Chapter 4
GAs: Selected Topics

Classical/Simple GA

Representation	Binary strings
Recombination	One-point with fixed probability
Mutation	Bit-flipping
Survivor selection	Fitness-Proportionate
Generation	All children replace parents

Why GA cannot to find the optimal solutions in the practical applications?

```
Procedure  Genetic Algorithm
Begin
    t ← 0
    Initialize P(t)
    Evaluate P(t)
    While (not termination-condition) do
    Begin
          t ← t + 1
          Select P(t) from P(t -1)
          Alter P(t)
          Evaluate P(t)
    End
End
```

- *Encoding problem*
- *Limit population size*
- *Limit number of iterations*
- …

Contents

```
Procedure  **Genetic Algorithm**
Begin
    t ← 0
    Initialize P(t)
    Evaluate P(t)
    While (not termination-condition) do
    Begin
        t ← t + 1
        Select P(t) from P(t -1)
        Alter P(t)
        Evaluate P(t)
    End
End
```

A. Sampling Mechanism

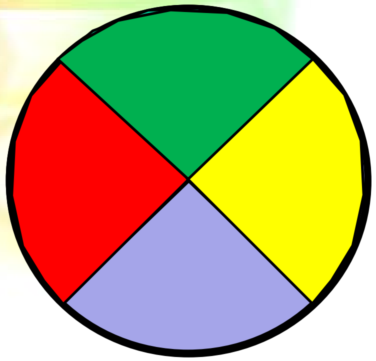
Slow and Premature Convergence

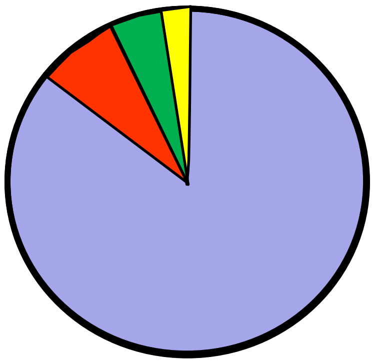
- Exploreing the search space (population diversity)

  Too much → random search → slow convergence

- Exploiting the best individuals (selective pressure)

  Too much → Hillclimbing → premature convergence

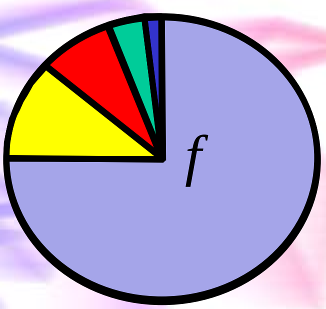




Survivor/Selection Mechanism

- Fitness-Based Selection
  - Proportionate-Based
  - Scaling-Based
  - Rank-Based
- Modified Genetic Algorithm
- Age-Based Selection

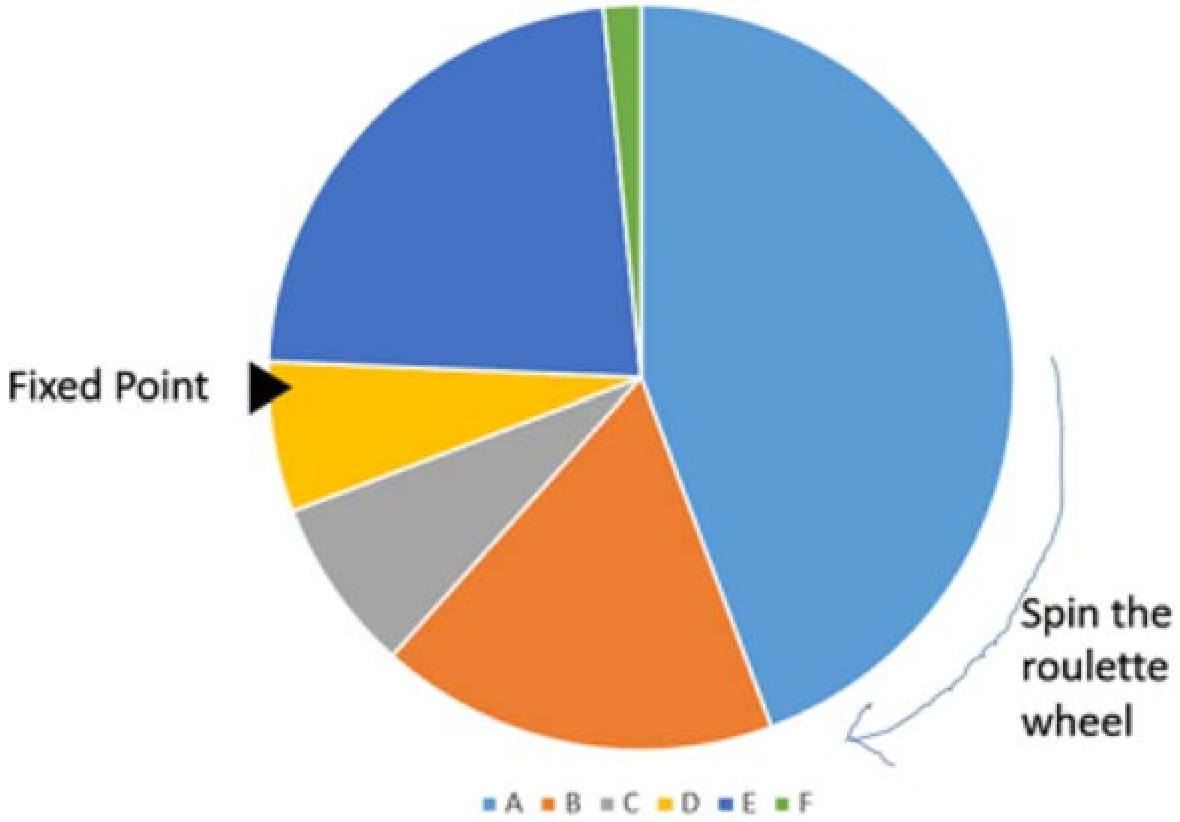
Proportionate-Based Selection



1. Proportional selection
2. Truncation selection
3. Brindle’s remainder stochastic sampling
4. Stochastic universal sampling
5. Crowding factor model (CF)
6. Tournament selection
7. Boltzmann tournament

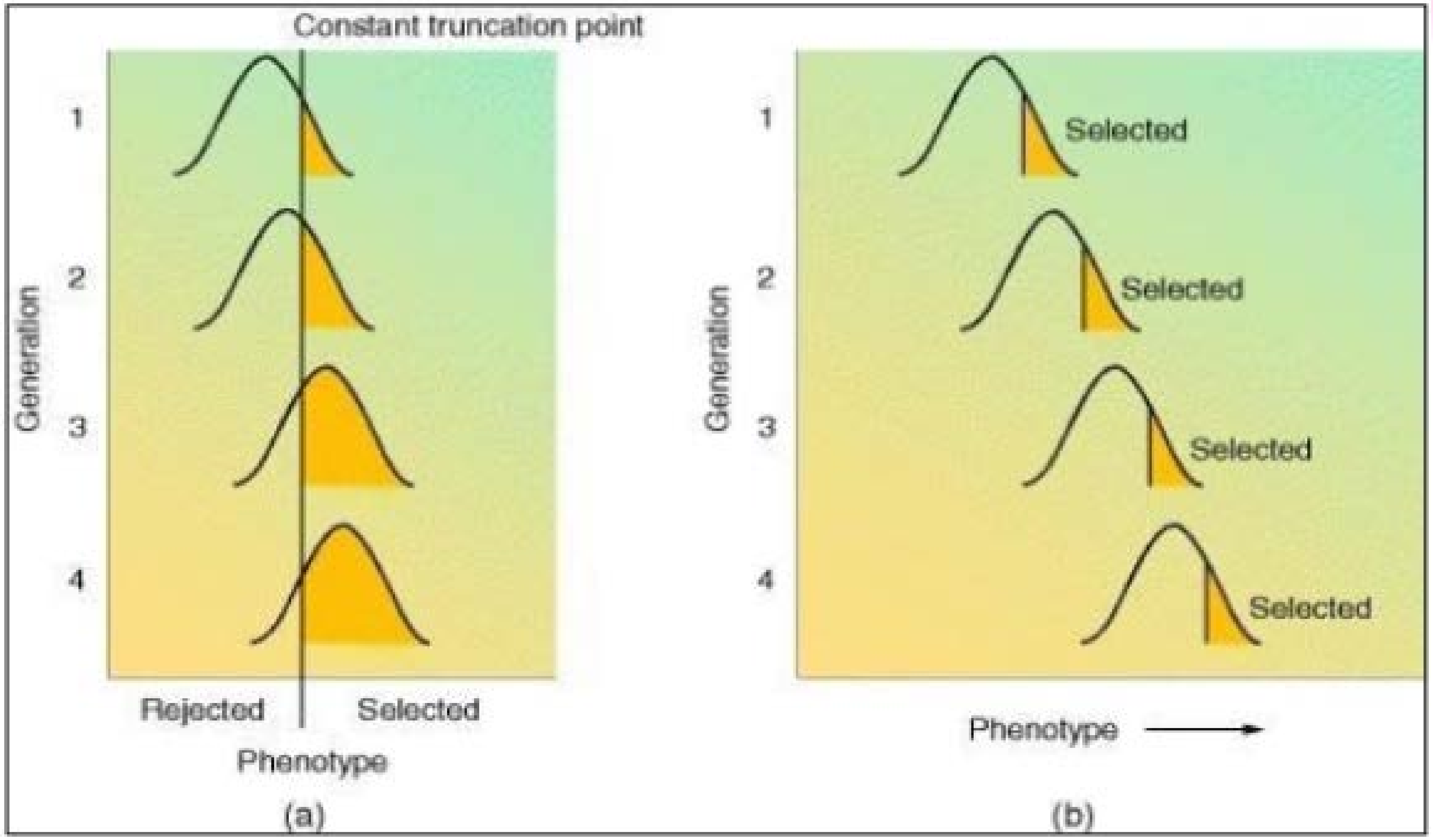
1. Proportional selection (SP_k)

(Elitist model)



Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

2. Truncation selection

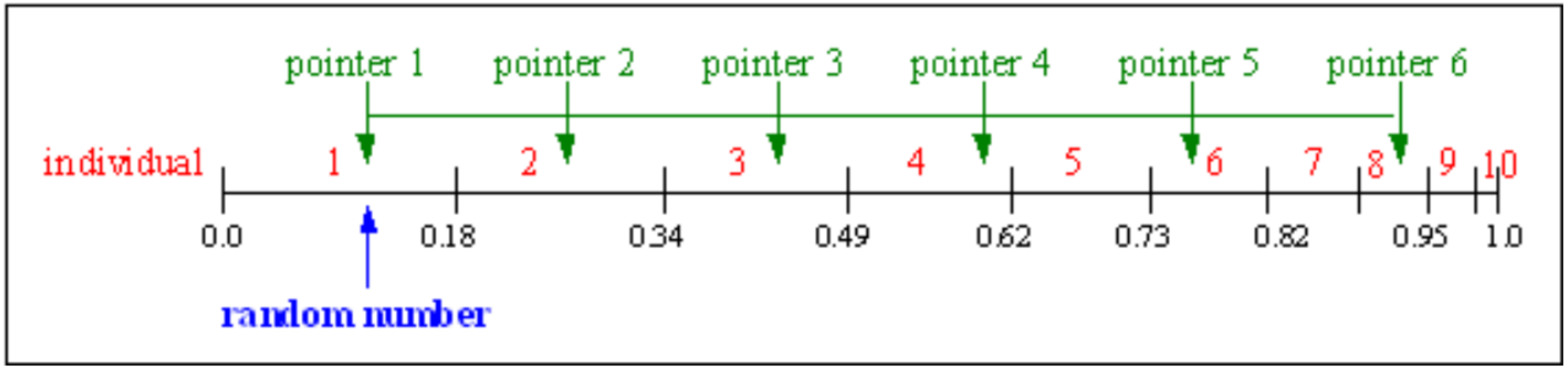


3. Brindle's remainder stochastic sampling

- $e_k = pop\_size \times p_k; \quad p_k = f_k \Big/ \sum_{j=1}^{pop\_size} f_j$
- The chromosome compete according to the fractional parts for the remaining place in the population.

4. Stochastic universal sampling

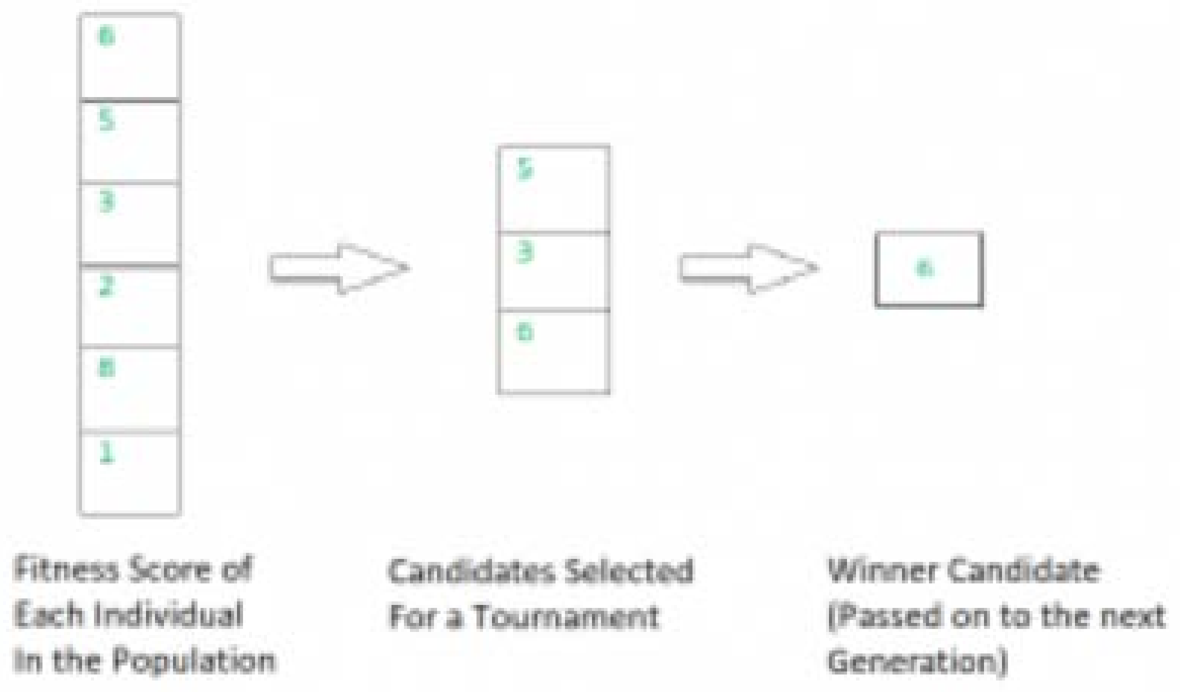
Number of individual	1	2	3	4	5	6	7	8	9	10	11
fitness value	2.0	1.8	1.6	1.4	1.2	1.0	0.8	0.6	0.4	0.2	0.0
selection probability	0.18	0.16	0.15	0.13	0.11	0.09	0.07	0.06	0.03	0.02	0.0



6. Crowding factor model (CF)

Crowding factor model An algorithm in which an offspring replaces a chromosome that closely resembles the offspring.

6. Tournament selection

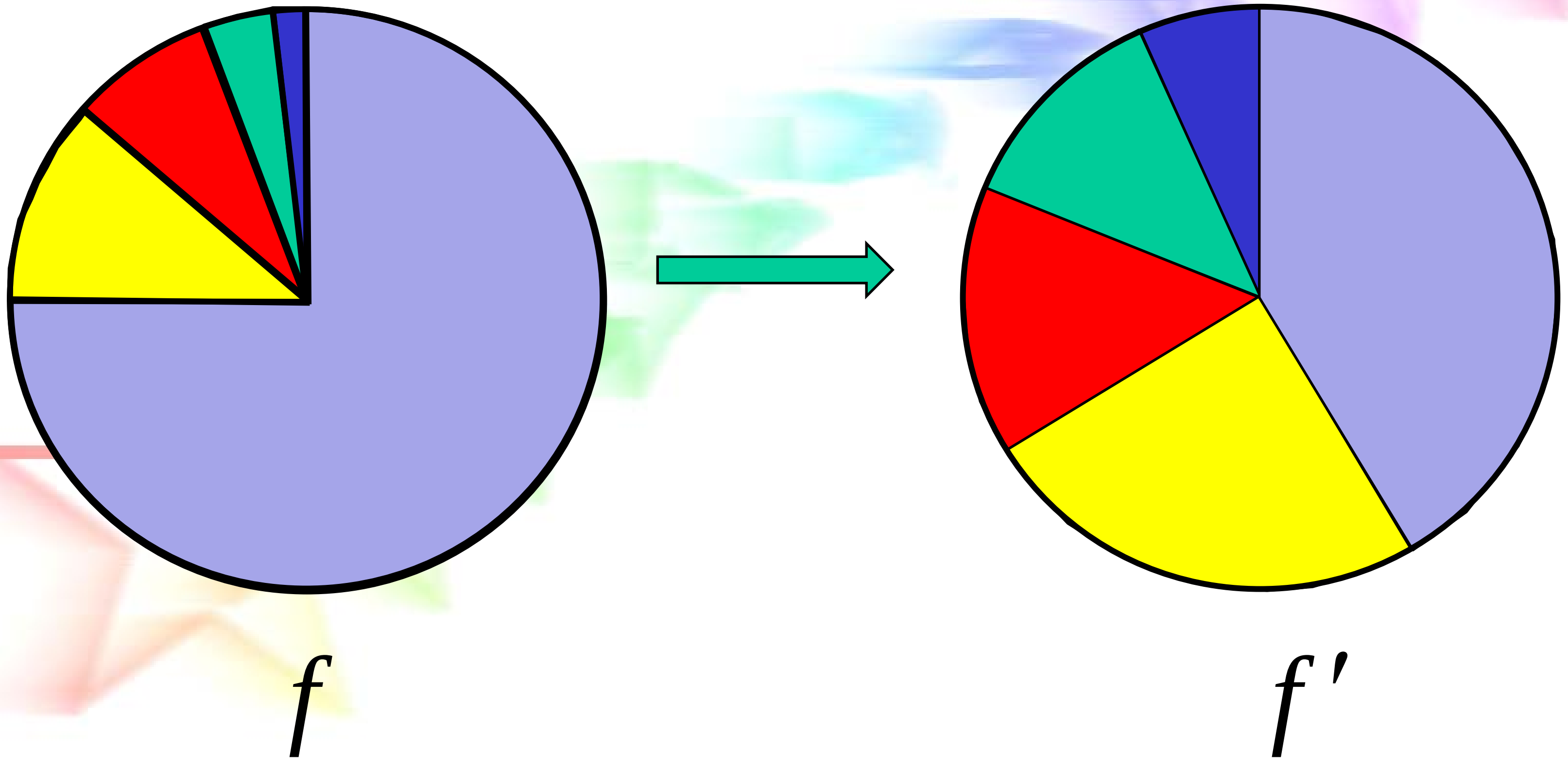

6
5
3
2
8
1
5
3
6
6
Fitness Score of Each Individual In the Population
Candidates Selected For a Tournament
Winner Candidate (Passed on to the next Generation)

7. Boltzmann tournament (tournament size = 2)

$$P(x) = \frac{1}{1 + e^{\frac{f(x) - f(y)}{T}}}$$ (for min problem)

f(x)	f(y)	d=(f(x)-f(y))	1/(1+exp(d/ 0.1))	1/(1+exp(d/ 0.01))	1/(1+exp(d/ 0.001))	1/(1+exp(d/ 0.0001))	1/(1+exp(d/ 0.00001))
0.00006	0.00007	(0.00001)	0.50002	0.50025	0.50250	0.52498	0.73106
0.00005	0.00007	(0.00002)	0.50005	0.50050	0.50500	0.54983	0.88080
0.00004	0.00007	(0.00003)	0.50007	0.50075	0.50750	0.57444	0.95257
0.00003	0.00007	(0.00004)	0.50010	0.50100	0.51000	0.59869	0.98201
0.00002	0.00007	(0.00005)	0.50012	0.50125	0.51250	0.62246	0.99331
0.00001	0.00007	(0.00006)	0.50015	0.50150	0.51500	0.64566	0.99753

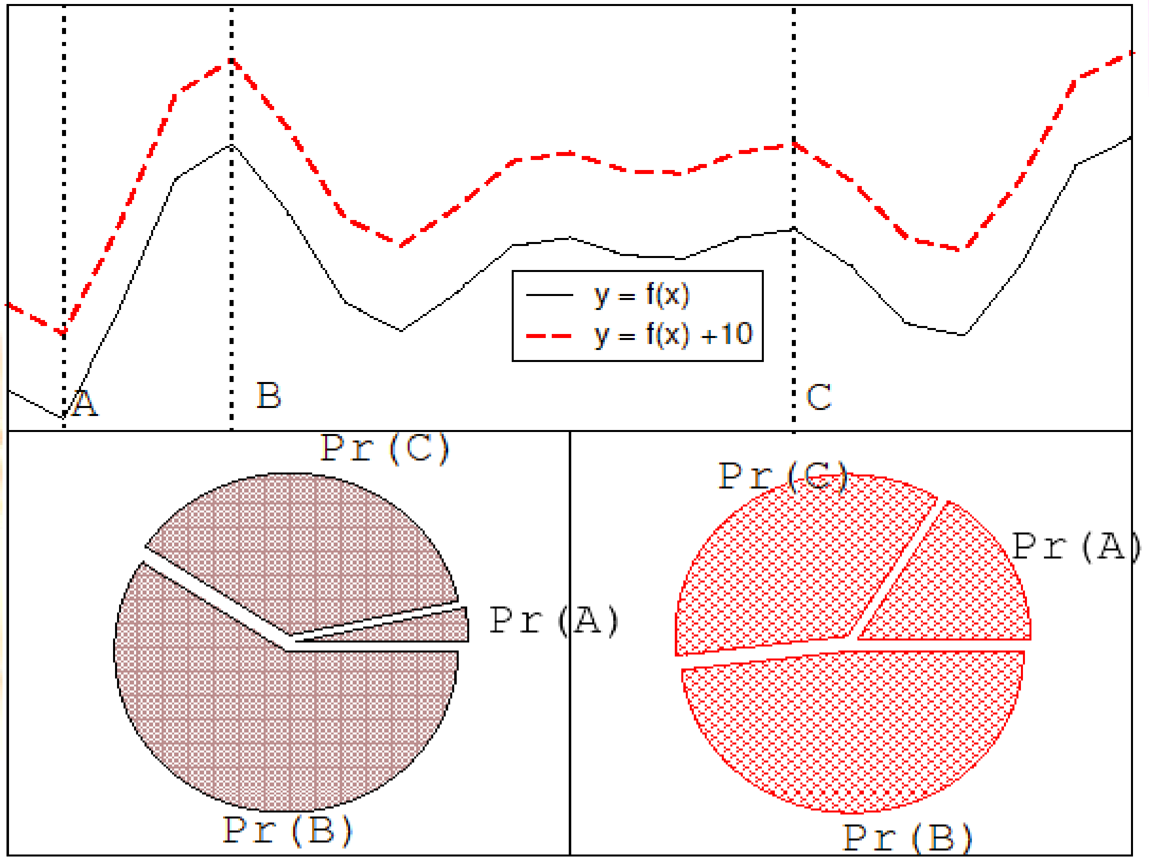
Scaling – Based Selection



Fitness-Proportionate Selection (FPS)

- Problems include
  - One highly fit member can rapidly take over if rest of population is much less fit
  - At end of runs when fitness are similar, lose selective pressure
  - Highly susceptible to function transposition
- Scaling can fix last two problems

Function transposition for FPS



Scaling – Based Selection

- Linear scaling

$$f_i' = a \times f_i + b \qquad (\hat{f}' = \hat{f})$$

where $a = \hat{f} \times \frac{c-1}{f_{\max} - \hat{f}}$ $\quad b = \hat{f}(1-a)$

- Sigma truncation

$$f_i' = f_i - (\bar{f} - c \times \sigma)$$
$$(f_i' = 0, \quad \text{if} \quad f' < 0)$$

where c is a constant(usually $1 <= c <= 3$), usually 2.

- Power law scaling

$$f_i' = f_i^{\,k}$$

where k should be problem depend, usually 1.005.

- Window Scaling

  - For maximum problem

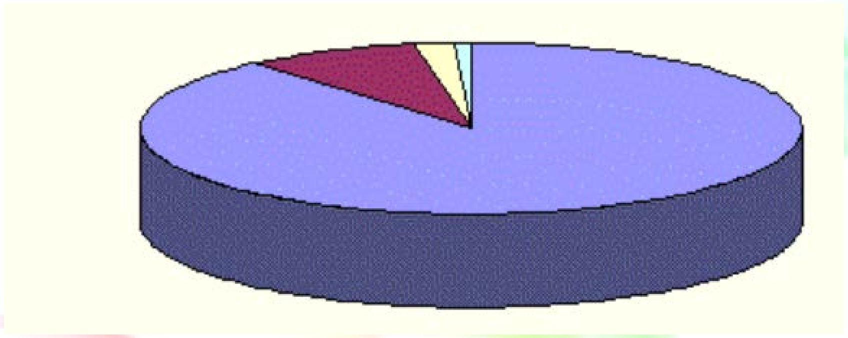
$$f'(x) = f(x) - \min_{y \in W} f(y)$$

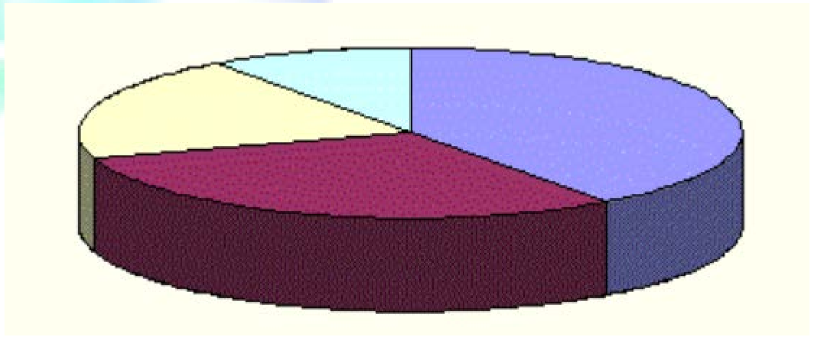
  - For minimum problem

$$f'(x) = \max_{y \in W} f(y) - f(x)$$

W : all individuals in the last *W* generations (*W* >= 0)

Rank – Based Selection





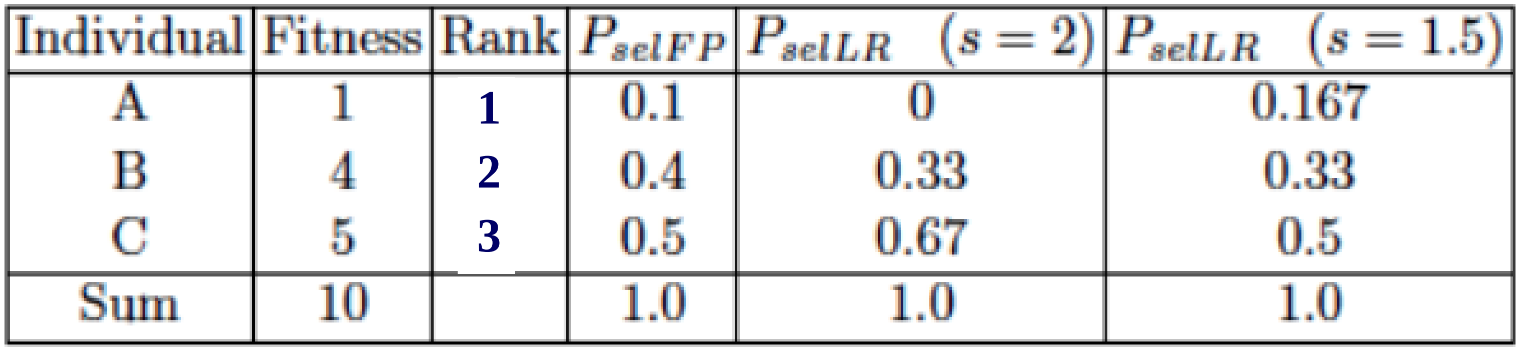
Rank – Based Selection

- Attempt to remove problems of FPS by basing selection probabilities on *relative* rather than *absolute* fitness
- Rank population according to fitness and then base selection probabilities on rank where fittest has rank μ and worst rank 1
- This imposes a sorting overhead on the algorithm, but this is usually negligible compared to the fitness evaluation time

Linear Ranking：

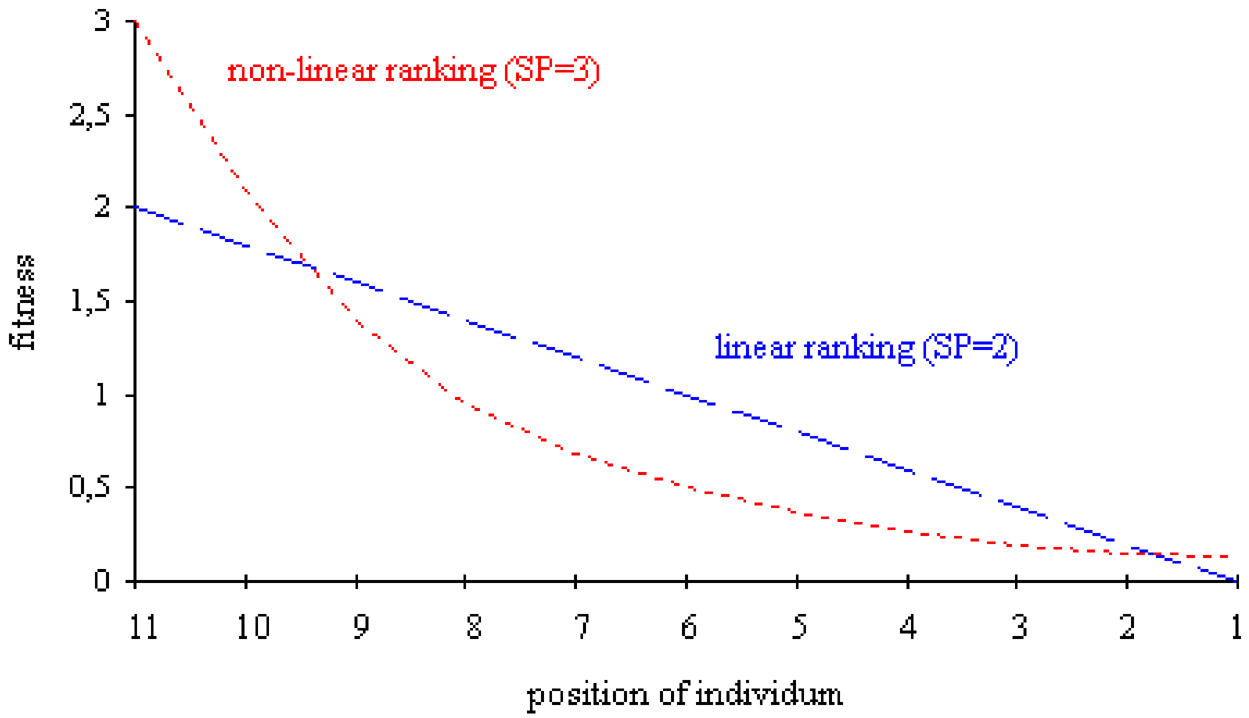
$$Prob(rank) = \frac{(2-s)}{N} + \frac{2(rank-1)(s-1)}{N(N-1)}$$

(s: $1.0 < s \leq 2.0$)



Individual	Fitness	Rank	P_{selFP}	P_{selLR} $(s=2)$	P_{selLR} $(s=1.5)$
A	1	1	0.1	0	0.167
B	4	2	0.4	0.33	0.33
C	5	3	0.5	0.67	0.5
Sum	10		1.0	1.0	1.0

Linear ranking vs. Nonlinear ranking :



$$Fitness(Pos) = 2 - SP + 2 \cdot (SP - 1) \cdot \frac{(Pos - 1)}{(Nind - 1)}$$

$$Fitness(Pos) = \frac{Nind \cdot X^{Pos-1}}{\sum_{i=1}^{Nind} X^{i-1}}$$

- **modGA Algorithm**

```
Procedure **modGA**
begin
    t ← 0
    initialize P(t)
    evaluate P(t)
    while (not termination-condition) do
    begin
        t ← t+1
        Select r parents from P(t-1) (non-distinct).
        Select (pop_size – r) distinct chromosomes
            from P(t-1) and copy them to P(t).
        Form P(t): r parents breed r offspring
        Evaluate P(t)
    end
end
```

B. Termination Condition

- Some number of evolution cycles
- The number of pre-defined fitness
- Some percentage of the converged alleles
- Some variation of individuals between different generations

C. Contractive Mapping GA

Contractive Mapping GA (CM-GA)

```
Procedure  CM-GA
Begin
    t ← 0
    Initialize P(t)
    Evaluate P(t)
    While (not termination-condition) do
    Begin (contractive mapping  f ( P(t) ) → P(t+1))
        t ← t + 1
        Select P(t) from P(t-1)
        Alter P(t)
        Evaluate P(t)
        if  Eval( P(t-1)) >= Eval( P(t) )
        then  t = t - 1
    End
End
```

$$Eval(P) = \frac{1}{n}\sum_{\bar{x}_i \in P} eval(\bar{x}_i)$$

- CM-GA satisfies the assumptions of Banach fixpoint theorem:
  - The distance δ

$$\delta(P_1,P_2)=\begin{cases}0 & if \quad P_1=P_2\\ |1+M-Eval(P_1)|+ & otherwise\\ |1+M-Eval(P_2)| & \end{cases}$$

  - The space of populations $\langle S,\delta\rangle$ is a metric space.

$$\delta(P_1,P_2)\geq 0;\ \text{ and } \delta(P_1,P_2)=0 \ \text{ iff } \ P_1=P_2$$
$$\delta(P_1,P_2)=\delta(P_2,P_1)$$
$$\delta(P_1,P_2)+\delta(P_2,P_3)\geq\delta(P_1,P_3)$$

  - The metric space $\langle S,\delta\rangle$ is complete.

$$P=\lim_{i\to\infty}P_i$$

  - The iteration $f:P(t)\to P(t+1)$ is contractive.

$$\delta(f(P_1(t)),f(P_2(t)))\leq\delta(P_1(t),P_2(t))$$
$$(\because Eval(P\ (t))<Eval(P\ (t+1)))$$

  - CM-GA converges to population P^*, which is a unique fixpoint in the space of all populations.

$$P^*=\lim_{i\to\infty}f^{\,i}(P(0))$$

Banach Fixpoint Theorem

Let

1. $\langle S, \delta \rangle$ be a *complete* metric space
2. $f : S \to S$ be a *contractive* mapping.

Then there f has a unique *fixpoint* $x \in S$ such that for any $x_0 \in S$,

$$x = \lim_{i \to \infty} f^i(x_0)$$

where $f^0(x_0) = x_0$ and $f^{i+1}(x_0) = f(f^i(x_0))$.

- A set S together with $\delta : S \times S \to R$ is a *metric space*, $\langle S, \delta \rangle$, if, for any elements $x, y \in S$,

$$\delta(x,y) \geq 0; \quad \text{and}\, \delta(x,y) = 0 \quad \text{iff} \quad x = y$$
$$\delta(x,y) = \delta(y,x)$$
$$\delta(x,y) + \delta(y+z) \geq \delta(x,z).$$

- If $\langle S, \delta \rangle$ is a metric space and $f : S \to S$, f is *contractive* iff there is a constant $\varepsilon \in [0,1)$ such that for all $x, y \in S$

$$\delta(f(x), f(y)) \leq \varepsilon \times \delta(x,y)$$

- The sequence $p_0, p_1, \cdots$ of metric space $\langle S, \delta \rangle$ is a *Cauchy sequence* iff for any $\varepsilon > 0$ there is k such that for all $m, n > k$, $\delta(p_m, p_n) < \varepsilon$.

- A metric space is complete if any Cauchy sequence $p_0, p_1, \cdots$ has a limit $p = \lim_{n \to \infty} p_n$.

D. Population size

- Too small → converge too quickly
- Too large → waste computational resources

- Varying population Size (GAVaPS)

Varying Population Size (GAVaPS)

```
Procedure  **GAVaPS**
begin
        t = 0
        initialize P(t)
        evaluate P(t)
        while  (not termination-condition)  do
        begin
                t = t+1
                increase the *age* to each individual by 1
                recombine P(t)        ‘ $AuxPopSize(t) = \lfloor PopSize(t) * p \rfloor$
                evaluate P(t)         ‘ *p : reproduction ratio*
                remove from P(t) all individuals
                        with *age* greater than their *lifetime*
        end  ‘ $PopSize(t+1) = PopSize(t) + AuxPopSize(t) - Dead(t)$
end
```

- Lifetime value:
  - Proportional allocation

$$\min\left(MinLT + \eta \frac{fitness[i]}{AvgFit}, MaxLT\right)$$

$$\text{where} \quad \eta = \tfrac{1}{2}(MaxLT - MinLT)$$

  - Linear allocation

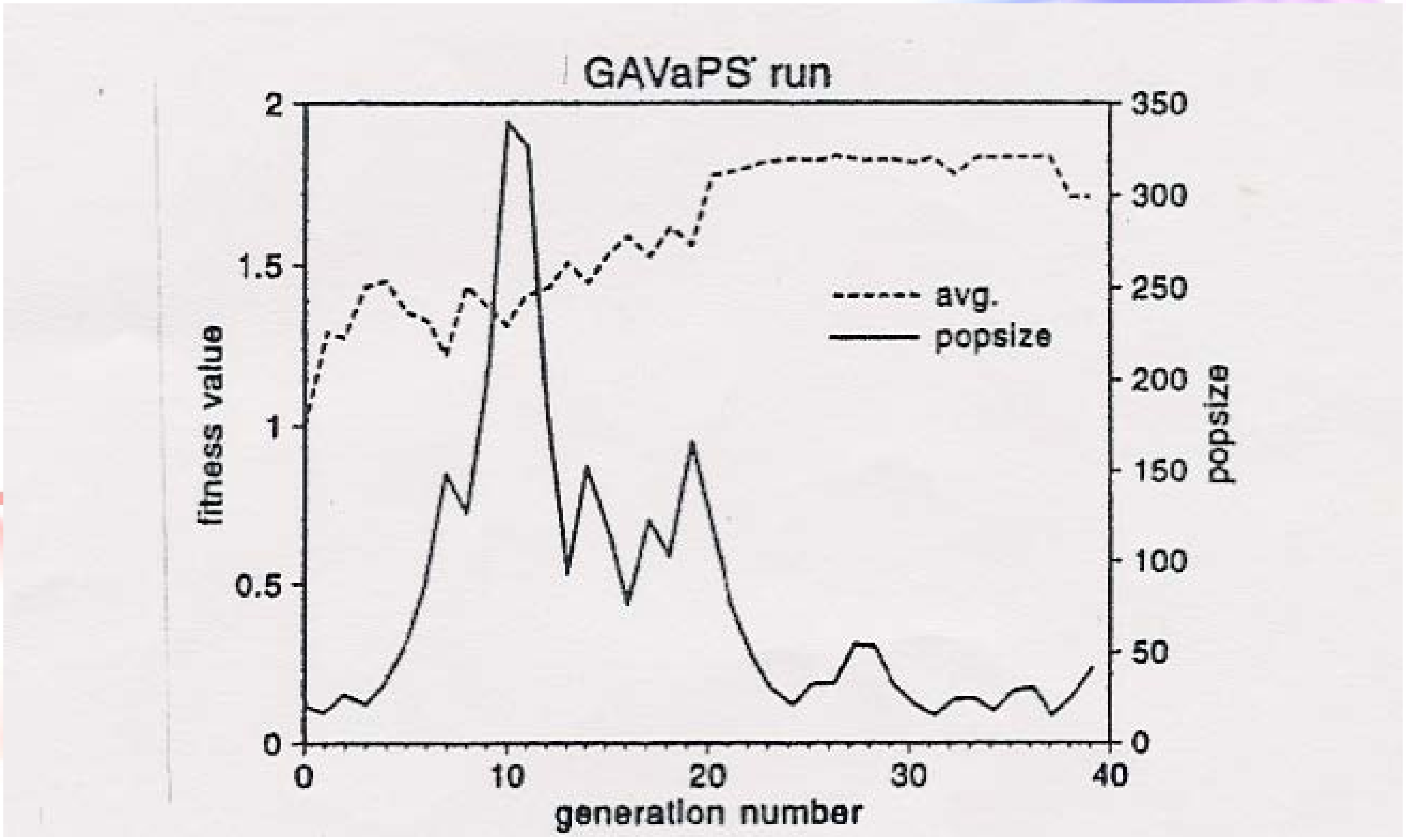
$$MinLT + 2\eta \frac{fitness[i] - AbsFitMin}{AbsFitMax - AbsFitMin}$$

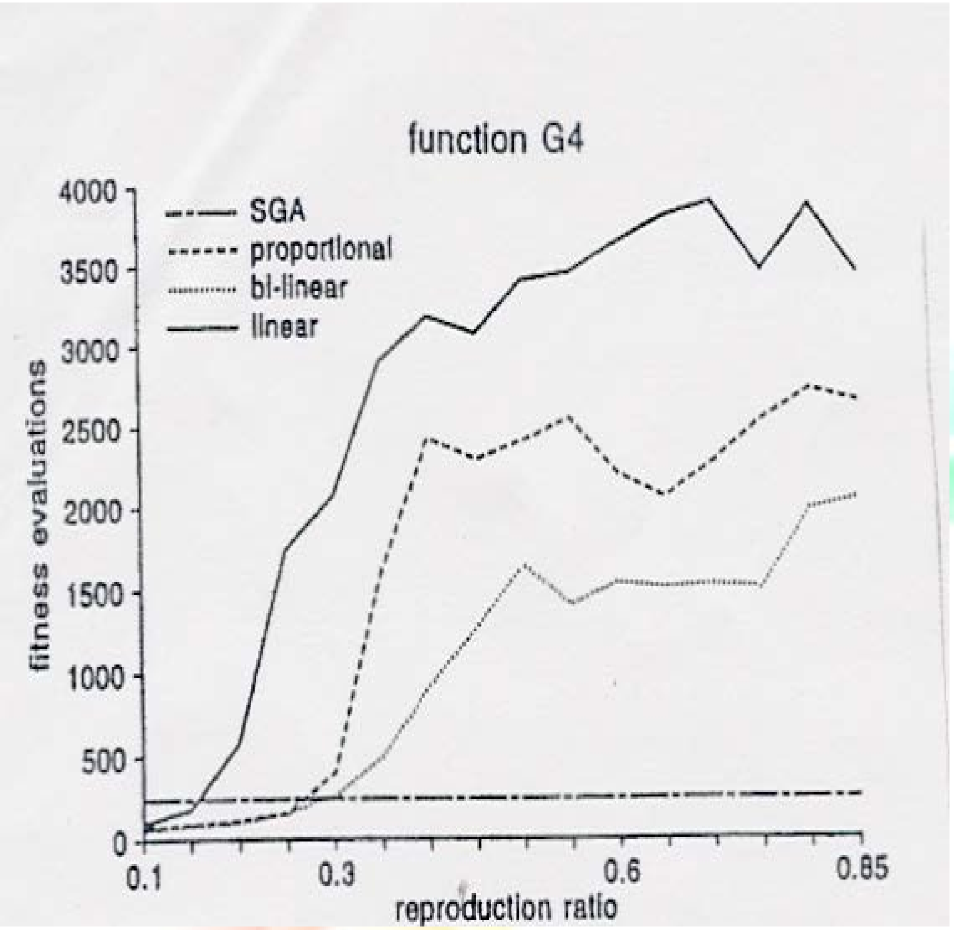
  - Bi-linear allocation

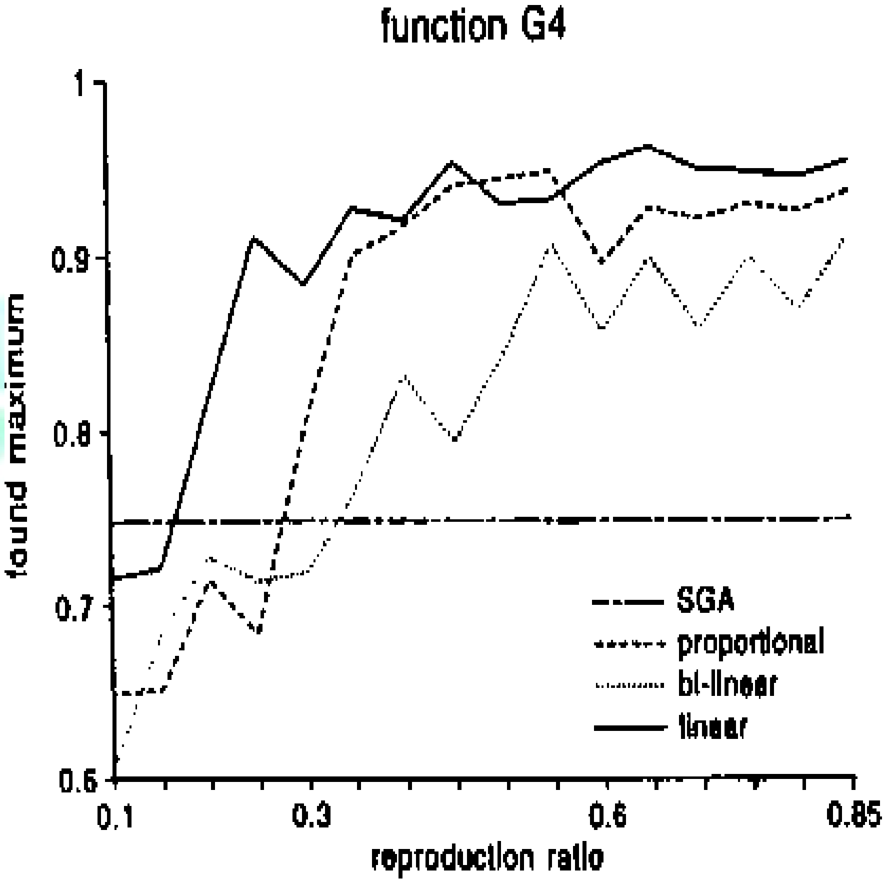
$$\begin{cases} MinLT + \eta \dfrac{fitness[i] - MinFit}{AvgFit - MinFit} & \text{if} \quad AvgFit \geq fitness[i] \\ \dfrac{1}{2}(MinLT + MaxLT) + \eta \dfrac{fitness[i] - AvgFig}{MaxFit - AvgFit} & \text{if} \quad AvgFit < fitness[i] \end{cases}$$

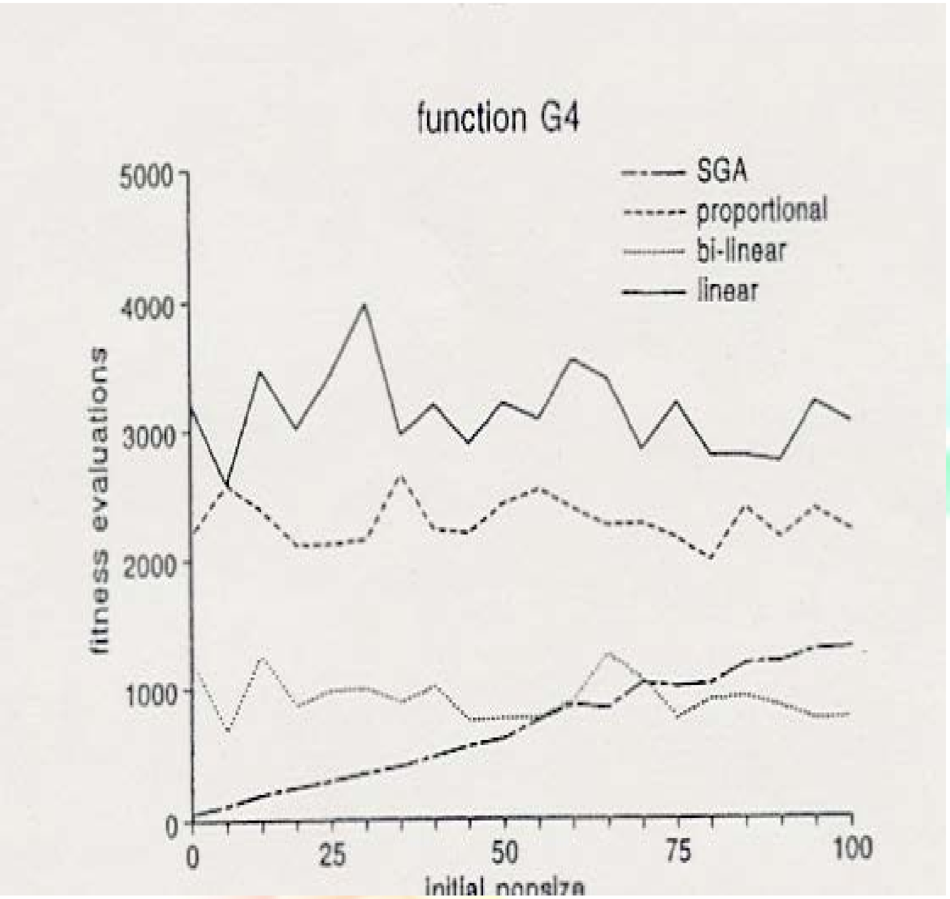
Example

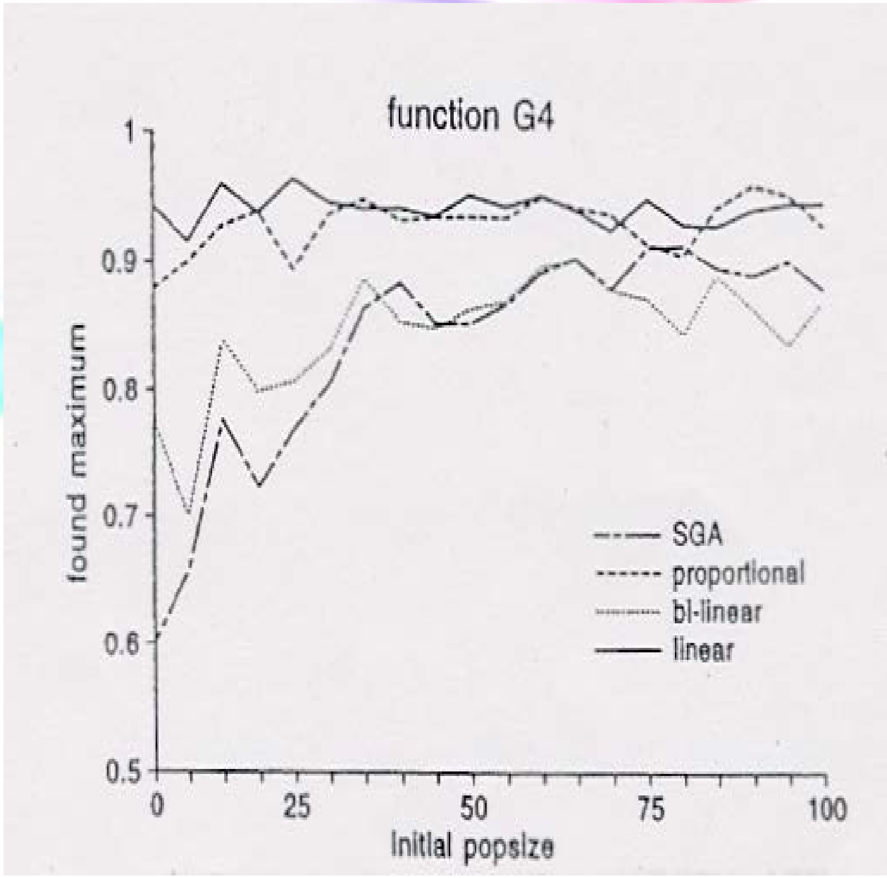
- Initial_size = 20
- Reproduction ratio = 0.4
- Mutation ratio =0.015
- Crossover ratio = 0.65
- Length of chromosome = 20
- MaxLT =7
- MinLT = 1
- Termination = no progress for consecutive 20 generations
- Independent run = 20 times

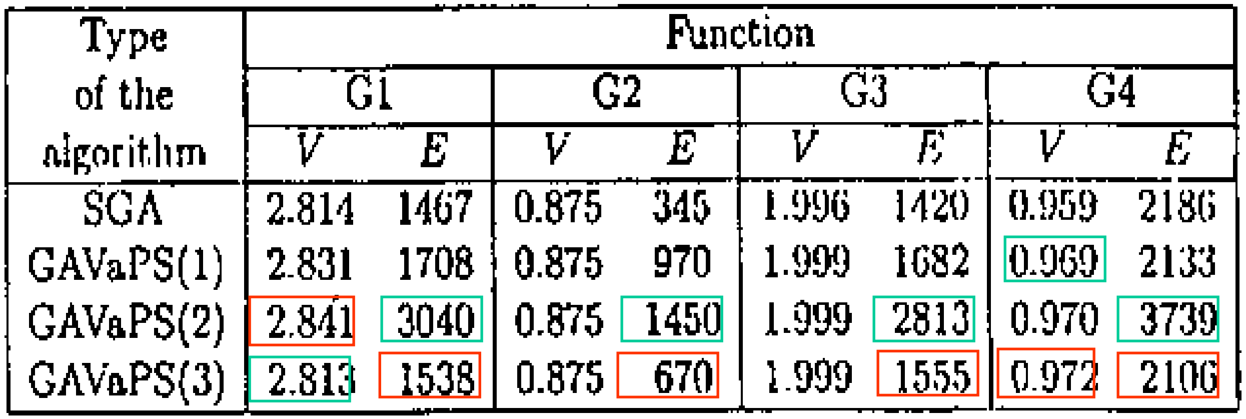

GAVaPS run
fitness value
popsize
generation number
avg.
popsize
2
1.5
1
0.5
0
350
300
250
200
150
100
50
0
0
10
20
30
40


function G4
SGA
proportional
bi-linear
linear
fitness evaluations
4000
3500
3000
2500
2000
1500
1000
500
0
0.1
0.3
0.6
0.85
reproduction ratio


function G4
found maximum
1
0.9
0.8
0.7
0.6
SGA
proportional
bi-linear
linear
0.1
0.3
0.6
0.85
reproduction ratio


function G4
5000
4000
3000
2000
1000
0
fitness evaluations
SGA
proportional
bi-linear
linear
0
25
50
75
100
initial popsize


function G4
1
0.9
0.8
0.7
0.6
0.5
found maximum
SGA
proportional
bi-linear
linear
0
25
50
75
100
initial popsize



Type of the algorithm	Function							
	G1		G2		G3		G4	
	V	*E*	*V*	*E*	*V*	*E*	*V*	*E*
SGA	2.814	1467	0.875	345	1.996	1420	0.959	2186
GAVaPS(1)	2.831	1708	0.875	970	1.999	1682	0.969	2133
GAVaPS(2)	2.841	3040	0.875	1450	1.999	2813	0.970	3739
GAVaPS(3)	2.813	1538	0.875	670	1.999	1555	0.972	2106

GAVaPS(1): proportional

GAVaPS(2): linear

GAVaPS(3): bi-linear

E. Initialization of Population

Appendix: Evolutionary Programming —— quasi-random sequences

Suppose that the natural numbers are expressed in the scale of notation with radax R, so that

$$n = a_0 + a_1 R + a_2 R^2 + \cdots + a_m R^m , \qquad 0 \le a_i \le R .$$

Write the digits of these numbers in reverse order, proceeded by a decimal point. This gives the numbe

$$\phi_R(n) = a_0 R^{-1} + a_1 R^{-2} + \cdots + a_m R^{-m-1} .$$

An example in the binary scale ($R = 2$) is shown in Table 1.

Holton [19] extended the two-dimensional result of Van Der Corput [10] to κ-dimensional, when R_1 R_κ are mutually coprime. We show an illustrative case in Table 2.

Since $\phi_R(n) < 1$, to satisfy this range, scaling any varying parameter (e.g., a real number ξ from $[\underline{\xi}, \overline{\xi}]$ to [0,1]) is required. Let the interval real (IR) matrix $X \in IR^{n\times m}$ be a set of degenerate real defined by

$$X = [L, U] = \{ [x_{ij}] \mid \ell_{ij} \le x_{ij} \le u_{ij} ;\ 1 \le i \le n,\ 1 \le j \le m \} ,$$

where L and U are constant real matrices. We introduce variables ξ_{ij}, $0 \le \xi_{ij} \le 1$ such that

$$x_{ij} = \ell_{ij} + \xi_{ij} (u_{ij} - \ell_{ij})$$

and use the notation

$$\xi = [\xi_{11}, \cdots, \xi_{1m}, \xi_{21}, \cdots, \xi_{2m}, \cdots, \xi_{n1}, \cdots, \xi_{nm}] .$$

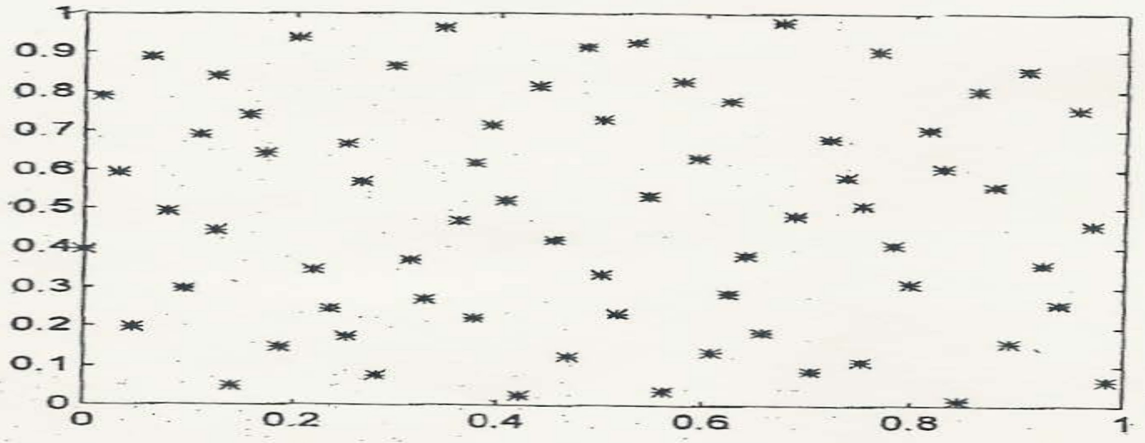
Then the interval matrix X can be denoted as $X(\xi)$. Let $\xi_{11} = \phi_2(n)$, $\xi_{12} = \phi_3(n)$, $\xi_{13} = \phi_5(n)$, and so on, to construct the desired initial population of size N (e.g., $N = 50$).

Table 1: Example of natual numbers in binary scale.

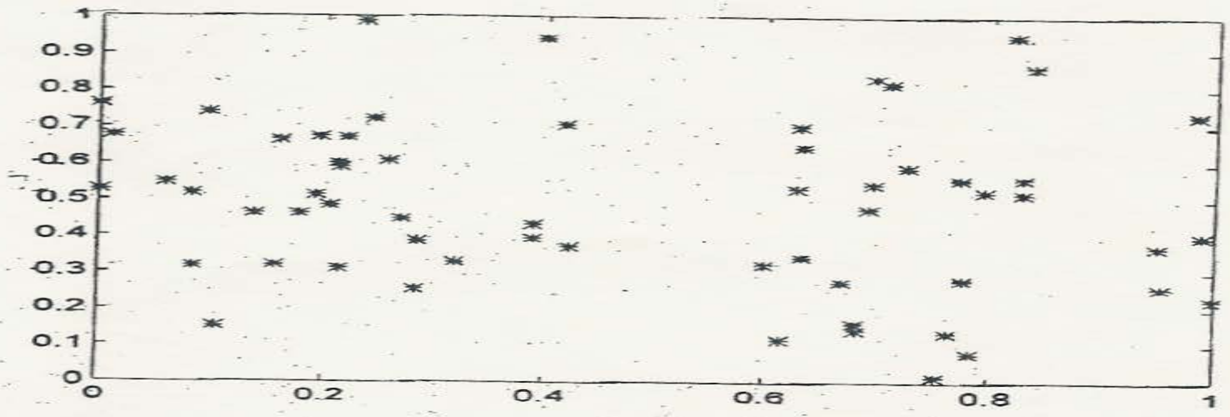
n (decimal)	(binary)	$\phi_2(n)$ (binary)	(decimal)
1	1	0.1	0.5
2	10	0.01	0.25
3	11	0.11	0.75
4	100	0.001	0.125
5	101	0.101	0.625
6	110	0.011	0.375
7	111	0.111	0.875
8	1000	0.0001	0.0625
...	...	...	...

Table 2: Example of quasi-random sequences.

$\phi_R(n)$	$R = 2$	3	5	7	11	13	17	...
$n = 1$	0.5000	0.333	0.2000	0.1429	0.0909	0.0769	0.0588	...
2	0.2500	0.6667	0.4000	0.2857	0.1818	0.1538	0.1176	...
3	0.7500	0.1111	0.6000	0.4286	0.2727	0.2308	0.1765	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
48	0.0469	0.1975	0.7680	0.9796	0.3967	0.7101	0.8304	...
49	0.5469	0.5309	0.9680	0.0029	0.4876	0.7870	0.8893	...
50	0.2969	0.8642	0.0160	0.1458	0.5785	0.8639	0.9481	...



a : QRS

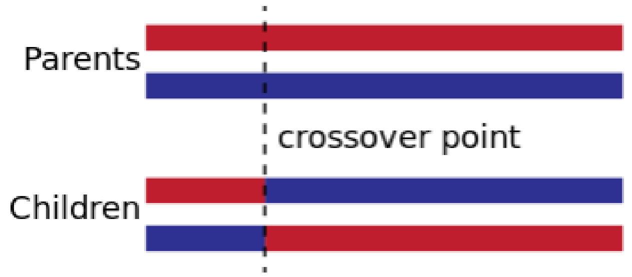


b: uniform random number

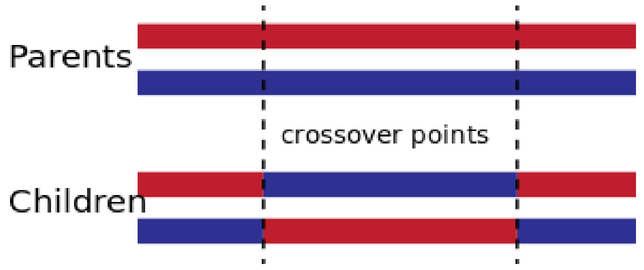
F. Crossover, Mutation

Crossover

- One-point crossover

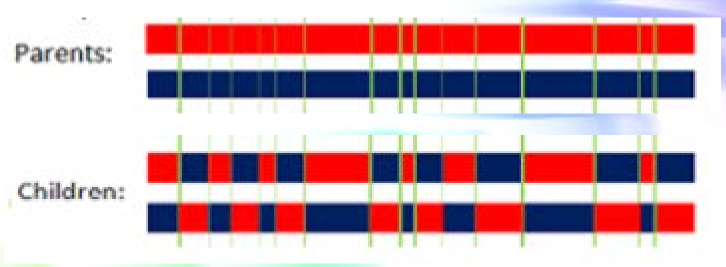


- Two-point crossover

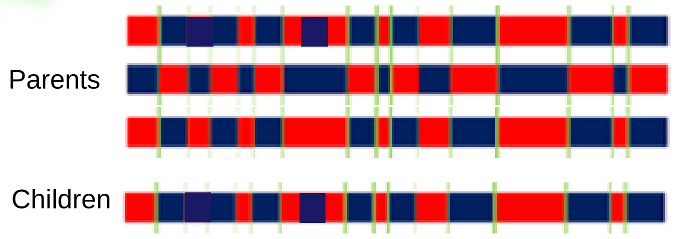


Crossover

- Uniform crossover (UX)



- Gene pool recombination (GPR)

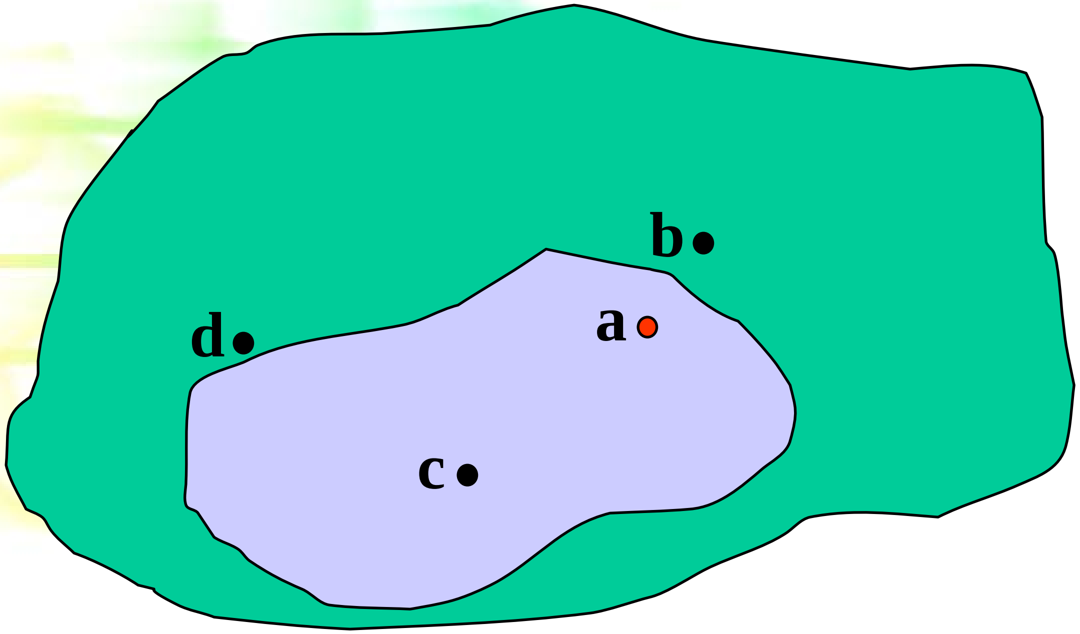


Mutation

- Uniform
- Non-Uniform

G. Constraints

$$\begin{aligned} &\max f(x) \\ \text{subject to} \quad & g_i(x) \le 0, \quad i = 1,2,\ldots,m_1 \\ & h_i(x) = 0, \quad i = m_1+1,\ldots,m \\ & x \in X \end{aligned}$$



Solution Space

Constraint-Handling Techniques

- Rejecting/Death Strategy
- Penalty Strategy
- Repairing Strategy
- Modifying GA Strategy (Decoder)

0-1 Knapsack Problem (example)

- A thief robbing a store finds n items; the *i*th item is worth P_i dollars and weighs W_i pounds, where P_i and W_i are integers. He wants to take as valuable a load as possible, but he can carry at most *C* pounds in his knapsack. What items should he take? (This is called the 0-1 knapsack problem because each item must either be taken or left behind; the thief cannot take a fractional amount of an item or take an item more than once.)

- Problem

Maximize

$$f(x) = \sum_{i=1}^{n} x[i] \bullet P[i]$$

Subject to

$$\sum_{i=1}^{n} x[i] \bullet W[i] \le C$$

- Penalty Strategy

$$eval(x) = f(x) + Pen(x)$$

Penalty Function:

$$Pen(x) = \begin{cases} 0 & \textbf{\textit{if}}\ \ x\ \ \textbf{\textit{is}}\ \ \textbf{\textit{feasible}} \\ </> \mathbf{0} & \textbf{\textit{otherwise (max/min)}} \end{cases}$$

- Penalty Strategies

$$eval(x) = f(x) - Pen(x)$$

Penalty Functions: $(\rho = \max_{i=1\cdots n}\{P[i]/W[i]\})$

- Linear:

$$Pen(x) = \rho(\sum_{i=1}^{n} x[i] \bullet W[i] - C)$$

- Logarithmic:

$$Pen(x) = \log_2(1 + \rho(\sum_{i=1}^{n} x[i] \bullet W[i] - C))$$

- Quadratic:

$$Pen(x) = (\rho(\sum_{i=1}^{n} x[i] \bullet W[i] - C))^2$$

- Repair Strategy

$$eval(x) = \sum_{i=1}^{n} x'[i] \bullet P[i]$$

Two different repair algorithms:

- Random repair
- Greedy repair (profit / weight)

Algorithm:

```
Procedure Repair (x)
begin
    knapsack-overfilled := false
    $x' := x$
    if $\sum_{i=1}^{n} x'[i] \bullet W[i] > C$
    then knapsack-overfilled := true
    while (knapsack-overfilled) do
    begin
        $i$ := select an item from the knapsack
        remove the selected item from the knapsack:
                i.e. $x'[i] := 0$
        if $\sum_{i=1}^{n} x'[i] \bullet W[i] \le C$
        then knapsack-overfilled := false
    end
end
```

- Decoder Strategy

Ordinal representation

- L = (1 2 3 4 5 6)
- *v* = (4 3 4 1 1 1)

Two different decoding algorithms:

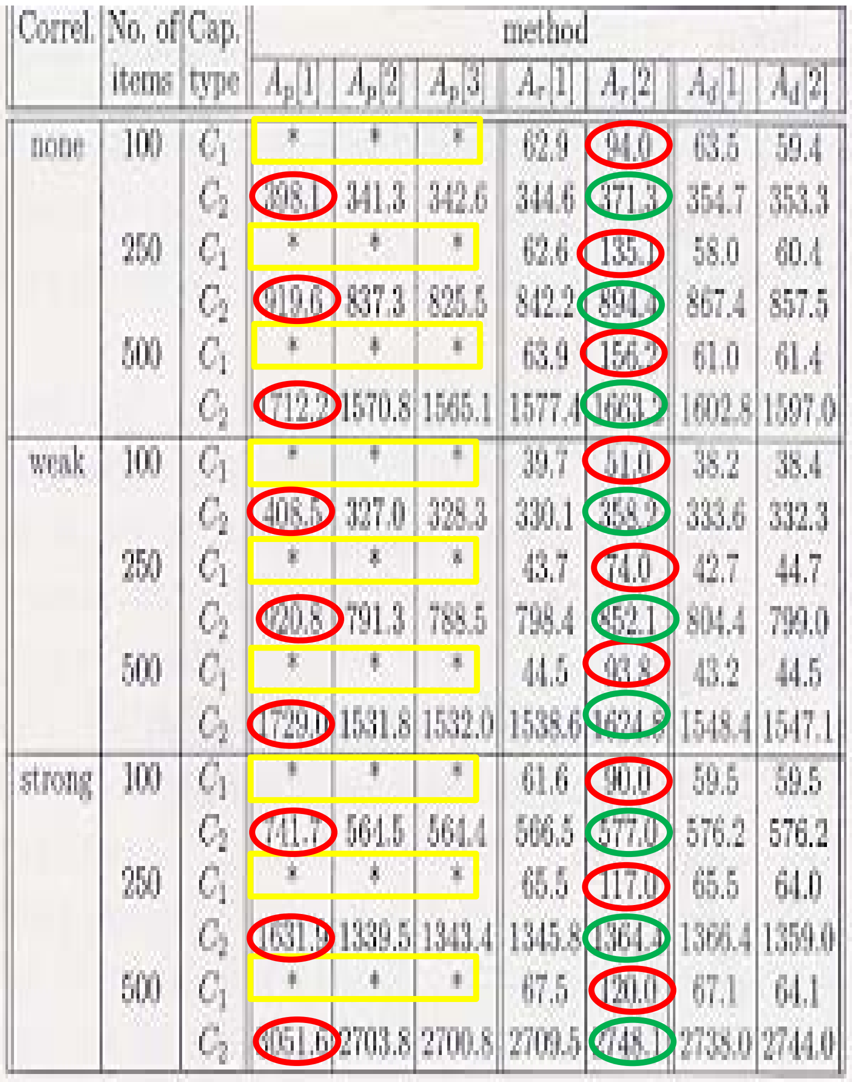
- Random decoding
- Greedy decoding (profit / weight)

Algorithm:

```
Procedure  **Decode** ($x$)
begin
        **build** a list $L$ of items
        $WeightSum := 0$
        $ProfitSum := 0$
        $i := 1$
        while   $i <= n$    do
        begin
                $j := x[i]$
                remove the $j$-th item from the list $L$
                if   $WeightSum + W[j] <= C$   then
                begin
                        $WeigthSum := WeightSum + W[j]$
                        $ProfitSum := ProfitSum + P[j]$
                end
                $i := i+1$
        end
end
```

Examples

G1:	$-x\sin(10\pi x)+1$	$-2.0 \le x \le 1.0$
G2:	$\mathrm{int}\,eger(8x)/8$	$0.0 \le x \le 1.0$
G3:	$x \cdot \mathrm{sgn}(x)$	$0.0 \le x \le 1.0$
G4:	$0.5+\dfrac{\sin^2\sqrt{x^2+y^2}-0.5}{(1+0.001(x^2+y^2))^2}$	$0.0 \le x \le 1.0$



Correl.	No. of items	Cap. type	method						
			$A_p[1]$	$A_p[2]$	$A_p[3]$	$A_r[1]$	$A_r[2]$	$A_d[1]$	$A_d[2]$
none	100	C_1	*	*	*	62.9	94.0	63.5	59.4
		C_2	398.1	341.3	342.6	344.6	371.3	354.7	353.3
	250	C_1	*	*	*	62.6	135.1	58.0	60.4
		C_2	919.6	837.3	825.5	842.2	894.4	867.4	857.5
	500	C_1	*	*	*	63.9	156.2	61.0	61.4
		C_2	1712.2	1570.8	1565.1	1577.4	1663.2	1602.8	1597.0
weak	100	C_1	*	*	*	39.7	51.0	38.2	38.4
		C_2	408.5	327.0	328.3	330.1	358.2	333.6	332.3
	250	C_1	*	*	*	43.7	74.0	42.7	44.7
		C_2	920.8	791.3	788.5	798.4	852.1	804.4	799.0
	500	C_1	*	*	*	44.5	93.8	43.2	44.5
		C_2	1729.0	1531.8	1532.0	1538.6	1624.8	1548.4	1547.1
strong	100	C_1	*	*	*	61.6	90.0	59.5	59.5
		C_2	741.7	561.5	564.4	566.5	577.0	576.2	576.2
	250	C_1	*	*	*	65.5	117.0	65.5	64.0
		C_2	1631.9	1339.5	1343.4	1345.8	1364.4	1366.4	1359.0
	500	C_1	*	*	*	67.5	120.0	67.1	64.1
		C_2	3051.6	2703.8	2700.8	2709.5	2748.1	2738.0	2744.0

Ap[1](logarithmic penalty)
Ap[2](linear penalty)
Ap[3](quadratic penalty)

Ar[1](random repair)
Ar[2](greedy repair)

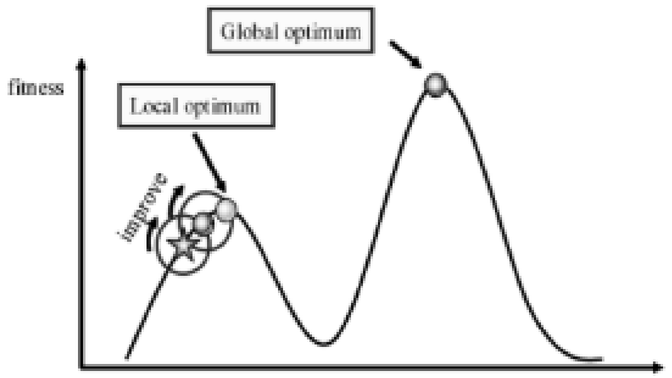
Ad[1](random decoding)
Ad[2](greedy decoding)

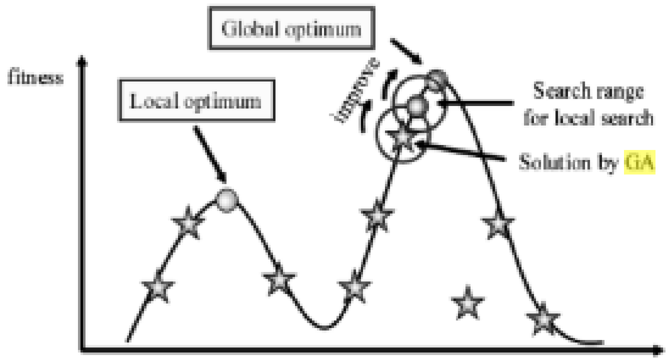
H. Other Ideas

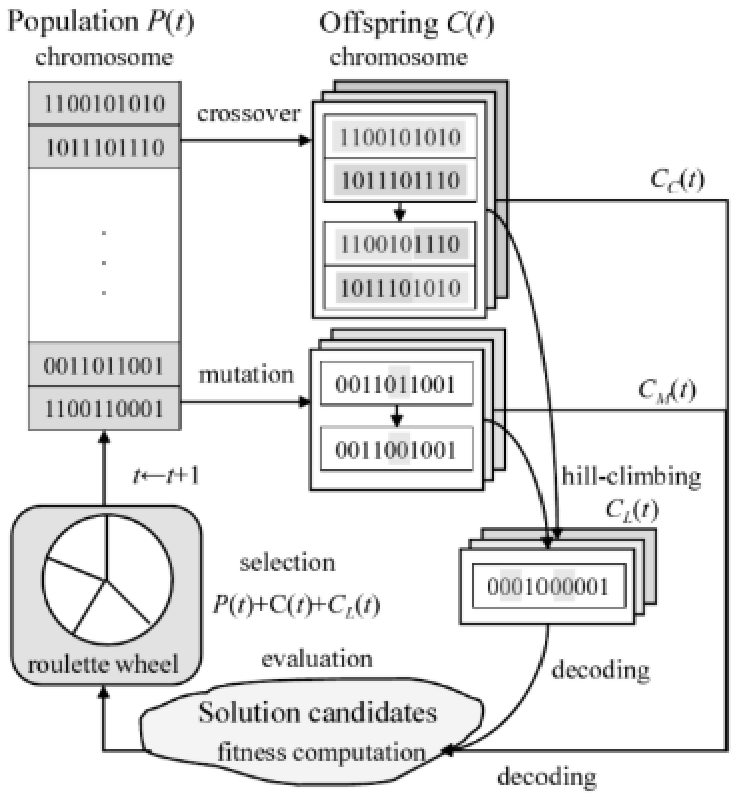
Messy Genetic Algorithm (mGA)

- Representation
  - tag, length, redundant and contradictory
    - v1 = ((7,1)(1,0))
    - v2 = ((3,1)(9,0)(3,1)(3,1)(3,1))
    - v3 = ((2,1)(2,0)(4,1)(5,0)(6,0)(7,1)(8,1))
  - overspecification – tie-breaking (first come, first serve)
  - underspecification – probability/competitive templates
- Selection
  - tournament
- Crossover
  - splice & cut
- Mutation
  - bit inversion

Hybrid GA


Global optimum
fitness
Local optimum
improve


Global optimum
fitness
Local optimum
improve
Search range for local search
Solution by GA


Population P(t)
chromosome
1100101010
1011101110
0011011001
1100110001
Offspring C(t)
chromosome
crossover
1100101010
1011101110
1100101110
1011101010
$C_C(t)$
mutation
0011011001
0011001001
$C_M(t)$
hill-climbing
$C_L(t)$
0001000001
$t \leftarrow t+1$
selection
$P(t)+C(t)+C_L(t)$
roulette wheel
evaluation
Solution candidates
fitness computation
decoding
decoding