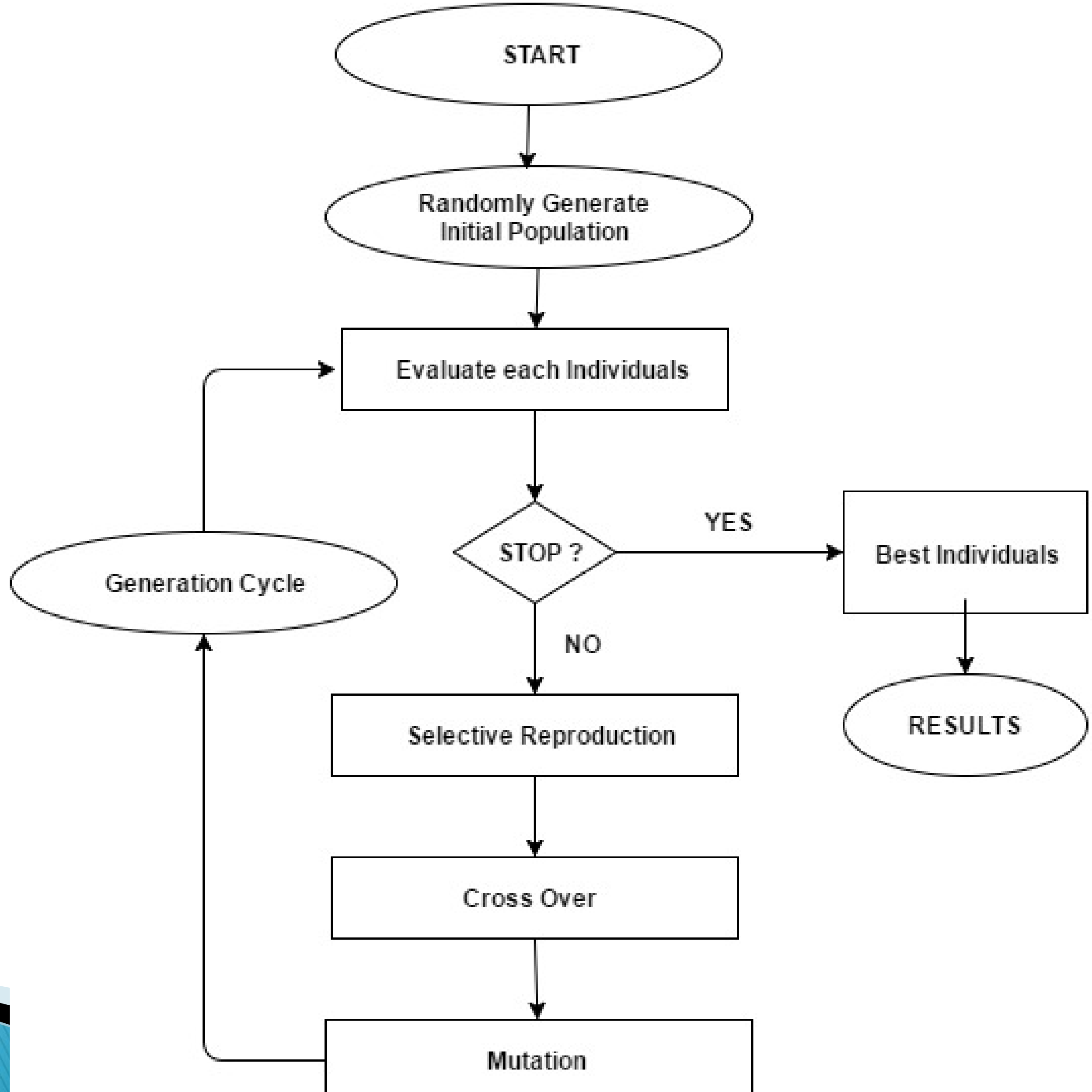
Genetic Algorithms

Prepared by
Md. Shah Fahad

Genetic Algorithm Flow Chart



Simple Genetic Algorithm

- produce an initial population of individuals
- evaluate the fitness of all individuals
- **while** termination condition not met **do**
- select fitter individuals for reproduction
- recombine between individuals
- mutate individuals
- evaluate the fitness of the modified individuals
- generate a new population
- **End while**

Basic principles 1

- Coding or Representation
  - String with all parameters
- Fitness function
  - Parent selection
- Reproduction
  - Crossover
  - Mutation
- Convergence
  - When to stop

Basic principles 2

- An individual is characterized by a set of parameters: Genes
- The genes are joined into a string: Chromosome

- The chromosome forms the genotype
- The genotype contains all information to construct an organism: the phenotype

- Reproduction is a “dumb” process on the chromosome of the genotype
- Fitness is measured in the real world (‘struggle for life’) of the phenotype

Population (Representations)

Chromosomes could be:

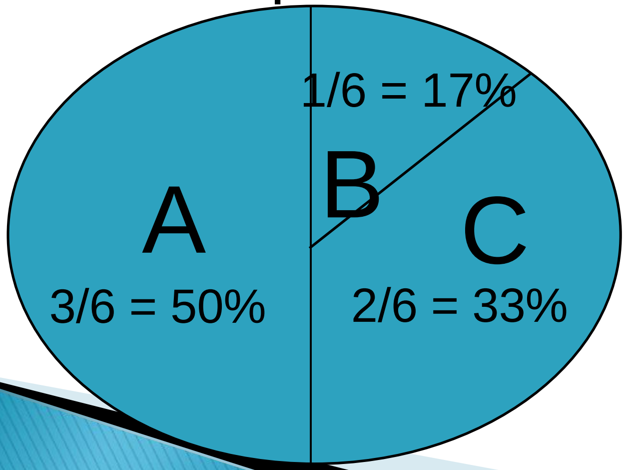
- Bit strings (0101 ... 1100)
- Real numbers (43.2 -33.1 ... 0.0 89.2)
- Permutations of element (E11 E3 E7 ... E1 E15)
- Lists of rules (R1 R2 R3 ... R22 R23)

Reproduction

- Crossover
  - Two parents produce two offspring
  - There is a chance that the chromosomes of the two parents are copied unmodified as offspring
  - There is a chance that the chromosomes of the two parents are randomly recombined (crossover) to form offspring
  - Generally the chance of crossover is between 0.6 and 1.0
  - Only crossover probability
- Mutation
  - There is a chance that a gene of a child is changed randomly
  - Generally the chance of mutation is low (e.g. 0.001)
  - Both mutation probability and rate

Parent/Survivor Selection

- Probability selection : proportional to their fitness
- Main idea: better individuals get higher chance
  - Chances proportional to fitness
  - Implementation: roulette wheel technique
    - Assign to each individual a part of the roulette wheel
    - Spin the wheel n times to select n individuals



fitness(A) = 3

fitness(B) = 1

fitness(C) = 2

Tournament Selection

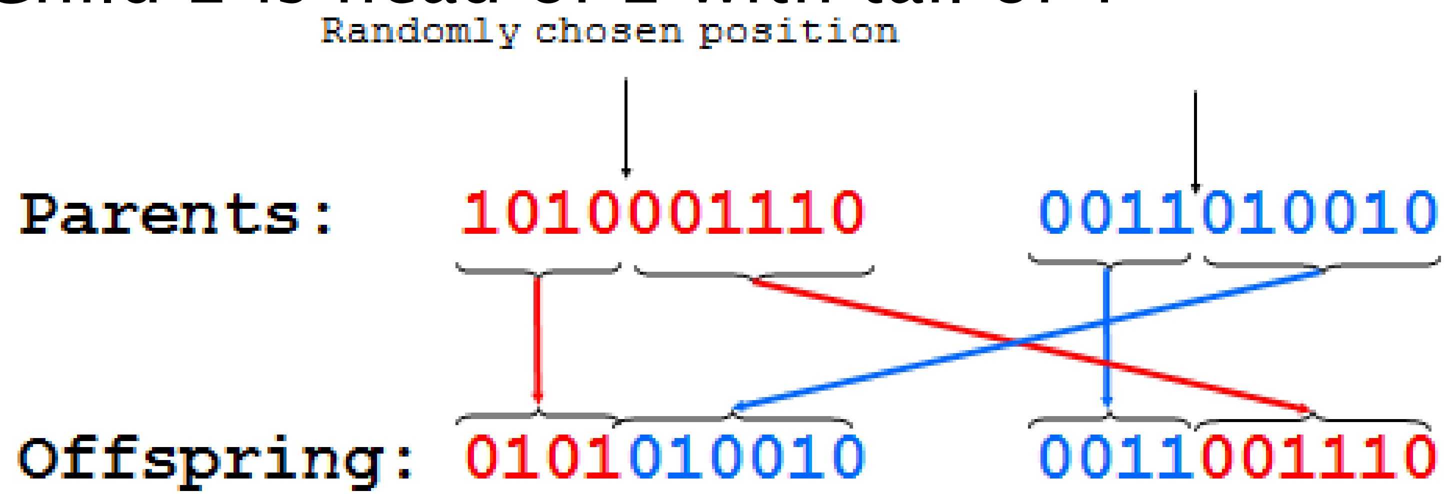
- Selecting an individual from a population of individuals
- involves running several "tournaments" among a few individuals (or 'chromosomes') chosen at random from the population.
- The winner of each tournament (the one with the best fitness) is selected for crossover.
- Selection pressure is easily adjusted by changing the tournament size.
- If the tournament size is larger, weak individuals have a smaller chance to be selected.

Global Optimal

- Trade off between
  - Exploration: introduction of new combination of features
  - Exploitation (Premature convergence): keep the good features in the existing solution

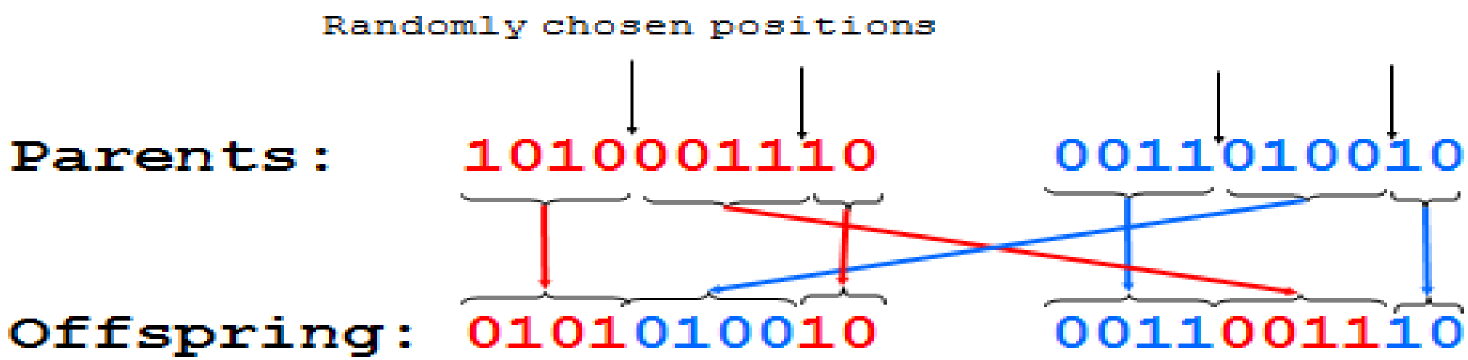
One-point crossover 1

- Randomly one position in the chromosomes is chosen
- Child 1 is head of chromosome of parent 1 with tail of chromosome of parent 2
- Child 2 is head of 2 with tail of 1



Two-point crossover

- Randomly two positions in the chromosomes are chosen
- Avoids that genes at the head and genes at the tail of a chromosome are always split



Uniform crossover

- A random mask is generated
- The mask determines which bits are copied from one parent and which from the other parent
- Bit density in mask determines how much material is taken from the other parent (takeover parameter)

Mask: 0110011000 **(Randomly generated)**

Parents: 1<u>01</u>000<u>01</u>110 <u>0</u>01<u>10</u>10<u>010</u>

Offspring: 0011001010 1010010110

Types of mutation

- **Flip Bit**–This mutation operator takes the chosen genome and inverts the bits.
- **Boundary**–This mutation operator replaces the genome with either lower or upper bound randomly.
- **Uniform**–This operator replaces the value of the chosen gene with a uniform random value selected between the user-specified upper and lower bounds for that gene.
- **Gaussian** –This operator adds a unit Gaussian distributed random value to the chosen gene. If it falls outside of the user-specified lower or upper bounds for that gene, the new gene value is clipped.

Last three are used for real and float genes.

An example

- Simple problem: max x^2 over $\{0,1,\ldots,31\}$
- GA approach:
  - Representation: binary code, e.g. 01101 $\leftrightarrow$ 13
  - Population size: 4
  - 1-point xover, bitwise mutation
  - Roulette wheel selection
  - Random initialization
- We show one generational cycle done by hand

X^2 example: crossover

String no.	Mating pool	Crossover point	Offspring after xover	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 | 1	4	0 1 1 0 0	12	144
2	1 1 0 0 | 0	4	1 1 0 0 1	25	625
2	1 1 | 0 0 0	2	1 1 0 1 1	27	729
4	1 0 | 0 1 1	2	1 0 0 0 0	16	256
Sum					1754
Average					439
Max					729

X^2 example: mutation

String no.	Offspring after xover	Offspring after mutation	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 0	1 1 1 0 0	26	676
2	1 1 0 0 1	1 1 0 0 1	25	625
2	1 1 0 1 1	1 1 0 1 1	27	729
4	1 0 0 0 0	1 0 1 0 0	18	324
Sum				2354
Average				588.5
Max				729

x^2 example: selection

String no.	Initial population	x Value	Fitness $f(x) = x^2$	$Prob_i$	Expected count	Actual count
1	0 1 1 0 1	13	169	0.14	0.58	1
2	1 1 0 0 0	24	576	0.49	1.97	2
3	0 1 0 0 0	8	64	0.06	0.22	0
4	1 0 0 1 1	19	361	0.31	1.23	1
Sum			1170	1.00	4.00	4
Average			293	0.25	1.00	1
Max			576	0.49	1.97	2

A Simple Example

The Traveling Salesman Problem:

Find a tour of a given set of cities so that

- each city is visited only once
- the total distance traveled is minimized

Representation

Representation is an ordered list of city numbers known as an *order-based* GA.

1) London	3) Dunedin	5) Beijing	7) Tokyo
2) Venice	4) Singapore	6) Phoenix	8) Victoria

CityList1 (3 5 7 2 1 6 4 8)

CityList2 (2 5 7 6 8 1 3 4)

Crossover (Ordered)

Crossover combines inversion and recombination:

			*			*		
Parent1	(3	5	7	2	1	6	4	8)
Parent2	(2	5	7	6	8	1	3	4)
Child	(5	8	7	2	1	6	3	4)

This operator is called the *Order1* crossover.

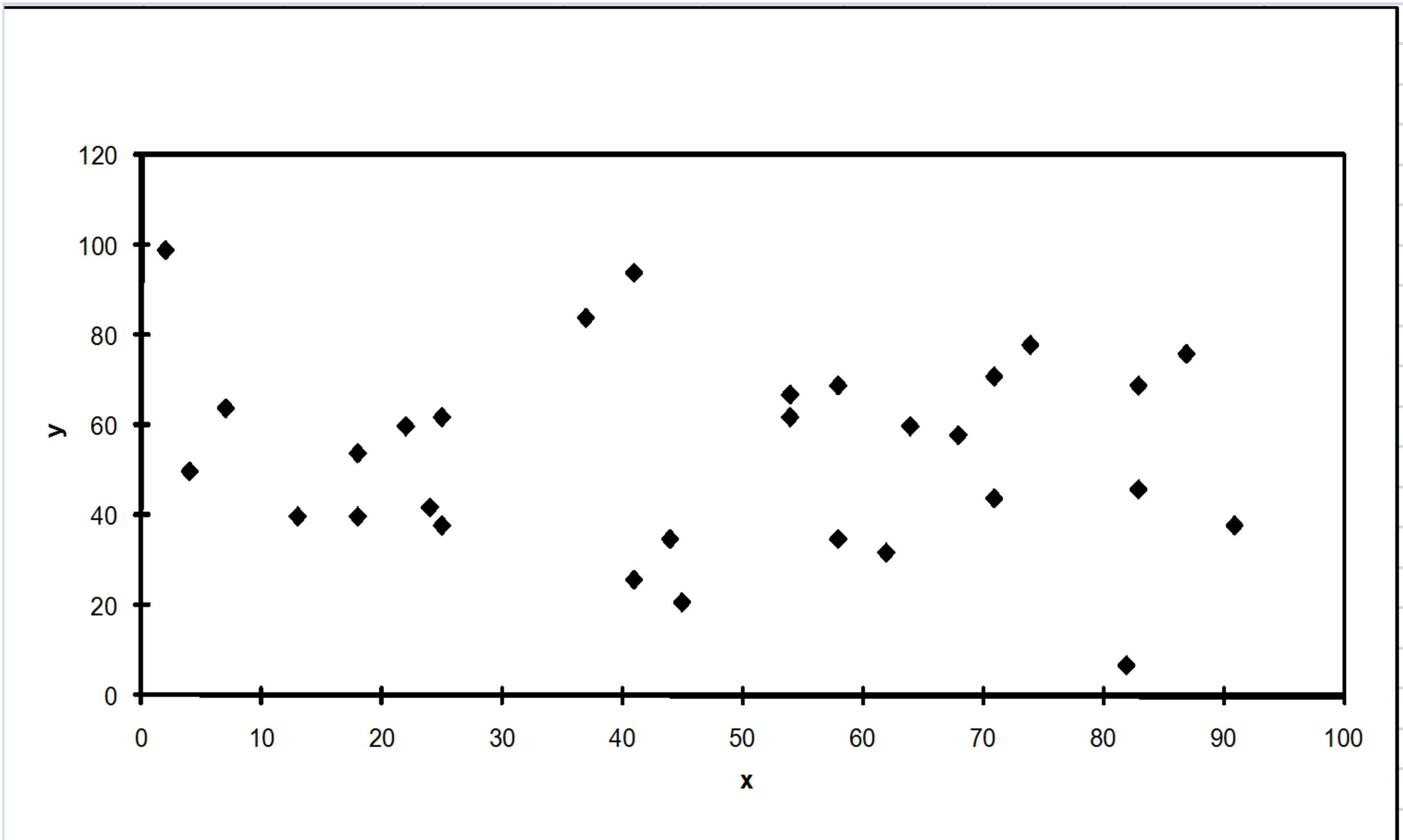
Mutation

Mutation involves reordering of the list:

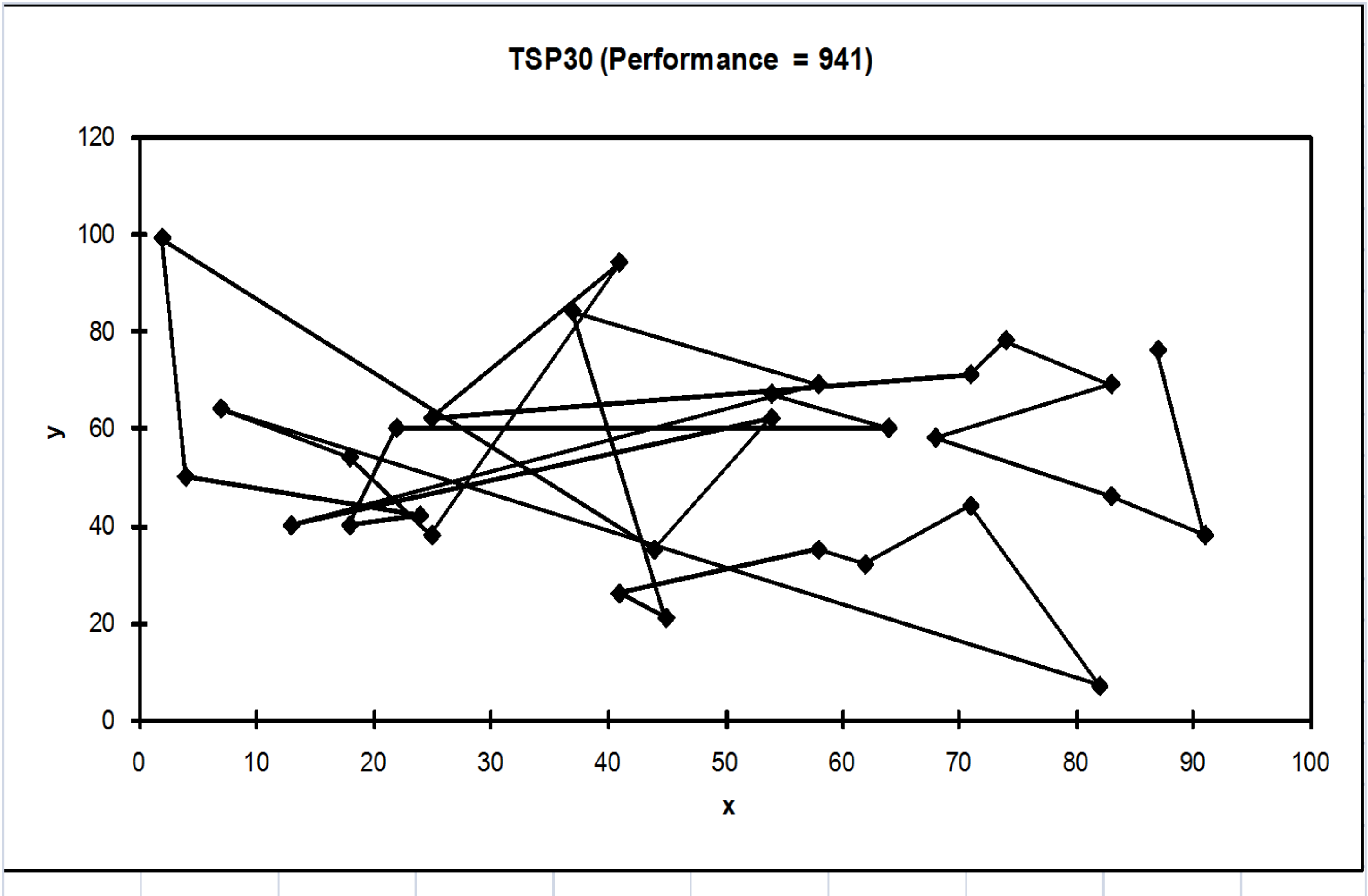
```
                 *           *
Before:    (5  8  7  2  1  6  3  4)

After:     (5  8  6  2  1  7  3  4)
```

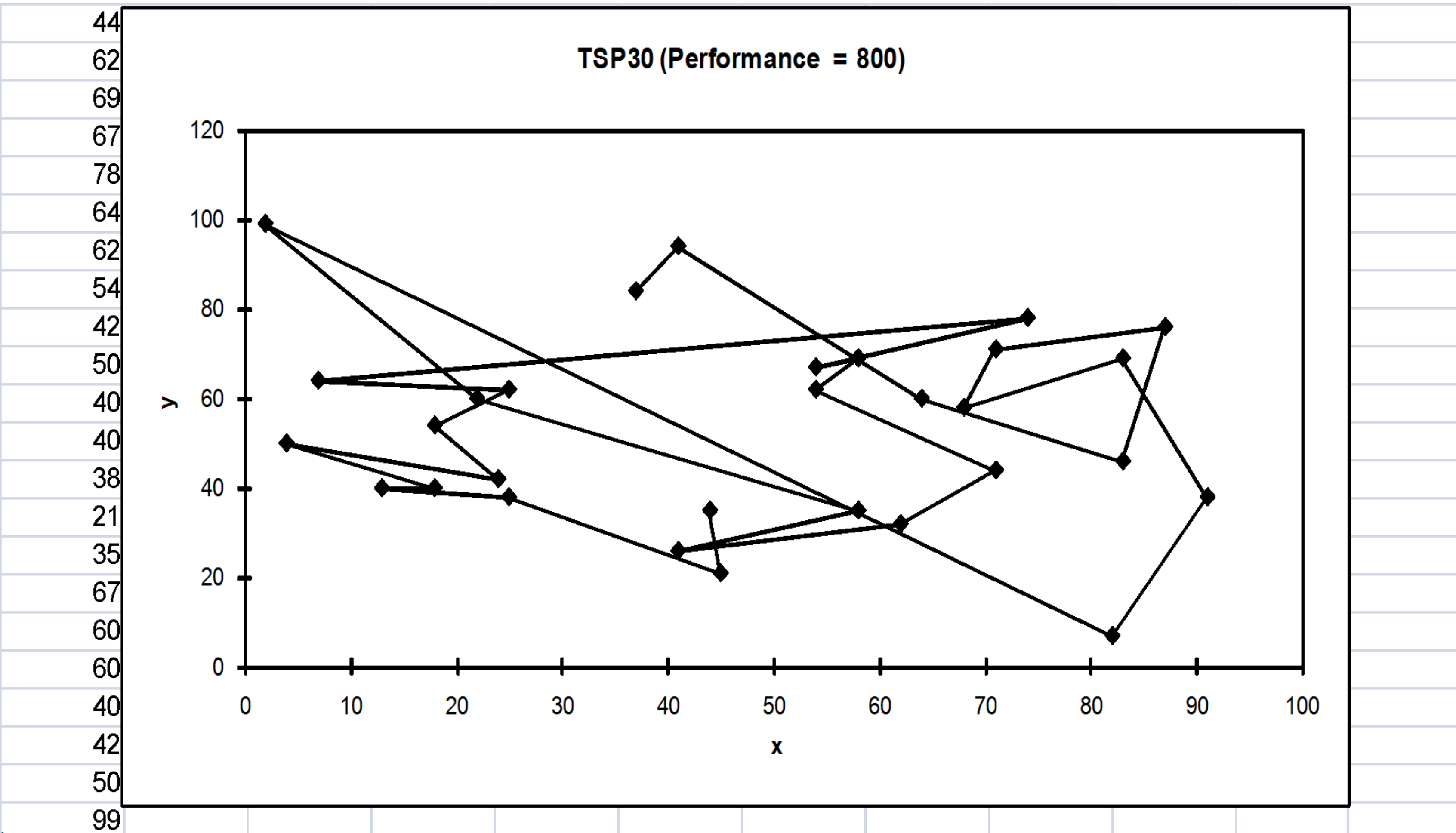
TSP Example: 30 Cities



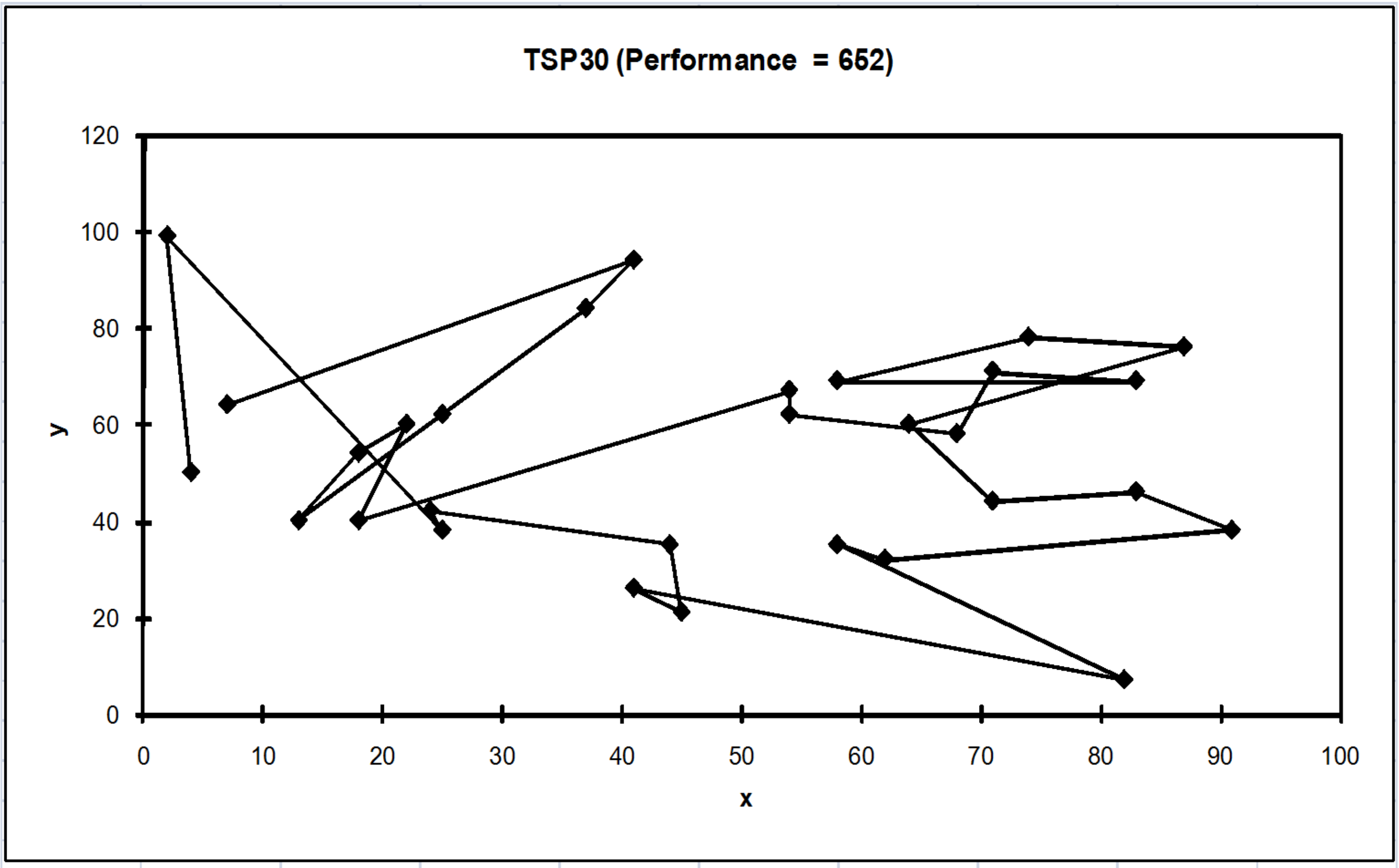
Solution ¡ (Distance = 941)



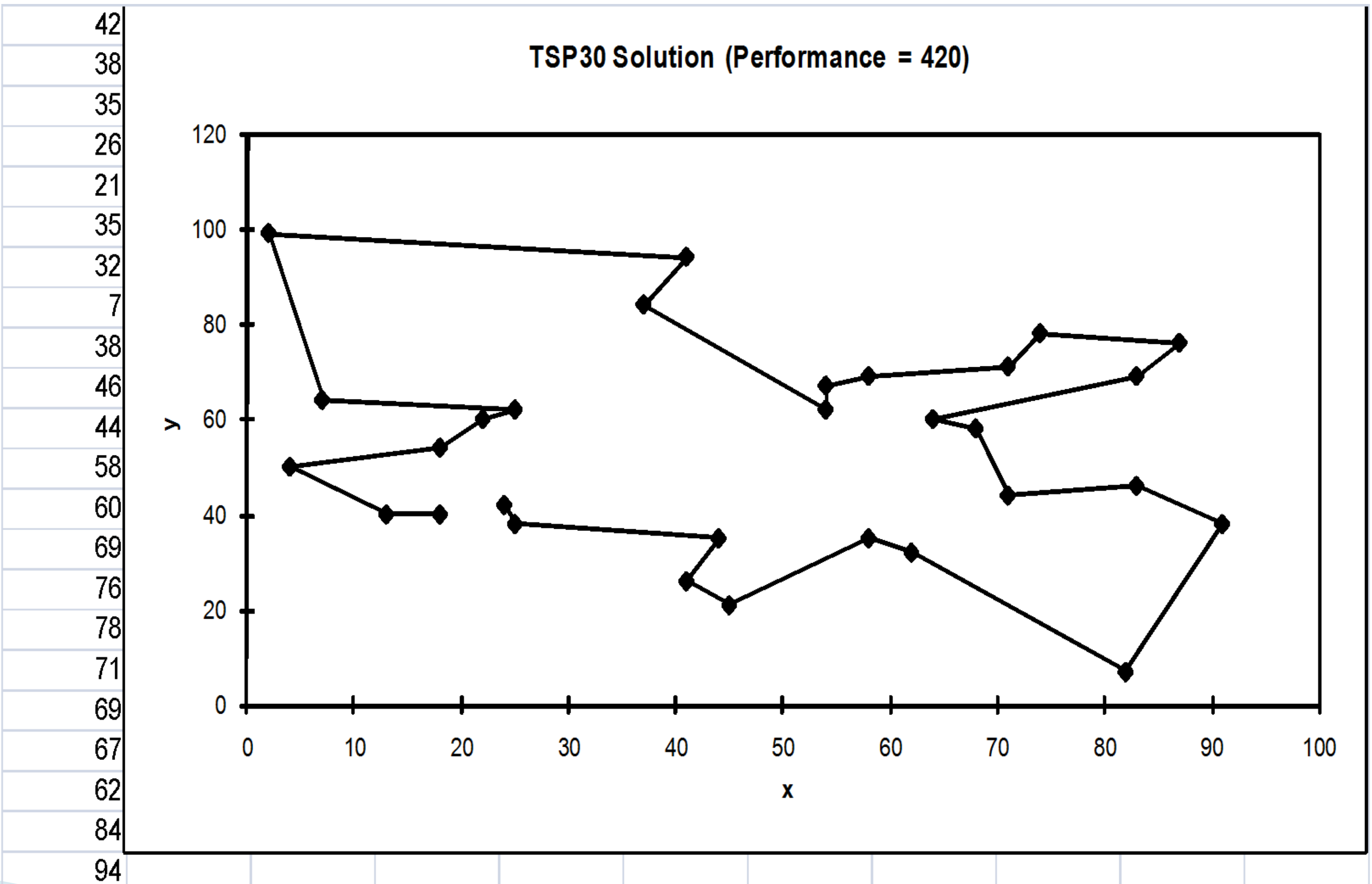
Solution $_j$(Distance = 800)



Solution $_k$(Distance = 652)



Best Solution (Distance = 420)



Issues for GA Practitioners

- Choosing basic implementation issues:
  - representation
  - population size, mutation rate, ...
  - selection, deletion policies
  - crossover, mutation operators
- Termination Criteria
- Performance, scalability
- Solution is only as good as the evaluation function (often hardest part)

Benefits of Genetic Algorithms

- Concept is easy to understand
- Modular, separate from application
- Supports multi-objective optimization
- Good for “noisy” environments
- Always an answer; answer gets better with time
- Inherently parallel; easily distributed

When to Use a GA

- Alternate solutions are too slow or overly complicated
- Need an exploratory tool to examine new approaches
- Problem is similar to one that has already been successfully solved by using a GA
- Want to hybridize with an existing solution
- Benefits of the GA technology meet key problem requirements

Some GA Application Types

Domain	Application Types
Control	gas pipeline, pole balancing, missile evasion, pursuit
Design	semiconductor layout, aircraft design, keyboard configuration, communication networks
Scheduling	manufacturing, facility scheduling, resource allocation
Robotics	trajectory planning
Machine Learning	designing neural networks, improving classification algorithms, classifier systems
Signal Processing	filter design
Game Playing	poker, checkers, prisoner’s dilemma
Combinatorial Optimization	set covering, travelling salesman, routing, bin packing, graph colouring and partitioning