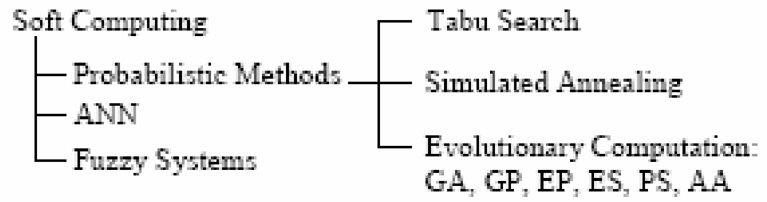
Soft Computing Techniques

- "Soft Computing differs from conventional (hard) computing in that, unlike hard computing, it is tolerant of imprecision, uncertainty and partial truth." Lotfi Zadeh
- Soft computing techniques exploit the tolerance for imprecision, uncertainty and partial truth to achieve tractability, robustness and low solution cost.



Biological Inspiration for Evolutionary Computation

- DNA: A-T, C-G
- Parents: ATCGGGGGTTTAAAACCCCCTGCACGTTT
  CCCCCCATTTTTTAAAAGGGGGTGACCCC
- Offspring: ATCGGGGGTTTAAAAAGGGGGTGACCCC
  CCCCCCATTTTTTAAACCCCCTGCACGTTT
- Nature, specifically the local environment, ensures that only fit individuals survive and pass on their traits to their offspring. Mutation, also plays and important role in the creation of better individuals for the environment.
- This process is called **evolution**.

Evolutionary Computation

- Computer based problem solving techniques using computational models based on the principles of *evolution*.
- *Population* based
- Use operators of *selection*, *mutation*, and *recombination*
- *Fitness* determines the survival of individuals
- Most operators use stochastic information
- Provide robust and powerful adaptive search mechanisms

Pseudo Code

```
initialize and evaluate population P (0);
while not done do
    P' (t) := selectparents P (t);
    recombine P' (t);
    mutate P' (t);
    evaluate P' (t);
    P (t+1) := survive P (t), P' (t);
end while
```

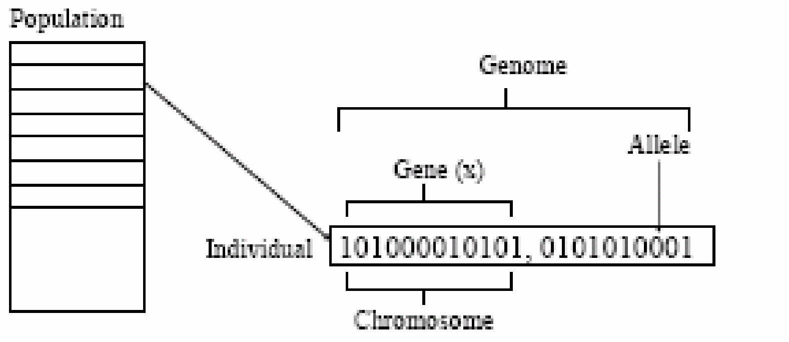
Terminology

- Population - A set of one or more individuals (organism) that interact during the application of a EC algorithm.
- Individual - A member of the population that represents a solution for the problem being solve. Its genome consist of one or more chromosomes.
- Genome - Set of genes in an individual.
- Chromosome - A data structure that represents the genes of an individual, usually binary strings.

Terminology (Cont'd)

- Gene - Part of a chromosome that encodes a parameter of a solution.
- Alleles - Possible values of a gene.
- Genotype - Genetic representation of an individual. Encoded value of the chromosome.
- Phenotype - The representation of the individual in the solution space. Decoded value of the chromosome.
- Fitness - A value that indicates the quality of an individual (solution).

Example: Find Highest F(x, y), 0 <x,y<1



Genotype of x is 101000010101= 2581 decimal

Phenotype of x is 2581/2^12

Fitness = F(x, y)

Genetic Algorithms

- A technique for searching the *fitness landscape* for highly fit individuals
  - Fitness landscape (Wright 1931) - Search space formed from the representation of all possible genotypes and their fitness value.
- Another optimization technique.
- Based on the principles of natural selection and genetic recombination.
- Work with a set of solutions.
- Problem independent, almost.
- Does well in noisy and complex search spaces.

Simple Genetic Algorithm

- Needed
  - Encoding: a solution (*individual*) corresponds to a *chromosome*.
  - Chromosome: character string, normally binary.
  - Fitness function: measures quality of a solution.
- Algorithm

```
Generate and evaluate initial population at random
For generation = 1 to MAX_GENERATIONS

    Selection step: Create the mating pool

    Recombination step: Apply crossover and mutation

    Replacement step: Replace entire population with offspring
end for
```

The Selection Step: Creation of Mating Pool

- Using *fitness proportionate reproduction* (FPR). An individual I has a probability $P(I) = F(I)/\sum F(I)$ of being chosen as a parent.
- Higher fitted individuals are more likely to be selected.
- Example:

No.	Individual	Fitness	Selection Probability	Mating Pool
1	10000	81	0.51	10000
2	00101	10	0.06	01011
3	01011	31	0.20	10000
4	01100	36	0.23	01100
Total		158	1.00	

The Recombination Step: Crossover and Mutation

- Crossover is applied with probability p_c between pairs of individuals.
  - Select a position in the string at random.
  - Swap the bits after this position in both strings.

```
Example: 1 0 | 0 0 0    -->   1 0 1 0 0
         0 1 | 1 0 0    -->   0 1 0 0 0
```

- Mutation is applied to every offspring.
  - Each bit in the string is changed with probability p_m .

The Recombination Step: Crossover and Mutation (Cont'd)

- Example: $p_c = 0.5$, $p_m = 0.05$

Mating Pool	Fitness	Offspring after Crossover	Fitness	Offspring after Mutation	Fitness
10000	81	10000	81	10001	82
01011	31	01011	31	01011	31
10000	81	10100	90	10100	90
01100	36	01000	27	01000	27
Total	229		229		230

Simple Genetic Algorithm

- Exploits individuals with high fitness to pass their traits to offspring.
- Suffers from *premature convergence*. More likely to converge to local optima.
- Cannot maintain diversity unless a high mutation rate is used.
- Population becomes homogeneous after many generations.

SWENG 584 - Genetic Algorithms

Schemata and GAs

Walter Cedeño

Penn State - Great Valley

Some Definitions

- Definition 1: An ordered arrangement of *r* distinct objects is called a *permutation*. The numbers of ways of ordering *n* distinct objects taken *r* at a time without replacement will be designated by the Symbol P^n_r.

$$P_r^n = \frac{n!}{(n-r)!}$$

- Definition 2: The number of combinations of *n* objects taken *r* at a time without replacement is the number of subsets, each of size *r*, which can be formed from the *n* objects. This number will be denoted by C^n_r.

$$C_r^n = \binom{n}{r} = \frac{n!}{r!(n-r)!}$$

Schemata

- A schema (Holland, 1975) over the binary alphabet A = {0, 1} is a string composed of the symbols in A plus an additional don't care symbol *.
- Example: The schema 11**0. This matches the four strings {11000, 11010, 11100, 11110}. That is, this particular schema matches all binary strings of length 5 starting with two 1's and ending with one 0.
- Example: The schema ***** matches all binary strings of length 5 and 11111 is the schema representing the binary string of all 1's.
- Thus a schema can be thought of as a template comprised of the symbols from an alphabet A and the don't care symbol *.

Schema Properties

- There are two important properties that must be defined for schemata. The order of a schema S, denoted o(S), and the defining length of a schema S, denoted δ(S).
- The order of a schema o(S) is the number of fixed positions, i.e., non * positions, in S.
- Example: 11111 is a schema of order 5, while **1** is a schema of order 1.
- Note that the value of o(S) is in the interval [0, l], where l is the number of positions in the string. The higher the order, the more specific the schema.

Schema Properties (Cont'd)

- The defining length of a schema δ(S) defines the number of crossover positions between the two outermost fixed positions in S.
- Example: the length of the schema 11*01 is 4, while the length for the schema ****1 is 0 since the two outermost fixed positions are the same, namely digit 1.
- The length of a schema is a value in the interval [0, l-1].
- The schema length δ(S) defines the compactness of the schema and it's different from the string length l which refers to the number of positions in the schema.

Schema Properties (Cont'd)

- In general, there are (k + 1)^l schemata for strings of length l in an alphabet of size k.
- Each string of length l in A is matched by exactly 2^l schemata.
- A schema with r don't care symbols matches exactly k^r strings.
- GA samples the schemata in the population giving and increasing number of mating slots to highly fit schemata.

The Schema Theorem

- The Schema Theorem (Holland, 1975) gives a lower bound on the effect of the genetic operators from one generation to the next in the classical GA.
- Recall that the classical GA consists of the following repeated steps; selection, crossover, mutation, and replacement.
- To be able to predict I^{t+1}, the population at time t + 1, we look at the influence of the genetic operators on the schemata of I^t, the population at time t.

The Schema Theorem: Selection

- In the first step, selection, individuals are chosen for mating according to the ratio of their fitness to the total fitness of the population. Let F_i be the fitness of individual I_i and

$$F^t = \sum_{i=1}^{n} F_i$$

the fitness of the population, then the probability of mating for individual I_i is

$$P_i = \frac{F_i}{F^t}$$

The Schema Theorem: Selection (Cont'd)

- To expand the notion of selection probability to schema, it is only necessary to define the fitness of a schema S.
- The fitness of a schema S at time t is defined as the average fitness of all the individuals matched by the schema. That is,

$$F_S = \sum_{i=1}^{\varphi} \frac{F_i}{\varphi_S^t} \qquad \text{, for } I_i \in S$$

where φ_S^t is the number of individuals matching schema S in the population at time t.

- Then schema S has probability $P_S = \frac{F_S}{F^t}$ to be selected.

The Schema Theorem: Selection (Cont'd)

- After selection takes place we expect to have

$$n \cdot \varphi_S^t \cdot \frac{F_S}{F^t}$$

individuals belonging to schema S in the mating pool prior to crossover.

- This result is obtained from the fact that schema S has φ_S^t individuals in I^t with an average probability P_S of being selected for any of the n slots in the mating pool.
- If we define the average fitness of the population $\overline{F}^t = \frac{F^t}{n}$ then we have

$$\varphi_S^{t+1} = \varphi_S^t \cdot \frac{F_S}{\overline{F}^t}$$

Observations

- Notice that the number of individuals matching schema S after selection grows as the ratio of the schema average fitness over the population average fitness. This means that schema with above average fitness tends to receive more mating opportunities than below average schema.
- From the value of φ_S^{t+1} we can also observe the long term effect of above average schema in the population. If we assume that a schema S remains above average in succeeding generations, then the number of individuals belonging to the schema will increase exponentially.
- This is a cause for premature convergence, where an above average individual drives the population towards a local minimum. This situation is not rare and is a direct result of using FPR.

The Schema Theorem: Crossover

- In the second step of the SGA crossover is applied to pairs of individuals from the mating pool. Lets see the effect of crossover on the expected number of schemata:

$$\varphi_S^t$$

- Crossover chooses uniformly at random a position in the chromosome and swaps the information to the right of this location between a pair of individuals.
- Since the mating pairs are chosen at random any two schema can undergo crossover.

Crossover and Schemata

- Let's assume that two individuals selected for crossover are representative of the following schemata; S1 = (11***) and S2 = (01**0) respectively.
- Suppose the second crossover position (from right to left) is selected.
- Then the offspring resulting from crossover of (11*|**) with (01*|*0) will belong to the schemata (11**0) and (01***) respectively.
- One of the offspring will still belong to schema S1, since 11**0 is contained in it, but the other offspring may or may not belong to the schema S2, because 01*** is not contained in S2.
- Schemata with large defining length are more probable of being destroyed than schema with shorter defining length.

Crossover and Schemata (Cont'd)

- In general, crossover is applied with probability χ and the crossover site is selected uniformly at random over the l - 1 positions in the string. Therefore the probability of destruction of a schema S is

$$PD_S = \chi \cdot \frac{\delta(S)}{l-1}$$

- Consequently, the probability of schema survival is

$$PS_S = 1 - \chi \cdot \frac{\delta(S)}{l-1}$$

- Notice also that there is still a chance for the schema to survive when the information passed from the other individual contain the same digits as the fixed positions in the schema.

Schemata and Crossover (Cont'd)

- In the prior example, S1 = (11***) and S2 = (01**0), if the individual belonging to S1 also had a 0 in its last digit then the schema S2 will survive the crossover.
- In the same manner two individuals not belonging to schema S2 can produce an offspring in S2. For example, (01001) and (11100) with the crossover position 1 will produce the offspring (01000) and (11101), with the first one belonging to S2.
- For this reason the value of PS_S is a lower bound estimate of the resulting survival probability:

$$PS_S \geq 1 - \chi \cdot \frac{\delta(S)}{l-1}$$

The Schema Theorem: Crossover (Cont'd)

- Combining the effect of crossover with the expected schema count obtained we get the equation:

$$\varphi_S^{t+1} \geq \varphi_S^t \cdot \frac{F_S}{\overline{F}^t} \cdot [1 - \chi \cdot \frac{\delta(S)}{l-1}]$$

- This equation captures the expected count for schema S in the next generation after selection and crossover are applied.

The Schema Theorem: Mutation

- Mutation works by randomly changing an allele in the chromosome with probability μ.
- It is easy to see that a schema S will only be affected if one of its fixed positions is modified.
- We know that the order of a schema defines the number of fixed positions in it. Therefore the probability that a schema S survives mutation is

$$PS_S = (1 - \mu)^{o(S)}$$

- For small values of μ (μ << 1), the survival probability due to mutation may be approximated by the expression

$$PS_S = 1 - \mu \cdot o(S)$$

The Schema Theorem: Mutation

- Adding the effect of mutation on schema count we get the equation

$$\varphi_S^{t+1} \geq \varphi_S^t \cdot \frac{F_S}{\overline{F}^t} \cdot [1 - \chi \cdot \frac{\delta(S)}{l-1}][1 - o(S) \cdot \mu]$$

- The addition of mutation changes our previous equation a little.
- Our previous observations about above average schema still apply.
- The next step, replacement does not change the equation since the new offspring replace the entire population.

The Schema Theorem

- Ignoring the small cross product terms in the equation we may conclude that for a particular schema S the expected number of individuals belonging to S after selection, crossover, and mutation is given by the following equation:

$$\varphi_S^{t+1} \geq \varphi_S^t \cdot \frac{F_S}{\overline{F}^t} \cdot [1 - \chi \cdot \frac{\delta(S)}{l-1} - o(S) \cdot \mu]$$

- This equation is known as the Schema Theorem, or the Fundamental Theorem of Genetic Algorithms.
- The implications of the theorem are very important; schemata with small defining length, low order, and above average fitness receive exponentially increasing trials in subsequent generations of the SGA.
- SGA exhibit a high degree of *implicit parallelism*. The ability to evaluate multiple schema at the same time

Example: Schemata Sampling

- Encoding: binary strings of length 3.
- Population size = 4.
- Fitness Function: Unknown
- Crossover probability = 1.0 and Mutation probability = 0.0.
- See handout for schemata count in mating pool and generation 1.

Two-Armed Bandit Problem

- Models the tradeoff between exploration and exploitation.
- Description: A gambler is given *N* coins with which to play a slot machine with two arms. Each arms have a different payoff. Their mean payoff is $\mu 1$ and $\mu 2$ respectively with a variance of $\sigma 1$ and $\sigma 2$. The goal is to maximize the total payoff after inserting the coins.
- Def. on-line performance: Performance metric that measures the system as a whole. Example: Population average fitness.
- Def off-line performance: Performance metric that measures one aspect of the system. Example: Population best fitness.
- GAs maximize on-line performance for the population.

SGA and Two-Armed Bandit

- TWB is a model for problems of resource allocations in the presence of uncertainty.
- SGA allocates number of trials to best schemas exponentially.
- The SGA does not sample schemata independently as for the TWB.
- The SGA maximizes cumulative payoff over time for any arbitrary landscape.
- The SGA is good for adaptive systems, control problems, learning, prediction, etc. Problems where the solution has to be good enough but not optimal.

Deceptive Functions

- Functions where the low order schemata contain misleading information about high order schemata.

Ex.

SCHEMA	FITNESS
00*	10
01*	8
10*	2
11*	20

- The bias created by FPR selection drives the population towards schemata with fitness that is above the population average fitness.
- On a population with high fitness variance the SGA would not be able to estimate appropriately the true fitness of the schema on a small sample.

Royal Roads Functions

- A general class of functions created to show the positive effects of single point crossover.
- Building block hypothesis: single point crossover combines short high performance schema to create high order schema with better fitness.
- Example: Figure 4.1 in page 128 in textbook

$$P_r^n = \frac{n!}{(n-r)!}$$

64 bits string, 8 schemas each with a fitness of 8.
(11111111******…****************************) = S_1
(********11111111*****************************) = S_2

Experimental Model

- SGA Parameters:
  - Population = 128
  - Crossover prob. = 0.7
  - Mutation prob. = 0.005
  - Selection: Sigma truncation - Each individual is selected 0,1, or 2 times.
- Steepest-ascent hill climbing (SAHC)
  1. Choose string at random. Call string current-hilltop.
  2. Flip each bit systematically and record the fitness of the resulting string.
  3. Go If any string has a higher fitness than current-hilltop, exchange it. Decide ties at random. Go to step 2 if exchange occurred.
  4. to step 1 if no increase in fitness is found.
  5. Stop when the maximum number of fitness evaluation is reached.

Experimental Model (Cont'd)

- Next-ascend hill climbing (NAHC)
  1. Choose string at random. Call string current-hilltop.
  2. From *i* = bit 1 to bit 64. Flip bit *i*. If fitness increases exchange with current-hilltop, otherwise flip bit *i* back.
  3. If no increases where found, goto step 1, otherwise go to step 2.
  4. Stop when the maximum number of fitness evaluation is reached.
- Random-mutation hill climbing (RMHC)
  1. Choose string at random. Call string best-evaluated.
  2. Choose bit at random. Flip bit and if fitness increases or stays the same then exchange string with best evaluated.
  3. Repeat step 2 until the global optima is found or the maximum number of fitness evaluation is reached.

Results

- Fitness Evaluations: All numbers averaged over 200 runs of each algorithm.

	SGA	SAHC	NAHC	RMHC
Mean	61,304	>256k	>256k	6179
Median	54,208	>256k	>256k	5775

- Why the SGA did so poorly?
  - Hitchhiking: low fit schema that benefits from individuals belonging to other high fit schema.
  - Selection: individuals from a schema take over the population easily.
- See figure 4.2 in page 133.

Ideal GA

- Independent sample: The population must be large enough, the selection process slow enough, and the mutation rate high enough to make sure that no single locus is fixed at a single value in the population or even in a large majority of strings.
- Sequestering desired schemas: Selection must be strong enough to preserve high fit schema and low enough to prevent hitchhiking.
- Instantaneous crossover: Crossover probability must minimize the time for recombination between two good schema.
- Speedup over RMHC: The string has to be long enough to make the factor of N speedup significant.

SWENG 584 - Genetic Algorithms

Simple & Steady State GAs

Walter Cedeño

Penn State - Great Valley

(Classical) Simple Genetic Algorithm (SGA)

- Properties
  - Single chromosome using binary encoding.
  - Single point crossover and bit mutation.
  - Fitness proportionate reproduction.
- Algorithm

```
Generate and evaluate initial population at random
For generation = 1 to MAX_GENERATIONS
    Selection step: Create the mating pool
    Recombination step: Apply crossover and mutation
    Replacement step: Replace entire population with offspring
end for
```

Practice Problem

- Problem
  - Finding the optimal parameters for the application of the Simple Genetic Algorithm to the maximization of a function F(x, y).
- Parameters to Optimize
  - Population size
  - Crossover probability
  - Mutation Probability
- What Encoding should we use?
- What will be the fitness function?
- Which Genetic operators should we use for recombination and mutation?

Prisoner's Dilemma

- Two players game. Each player makes a decision on whether to cooperate or defect in every move. Score of every move is given by the following payoff matrix:

		Player 1	
		Cooperate	Defect
Player 2	Cooperate	3, 3	0, 5
	Defect	5, 0	1, 1

- Goal is to get as many points as possible
- Tournament of computer programs going against each other. TIT FOR TAT strategy won. Cooperates in the first game and then does whatever the previous player did in the previous move.
- Can a GA evolve the best strategy if it can remember the last 3 moves made by the opponent? (Axelrod '84).

GA Application to Prisoner's Dilemma

- There are four possibilities for every move {CC, CD, DC, DD}.
- There are 64 = 4^3 possible sequences of 3 moves.
  - CC CC CC, CC CC CD, ...
- A strategy consist of table indicating what to do for each of the 64 sequences.
- Encoding: 64 letter string + 6 initial letters = 70 letters.
- A set of 8 strategies (without TIT FOR TAT) was used to evaluate the quality of the individual.
- Fitness was the average score of the individual against the 8 strategies.
- GA evolved strategies that score better than TIT FOR TAT.
- GA can evolve highly specialized strategies for the environment.

Steady State Genetic Algorithm (SSGA)

- Introduced independently, by Whitley ('88) and by Syswerda ('89).
- Only a subset of the individuals in the population, called the *generation gap* (De Jong, 1975), are replaced in every generation.
- This generation gap is often expressed as a fraction in the interval [0.0, 1.0].
- The mating pool size and the number of offspring generated in every generation is equal to the value of *G* * *n*, where *G* denotes the generation gap and *n* the population size.

SSGA (Cont'd)

- The selection and mating steps are applied as in the SGA. During replacement, only those individuals *G* * *n* at the bottom of the fitness ladder are replaced with the offspring.
- The SSGA is equivalent to the SGA when the value of *G* is set to 1.0, i.e., when the entire population is replaced in every generation.
- In the simplest case, *G* = 2/n, two individuals are selected using FPR, then crossover and mutation are applied to generate two offspring. The two lowest fitness individuals in the population are replaced with the offspring.

SSGA (cont'd)

- In this scenario the new offspring are part of the population and are ready for selection the next time around.
- The classical GA and the steady state GA with $G = 2/n$ are analogous to the Jacobi (simultaneous replacement) and Gauss-Siedel (successive replacement) methods respectively in numerical analysis.
- In general when we refer to the SSGA we are assuming the simple case with $G = 2/n$.

SSGA Pseudo Code

```
Generate initial population of size n at random.
Evaluate initial population.
For gen = 1 to MAX_TRIALS
  Select 2 individuals from the population using FPR.
  Apply crossover and mutation with probability Pc and Pm
   respectively.
  Replace 2 individuals from the population with offspring.
end for
```

SSGA Replacement Step

- The manner in which individuals are chosen from the population for replacement, so that the offspring can be inserted into vacated slots, differs from method to method.
- The simplest way is to eliminate the two lowest fitness individuals from the population.
- Another method, called *inverse ranking* (Syswerda, 1989), has been used for the replacement step. In inverse ranking the population is sorted and starting with the lowest fitness member in the population, two individuals are selected for deletion with probability $> 1/n$. With this operator, low fitness individuals are more likely to be deleted but can still be parents if they are lucky.

SSGA Benefits

- Closer to what happens in nature than the SGA.
- Low fitness individuals disappear faster from the population. Higher fitness individuals survive for more than one generation (generation = n/2 trials).
- Shown to sample the search space (Whitley, 1988) faster than the SGA but at the same time tends to exhibit *premature convergence* faster.
- Can work with a single population.

SSGA Drawbacks

- Not easy to parallelize.
  - By applying all three genetic operators sequentially the algorithm works with only one population and therefore the selection, mating, and replacement steps cannot be applied in parallel for each pair of individuals. Nevertheless, the steady state GA is used by many researchers in the field.
- Tends to converge (in the GA sense) to local optima faster.

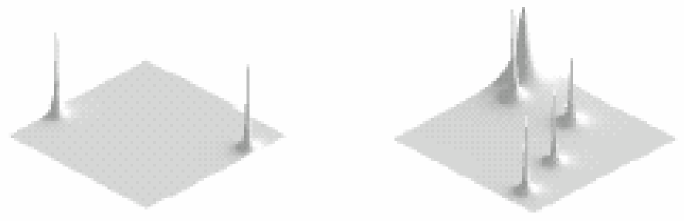
SGA & SSGA Hybrid

```
Generate and evaluate initial population at random
For generation = 1 to MAX_GENERATIONS
  Selection step: Create the mating pool
  Recombination step: Apply crossover and mutation
  For index = 1 to #Offspring
    Replacement step: Insert an offspring at a time in population
  end for
end for
```

Parallel GA Models

- There are basically three parallel GA models (Gordon et. al., 1992) exhibiting different degrees of parallelism;
- Fine grain model (Davidor, 1991; Gorges-Schleuter, 1989; and Spiessens & Manderick, 1991), each solution in the population is mapped to a processor with genetic operators applied between nearest neighbors.
- Distributed model (Mühlenbein et. al., 1991; Tanese, 1989), processors are assigned sub-populations, which converge locally and exchange genetic material among them at fixed interval.
- Direct models (Grefenstette, 1981), exploit the parallelism inherent in the GA operators and the GA structure while having the same properties of a sequential GA.

Home Made Test Functions

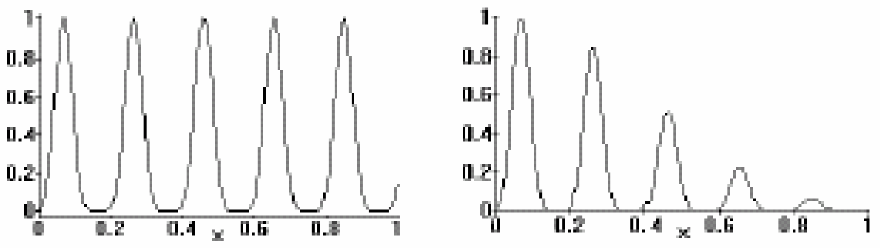


$$F(x,y) = \sum_{i=1}^{np} H_i / 1 + W_i[(x - X_i)^2 + (y - Y_i)^2]$$

Peak Parameters

Function	Peak Location	Width	Height
$F_4(x,y)$ $0 \le x,y \le 65{,}535$	(45k, 2k)	0.0004	100
	(15k, 62k)	0.0004	100
$F_5(x,y)$ $0 \le x,y \le 65.535$	(17.1, 34.4)	1.9	50.5
	(8.7, 4.1)	0.17	44.8
	(20.3, 11.0)	3.6	87.1
	(38.3, 47.9)	3.1	51.5
	(57.9, 54.1)	3.7	55.5

Goldberg's Sine Functions



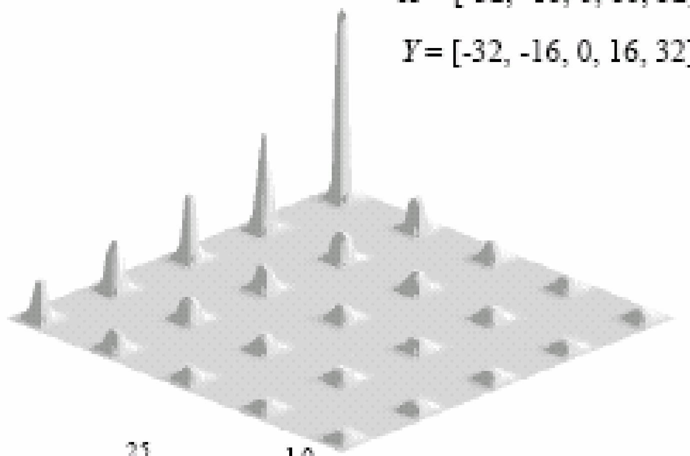
$$F_1(x) = \sin^6(5.1\pi x + 0.5)$$

$$F_2(x) = \exp^{-4(\ln 2)(x-0.0667)^2/0.64} F_1(x)$$

Inverted Shekel's Foxholes

$X = [-32, -16, 0, 16, 32]$ and

$Y = [-32, -16, 0, 16, 32]$.



$$F(x,y) = 0.002 + \sum_{i=1}^{25} \frac{1.0}{(x - X_i)^6 + (y - Y_i)^6 + i}$$

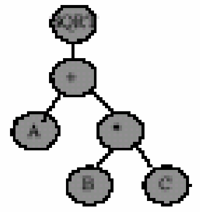
Other Encodings

- Gray Coding - Minimizes the effect of mutation
  - 000, 001, 011, 010, 110, 111, 101, 100
- Real Value coding - Represent genes using floating point values
  - In function optimization or parameter optimization
- Charater encoding (Problem encoding)
  - To represent permutations or multi-character strings
- Self adapting encodings
- Haploid vs Diploid encodings

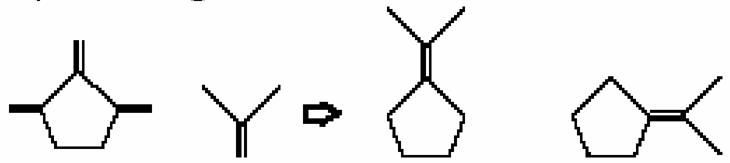
Encodings (Cont'd)

- Tree Encoding

  (SQRT (+ A (* B C)))



- Graph Encoding



Scaling

- A way to alleviate premature convergence.
- Linear scaling: *F' = aF + b*, where *a* and *b* are chosen to scale down or up the maximum fitness to twice the population average fitness.
- Sigma scaling: *F' = F - (avgF - c*variance)*, *c* is chosen between 1-3 and any negative fitness values are set to zero.
- Power law: *F' = F ^ k*, *k* = 1.005. Problem dependent

Selection Operators

- Random Selection
- Tournament Selection
- Boltzmann Selection

$$Sel(i,t) = \frac{e^{f(i)/T}}{\sum e^{f(i)/T}}$$

- Rank Selection

$$Sel(i,t) = \frac{rank(i,t)}{N*(N+1)/2}, \text{where N is the best rank}$$

Crossover Operators

- Single Point Crossover
- Uniform Crossover
- Multi-Point Crossover
- Linear Crossover (Interval Crossover)
- Permutation based Crossover (Next slide) - Minimizes search space.
- Problem dependent Crossover

Permutation Based Crossover

- Partially Matched Crossover (PMX) - Two crossing sites are picked at random. Middle cities are exchange. Map cities between offspring.

```
Ex.  9 8 4|5 6 7 |1 3 2 10 -> 9 8  4|2 3 10|1 6 5 7
     8 7 1|2 3 10|9 5 4 6  -> 8 10 1|5 6 7 |9 2 4 3
```

- Order Crossover (OX) - Starts as PMX. The cities in section are represented as H in the offspring of the other parent. Slide cities in middle section to the left (circularly) until all H are in the middle section. Exchange crossover section between parents. Using same parents as above.

```
          Parent 1                 Parent 2
a. 9 8 4|5 6 7 |1 H H H      8 H 1 |2 3 10|9 H 4 H
b. 5 6 7|H H H |1 9 8 4      2 3 10|H H H |9 4 8 1
c. 5 6 7 2 3 10 1 9 8 4      2 3 10 5 6 7  9 4 8 1
```

Permutation Based Crossover (Cont'd)

- Cycle Crossover (CX) - Select first city from parent 1. Set cities according to parent 1 until it comes back to a placed city. Start with unassigned city in parent 2 and repeat process until all cities are placed.

```
Ex.   9 8 2 1 7 4 5 10 6 3
      1 2 3 4 5 6 7  8 9 10
  Offspring 1:            Offspring 2:
9 - - - - - - - - -     1 - - - - - -  - - -
9 - - 1 - - - - - -     1 - - - - - -  - 9 -
9 - - 1 - 4 - - - -     1 - - - - 6 -  - 9 -
9 - - 1 - 4 - - 6 -     1 - - 4 - 6 -  - 9 -
9 2 - 1 - 4 - - 6 -     1 8 - 4 - 6 -  - 9 -
9 2 - 1 - 4 - 8 6 -     1 8 2 4 - 6 -  - 9 -
9 2 - 1 - 4 - 8 6 10    1 8 2 4 - 6 -  - 9 3
9 2 3 1 - 4 - 8 6 10    1 8 2 4 - 6 - 10 9 3
9 2 3 1 5 4 - 8 6 10    1 8 2 4 7 6 - 10 9 3
9 2 3 1 5 4 7 8 6 10    1 8 2 4 7 6 5 10 9 3
```

Permutation Based Crossover (Cont'd)

- Edge Recombination - Link based.
- Modified Edge Recombination - Link and order based.

Parent 1	Common links	Offspring 1
(6 7 8 1 2 3 9 4 5 0)	(- 7 8 1 - - - - 5 0)	(3 7 8 1 4 6 2 9 5 0)

Parent 2	Common links	Offspring 2
(1 8 7 9 5 0 2 6 3 4)	(1 8 7 - 5 0 - - - -)	(1 8 7 3 5 0 6 2 4 9)

Applying to TSP

- TSP - Travelling Salesperson Problem
- Encoding - Permutation of cities with repeating start and end
  - L A B C M O P R V W Z L
- Crossover: Uniform order based crossover

– Parent 1	L A B C M O P R V W Z L
– Parent 2	L Z W V R P O M C B A L
– Child (step 1)	L - - C M O R - - Z L
– Child (step 2)	L W V C M O P R B A Z L

- Fitness given by distance of tour length from worst tour
- Ex. GA Playground - http://www.aridolan.com/ga/gaa/gaa.html

Mutation Operators

- Bit Mutation
- Inversion
- Problem Dependent
- Regeneration
- Random Exchange

SWENG 584 - Genetic Algorithms

Evolving Neural Networks

Walter Cedeño

Penn State - Great Valley

Neural Networks 101

- Simulate the way in which brain cells transmit information.
- Consist of set of neurons and weighted connections.
- Hopfield Nets, Feed Forward, Recurrent Networks, etc...
- Each neuron has a function that determines when it "fires". Usually is a step function with a given threshold.
- Example on page 66, Fig 2.16

Evolving Neural Networks

- Evolving Weights - Montana and Davis (1989)
- Evolving Network Architectures - Miller, Todd, and Hegde's (1989)
- Evolving Network Architectures - Kitano (1990)
- Evolving Learning Rules - Chalmers (1990)

Evolving Weights

- Montana and Davis
- Chromosome consist of a fixed number of weights between -1.0 and 1.0.
- Crossover consist of applying uniform crossover on non-input nodes and copying the weights of the incoming links.
- Mutation is applied by the weights of the incoming links of a node that is selected.
- See figures 2.18, 2.19 in page 69.
- Results in fig. 2.20 page 70.

Evolving Network Architectures: Direct Encoding

- Miller et. al.
- Chromosome was represent by an N x N matrix, where N is the number of neurons in the fixed network. Connections represented by a '1' in the correct position in the matrix.
- Connections restricted to feedforward networks.
- Crossover will exchange a row between two parents. Row selected at random. Bit mutation was used
- See Figure 2.21 in page 71.

Evolving Network Architectures: Grammatical Encoding

- Kitano
- Use of grammar rules to encode the architecture
- GA will evolve the non-terminal grammar rules given a set of terminal rules and a start symbols S.
- The grammar will represent a matrix that indicates the nodes present in the network and the connections between the nodes. A '1' in the diagonal will indicate if the node is in the network. Non- existent nodes and recurrent connection were ignored.
- Crossover occurred between some points using multi-point crossover.
- Mutations consisted of replacing a symbol.
- See Figure 2.22 in page 74.

Evolving a Learning Rule

- Chalmers
- Evolved a learning rule that can be use in a fully connected feedforward network to calculate the weights. The learning rule is used during learning to adjust the weights based on inputs, network outputs, and correct output.
- DeltaWij = k0(k1 Wij + k2 Ai + k3 Oj + k4 Tj + k5 Wij Ai + k6 Wij Oj + k7 Wij Tj + k8 Ai Oj + k9 Ai Tj + k10 Oj Tj)
- Evolved the coefficients k.
- Used 20 test problems to evolve a learning rule for all 10 of them.
- Try it using only 10 test problems and results are shown in Fig 2.25, page 78.

SWENG 584 - Genetic Algorithms

Multi-Objective Problems

Walter Cedeño

Penn State - Great Valley

Multi-Objective and Combinatorial Problems

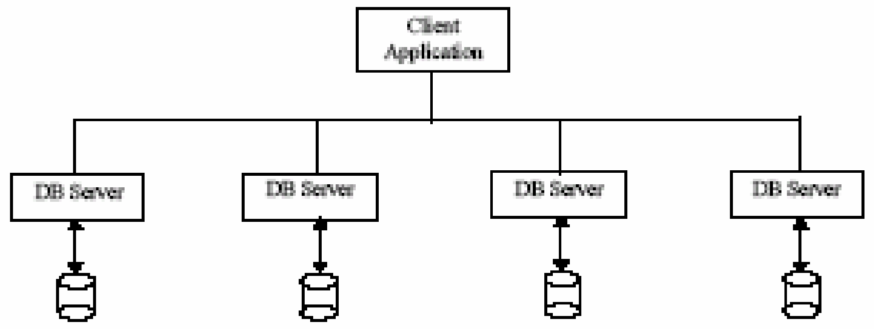
- Multi-Objective problems are characterize by the optimization of multiple conflicting objectives.
- The GA model must balance between the objectives.
- Pareto Optimality: An *n* objective solution (c1, c2, …, c*n*) dominates a solution (d1, d2, …, d*n*) if for all *i* (c*i* >= d*i*) and exist *i* such that (c*i* > d*i*).
- Combinatorial problems are mostly NP-Hard and require special operators for crossover and mutation.
- These type of problems exhibit a complex search space with the presence of multiple local optima.

The File Design Problem

- The design of distributed databases requires a configuration of the data such that queries are satisfied by accessing a minimum number of locations and the system load is equitably distributed among all locations.
- This problem is NP Hard and requires the optimization over conflicting objectives.
- A genetic algorithm based on multi-niche crowding combines heuristics with parallel processing to provide a suitable approach to solve this problem.
- Cedeño, W. and Vemuri, V. (1997). Database design with genetic algorithms. D. Dasgupta and Z. Michalewicz (eds), *Evolutionary Algorithms in Engineering Applications*, Springer Verlag, 3/97, ISBN: 3-540-62021-4.

Distributed Database System

- Goal: Find an assignment of database records to files that minimizes the average number of files examined over all single attribute queries.
- All nodes must handle about the same number of queries on average.
- A query must be satisfied by accessing only a small number of nodes.



Parallel Implementation of the MNC GA

```
• Generate initial population of size n at random.
  Evaluate initial population.
  For gen = 1 to MAX_GENERATIONS
      Use crowding selection to find mate for all individuals
      Mate and mutate all pairs
      Insert offspring sequentially in population using
      WAMS replacement
```

- Special crossover operator that uses heuristics.
- Multiple test done on different platforms.
- Implementation done using parallel functional language called SISAL.

Mathematical Definition of the File Design Problem

- Given a set of *N* records, and each is characterized by a single attribute *A* that takes *h* different values { a1, a2, ..., a*h*}.
- There are *ni* records corresponding to attribute a*i*, i.e., n1 + n2 + ... + n*h* = *N*.
- Assumption: Queries for records of any given attribute are equally likely.
- We also have *K* files of size *b* such that *K* * *b* = *N*. Different file sizes can also be accomodated to represent realistic configurations. The approach presented here is not limited by the simplifications made to the problem.
- The constants *K*, *b*, n*i*, *N*, and *h* are all positive integers.
- Find an assignment of the *N* records to the *K* files such that the average number of files (*ANF*) accessed over all possible single-attribute queries is minimized. In other words, an assignment of the records to the files must be found such that (on average) queries for the records with the same attribute can be satisfied by reading from as few files as possible.

Sample Problem

- Given 12 (*N=12*) records in 2 files of size 6.
- Four attributes values (*h=4*) are {B, C, F, V} and have {2, 7, 1, 2} records respectively. (n1=2, n2=7, n3=1, n4=2)
- Insert the records in 2 (*K=2*) files of size 6 (*b=6*).

File 1	File 2	fex(B)	fex(C)	fex(F)	fex(V)	ANF
C C C C C C	C B B V V F	1	2	1	1	1.25
C C C B V V	C C C C B F	2	2	1	1	1.50
C C C C B V	C C C B V F	2	2	1	2	1.75
C C C B B F	C C C C V V	1	2	1	1	1.25

ANF Calculation

- The formua is given by:

$$\sum_{i=1}^{h} fex(a_i) \bigg/ h$$

- where *fex(ai)* returns the number of files containing records with attribute *ai*.

File 1	File 2	fex(B)	fex(C)	fex(F)	fex(V)	ANF
C C C C C C	C B B V V F	1	2	1	1	1.25
C C C B V V	C C C C B F	2	2	1	1	1.50
C C C C B V	C C C B V F	2	2	1	2	1.75
C C C B B F	C C C C V V	1	2	1	1	1.25

Chromosome Encoding

- A chromosome represents a valid solution to the problem.
- It consists of an array of *N* alleles corresponding to each of the records in the problem.
- Each allele may assume a value between 0 and *K* - 1 inclusive, indicating the file containing the record. A valid encoding is a *N* digit number in base *K* where all digits appear exactly *b* times.
- Example for the records shown in last slide. To make clear that the chromosomes are not binary we selected in this case *K* = 3 files of size *b* = 4.

```
   Rec. number:    1   2   3   4   5   6   7   8   9  10  11  12
Rec. attribute:    B   B   C   C   C   C   C   C   C   F   V   V
  Chromosome 1:    0   0   0   0   1   1   1   1   2   2   2   2
  Chromosome 2:    0   0   1   1   2   2   2   0   0   1   1   2
```

Similarity Metric

- Similarity between two solutions is measured from the number of records assigned to the same file.
- The following example shows two chromosome using the data from the previous slide. As in the previous slide we have 3 files of size 4.
- Each digit indicates the file where a record is located. In this example we have a similarity value of 8, that is 8 records have been assigned to the same file.

```
           Records:  B B C C C C C C C F V V
      Chromosome 1:  1 0 2 2 0 1 0 1 1 0 2 2
      Chromosome 2:  2 0 1 2 0 1 0 1 2 0 1 2
Similar assignments:   1   2 3 4 5 6   7   8
```

Crossover Operator

- Designed with two goals in mind:
  1. The characteristics expressed in both parents will be expressed in the offspring, thus preserving the schemata in both solutions.
  2. Fitness should be improved when combining two similar solutions.
- First, transfer similar characteristics from the parents to the offspring.
- Those records not assigned are counted for each attribute and sorted in decreasing order.
- Offspring inherits similar alleles from parents:

```
Record Attribute: B B C C C C C C C F V V
       Parent 1:  0 0 1 2 2 2 0 1 0 1 2 1
       Parent 2:  0 1 0 1 2 2 1 2 0 0 1 2
      Offspring:  0 - - - 2 2 - - 0 - - -
```

- Unassigned records by attribute: B:1, C:4, F:1, V:2

Heuristic Based Crossover

- One offspring is created using a "best fit" method based on the contents of files. In this approach unassigned records are located into files where records with the same attribute reside. The main idea is to group files with the same attribute in the same file as much as possible.
- The second offspring is created using a "first fit" method based on the empty space in the files. Here the unassigned records will be located where file space is available for records with the same attribute.
-

```
           Offspring 1                 Offspring 2
         Best Fit Method             First Fit Method
     B B C C C C C C C F V V     B B C C C C C C C F V V
     0 - - - 2 2 - - 0 - - -     0 - - - 2 2 - - 0 - - -
C:4  0 - 2 2 2 2 0 0 0 - - -     0 - 1 1 2 2 1 1 0 - - -
V:2  0 - 2 2 2 2 0 0 0 - 1 1     0 - 1 1 2 2 1 1 0 - 0 0
B:1  0 1 2 2 2 2 0 0 0 - 1 1     0 2 1 1 2 2 1 1 0 - 0 0
F:1  0 1 2 2 2 2 0 0 0 1 1 1     0 2 1 1 2 2 1 1 0 2 0 0
```

Mutation Operator

- Mutation is applied with a fixed probability for each allele.
- When an allele is selected for mutation another position in the chromosome is selected at random and the two values are interchanged.
- Such mutations may introduce a new configuration in succeeding generations.
- Example

```
                   B B C C C C C C C F V V
Before Mutation:   0 1 2 2 2 2 0 0 0 1 1 1
After Mutation:    0 1 2 2 2 2 1 0 0 1 0 1
```

Fitness Function

- The fitness function captures 3 important objectives of an optimal solution:
  1. Minimize ANF
  2. Maximize grouping of records with the same attribute.
  3. Spread equally the number of records with the same attribute among the minimum number of files needed to store them.
- The last two points are captured in a grouping term (GT) and balancing term (BT) respectively.
- The two terms are contradictory in the sense that GT wants to group records together, while BT wants to spread records equally across files.
- These three terms are added together, with different weight values, to get the fitness of a solution. The GT value is given a higher weight over the BT value because it promotes lower ANF values in the solution.

Fitness Function (Cont'd)

- ANF Term: $ANF = \sum_{i=1}^{h} fex(a_i) \Big/ h$
- Lower and Upper Bounds of ANF:

$$min\_anf = \sum_{i=1}^{h} \lceil n_i / b \rceil \Big/ h \leq ANF \leq max\_anf = \sum_{i=1}^{h} \min(n_i, K) \Big/ h$$

- GT Term: $GT = \sum_{i=1}^{h} \sum_{j=1}^{K} \left(attr(a_i, j) / n_i \right)^2$

  where *attr*(a_i,*j*) = number of records of attribute a_i in file j

- BT Term: (Closer to zero are better values)

$$BT = \sum_{i=1}^{h} \sum_{j=1}^{K} \left\lfloor \left| attr(a_i, j) - n_i / \lceil n_i / b \rceil \right| \right\rfloor, \ when\ attr(a_i, j) \neq 0.$$

Fitness Function (Cont'd)

- The three terms *ANF*, *BT*, and *GT* are used to define the fitness value for a solution.
- Since higher positive values are used to indicate a better solution, the terms are normalized to return values between 0.0 and 1.0.
- A weighted value for each term is then added to form the final fitness value as indicated by the following formula

$$fitness = 0.70 * \frac{GT}{h} + 0.25 * \frac{max\_anf - ANF}{max\_anf - min\_anf} + 0.05 * \frac{1.0}{1.0 + BT}.$$

Experimental Data

- A test set consisting of various configurations was assembled.

Case Num.	Num. Files	File Size	Num. Attr.	Number of Records per Attribute n_1 n_2 n_3 n_4 n_5 ... n_n	min anf exist
1	5	20	10	7, 2, 3, 1, 5, 17, 18, 13, 15, 19	Yes
2	10	10	10	7, 2, 3, 1, 5, 17, 18, 13, 15, 19	Yes
3	5	20	10	7, 4, 3, 8, 6, 11, 18, 15, 10, 18	No
4	10	10	10	7, 4, 3, 8, 6, 11, 18, 15, 10, 18	No
5	5	20	21	7, 4, 3, 8, 6, 1, 8, 5, 10, 8, 1, 2, 4, 9, 5, 1, 6, 2, 3, 3, 4	Yes
6	5	20	15	7, 4, 9, 7, 7, 4, 9, 5, 9, 7, 9, 5, 6, 7, 5	No

GA Parameters

- Initial population generated at random. Record where placed in files at random for each individual.

Population size:	100
Number of generations:	50
Mating probability:	0.95
Mutation probability:	0.01
Cs for selection:	4
Cf for replacement:	3
s for replacement:	5

- To evaluate the performance of the implementation three different platforms were used: the SGI Iris 4D, Cray Y-MP, and Cray C90.

Results

Case	Gen	Best Solution
1	3	4444444, 11, 333, 0, 22222, 33333333333333333, 11111111111111111, 4444444444444, 222222222222222, 000000000000000000
2	4	9999999, 77, 111, 6, 88888, 333333333113311111, 2222222222777777777, 5555555555999, 4444444444488888, 0000000000666666666
3	25	2222222, 0000, 222, 44444444, 333333, 44444444444, 000000000100001000, 33333343333333, 2222222222, 11111111111111111
4	15	8888888, 9999, 888, 55555555, 999999, 22222222227, 000000000066666666, 111111111155667, 3333333333, 444444444477777777
5	5	0000000, 3333, 222, 33333333, 222222, 0, 00000000, 11111, 2222222222, 44444444, 2, 44, 4444, 11111111, 3333, 4, 111111, 44, 333, 444, 0000
6	5	0000000, 2222, 222222222, 4444444, 222222, 3333, 333333333, 44444, 00040000, 3333333, 111111111, 11111, 111111, 4444444, 00000

Performance Results

- MNC GA execution time in seconds for 50 generations

Platform	1 Proc.	2 Proc.	3 Proc.	4 Proc.
Y-MP C90	12.9799	8.6748	6.4425	5.4930
Y-MP	18.6106	10.4642	8.9153	6.3648
SGI Iris	25.8900	16.5300	13.4200	11.9000

- A speedup between 2.2 and 2.9 was achieved with four processors in the three different platforms.

Conclusions

- Parallel version of the MNC GA model performed equally wells as the steady state counterpart. Diversity was maintained during the run and the last generation contained multiple optima.
- Creating genetic operators that use heuristics enhanced the convergence of the algorithm while at the same time allowing multiple solutions to coexist.
- A better speedup can be achieved using more complex fitness functions or by introducing a parallel version of the *WAMS* replacement operator. The parallel version must retain the important properties of WAMS. Competition among solutions within the same peak is encouraged while allowing competition among the multiple peaks as well.

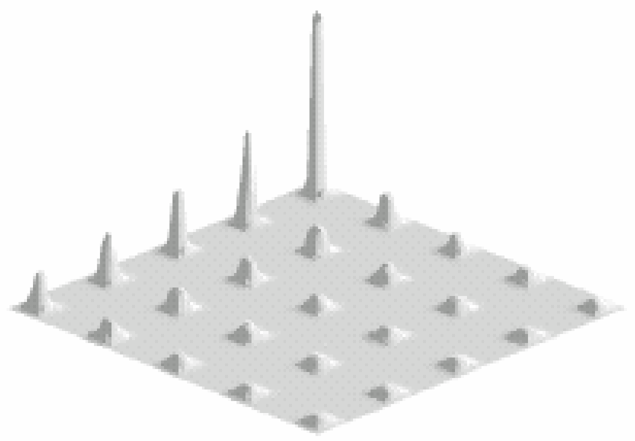
SWENG 584 - Genetic Algorithms

Multi-Modal Search, Niching, and Speciation

Walter Cedeño

Penn State - Great Valley

Inverted Shekel's Foxholes



Genetic Algorithms for Multimodal Search

- Must avoid converging to only one of the optima.
- Many applications where the location of other local optima and their values are of interest.
- Promote the creation of species, individuals with common characteristics.
- Converge to niches, subdomains of the search space.
- Multimodal spaces are deceptive due to the number of optima in the search spaces.
- Today we will describe multiple techniques for Multimodal Function Optimization.

Preselection (Cavicchio, 1970)

- Modification to the replacement step of a classical GA.
- In preselection not all the generated offspring are chosen for the new population. Only the offspring with higher fitness than their parents, replace their parents in the next generation.
- Cavicchio claims to have maintained a more diverse population with relatively small population sizes (n = 20). No significant added complexity and convergence of the algorithm was improved.
- Like the other traditional GAs, this stratagem does not keep stable species subpopulations for many generations and it only converges to one optimum.

Crowding (De Jong, 1975)

- Generalization of preselection where offspring replace similar individuals from the population. An offspring is inserted in the population as follows.
- First, a group of CF individuals is selected at random from the population. CF, called the crowding factor, indicates the size of the group. (2 or 3)
- Second, the bit strings in the offspring chromosome are compared with those of the CF individuals in the group using the Hamming distance, which measures the number of differing bits, as a measure of similarity.
- The group member that is most similar to the offspring is now replaced by the offspring.
- The only limitation is that it does not converge to multiple solutions and after many generations one of the peaks takes over.

Sharing (Goldberg and Richardson 1987)

- In sharing, the fitness value of an individual is adjusted downwards according to the number of individuals in its neighborhood or niche.
- A sharing function assumes a value between 0 and 1 for any distance value dij between any two individuals Ii and Ij in the population.
- Triangular sharing function counts the individuals in a given neighborhood.

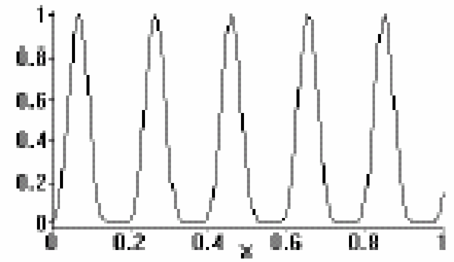
$$Sh(d_{ij}) = 1 - d_{ij} / \sigma_{share}$$

- New fitness given by:

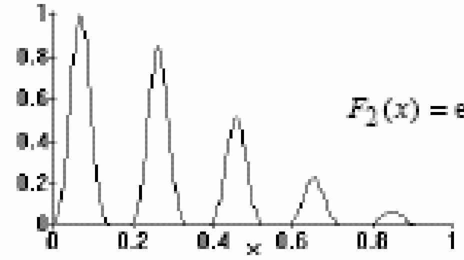
$$F_i' = F_i \Big/ \sum_{j=1}^{n} Sh(d_{ij})$$

- Complexity increased to *n*^2 steps.
- Problem dependent: Must define good value for neighborhood to avoid overlapping niches.

Goldberg's Sine functions



$$F_1(x) = \sin^6(5.1\pi x + 0.5)$$



$$F_2(x) = \exp^{-4(\ln 2)(x-0.0667)^2 / 0.64} F_1(x)$$

Improvements to Sharing

- Mating Restriction (Deb and Goldberg 1989).
  - Improvements to scaling using phenotypic distance.
  - Individuals in the neighborhood were allowed to participate in crossover.
  - Same drawbacks as before. Only better convergence was observed.
- Technique from cluster analysis (Yin and Germay 1993).
  - Reduce the complexity of sharing to O(n).
  - After each generation, an adaptive clustering algorithm groups the individuals in the population into a number of clusters dynamically.
  - These clusters are then used to compute the sharing value and the derated fitness for each individual in the population.
  - On the other hand the algorithm exhibits the same behavior as sharing alone in the sense that the parameters of the clustering algorithm are problem dependent.

Deterministic Crowding (Mahfoud, 1992)

- Selection pressure is eliminated by allowing individuals to mate at random.
- Pressure is applied, however, during the replacement step using preselection.
- Each of the two offspring is first paired with one of the parents; this pairing is not done randomly, rather the pairings are done in such a manner that the offspring is paired with the most similar parent.
- Then each offspring is compared with its paired parent and the individual with the higher fitness is allowed to stay in the population and the other is eliminated.
- Deterministic crowding adds little complexity to the classical GA.
- Cannot maintained multiple solutions for many generations.

Fitness derating (Beasley et. al., 1993)

- Allows unimodal optimization methods to be applied to multimodal problems by using the knowledge gained in a run.
- In fitness derating, a classical GA is first applied to the problem at hand. The solution obtained is used to derate the fitness values of the individuals in the neighborhood of the solution. The GA is then applied again.
- The authors used sharing functions to derate the values at every peak solution found so far. They modified the fitness function after each run by adding a derating term to lower the fitness values of individuals around a radius selected by the user.
- The additional term and the repetitive application of the algorithm makes the complexity of fitness derating similar to that of sharing.
- As in sharing, fitness derating can succeed only with knowledge from the search space.

Subpopulations Schemes (Spears, 1994)

- Diversity is maintained by creating subpopulations in a classical GA using tag bits. A similar tag number identifies individuals in the same subpopulation.
- Mating and selection is restricted to individuals belonging to the same subpopulation, i.e. having the same tag value.
- This method replaces the concept of distance between individuals with labels identifying them as members of the same subpopulation. It also uses fitness sharing during each generation, but uses the number of individuals within each subpopulation as the niche count.
- This approach reduces the complexity of the sharing scheme while obtaining similar results.
- Not able to sustain multiple solutions in lower peaks for many generations.

Restricted Tournament Selection (Harik, 1995)

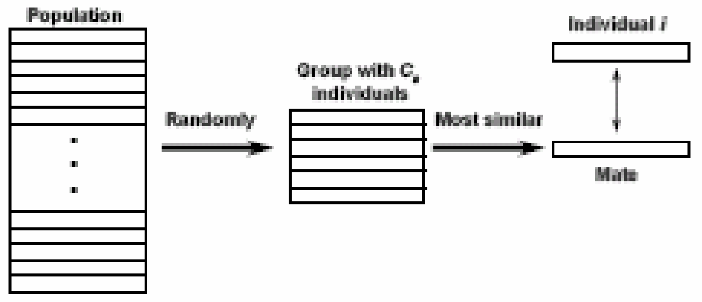
- Modifies the selection and replacement steps in the SSGA.
- In this approach selection is done at random. Both parents are selected at random from the population.
- Offspring are inserted in the population by chosing a group of w individuals from the population at random with replacement. Then the individual in the group which is most similar to the offspring is selected.
- The offspring replaces the chosen individual in the population if its fitness is higher, otherwise the offspring is eliminated. As we will see later this approach is a special case of our multi-niche crowding GA with replacement done greedily.
- The major flaw of this approach is its greedy nature. This greedy stratagem prevents possible new areas of the search space to be discovered, a necessary property when applying the technique to a dynamic landscape.

Multi-Niche Crowding (MNC): A GA suitable for Multimodal Landscapes

- Modifications to classical GA:
  - Selection step: *Crowding selection.*
  - Replacement step: *Worst among most similar* replacement.
  - Similarity measure: To compare individuals.
- Goals behind the modifications:
  - To encourage mating among members of the same niche.
  - Offspring should replace low fitted individuals from the same niche.
- Properties:
  - Competition occurs naturally within and between niches.
  - Stable subpopulations are maintained in every niche.

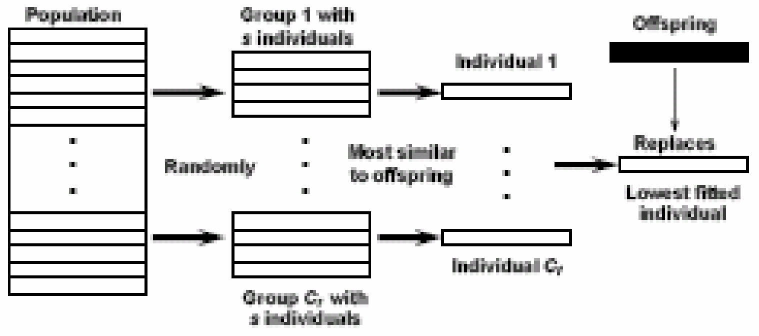
Crowding Selection

- Select individual *i* either randomly or sequentially.
- Form a group of C_s individuals at random from the population.
- Select most "similar" one to individual *i* for mating.



Worst Among Most Similar Replacement

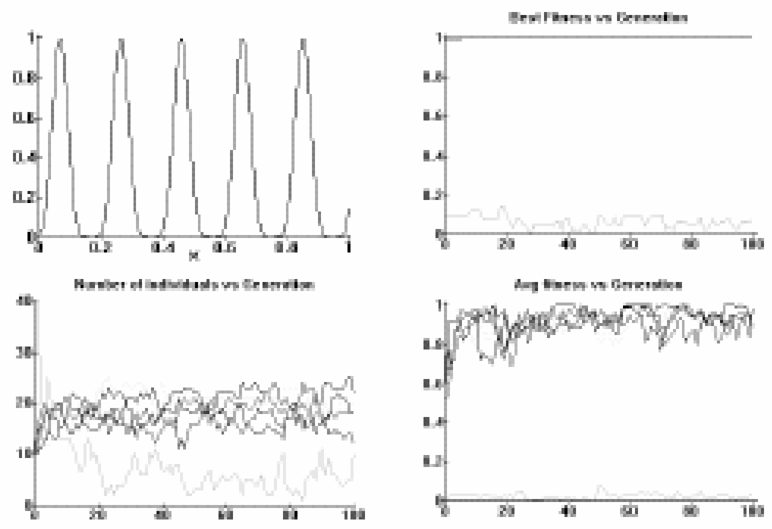
- Form C_f groups with *s* individuals selected at random from the population.
- Identify individual most similar to offspring in each group.
- Select individual with the lowest fitness to die.



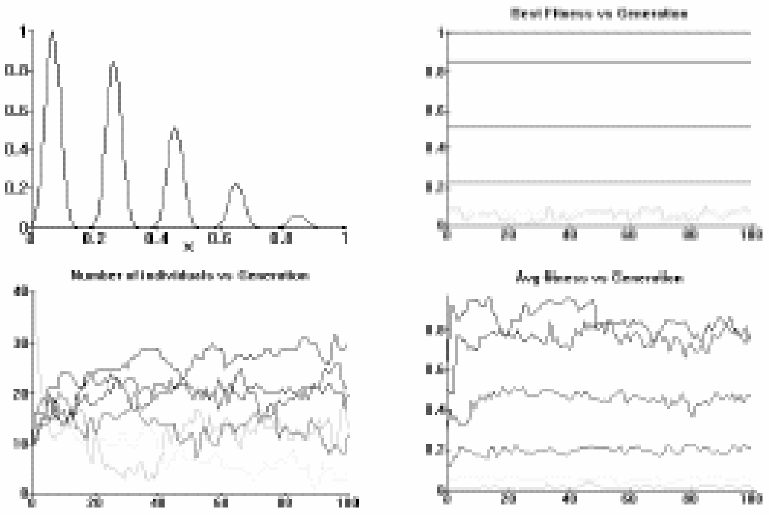
Complexity Analysis

GA Method	Selection steps	Mating steps	Replacement steps	Other	Total
Classical	n	$n/2 + nl$	0		$n(l + 3/2)$
Crowding	n	$n/2 + nl$	$n*C_f$		$n(l + C_f + 3/2)$
Sharing	n	$n/2 + nl$	0	$n(n - 1)/2$	$n^2/2 + n(l + 1)$
Deterministic Crowding	$2n$	$n/2 + nl$	n		$n(l + 9/2)$
MNC	$n*C_s$	$n + nl$	$n*C_f*s$		$n(l + C_s + C_f*s + 3)$

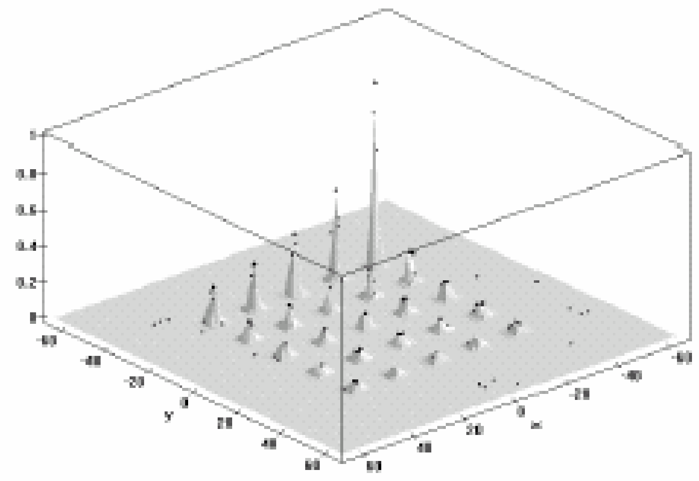
Results using Function F_1



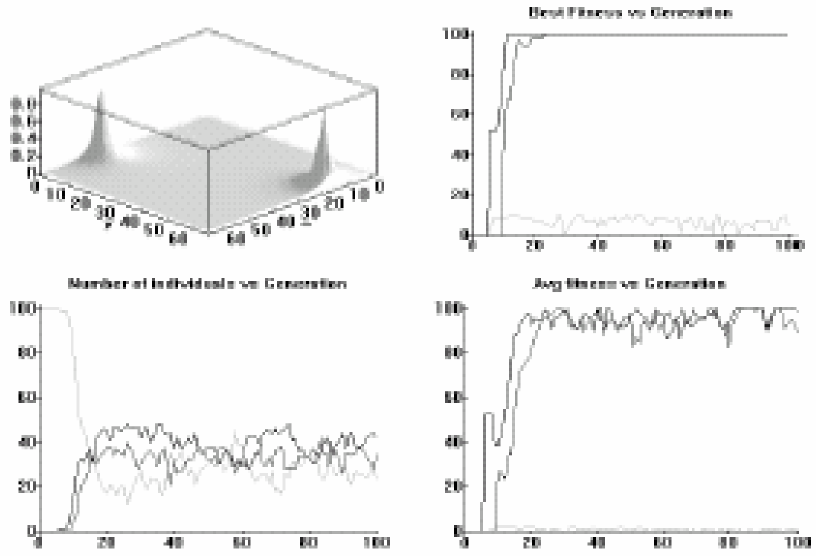
Results using Function F_2



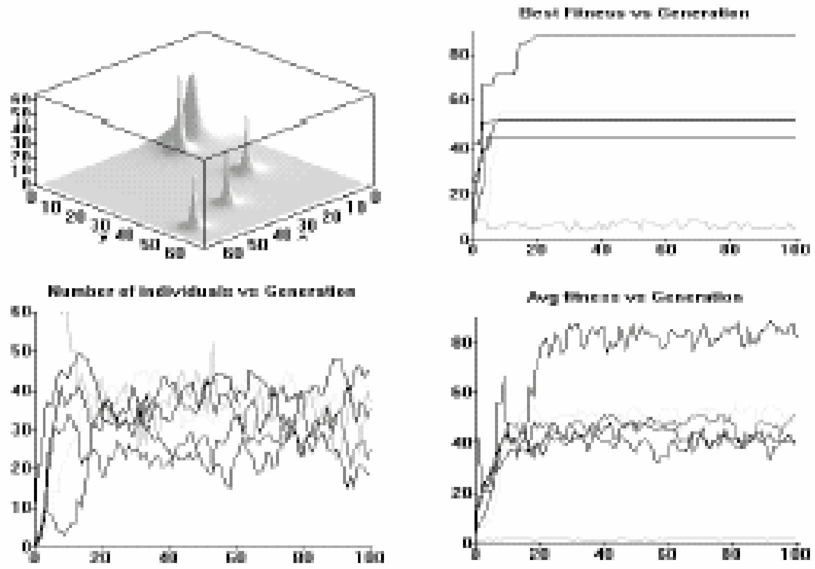
Results on Function F_3



Results using Function F_4



Results using Function F_5



Comments on MNC Method

- Diversity is maintained throughout the search.
- Maintains stable subpopulations.
- Converges to multiple optima.
- No prior knowledge about the search space is needed.
- More analysis is needed to determine factors affecting the search.

Function F_5 Parameters

Peak Location	Width (w)	Height (h)	Color
(17.1, 34.4)	1.9	50.5	black
(8.7, 4.1)	0.17	44.8	blue
(20.3, 11.0)	3.6	87.1	brown
(38.3, 47.9)	3.1	51.5	red
(57.9, 54.1)	3.7	55.5	yellow

Comparison with other Methods

- Comparison of parallel hillclimbing (PH), fitness derating (FD), sharing (SH), deterministic crowding (DC), and multi-niche crowding (MNC).

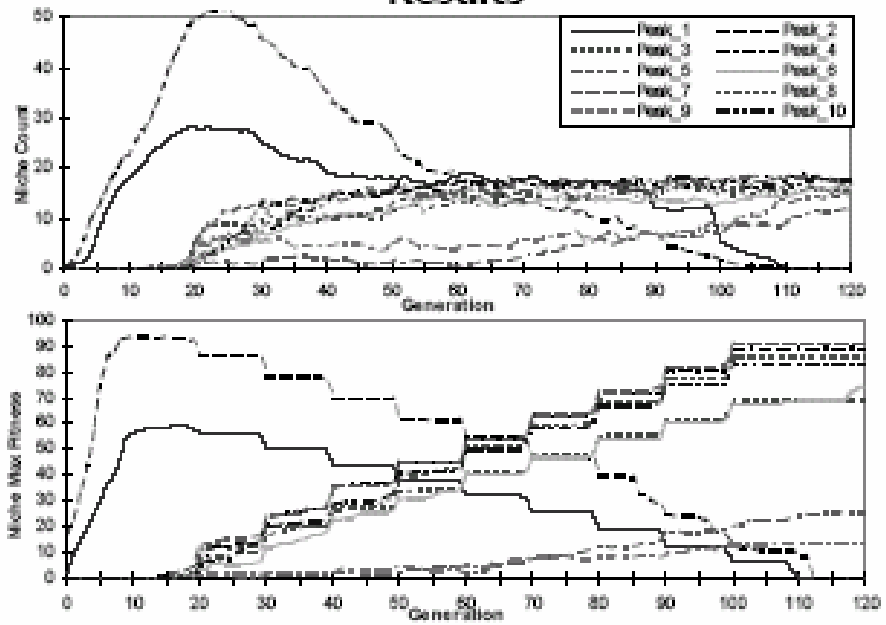
Method	Average Niche Count	Average Number of Generations	Average Function Evaluations for GA	Average Function Evaluations for GA + PH
			Function F_1	
PH	2.72			1,017
FD	3.68	46.40	738	4,112
SH	5.76	8.00	264	2,431
DC	2.40	28.00	380	1,246
MNC	3.14	12.30	197	1,026
			Function F_2	
PH	2.72			1,021
FD	4.64	75.60	1,770	8,632
SH	8.96	8.70	442	3,827
DC	2.40	27.40	372	1,264
MNC	3.16	12.30	197	1,027
			Function F_3	
PH	12.29			29,017
FD	3.58	146.30	12,202	46,657
SH	5.12	11.80	1,638	12,910
DC			> 1,500,000	
MNC	3.07	21.50	2,150	12,133

Dynamic Landscapes

- A multimodal dynamic landscape presents a very challenging problem to any search technique for various reasons.
  - First, the heights and widths are constantly changing.
  - Second, new peaks that are constantly emerging must be located quickly to avoid missing solutions that could potentially be useful.
  - Third, solutions already converged to a peak that has since flattened (become a peak with a small height) must be diverted to other areas of the search space in search of better solutions.
  - Finally, solutions that have been found previously must be re-evaluated after some time to maintain an accurate value of its fitness.
- Test Function:

$$f(x,y) = (1 - 0.1\left\lfloor \frac{k}{g} \right\rfloor) h_1(x,y) + 0.1\left\lfloor \frac{k}{g} \right\rfloor h_2(x,y)$$

Results


Peak_1
Peak_2
Peak_3
Peak_4
Peak_5
Peak_6
Peak_7
Peak_8
Peak_9
Peak_10
Niche Count
Generation
Niche Max Fitness
Generation