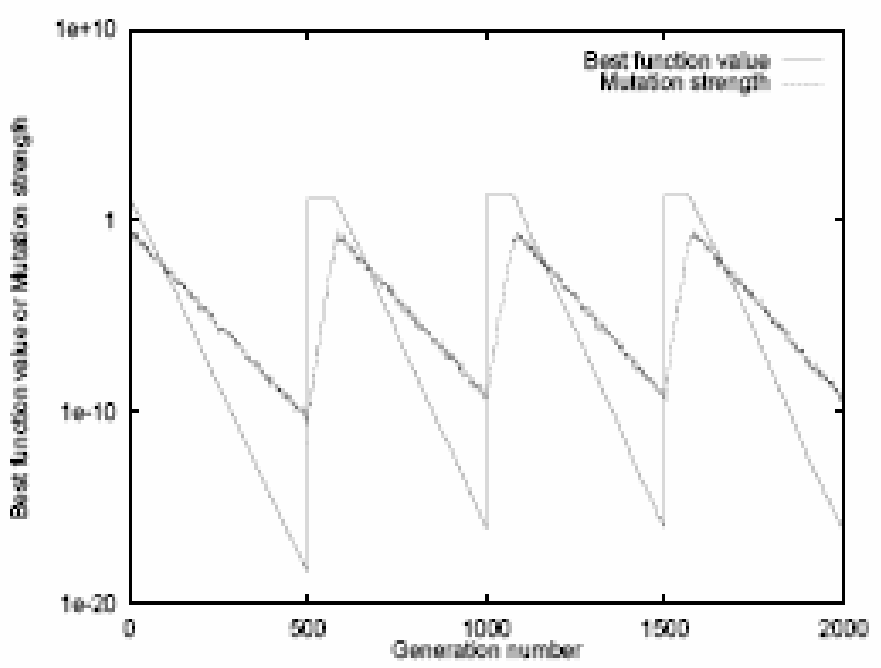


Time-varying Sphere Model

- Sphere model:

  $f(x) = |x - x^*|^2$

- Optimum x* is shifted after every 150 generations
- (15, 100)-ES, no recombination
- Simple model of a dynamic environment

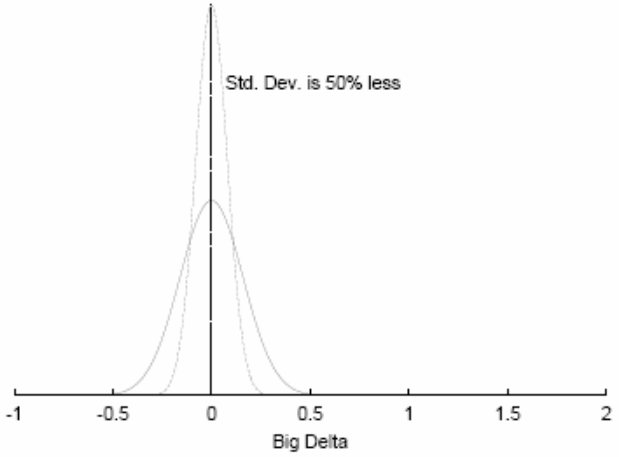


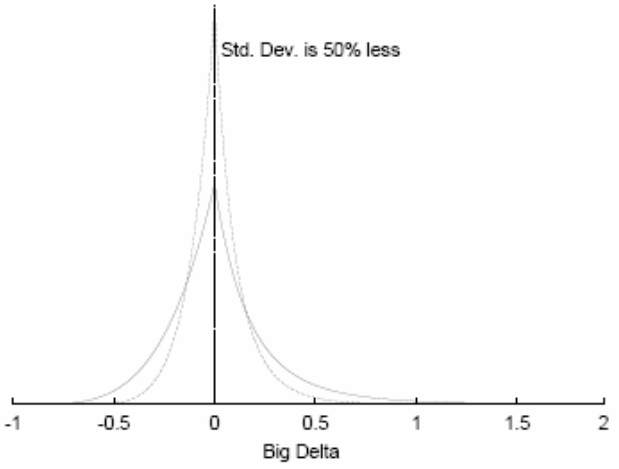
Real-Coded GAs with SBX Operator

- Recall:
  - Children are assigned in proportion to difference in parent
  - Children closer to parents are monotonically more probable to be created
- Difference in a child and a parent

$$\Delta = (\delta_p/2)(\beta_q - 1)$$

- δ_p is the absolute difference between parents
- β_q has its mode at 1
- Thus, distribution of Δ has its mode at 0 and std. dev. proportional to δ_p

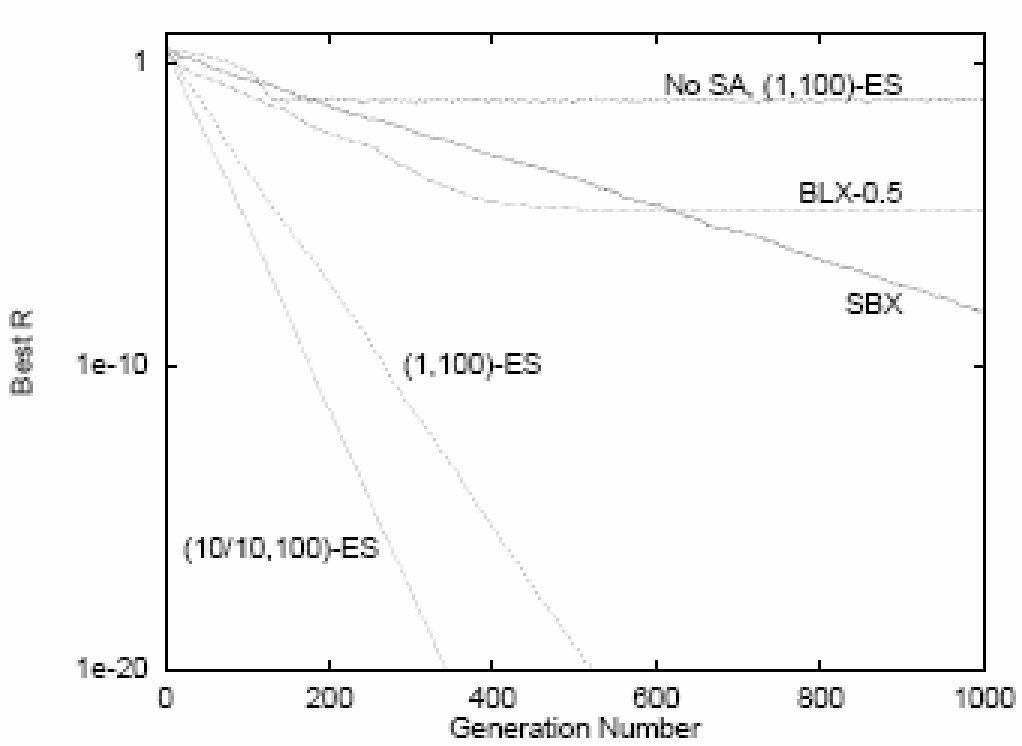




Comparison of Different Self-Adaptive Methods

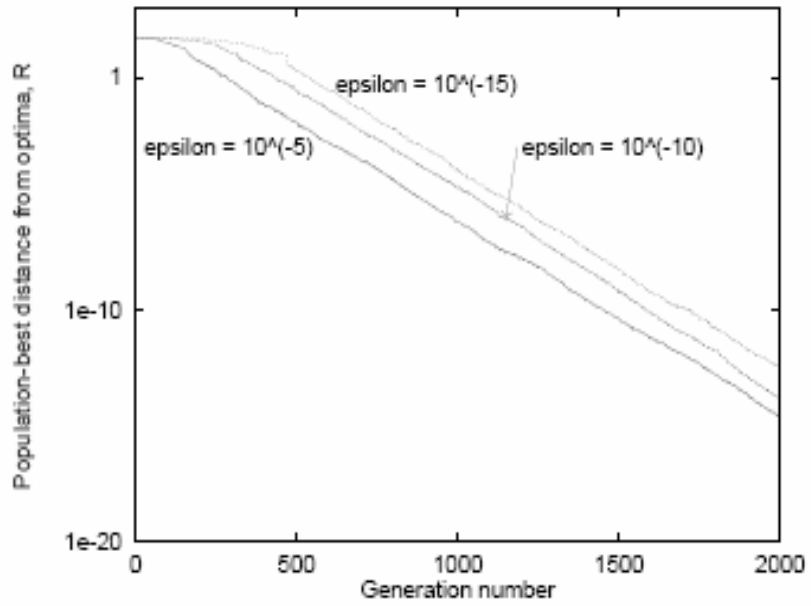
- Sphere model
- R: distance from optimum

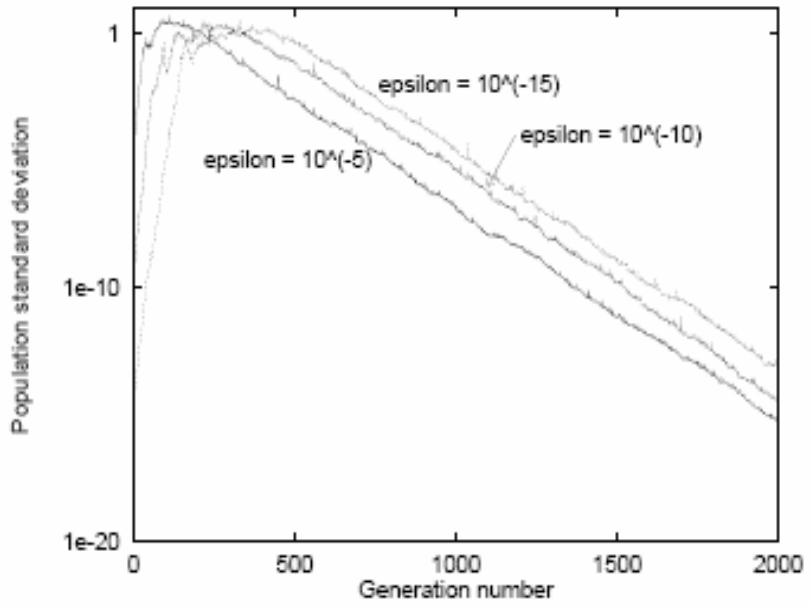
ES has an advantage on an isotropic problem



With Biased Population

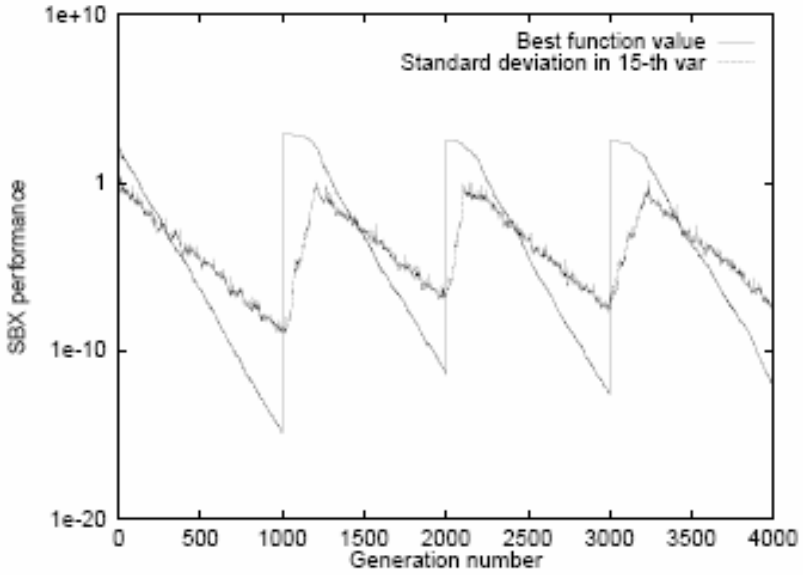
- Sphere model with x_i in [10-ε,10+ε]
- ε = 10^{-5}, $10^{-10,}$ 10^{-15}
- Population best distance and standard deviation





Time-varying Sphere Model

- 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

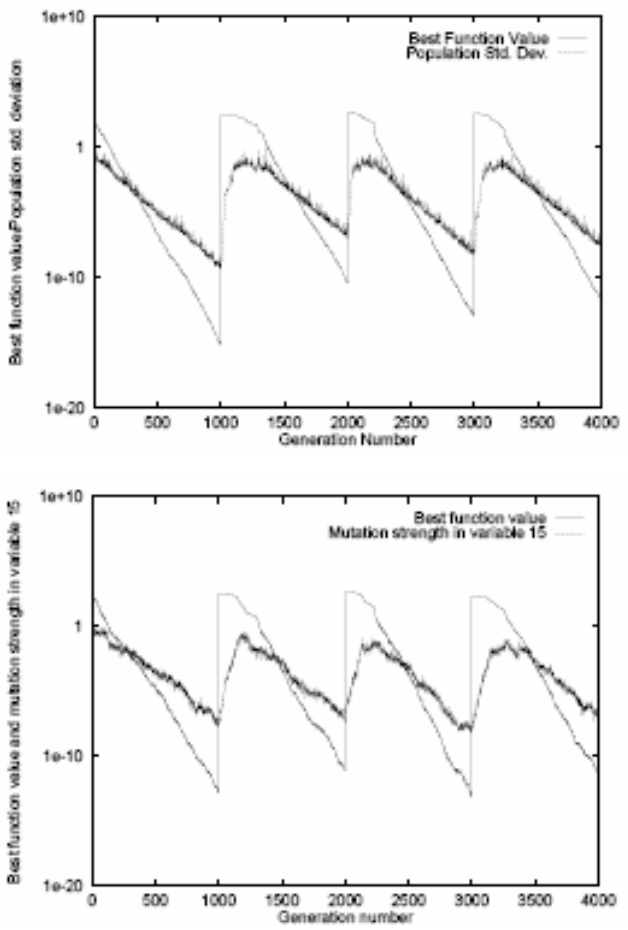


Time-Varying Functions (cont.)

- Elliptic function

$$f(\vec{x}) = \sum_{i=1}^{30} r_i x_i^2$$

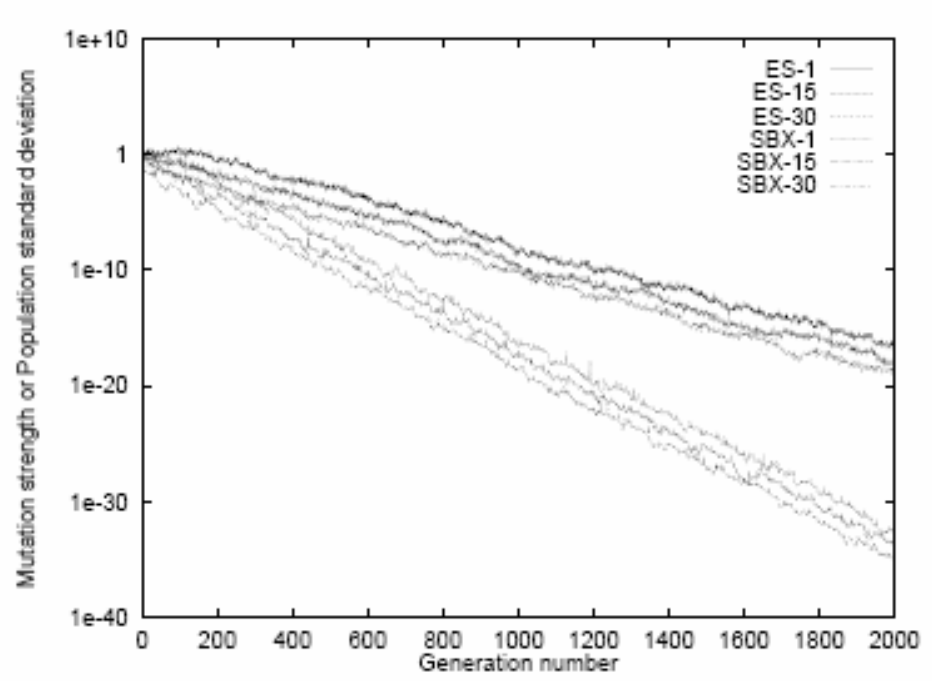
- R is an array of permutation of integers within [1,30]
- GAs with SBX and self-adaptive (15/15,100)-ES
- Very similar performance



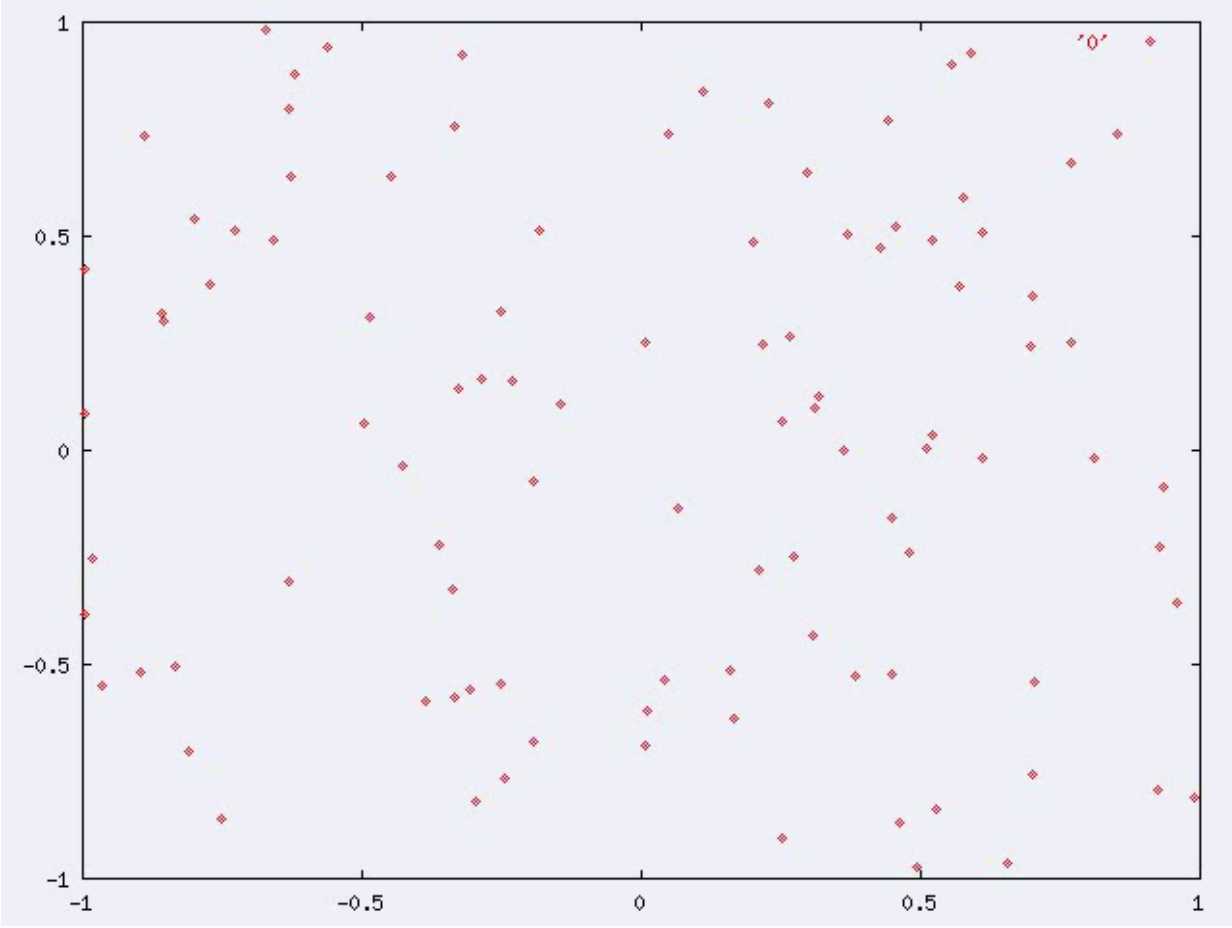
More Results

- Elliptic function
- 1st, 15th and 30th variables
- SBX (η=1) and (15/15,100)-ES compared
- Similar variations

$$f(\vec{x}) = \sum_{i=1}^{30} 1.5^i x_i^2$$

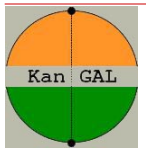


Simulation on a Moving Function



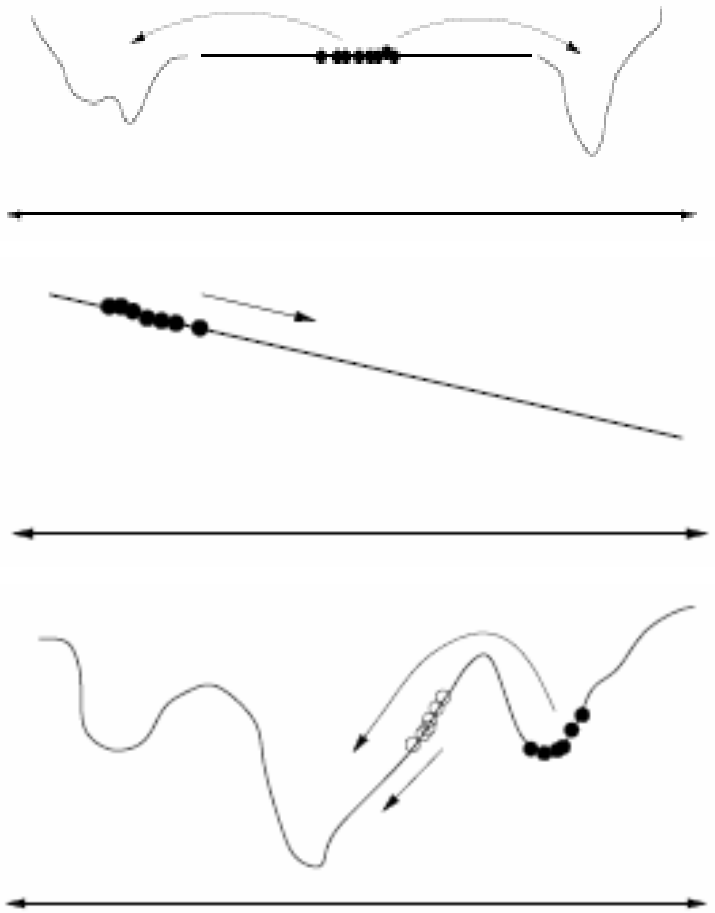
Postulate on Expected Population Mean

- Variation operator usually do not use fitness information
- No motivation for bias in any particular direction
- Standard ES/EP mutation operators use zero-mean prob. distributions
- Binary crossovers conserve expected population mean
- Crossovers in real-coded GAs also use zero-mean prob. distributions
- Creep mutation or non-uniform mutation also support



Postulate on Expected Population Variance

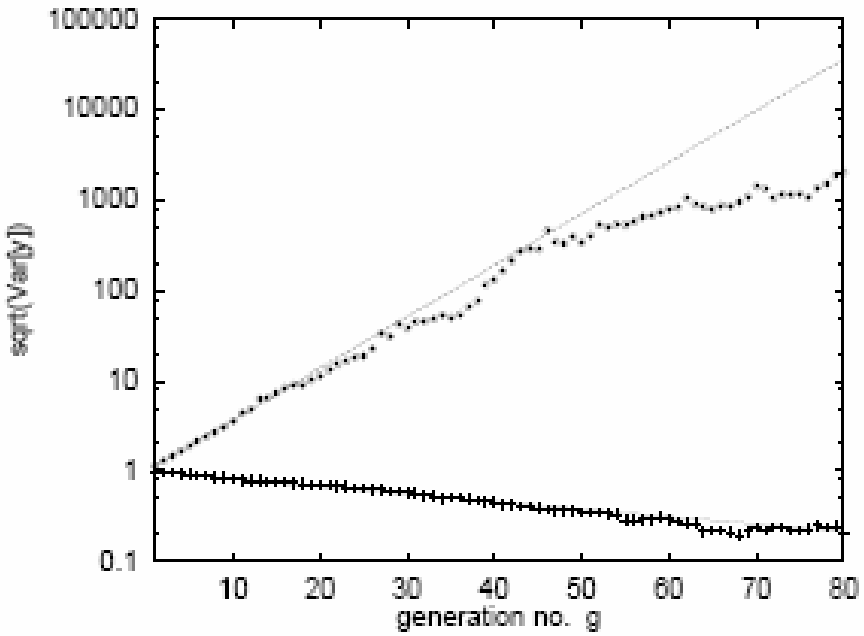
- Flat fitness behavior:
  - Increase exponentially quickly to leave the region of constancy
- Linear fitness behavior:
  - Should approach the optimum exponentially quickly
- Unimodal fitness behavior:
  - If population brackets the true optimum, reduction in variance is desired
  - Else (most likely the case) an increase is recommended
- Multi-modal fitness behavior:
  - Increase in general



Growth of Population Variance Under SBX-η

$$\sigma^2[y]^{(g)} = \sigma^2[y]^{(0)} \left(1 + \frac{2}{(\eta+3)(\eta-1)} - \frac{1}{\mu}\right)^g$$

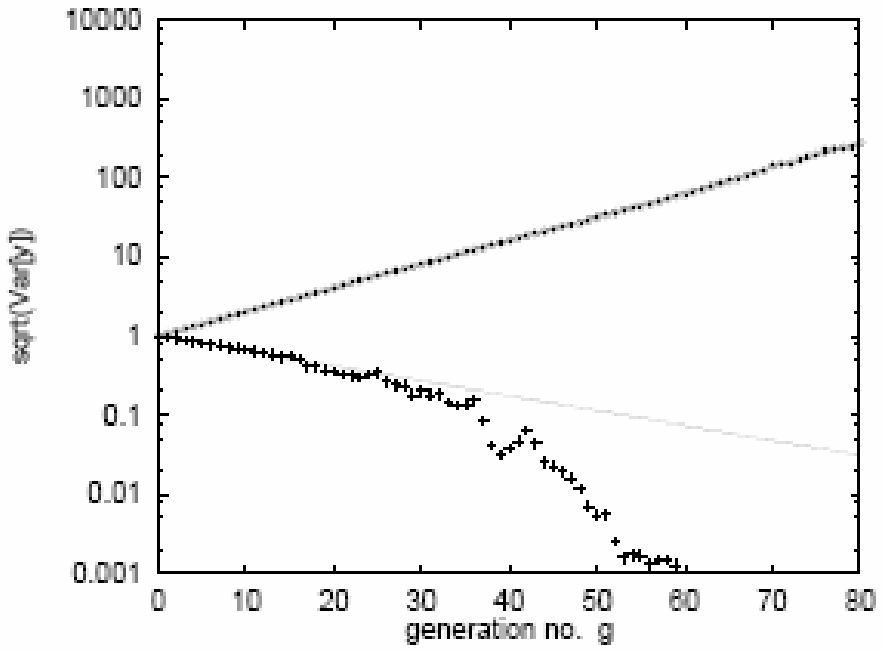
- 10 parents and η=2 (increasing)
- 10 parents and η=5 (decreasing)



Growth of Population Variance Under BLX-α

$$\sigma^2[y]^{(g)} = \sigma^2[y]^{(0)} \left(1 + \frac{1}{3}(2\alpha^2 + 2\alpha - 1) - \frac{1}{\mu}\right)^g$$

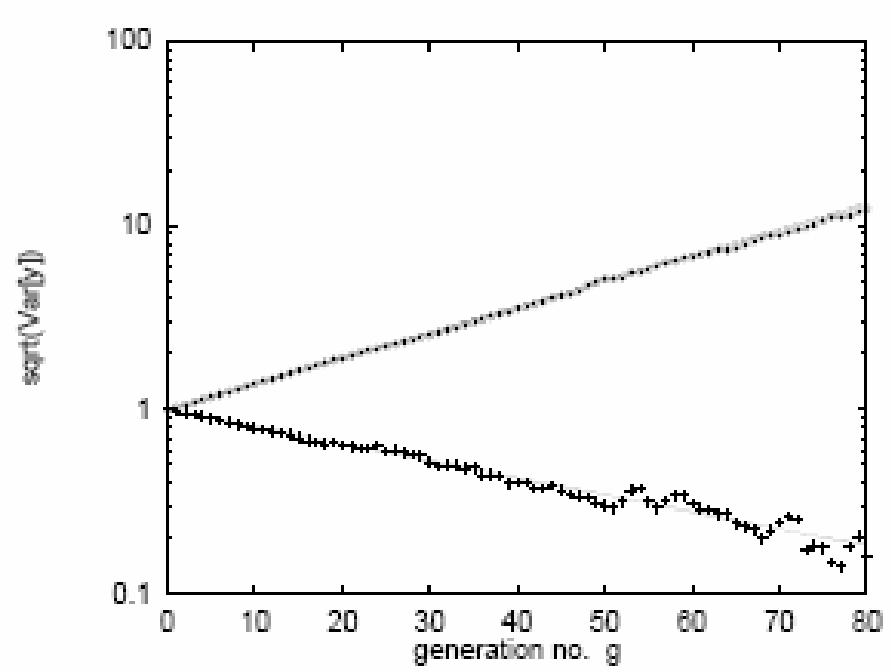
- α=0.5 with 60 parents (increasing)
- α=0.5 with 4 parents (decreasing)



Growth of Population Variance Under FR-d

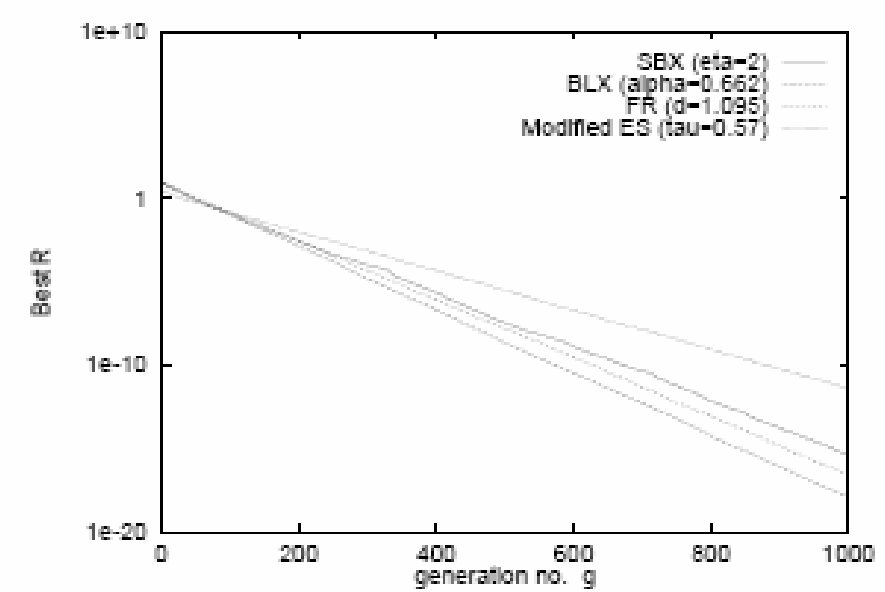
$$\sigma^2[y]^{(g)} = \sigma^2[y]^{(0)} \left(1 + \frac{1}{3}d^2 - \frac{1}{\mu}\right)^g$$

- d=0.5 with 60 parents (increasing)
- d=0.5 with 8 parents (decreasing)



Identical Variance Properties

- 20-dimensional sphere
- μ=100, x_i=[-5,5],
- Average of 10 runs



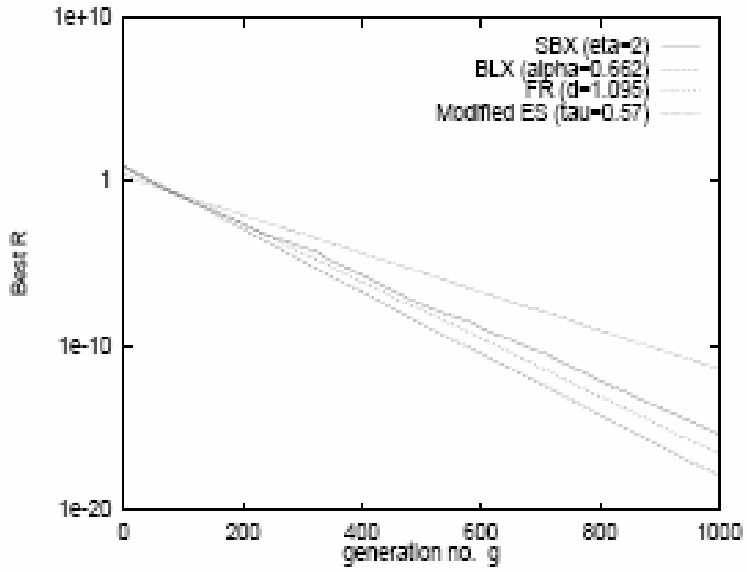
SBX η	BLX α	FR d
2	0.662	1.095
3	0.500	0.707
5	0.419	0.433
10	0.381	0.226

Self-Adaptive GAs and ESs

- Equating variance growth terms
- ES-τ with FR-d

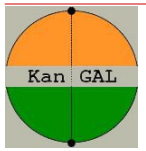
$$\frac{\sigma^2[y]^{(g)}}{\sigma^2[y]^{(g-1)}} = e^{\tau^2} + \frac{1}{\mu_{ES}} e^{\tau^2}\left(e^{\tau^2} - 1\right) = 1 + \frac{d^2}{3} - \frac{1}{\mu_{GA}}$$

- Results in $\tau \approx \sqrt{\ln\left(1 + \frac{d^2}{3}\right)}$
- (50/50,100)-ES with τ=0.57
- d_{eq} = 1.095



Comments on Self-Adaptive EAs

- Self-adaptive properties can be introduced with various EAs
- One key approach:
  - Create children based on difference in parents
  - Real-coded GAs, ESs, DE, PSO etc.
- Self-adaptations in operators, their probabilities, population size, coding etc. are all possible
- An interesting area of research and application

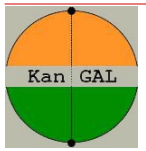


Overview of Lecture 12

- Scheduling EAs
- EAs with other soft-computing methods
  - ANN
  - Fuzzy logic approaches
- Conclusion of the course

Scheduling EAs

- Traveling salesperson problem (TSP)
  - Find shortest tour of n cities
- Quadratic assignment problem (QAP)
  - Find cheapest placement of n objects
- Graph coloring, aircraft scheduling etc.
- Blind approaches
  - Davis (1985), Goldberg and Lingle (1985), Gorges-Schlueter (1989), Muhlenbein (1989), Oliver, Smith and Holland (1987) etc.
- Hybrid approaches
  - Brown, Huntley and Spillane (1989)
- Knowledge-augmented approaches
  - Grefenstette et al. (1985), Suh and Van Gucht (1987)



Representation and GA Operators

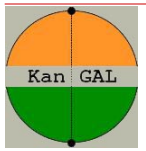
- A permutation of entities (cities, objects, etc.)
  - (d c a e b)
- Other representations exist
- Simple crossovers do not work!

  a b|c d e a b|a e b

  d c|a e b d c|c d e
- Need to create meaningful crossovers

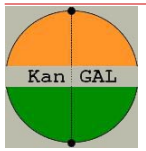
  a b|c d e a b|d c e

  d c|a e b d c|a b e



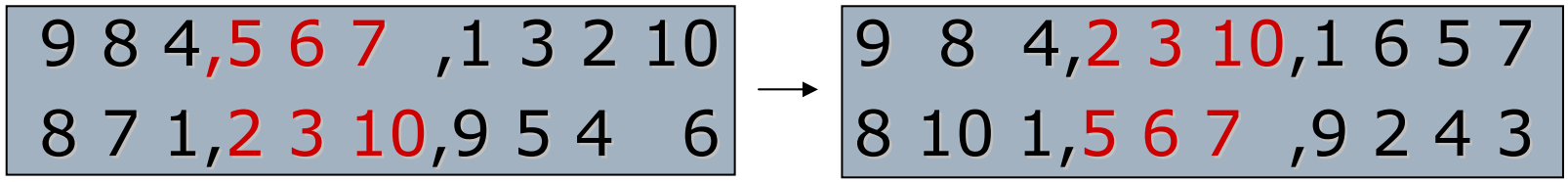
GA over Permutation

- Create tour-friendly crossovers
- O-schemeta (Goldberg & Lingle, 1985)
- Recombination operators
  - Partially matched crossover (PMX): respects absolute o-schemata
  - Cycle crossover (CX): respects cycles
  - Order crossover (OX): respects relative o-schemata
  - Edge recombination: respects edges
  - Other graph-based operators



Partially-Matched Crossover

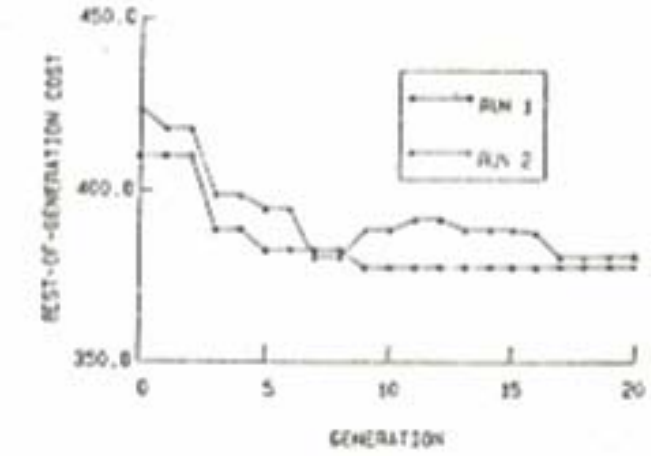
- Use pair-wise exchanges to match matching sections

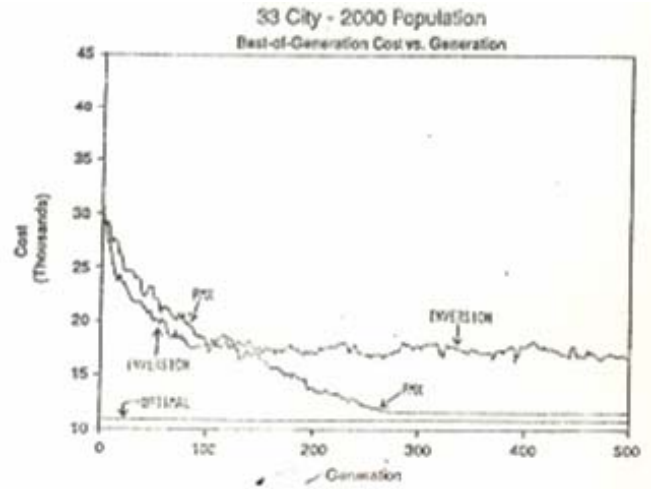


- A portion remains fixed, others as per their positions in the second schedule
- Respects absolute o-schemata
- Good for scheduling problems

TSP: PMX Versus Inversion

- Inversion: invert a partial permutation
- 33-city TSP
- Goldberg and Lingle (1985)
  - PMX is better
- Thousands of cities tried





Order Crossover (OX)

- A portion from one parent and others as per their order from second

  9 8 4 5 6 7 1 3 2 10

  8 7 1 2 3 10 9 5 4 6

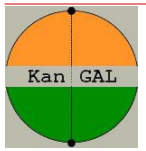
  0 0 0 1 1 1 0 0 0 0 Mask

- Offspring solutions:

  8 1 2 5 6 7 3 10 9 4

  9 8 4 2 3 10 5 6 7 1

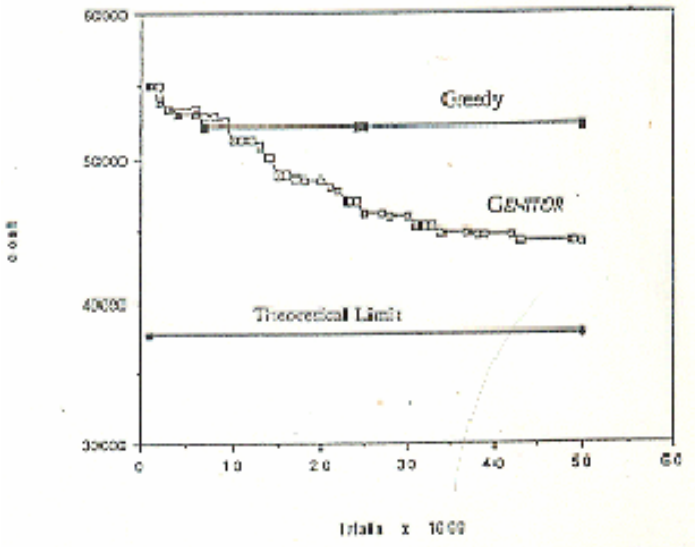
- Respects relative o-schemata
- Good for TSP type of problems



Edge Recombination

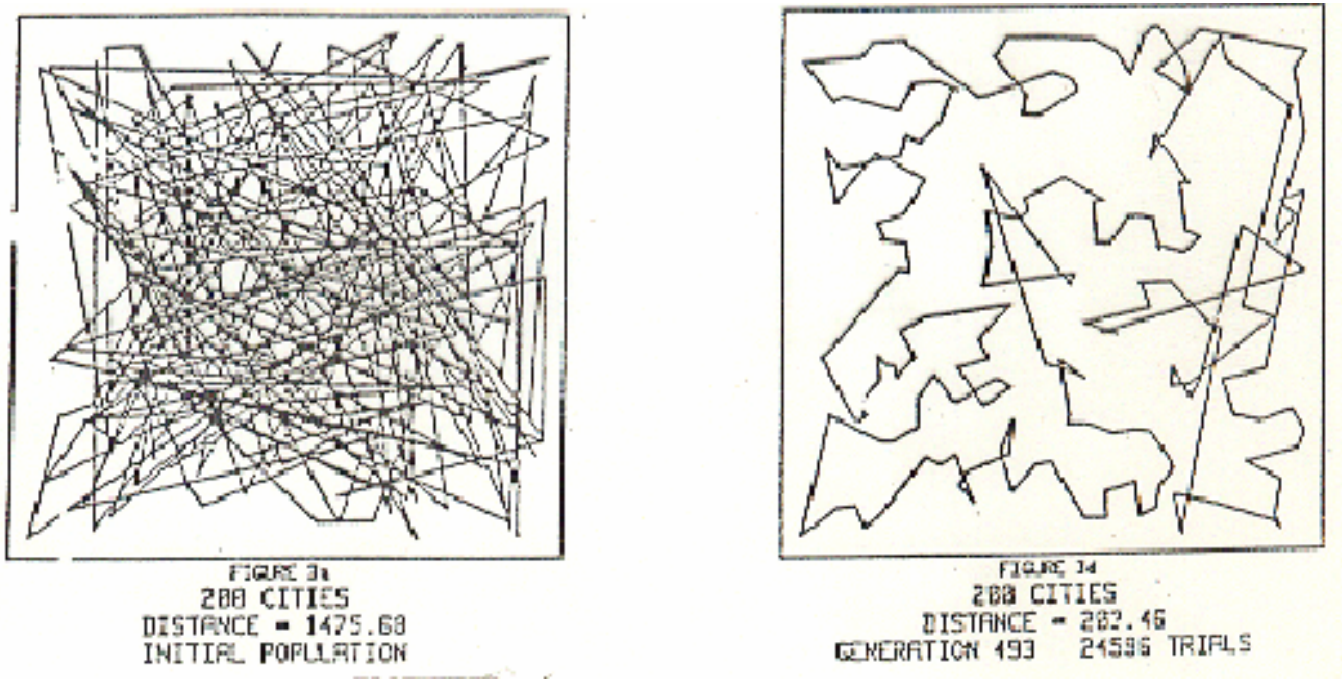
- Whitley et al. (1989)
- Tours: (A B C D E F) and (B D C A E F)
- Edge map
- Choose edge with least alternatives, break ties randomly
- Child: (B C D E A F)
- Respect edges (without regard to order)
- Job shop scheduling

City	Edges
A	B F C E
B	A C D F
C	B D A
D	C E B
E	D F A
F	A E B



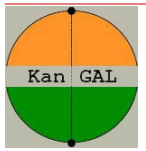
Knowledge-Augmented Operators

- Greedy crossover: use shorter edge
- Not a blind operator – may limit scope of use



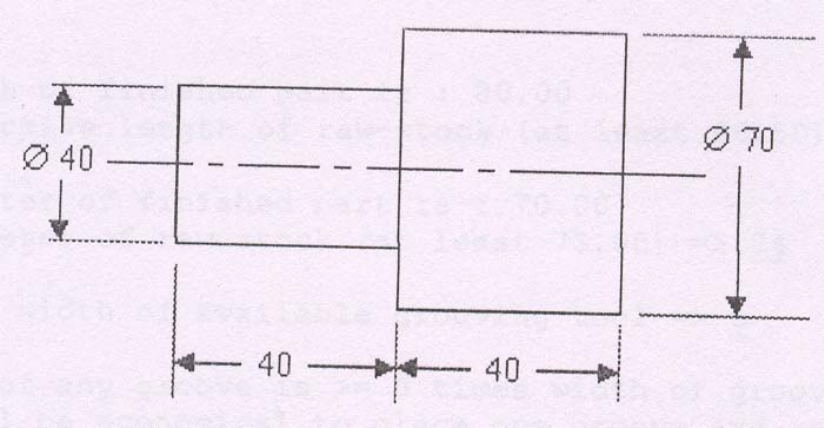
Machining Operation Scheduling

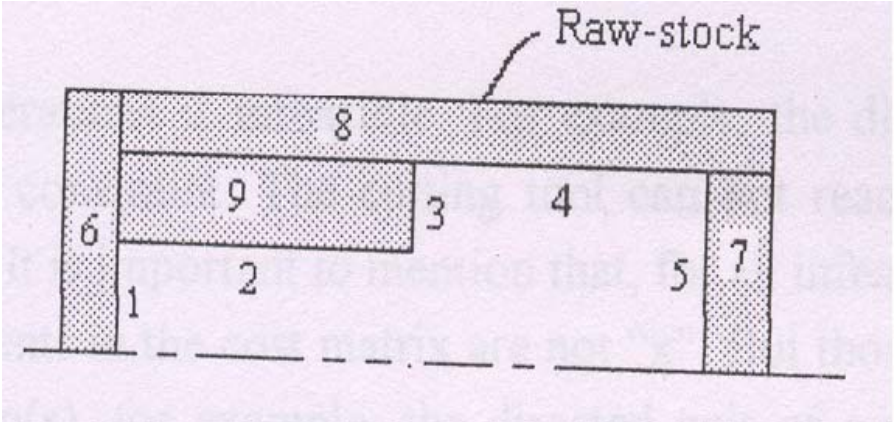
- A component requires multiple machining operations
- Automated sequencing is important
- An optimization task: Minimize overall cost
- A solution: A operation sequence
- A combinatorial optimization
- Employ a GA with special crossover and mutation operators



Case Studies

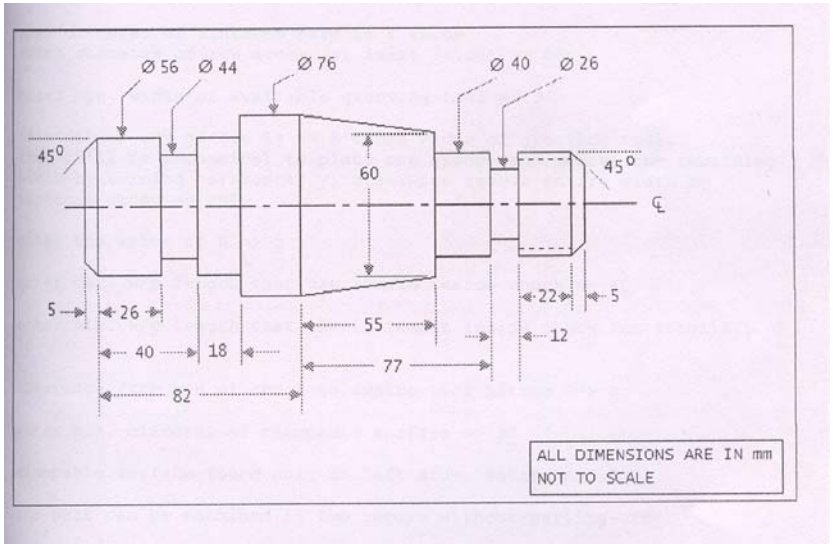
Large cost is added if an operation is not feasible

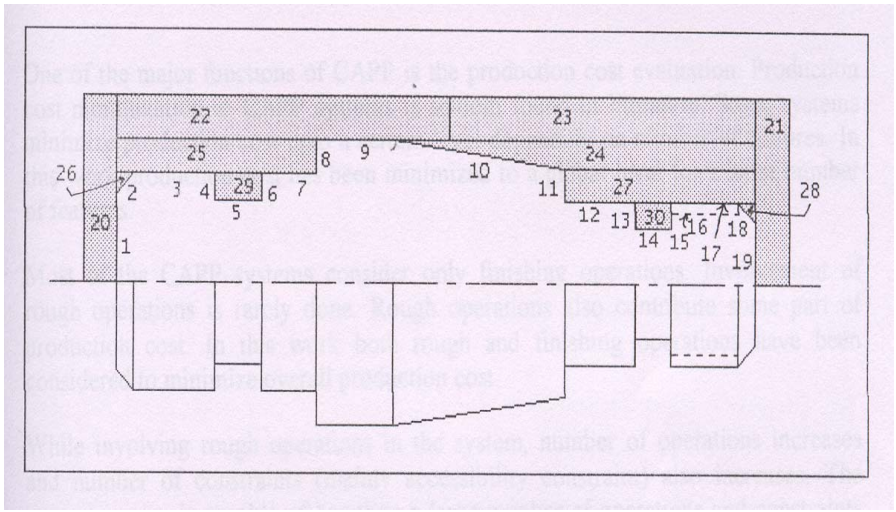




Sequences:
6-1-8-9-3-2-4-7-5 Cost: (Seq 78, Penalty 297)
6-1-8-9-2-4-3-7-5 Cost: (Seq 78, Penalty 297)
6-1-8-9-3-4-2-7-5 Cost: (Seq 78, Penalty 297)

More Case Studies





Sequences (40 in total)
23-21-25-26-28-22-27-36-31-34-29-33-37-8-18-2-14-35-30-32-16-11-5-17-6-12-13-7-9-19-3-20-4-10-15-24-1
Seq 167 Penalty 6582

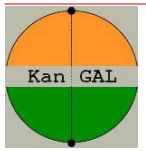
Another Representation Scheme

- A permutation is changed to an integer string
- (0 0 2 1 3 2) represents (b d f a e c)
  - i-th position can take [0,i-1]
  - Start with (a b c d e f) and item (_a_)
  - 2nd position 0 means b before a (_b_a_)
  - 3rd position 2 means (_b_a_c_)
  - 4th position 1 means (_b_d_a_c_)
  - 5th position 3 means (_b_d_a_e_c_)
  - 6th position 2 means (b d f a e c)
- Standard crossover (e.g. 1-pt) can be used

GAs, ANN and Fuzzy Logic

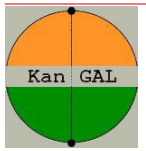
- They are not competing to each other
- A not-so-relevant question:

  Which is better, GAs or ANN?
- They each have their niche
  - GAs: Optimization and search
  - ANN: Function modeling
  - Fuzzy logic: Representation of uncertain variables and computation of functions
- They all are complementary to each other



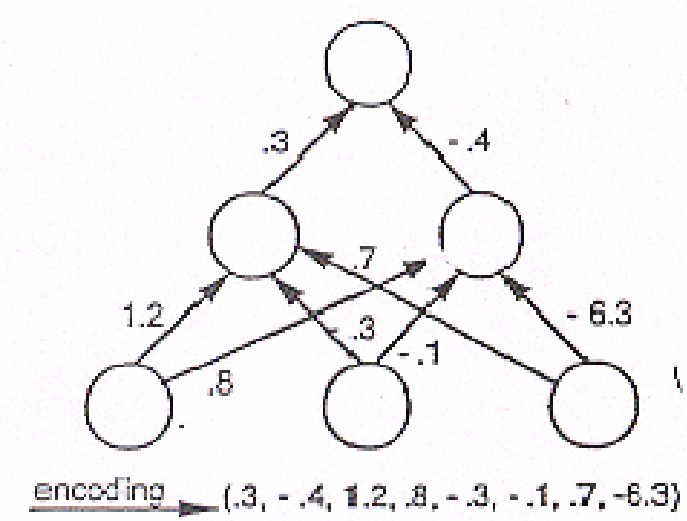
GAs with Neural Nets

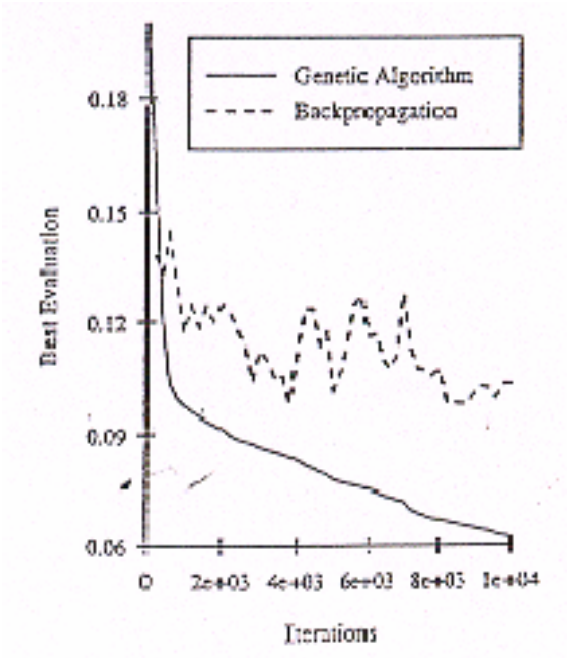
- Difficulties in ANN approaches
  - Weight optimization through steepest-descent methods
  - Network architecture design
- How can GAs help?
  - Replace back-propagation or other learning rules
  - Generate architecture on the fly
  - Least-mean-square error need not be goal
  - Optimal node transfer function: sigmoidal, Gaussian, polynomial etc.
  - Simultaneous network and weight optimization
  - Variable training sets



ANN Weight Optimization

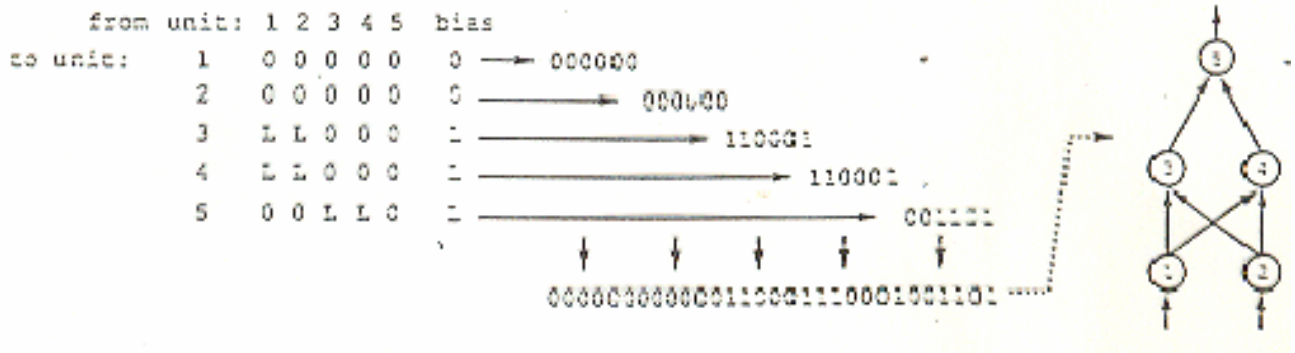
- Montana and Davis (1989)
- Use real weights and various operators
- Comparison to back-propagation (X-OR problem)



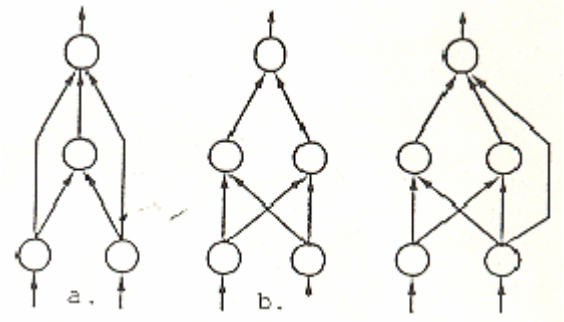


ANN Architecture Optimization

- Miller, Todd and Hegde (1989)
- n(n+1) learnability matrix converted to bit string

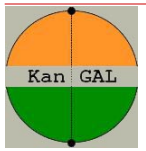


- Faster learning with extra connection in X-OR



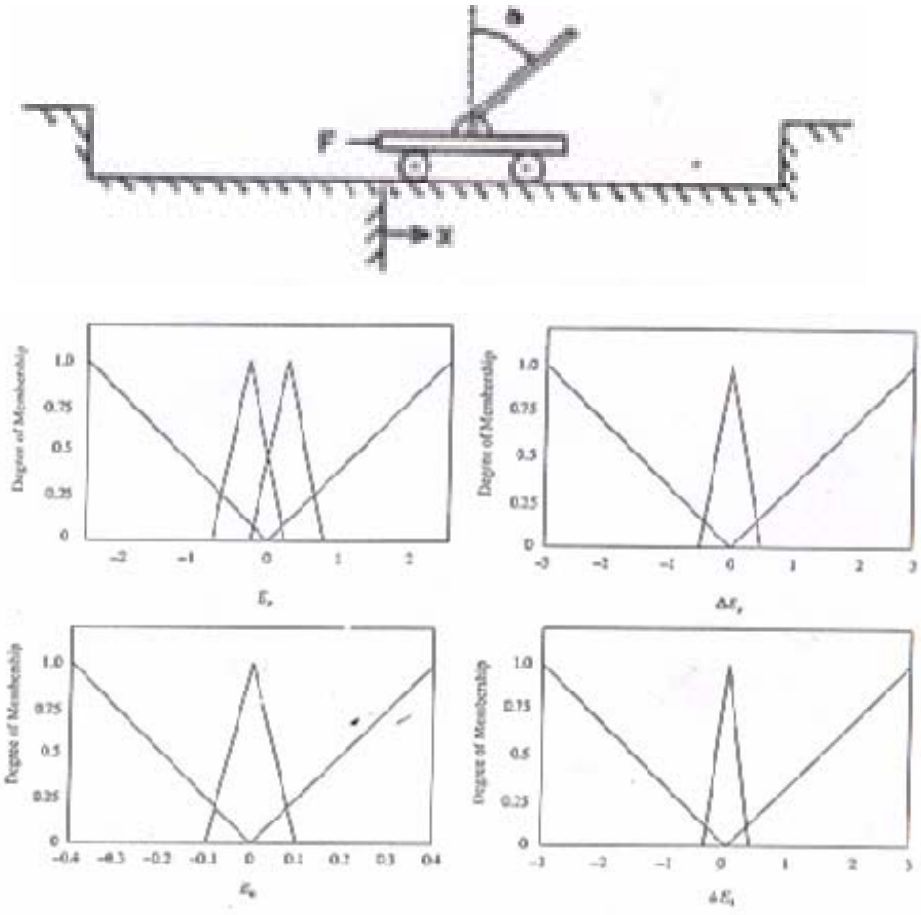
EAs with Fuzzy Logic

- Use EAs to find optimal fuzzy controller
  - Optimal fuzzy membership functions
  - Optimal rule base
  - Optimal rule base and membership functions
  - Simultaneous with classical optimization
  - Above in stages
- Use fuzzy technique to improve the performance of EAs
  - EA parameter setting
  - EA operator setting



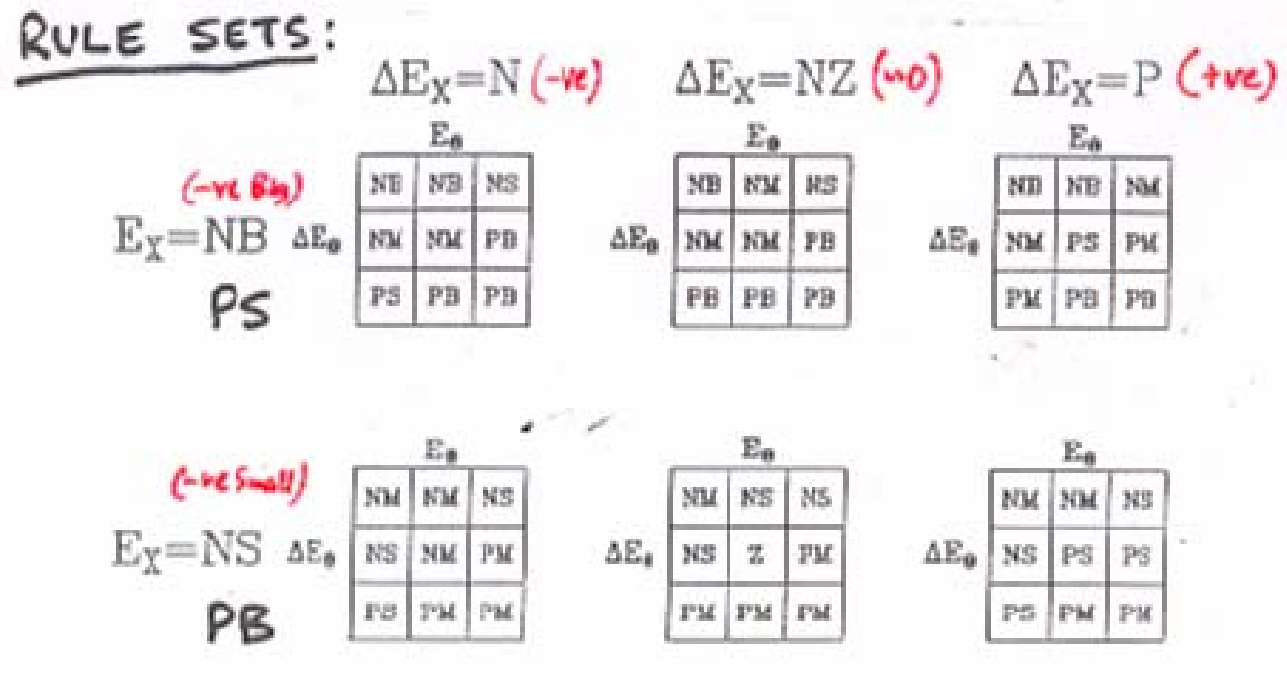
GAs for Optimal Fuzzy Controller

- Karr (1991): Cart pole balancing
- Four state variables: x,v,θ,α
- One action variable: F
- Optimal membership functions for maximum performance



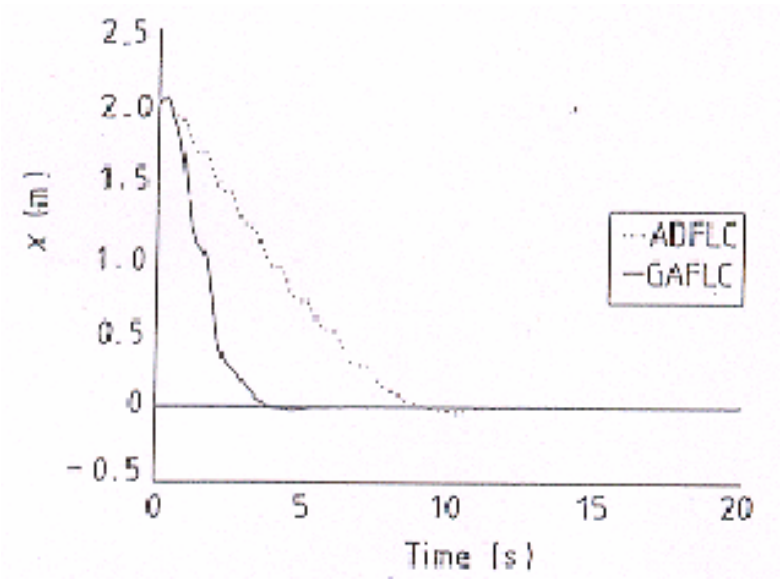
GAs for Optimal Fuzzy Controller (cont.)

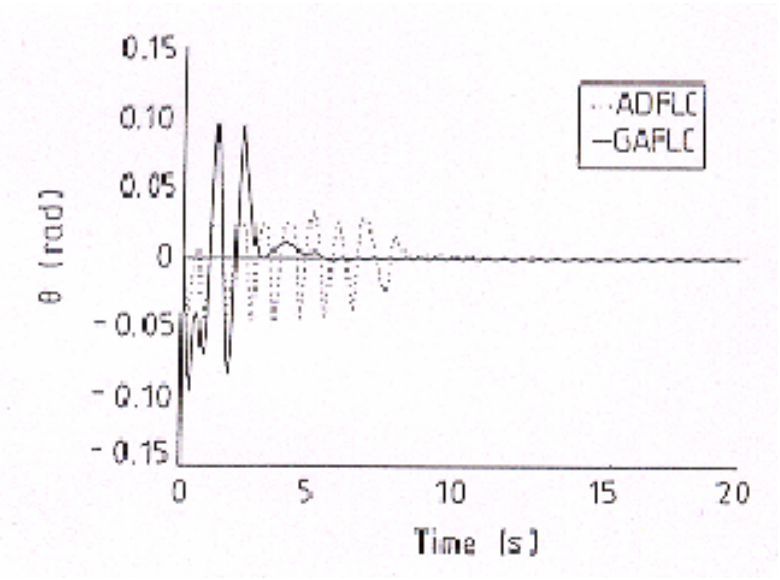
- x has 4 possibilities; others 3 each
- Total 4x3x3x3=108 rules (fixed)



GAs for Optimal Fuzzy Controller (cont.)

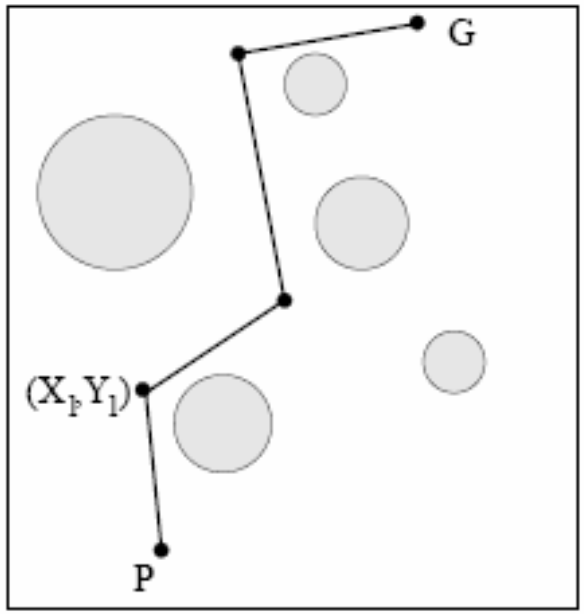
- Objective: $\sum_{\text{case } 1}^{\text{case } 4} \sum_{t=0}^{20} |x_{ij}| + 10|\theta_{ij}|$
- 4-bits for each variable (total 60 bits)
- Compared with author-defined rule base





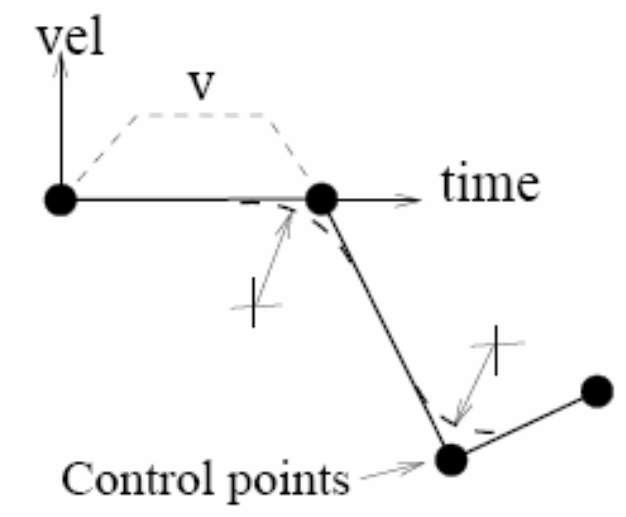
GAs with Fuzzy Evaluation

- Help reduce computational time
  - Can be applied to computationally expensive problems
- Example: Navigation among static obstacles
- Location of obstacles are known a priori
- Find a path which minimizes total traveling time by avoiding obstacles



Robot Navigation on Static Obstacles

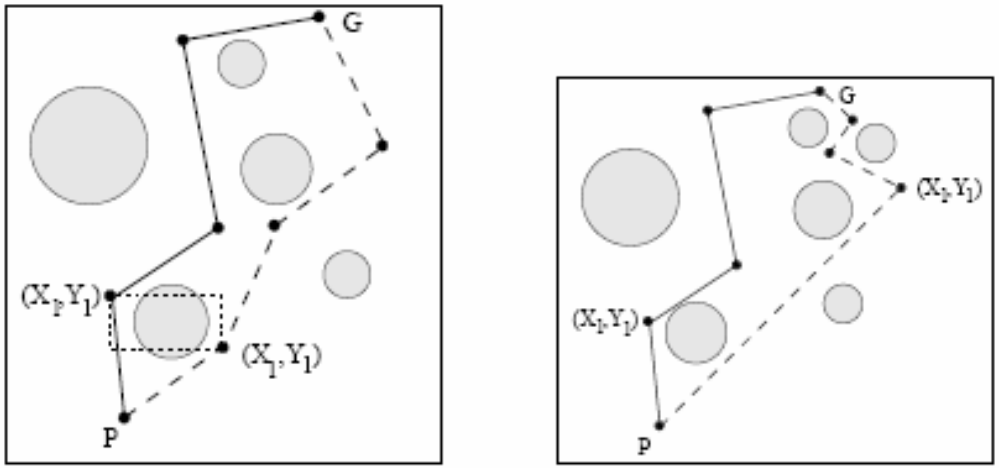
- A path contains straight lines and curves defined by a number of control points
- On each line segment, robot accelerate (a), maintain a velocity v, and then decelerate (-a)
- Curves with a fixed radius used to make a smooth transition
- Minimize Time: D/v+v/a
  - D: Total distance travelled
  - Penalty for collison



Static Obstacles (cont.)

- (X_i,Y_i) are coordinates of control points
- A naive representation:

  (X_1,Y_1), (X_2,Y_2), …, (X_N,Y_N)
- Although parents are feasible, children may not
- Coordinates of two parents may represent different region in the search space

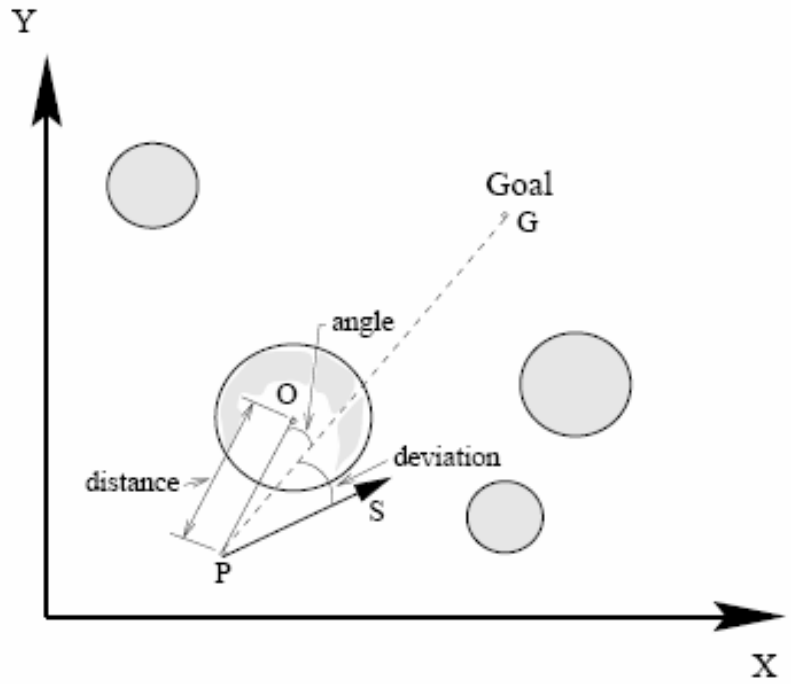


Representation and FLC

- Distance along line segments are coded, along with velocities and accelerations

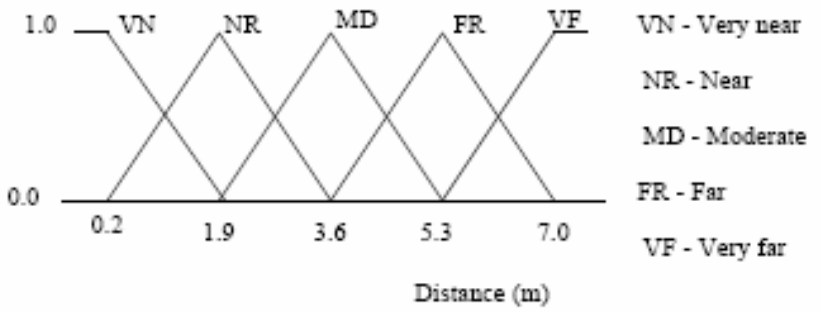
(d_1,d_2,…,d_N | v_1,v_2,…,v_N | a_1, a_2,…,a_N)

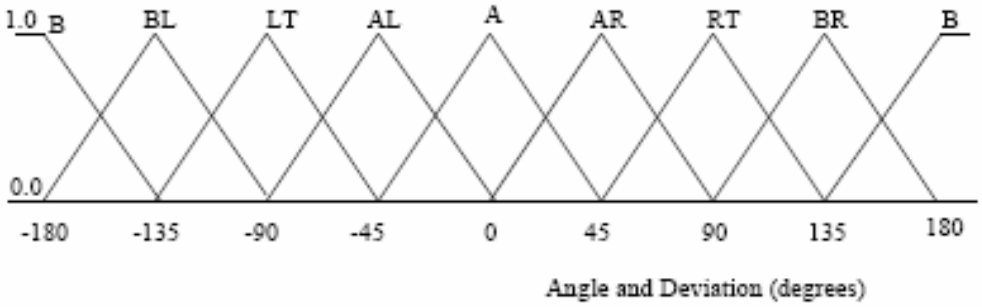
- Obstacle-free directions are found using Fuzzy logic controller (FLC)
- GAs are used to find optimal d_j, v_j and a_j



Fuzzy Membership Functions

- Two input: distance and angle
- One output: deviation angle from target path





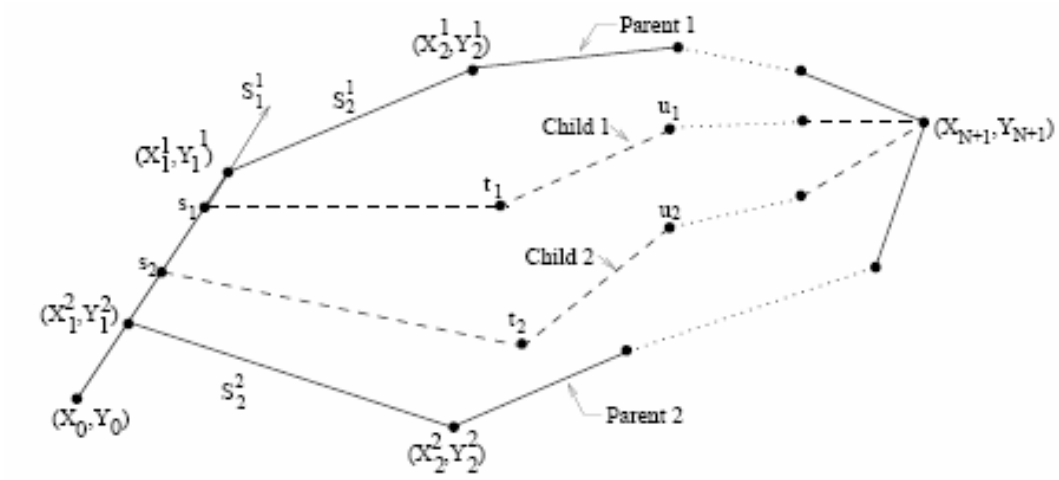
A - Ahead
AR - Ahead right
RT - Right
BR - Behind right
B - Behind
AL - Ahead left
LT - Left
BL - Behind left

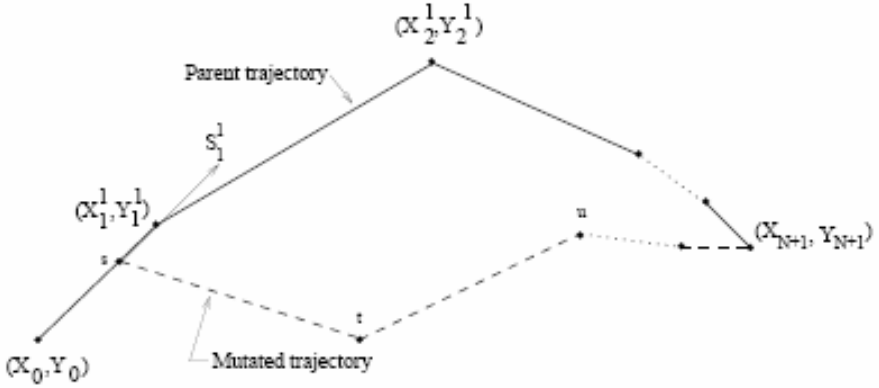
distance \ angle

	A	AL	LT	BL	B	BR	RT	AR
VN	RT	AR	A	A	A	A	A	AL
NR	AL	A	A	A	A	A	A	AL
MD	AR	A	A	A	A	A	A	A
FR	AR	A	A	A	A	A	A	A
VF	A	A	A	A	A	A	A	A

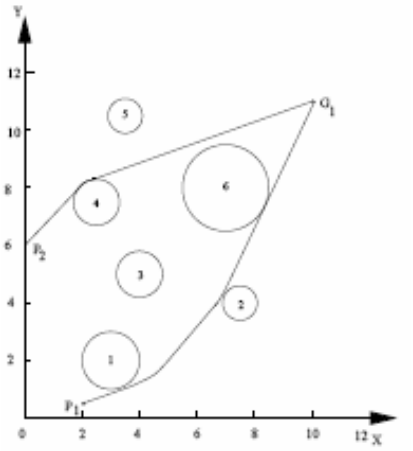
Modified GA Operators

- d_1^1 and d_1^2 are blended to find s1 and s2
- FLC finds two safe directions
- d_2^1 and d_2^2 SBX-ed to find t1 and t2
- Mutation: similar
- Problem-specific GA operators

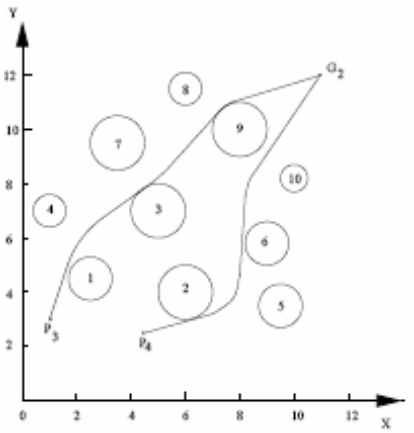


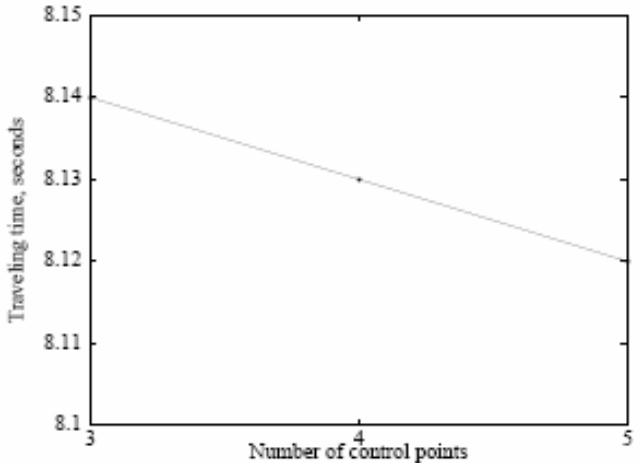


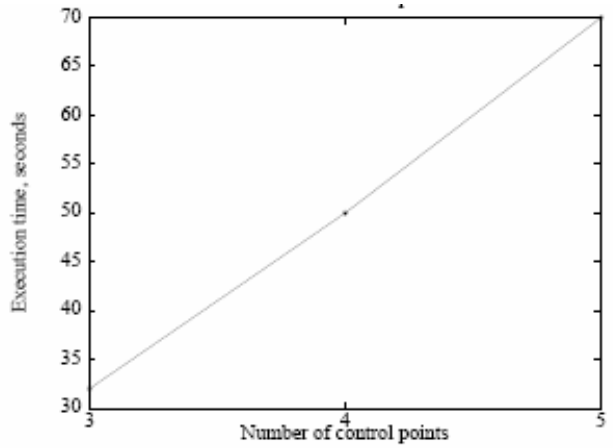
Simulation Results



Sl. No.	Case	Path		Traveling time, T (sec)	
		Start	Goal	Proposed	T-graph & A*
1	1	P_1	G_1	7.93	7.86
2	1	P_2	G_1	6.72	6.65
3	2	P_3	G_2	8.14	8.02
4	2	P_4	G_2	7.31	6.81

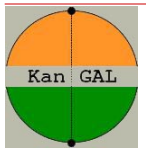






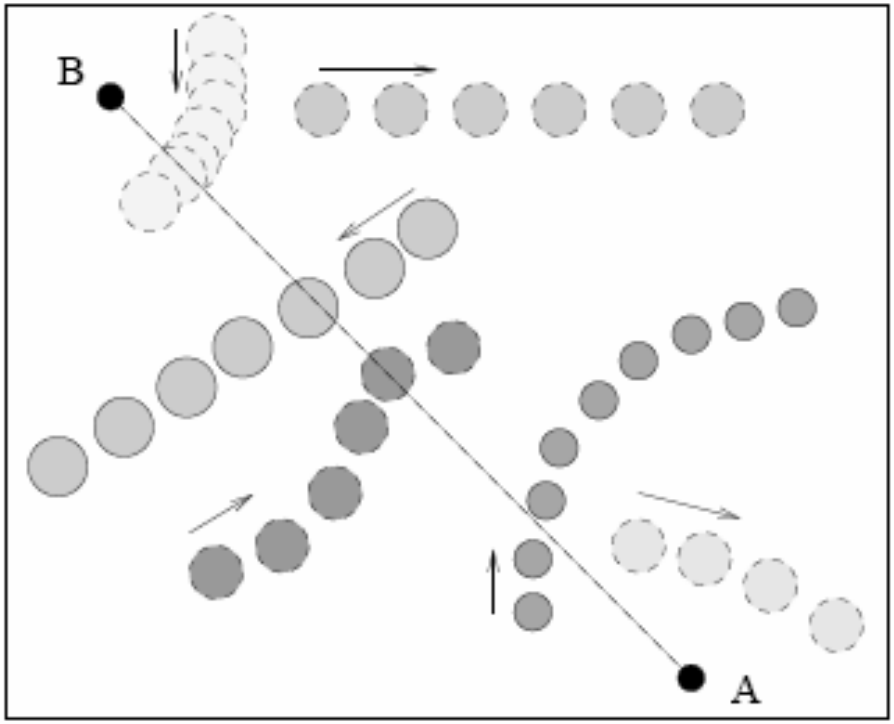
Lessons Learned

- Fuzzy logic controller to find safe directions quickly
- Genetic algorithms to find distance traveled along those directions
- Innovative GA operators maintain obstacle-free solution
- Tangent graph with A* best on circular obstacles and has quadratic
- time complexity
- Fuzzy-GA approach finds near-optimal solution with linear time complexity
- For large number of obstacles, fuzzy-GA approach will pay off



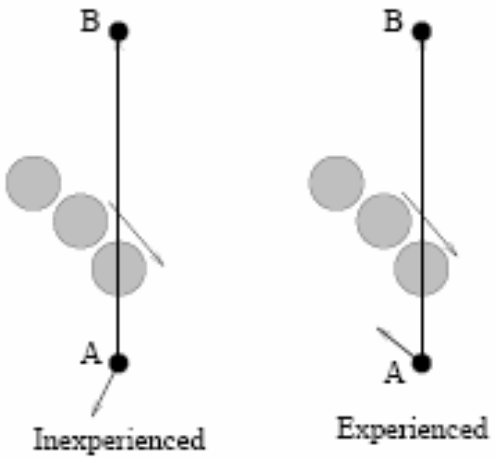
Robot on a Dynamic Environment

- A robot has to move from Point A to Point B
- By avoiding moving obstacles with unknown velocity and direction
- In a quickest possible way



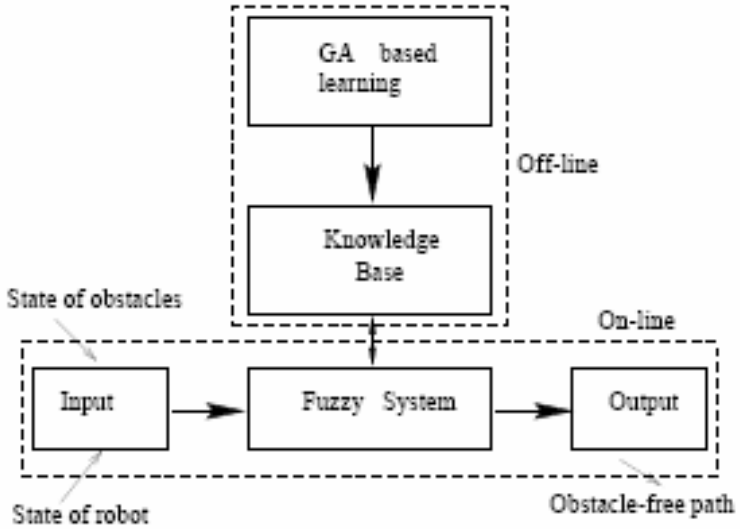
Natural Metaphor

- Decompose into two tasks:
  - Learn to find any obstacle-free path
  - Learn to find quickest obstacle-free path
- Natural metaphors:
  - Similar to a child's learning to avoid obstacles: Avoidance is important, not by precise angles
  - Similar to an adult's learning to avoid obstacles: Learning through experience



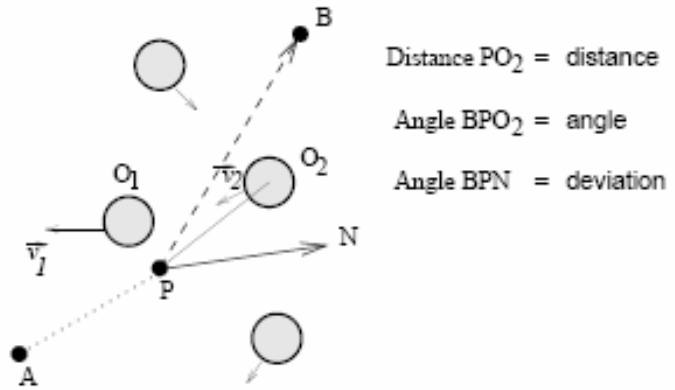
Genetic-Fuzzy Approach

- First task through fuzzy rules, second task through GAs
- An optimal FLC is found off-line by simulating on *S* scenarios of moving obstacles
- Optimal FLC becomes the brain of robot
- Navigate on-line using optimal FLC and current state space



Fuzzy Rule Base

- Input: distance, angle
- Output: deviation angle
- Fuzzy rule base matrix user-specified



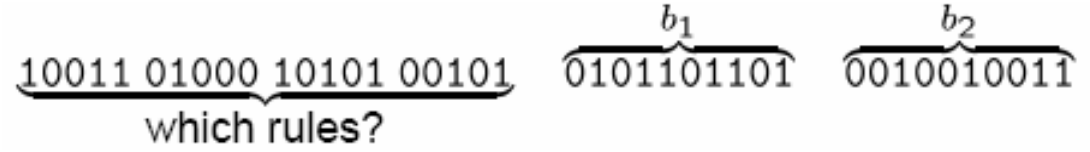
☞ distance: VN, N, F, VF; angle: L, AL, A, AR, R; deviation: L, AL, A, AR, R

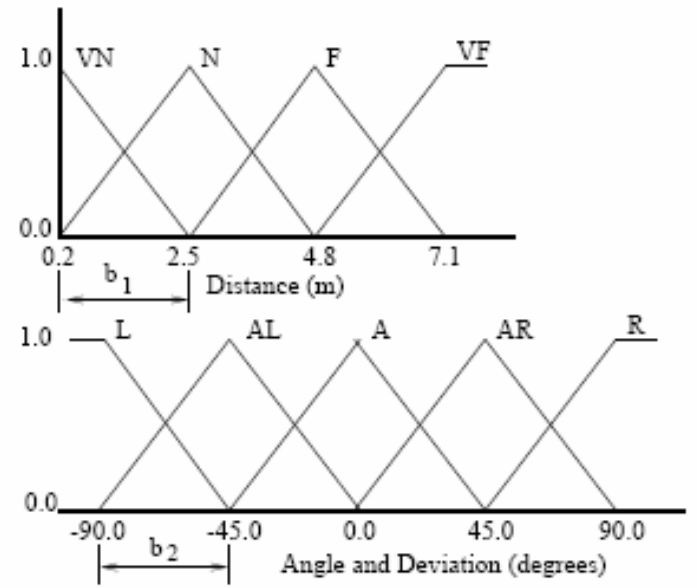
distance \ angle	L	AL	A	AR	R
VN	A	AR	AL	AL	A
N	A	A	AL	A	A
F	A	A	AR	A	A
VF	A	A	A	A	A

If distance is VN & angle is A, then deviation is AL

GA Representation

- Fuzzy membership function
- A typical solution:



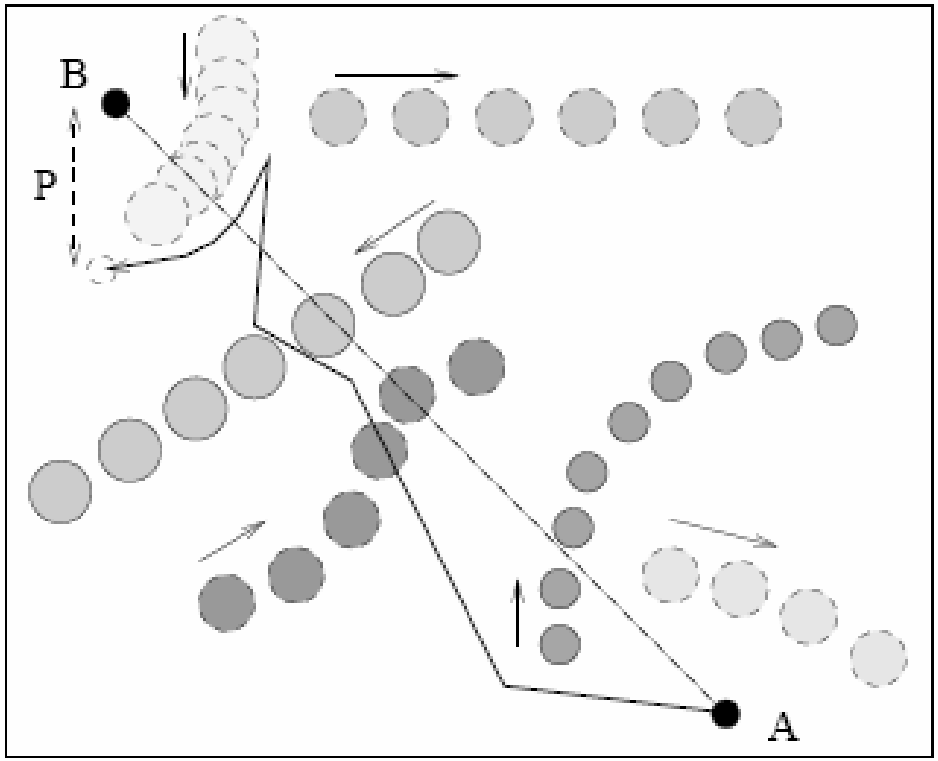


distance \ angle	L	AL	A	AR	R
VN	A			AL	A
N		A			
F	A		AR		A
VF			A		A

- b1, b2 and rules are evolved using a BGA

Evaluation of a Solution

- Simulate motion of a robot in *S* scenarios
- A maximum time T_{max} allowed to reach goal
- A penalty proportional to time to travel Euclidean distance from final point till destination is added



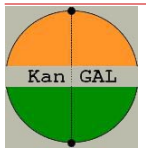
Simulation Results

- 60 population members, 100 iter.
- S=10
- Scenarios 4-6 are not used during optimization
- Optimized solutions are better

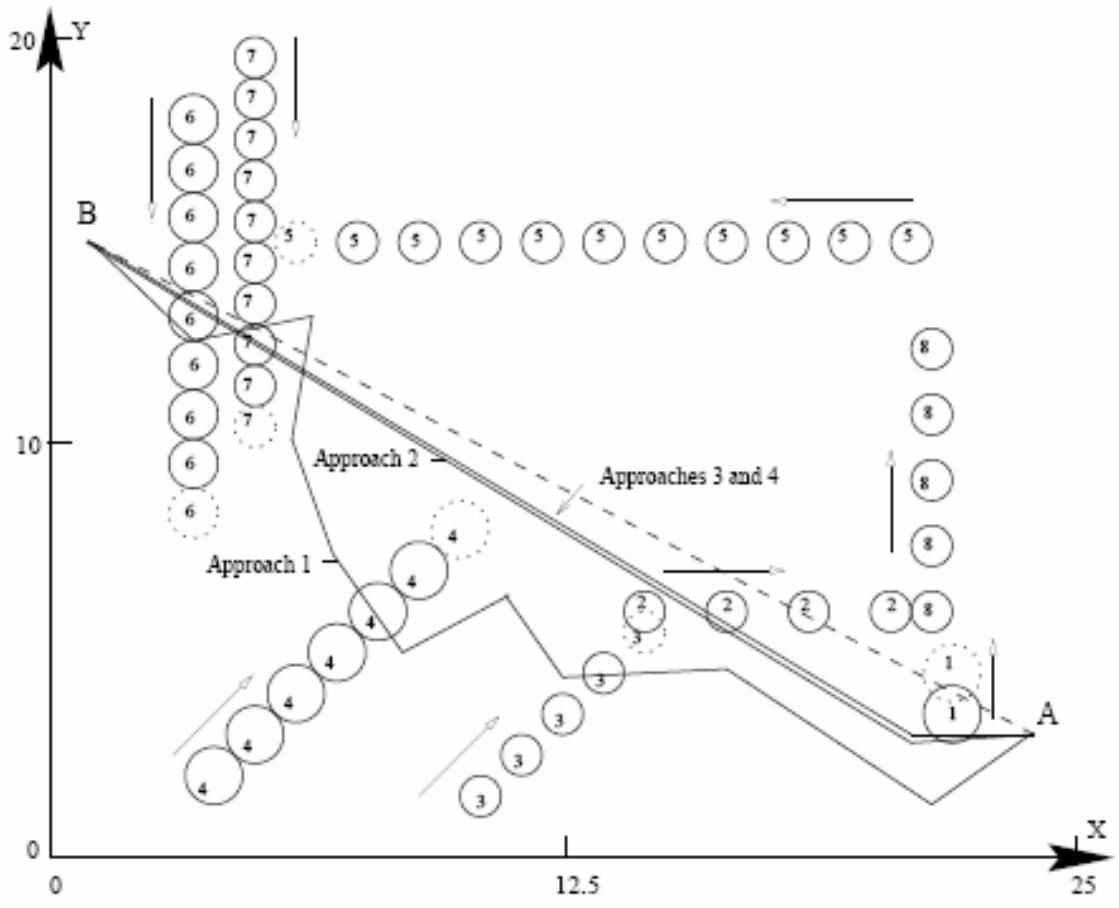
Rule Base	Membership Functions Optimized	
Optimized	No	Yes
No	Approach 1	Approach 2
Yes	Approach 3	Approach 4

	Approach 1		Approach 2	
Scenario	D	T	D	T
1	27.203	28.901	26.077	27.769
2	26.957	28.943	25.966	27.622
3	29.848	36.798	28.623	35.164
4	33.465	43.365	26.396	27.907
5	32.836	41.781	27.129	33.000
6	33.464	43.363	28.001	31.335

	Approach 3		Approach 4	
Scenario	D	T	D	T
1	26.154	27.872	26.154	27.872
2	26.026	26.546	26.026	26.546
3	26.660	34.547	27.139	35.000
4	26.243	27.512	26.243	27.512
5	26.543	32.390	27.041	33.000
6	27.164	31.000	27.164	31.000



Navigation Through Unknown An Scenario



Discussions

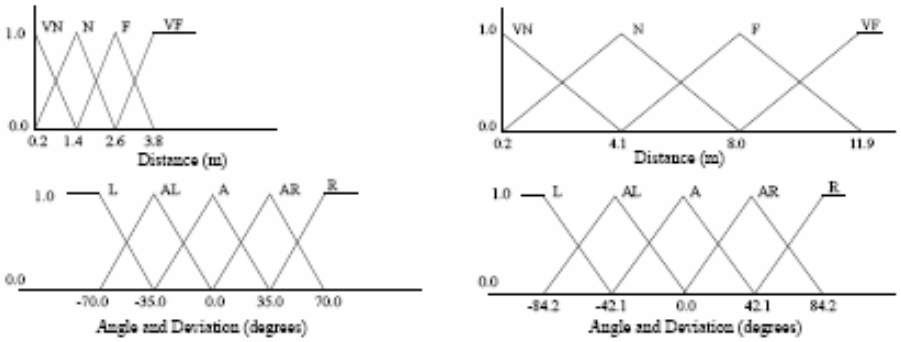
- Approach 4:
  - Elongates MF for distance to have about one rule per distance value
- Approach 2:
  - Have shortened MF for distance so mostly move ahead

☞ Approach 3 versus Approach 4

distance \ angle	L	AL	A	AR	R
VN					
N		A		A	
F	A	A			
VF	A	A	A	A	A

distance \ angle	L	AL	A	AR	R
VN		AR			
N	A	A			
F				A	
VF				A	

☞ Approach 2 versus Approach 4



Automatic Rule Generation

- Base rule base is not predefined
- For each matrix element, 4 options
  - One bit for presence or absence of the rule
  - Two bits to define deviation
  - Example: distance = VN, angle = AL
  - 100: L, 101: A, 110: AR, 111: R
- 60 bits to represent a GA solution
- Approach 5

distance \ angle	L	AL	A	AR	R
VN	A	AR	AL	AL	A
N	A	A	AL	A	A
F	A	A	AR	A	A
VF	A	A	A	A	A

Results

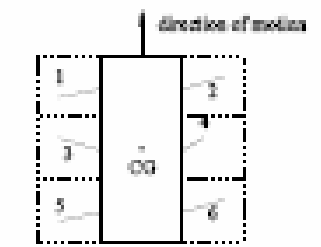
- Better solutions through Approach 5,
- Discovery: when object is in left move ahead or right, and vice versa

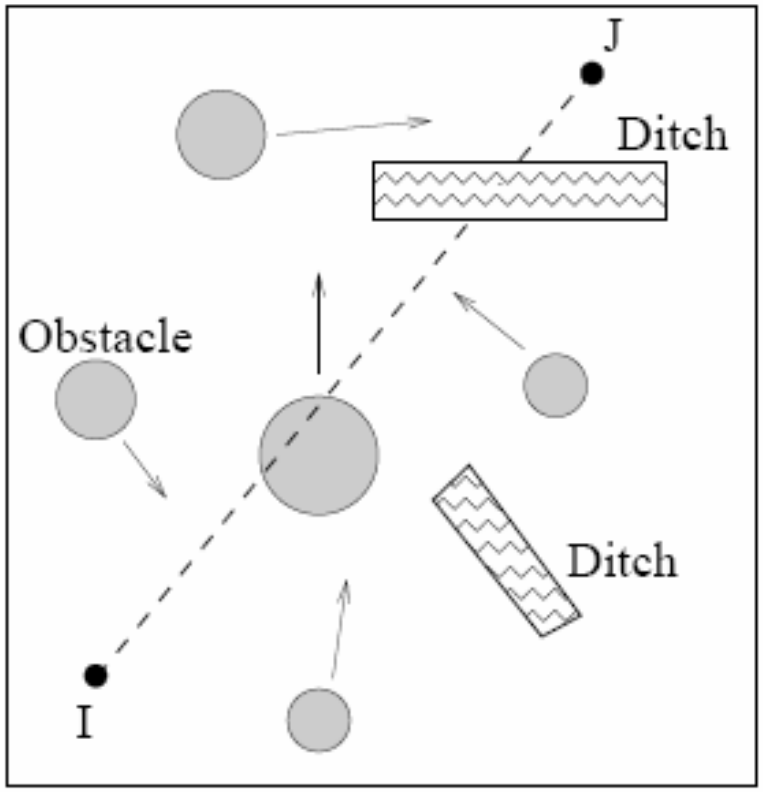
Scenario	Approach 3		Approach 5	
	D	T	D	T
1	26.154	27.872	26.154	27.872
2	26.026	26.546	26.026	26.546
3	26.660	34.547	27.634	31.592
4	26.243	27.512	26.243	27.512
5	26.543	32.390	26.528	32.370
6	27.164	31.000	27.164	31.000
F		24.091		23.717

distance \ angle	L	AL	A	AR	R
VN	A	AR		A	
N		A			L
F		AR		A	AL
VF		A			A

Six-Legged Robot Crossing a Ditch

- Path generation of a legged robot
- Gait generation of a legged robot
  - Periodic gait
  - Ditch-crossing gait
  - Turning gait
- Travel from I to J in a minimum time by avoiding all hurdles

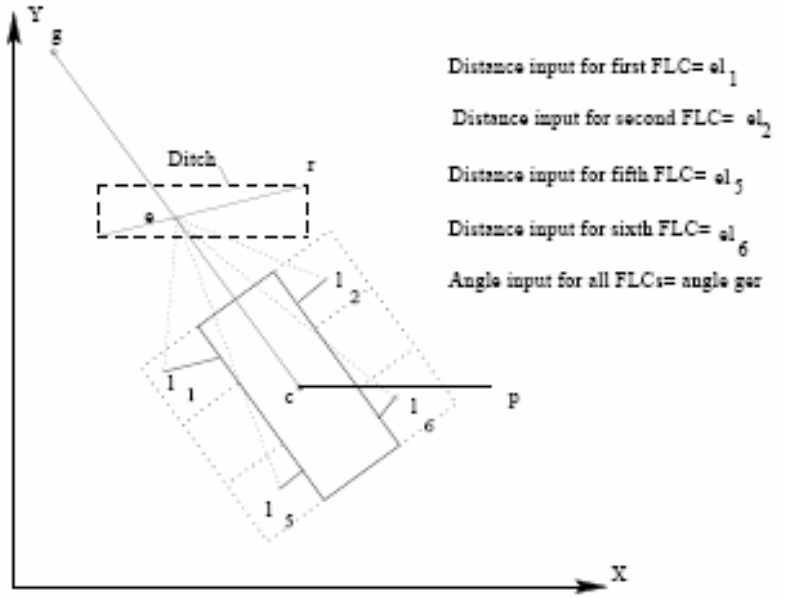


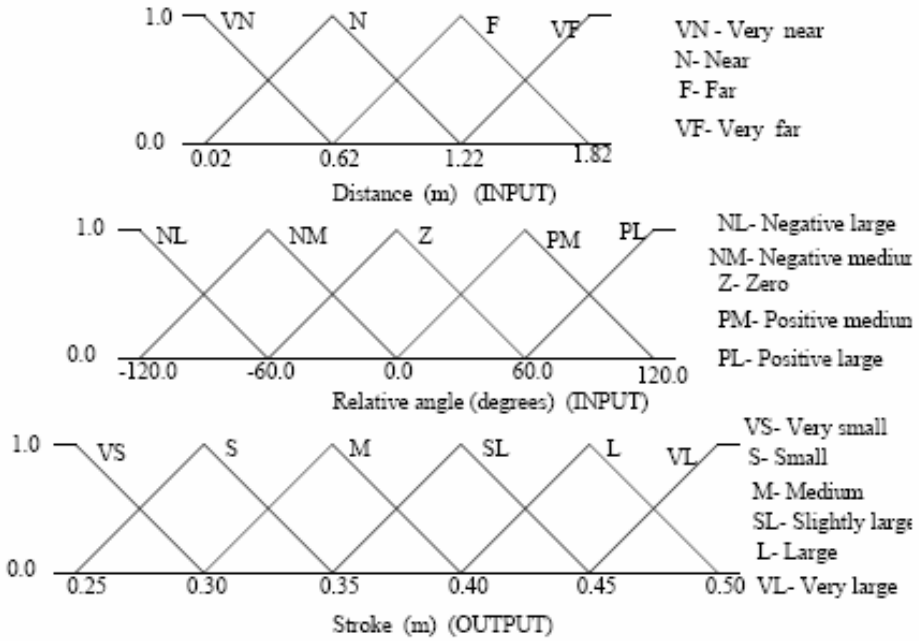


Fuzzy Rule Base for Ditch Crossing

- Input: Distance and relative angle
- Output: Leg stroke

distance \ relative angle	NL	NM	Z	PM	PL
VN	M	SL	L	SL	M
N	S	SL	L	SL	S
F	S	M	L	M	S
VF	S	S	SL	S	S

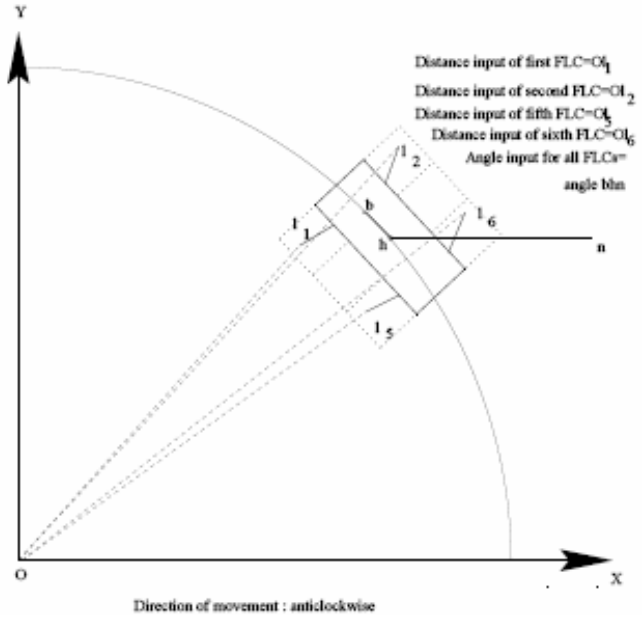


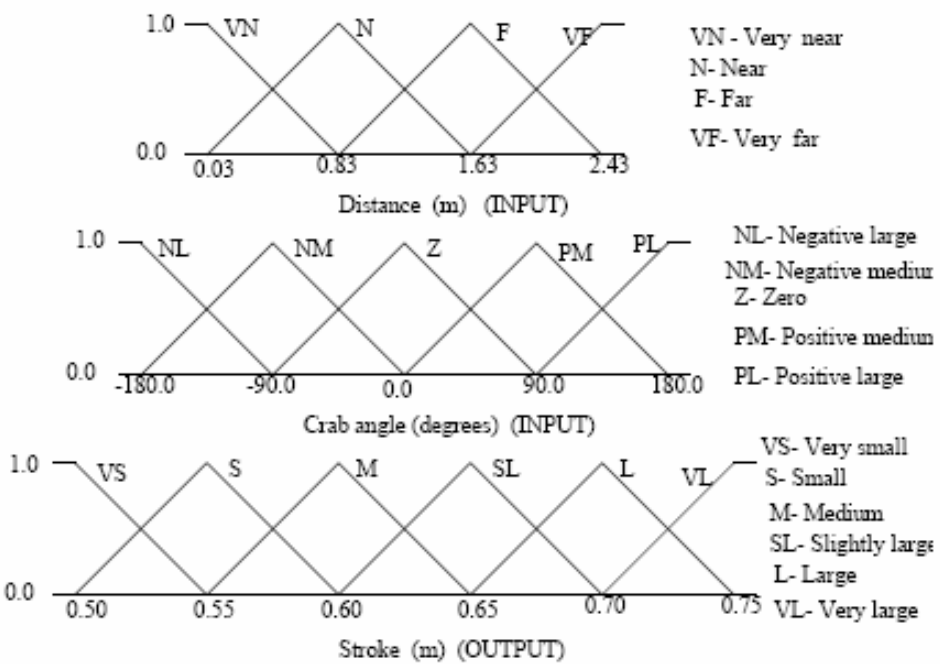


Turning Gait

- Input: Distance and crab angle
- Output: Stroke

distance \ crab angle	NL	NM	Z	PM	PL
VN	VS	S	M	S	VS
N	S	M	SL	M	S
F	M	SL	L	SL	M
VF	SL	L	VL	L	SL

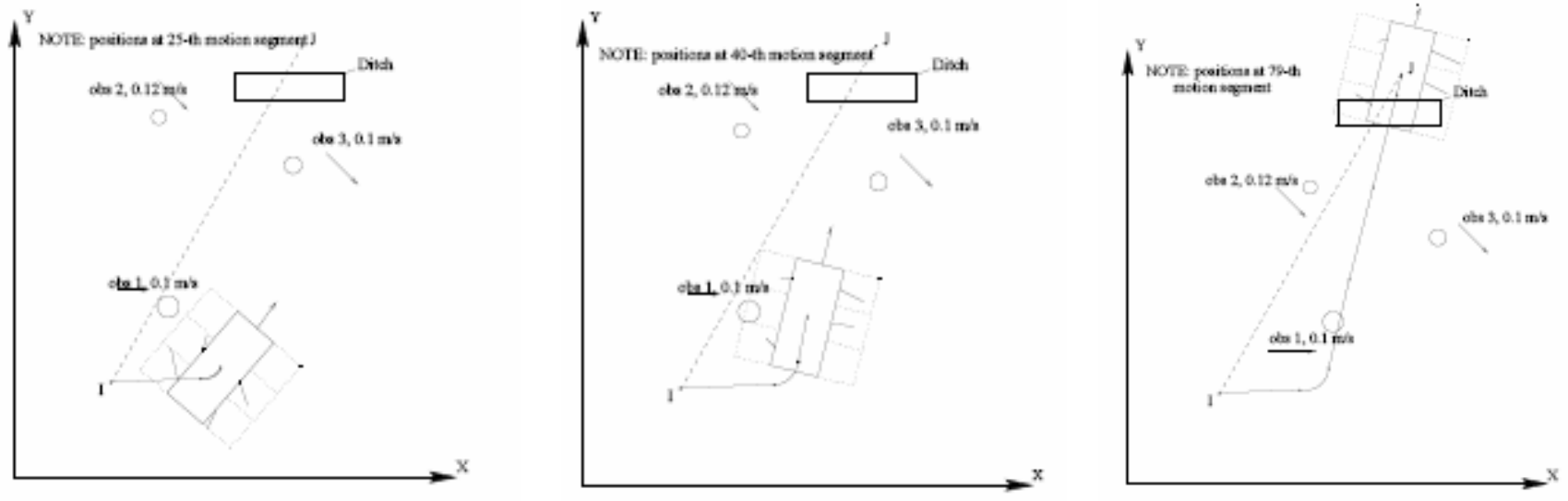




Simulation Results

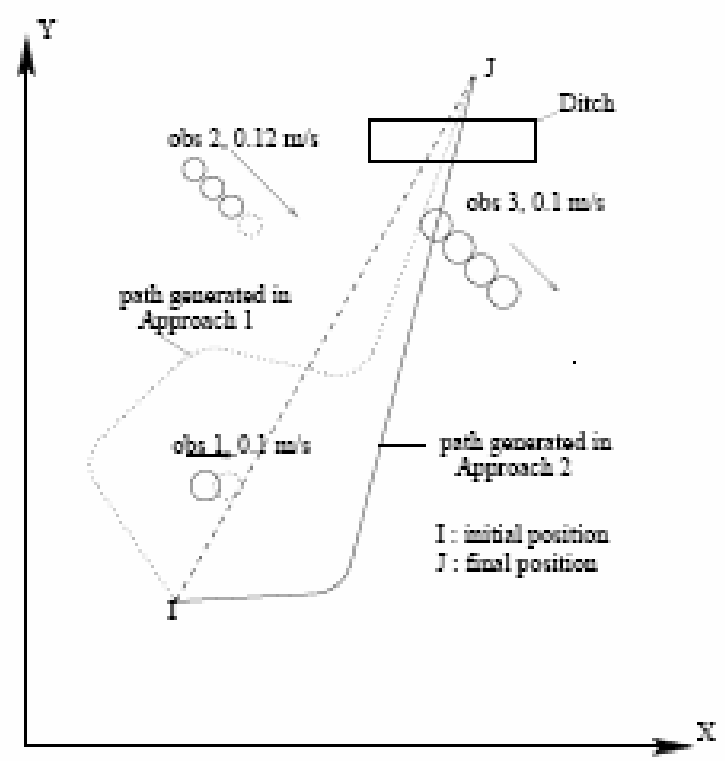
- Overall 260-bit GA string

$$\underbrace{10111\ldots1}_{\text{path}}\underbrace{01100\ldots01100111}_{\text{ditch crossing gait}}\underbrace{11001\ldots10001101}_{\text{turning gait}}$$



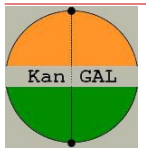
Simulation Results (cont.)

- Author-defined FLCs (approach 1) may not be optimal
- Approach 1 does not find a stable solution for 6th scenario
- GA finds 115 (6 and 109) out of 260 rules necessary
- Approach 2 takes 79 segments, while approach 1 takes 120



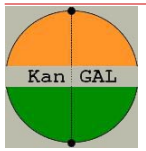
Conclusions of Lecture 12

- Genetic Programming good for
  - Regression, automatic discovery
  - Computational issues
- Self-adaptive EAs more powerful
- Scheduling EAs exist and applied
- EAs are ideal and complimentary to ANN and Fuzzy approaches
- They are better applied by incorporating problem knowledge



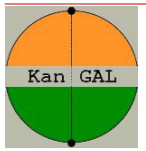
Concluding Remarks of the Course

- GAs were around for past two decades
- Major contributions so far
  - Population based approach for optimization is revolutionary
    - Constraint handling, coevolution etc.
  - Multi-modal and multi-objective optimization
  - Combined (discrete+continuous) problem solving
  - Flexibility and ease of use



Concluding Remarks (cont.)

- EAs are here to stay
- More attention needed
  - Theoretical understanding
  - Unified approaches to various EAs
  - More case studies
  - Hybrid approaches with other methods
  - More software development
  - Experimental EAs



Happy EA Computing …

- Thanks for your participation and patience
- In future, contact:
  - Email: deb@iitk.ac.in
  - Web: http://www.iitk.ac.in/kangal/
  - Technical Reports: http://www.iitk.ac.in/kangal/pub.htm
  - Computer Codes http://www.iitk.ac.in/kangal/soft.htm

