CISC452/CMPE452/COGS 400 Optimization Methods

Simulated Annealing and Genetic Algorithms

Ch. 7 Textbook

Farhana Zulkernine

Optimization Methods

- In many interesting problems, optimal solutions cannot be guaranteed by reasonably fast algorithms
  - because the space and time requirements of the algorithm are bounded by polynomial functions of the problem size.
- Solution: Provide near-optimal solutions in much less time using some optimization method.

Optimization Algorithms

- Optimization algorithms conduct a *state space search*.
  - The *state space* is a set of states and depends on the representation chosen for the data.
  - A *state* represents possible assignment of values to each variable relevant to the problem.
- Goal is to find a “feasible solution” which is an acceptable solution.
- Problems often specify *constraints* that dictate that the **feasible solution space is much smaller than the entire search space**.

Using Energy Function

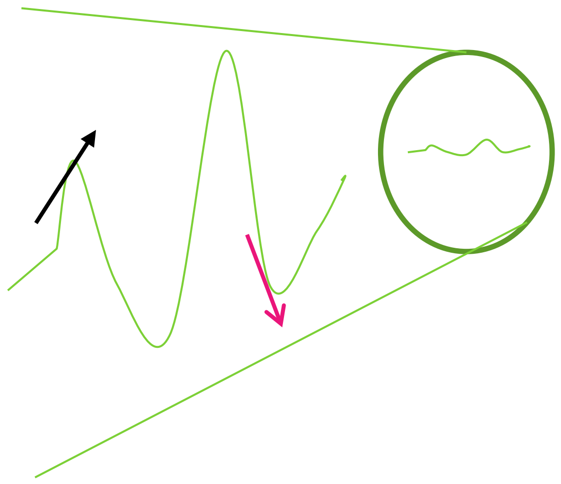
- Many optimization algorithms define an Energy function *E* that represents **cost and performance criteria to measure quality** of a candidate solution with **penalty functions** that implement constraints needed to ensure *feasibility*.
  - Infeasible solutions have higher energies than feasible solutions.
  - **The lower the energy of a *feasible solution* the better**.
  - General procedure: Move in the state space of the energy function to find a feasible solution that reduces the energy until an acceptable level is reached.

General Search Algorithm

```
While termination criteria is not met
        Generate a move in the state space of the
        energy function
        Decide if the move can be accepted
        If acceptable
                Update current state using the move
        End if
End while
```

Local and Global Optimum

- In hill climbing (or gradient descent), a move is made in the direction that is best but local maxima (or minima) can result (stopped at the wrong hill).



Local and Global Optimum (cont…)

- A global optimum is a candidate solution of the energy function whose **quality is better than or equal to the quality of every other candidate** solution.
- A local optimum is a candidate solution of the energy function whose quality is better than other solutions reachable in the state space with a single move.
  - One option is to define every local optimum as a feasible solution since most algorithms are prone to get stuck at a local optimum.

Non-neural Optimization Approaches

- We will discuss
  - **Simulated Annealing:** Allows worse moves to be accepted with an acceptance probability that decreases over time to escape local minima.
  - **Genetic Algorithms:** Instead of trying to improve a single candidate solution, it tries to generate new populations of candidate solutions based on prior results.

Why discuss non-neural approaches?

- Simulated annealing and genetic algorithms are non-neural approaches because there are no nodes and connections
- Features
  - Numerical methods
  - Rely on local information and computations rather than centralized processing
  - Often provide better solutions than hill climbing or gradient descent
  - Simulated annealing can be used in neural networks instead of gradient descent. E.g. Boltzmann machine

Simulated Annealing in Metallurgy

- Kirkpatrick et al (1983) suggests an approach based on the metallurgical process of annealing.
  - A metal or alloy is very slowly cooled and maintained at each temperature until some kind of an equilibrium is achieved.
  - Due to greater kinetic energy structures can change more easily at high temperatures even after reaching an optimal structure.
  - Best structures are stably obtained at very low temperatures, but rapidly cooling a metal can result in brittle structure.
  - Therefore, the cooling process must be slow and carefully controlled.

Simulated Annealing Algorithm

- Simulated annealing is a probabilistic algorithm for optimization.
  - An energy function is defined which is to be minimized by the algorithm.
  - A set of candidate moves are generated from which a candidate is selected
  - Acceptance of the candidate move depends on **current temperature and resulting energy change** defined as a probability which is
    - Always 1 if the move decreases energy
    - Otherwise the probability is defined by

$$\boxed{\Pr(\text{acceptance}) = \exp(E(\text{current state}) - E(\text{new state}))/T}$$

    - A move to higher energy is made if this probability is > a random number [0,1]

Simulated Annealing Algorithm

```
• Repeat
      Repeat
            Generate a move set from current state
            Select a candidate move
            If (Pr(acceptance) > random[0,1]) then
                  current-state = new-state
                  If (E(current-state) < E(best-so-far)) then
                        best-so-far = current-state
      Until Quasi-equilibrium is reached
      Decrease temperature by a cooling rate
• Until temperature is small enough
• Return the best-so-far equilibrium structure
```

Simulated Annealing Alg. (cont…)

- Move Generation: Generate a move that is reachable in one step from the current step (from 10011→10111 and NOT 10011→ 11111)
  - Selection probability does not depend on cost or temperature
- Move Acceptance:
  - Moves that decrease energy → always accepted (as in gradient descent)
  - Moves that increase energy → accepted with an acceptance probability that **depends on current temperature and energy change**.

Move Acceptance Probability

- The higher the temperature the greater the probability of accepting the move.
  - Analogous to greater temperature implies higher probability of change in structure
- The smaller the energy difference is between the new and the current state, the greater the probability of accepting the move.

Probability of Energy at Stable State

- After running long enough at a temperature, the inner loop stabilizes and reaches Boltzmann distribution of states which is irrespective of the initial state,

  Probability that the current state has energy e_l

  $$\Pr(e_l) = \frac{\exp(-e_l / T)}{\sum_j \exp(-e_j / T)}$$

  assuming

  a) every state is reachable from every other state in a number of moves and

  b) at high temperature, all states are equally likely but at low temperatures, there is more probability of reaching states with low energies (our goal).

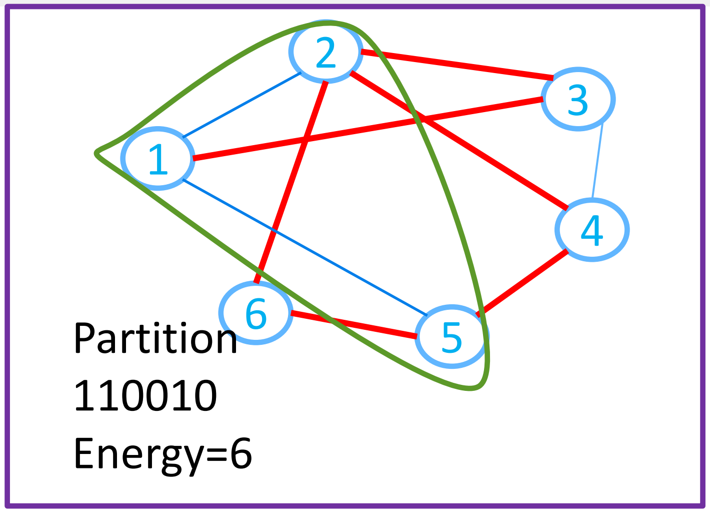
Probability of Energy (cont…)

- Testing whether a state is in quasi-equilibrium requires much computation.
- A practical alternative is to let the system run longer at lower temperatures than at higher temperatures (the inner loop).
- The outer loop steadily lowers the temperature allowing each the system to reach quasi-equilibrium at each temperature.
- Ideally when Boltzmann distribution is reached at temperature 0, global optimum is reached with probability 1.
- Cooling rate = Rate of lowering temperature

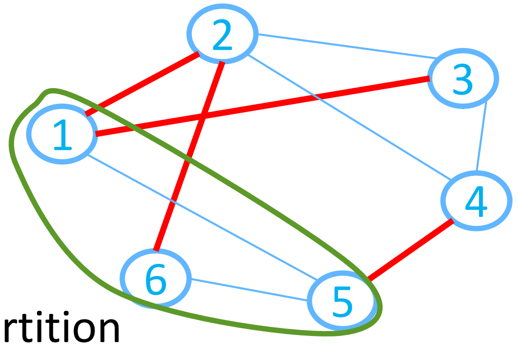
Example – Graph Partitioning

- In the problem of graph partitioning, the goal is to divide the nodes of a given graph into two equal-sized groups in such a way as to minimize the total cost (number of edges between the two-groups). The problem is known to be NP-complete; no deterministic polynomial-time algorithm is known to exist to solve the problem. We illustrate the graph partitioning problem with simulated annealing.

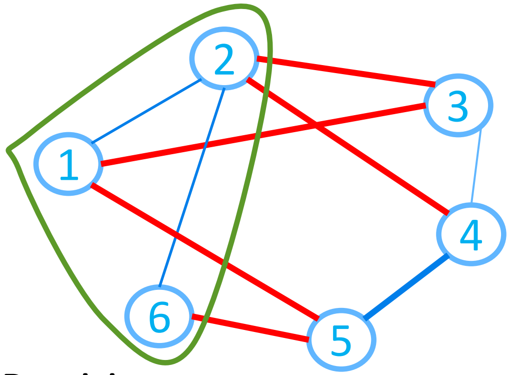
Example – States and Energy



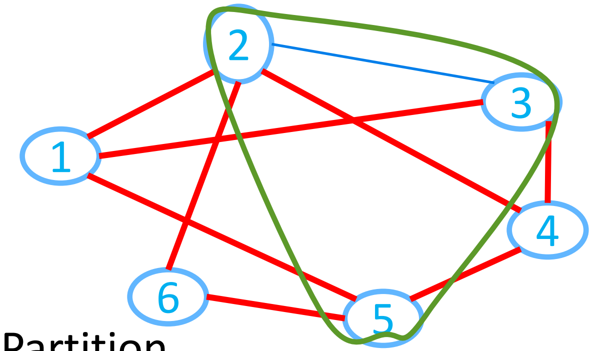
Partition
110010
Energy=6



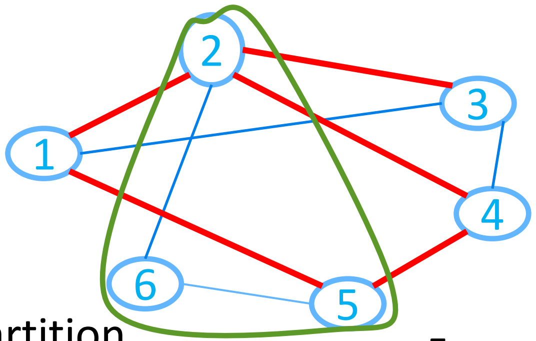
Partition
100011
Energy=4



Partition
110001
Energy=5



Partition
011010
Energy=8



Partition
010011
Energy=5

Energy = No. of external connections
Partition = Nodes in a partition have same value

Example - Moves

- Moves should be easy to compute involving small changes to the current state
  - Example: simultaneous *change of two digits*, $1 \longrightarrow 0$ and $0 \longrightarrow 1$
  - Move set for 110010 (6): 100011 (4), 110001 (5), 010011 (5), 011010 (8)
  - Energy for each state: 4, 5, 5, 8
  - Move to lower energy 100011 (4) will always be accepted
  - Move to higher energy state 011010 (8) will be accepted with probability $e^{(6-8)/T} = e^{-2/T}$ => dependent on temperature

Example - Moves

- For T=50, acceptance prob. $e^{-2/T}$ => $e^{-2/50}$ = 0.96
- For T=10, acceptance prob. $e^{-2/10}$ = 0.82
- For T=1, acceptance prob. e^{-2} => 0.13
- Therefore, the higher the temperature the higher the probability of acceptance of a move to a higher energy state, and
- The probability of acceptance of a move to a lower energy state = 1

Summary

- Other versions of simulated annealing exists such as
  - Adaptive simulated annealing (Ingber 1993)
  - Threshold annealing (Dueck and Scheuer 1990) – use a threshold (e.g. 0.5) instead of a random number
- Instead of using an extremely slow cooling rate
  - Use faster cooling rate with gradient descent
    - ❑ Execute several iterations of the simulated annealing (SA) with fast cooling rate,
    - ❑ Apply gradient descent (GD) at the end of each run
    - ❑ Select the best result obtained in these iterations
  - Perform SA many times steadily decelerating the cooling schedule and applying GD in the end of every process as long as solution improves.
- Applications in Hopfield network, Boltzmann machine

Natural Selection in Evolution

- Evolutionary Computation was proposed based on Darwin’s (1859) theory of natural selection. It requires three elements:
  - *Variation* in traits within a species.
  - The passing of genetic material from one generation to the next, called *inheritance*.
  - *Selection*, which is a change in the environment that favors one trait over another - “Survival of the Fittest”

Aspects of Evolutionary Processes

- The stronger genes from the parents live on within the offspring because of greater ‘fitness’ and they define characteristics in the offspring.
- Weaker genes have less probability of survival and reproduction.
- So the basic procedures and measures are:
  - Fitness of individuals which influences selection
  - Selection of individuals as parents for reproduction
  - Reproduction transfers features from parents to offspring in the next generation
  - Not every reproduction improves fitness but over multiple generations in a stable environment, the process ensures that the best offspring achieve higher fitness and a better population

Evolutionary Computations

- Motivated by the theories of evolution and genetics.
- **Individuals** in the **population** are (potential) solutions.
- The effectiveness of each solution is the **fitness** measure which is used to **select** candidate solutions for **reproduction**.
- New solutions are constructed from existing solutions, called ‘Reproduction’ which ensures:
    a) Continuation and propagation of successful aspects of the solutions,
    b) Diversity, so that alternative directions for solutions will be considered.
- Through the generations, the fitness of the best individual improves.

Evolutionary Algorithms

- Maintains a *population* of candidate solutions whose *fitness* increases in subsequent *generations*.
- The fact that the new candidate solutions (offspring) are created (*reproduction*) from multiple existing members of the population (parents), increases the quality of the new solution.
- *Selection* mechanism helps to weed out poor quality members from the population.

Algorithm

- Algorithm: Evolutionary Computation

```
Initialize the population P(0)
While terminating criteria is not satisfied
    Determine potential parents from current population P(t)
    Apply evolutionary operators to reproduce offspring Ω(t)
    Obtain P(t+1) by selection from (P(t) U Ω(t))
End while
Return best candidate solution from current population
```

Evolutionary Algorithms (cont…)

- Algorithms are categorized into three broad classes:
  - Evolutionary Programming (EP) – Focuses on representation and reproduction
  - Evolutionary Strategies (ES) – Focuses on adaptive strategies for selection.
  - Genetic Algorithms (GA) – Specific approach to representation (fixed length binary string) and reproduction (‘crossover’ to combine segments of strings from parents)

Genetic Algorithms (GA)

- Modeled on natural evolutionary mechanisms.
- Individuals are specified as strings (as DNA).
- Terminology has been generally borrowed from the field of Genetics. Each location in the string is a *gene*, and the string itself is a *chromosome*, and value that the gene can take on is an *allele* (alternative forms of the same gene).

Example

- Consider the task of maximizing the function $f(x) = -x^2+12x+300$ where $x \in \{0, \ldots, 31\}$. Potential solutions are bit-strings of length 5 (can represent values from 1~31).
- To carry out the Genetic Algorithm, we decide on N_p , the *size of the population*, and then N_g , the *number of generations* that we will run.
- We generate random solutions, and evaluate their fitness.
- We normalize fitness so that it can be used as a probability.

Example (cont…)

X	String	Fitness	Probability
22	10110	80	0.08
17	10001	215	0.22
24	11000	12	0.01
26	11010	320	0.32
3	00011	335	0.34

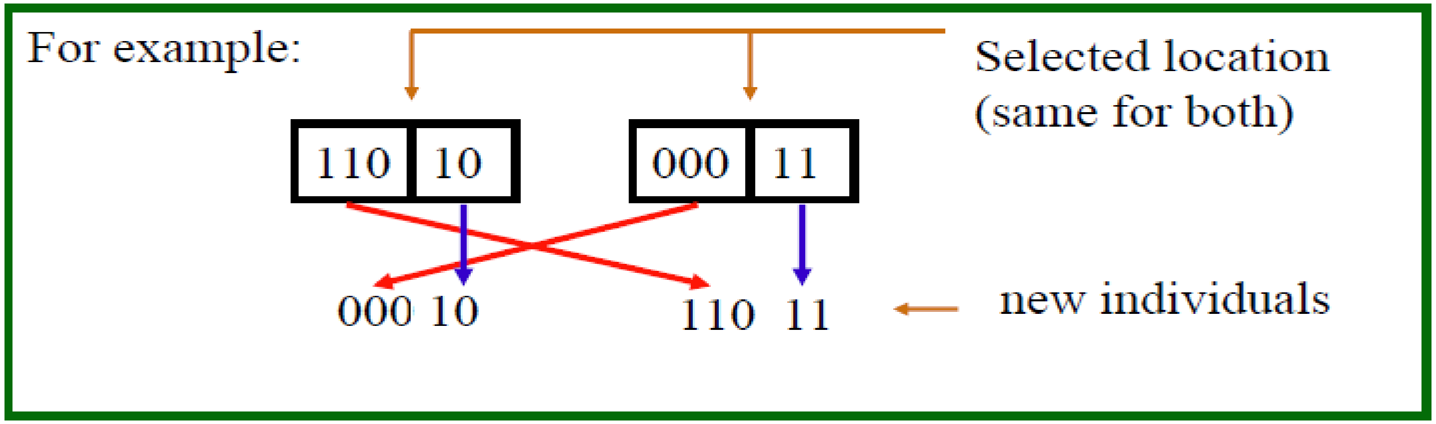
The fitness function is defined based on the type of the problem.

Genetic Operators

- For each new generation, individuals are chosen based on their fitness, to enter into the reproduction step.
- At the reproduction step, a genetic operator is applied to *one or more parent candidate* to produce the next generation of population.
- Many operators have been proposed in the literature – 3 are used more often.

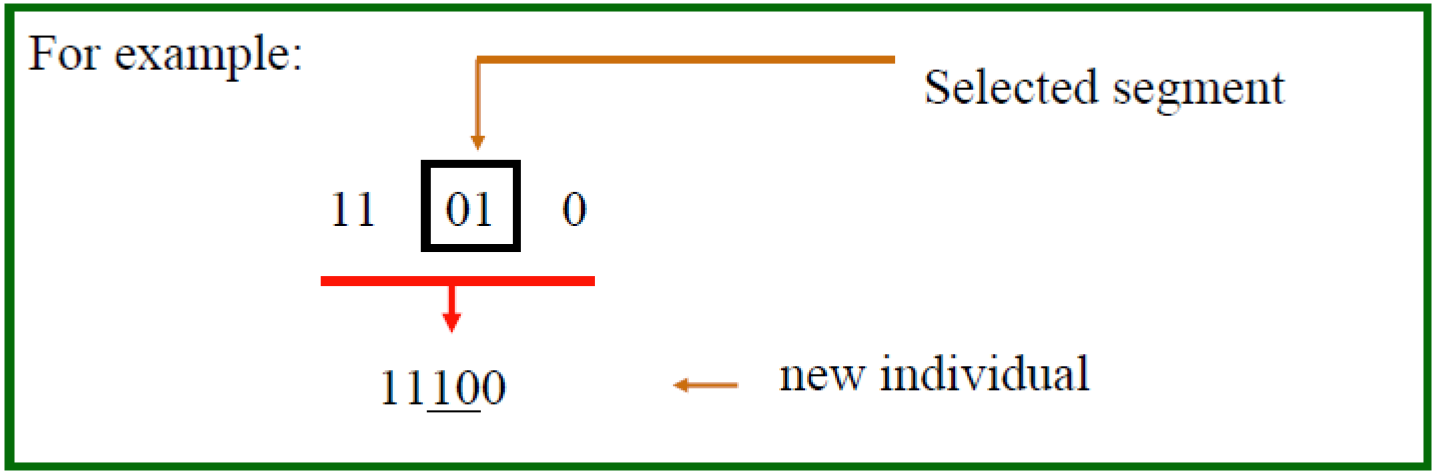
1. Crossover

- 2 individuals are selected.
- A point within the chromosome is selected at random (called 1 point crossover).
- Segments are switched to produce new individuals.



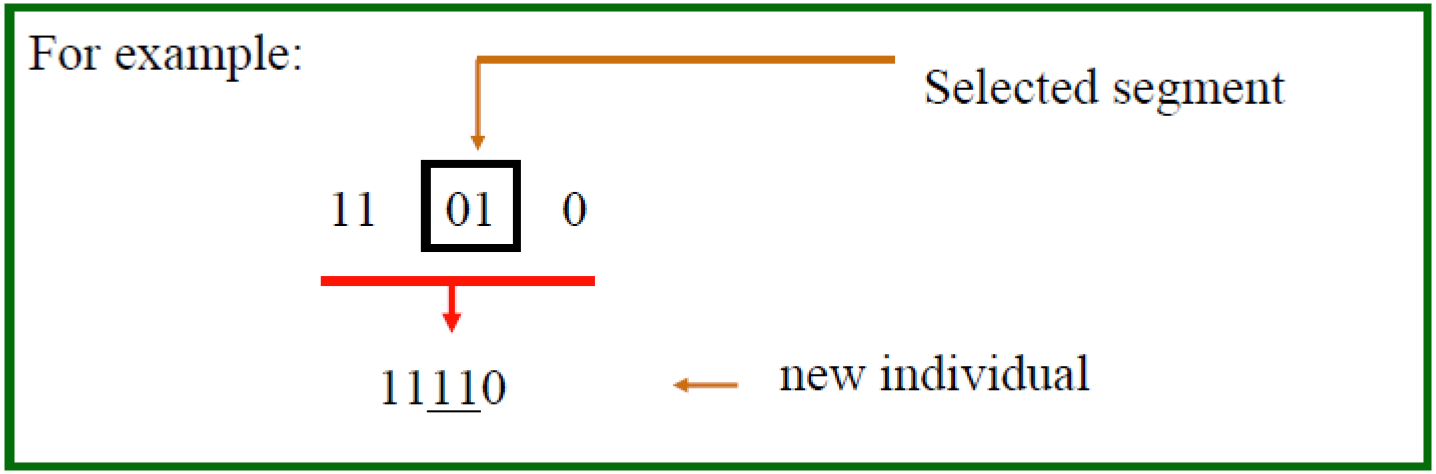
2. Inversion

- An individual is selected (based on fitness)
- A segment of the chromosome is selected at random
- That segment is reversed



3. Mutation

- An individual is selected (based on fitness)
- A segment of the chromosome is selected at random
- Randomly replace the segment



GA (cont…)

- **Representation and choosing parents:** Select parents for reproduction based on some probability – fitness
- **Reproduction:** Create new individuals.
- **Selection:** After reproduction, new individuals replace randomly selected members of the population.
- On average, the fitness will improve - and it will do so more quickly than could be obtained by systematically searching through all the possible solutions.
- Note: Not all genetic operators apply well to all problems.
  - E.g. Meet other constraints of the problem such as maintain equal number of 1s and 0s for which inverse will work but not mutation.

Selection and Fitness

- Why not always replace the least fit individuals rather than selecting the ones to be replaced at random?
  - It is important to retain *diversity*. Individuals that appear to have poor fitness may possess some aspect that will be important to fine tune the solution to the best possible.
- Is fitness for survival the same as fitness for reproduction?
  - Typically ‘yes’ – fitness function depends on the problem

Why not only use the best?

- In our example of maximizing $f(x) = -x^2+12x+300$ where $x \epsilon \{0,..,31\}$, may be 6 (110) is the optimum solution.
- Suppose we were using only crossover, and all the most fit population had the rightmost bit on (all odd numbers). Then we could never reach the best solution of x=6.

A Genetic Algorithm Example: Satisfaction of Conjunctive Normal Form

- CNF is a form of logic statement such as
- $(\neg a \lor c) \land (\neg a \lor c \lor \neg e) \land (\neg e \lor f) \land (\neg b \lor c \lor d \lor \neg e) \land (a \lor \neg b \lor c)$
- We wish to find logic values of a,b,c,d,e,f that satisfy the expression
  1. Representation: use bitstrings 101010 (position 1=a, 2=b, etc)
  2. Genetic operators: any string is a possible solution. Probably just use cross-over and mutation.
  3. Fitness: *The more clauses are satisfied the better the solution is*. There are 5 clauses, so use 0 to 5 indicating how many clauses are satisfied (perhaps weighted by the length of each clause)
  4. Selection strategy: Choose the best fit plus random members from the parent generation.

A Genetic Algorithm Example: Travelling Salesman

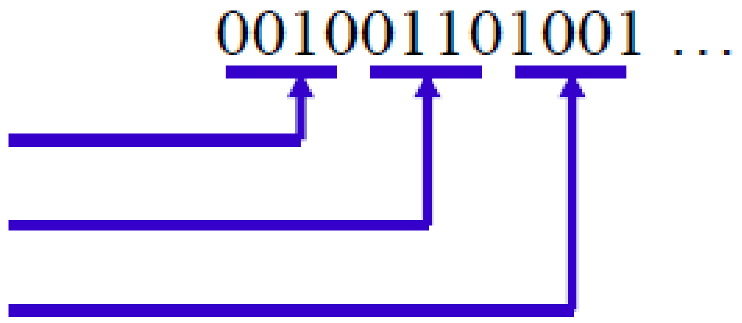
1. **Representation:** We need to represent position in the tour for each city. Since there are more than just two alternatives for each position in the tour, we might think to use several bits for each position, such as:

   001001101001 …

   City A in position 2

   City B in position 6

   City C in position 9



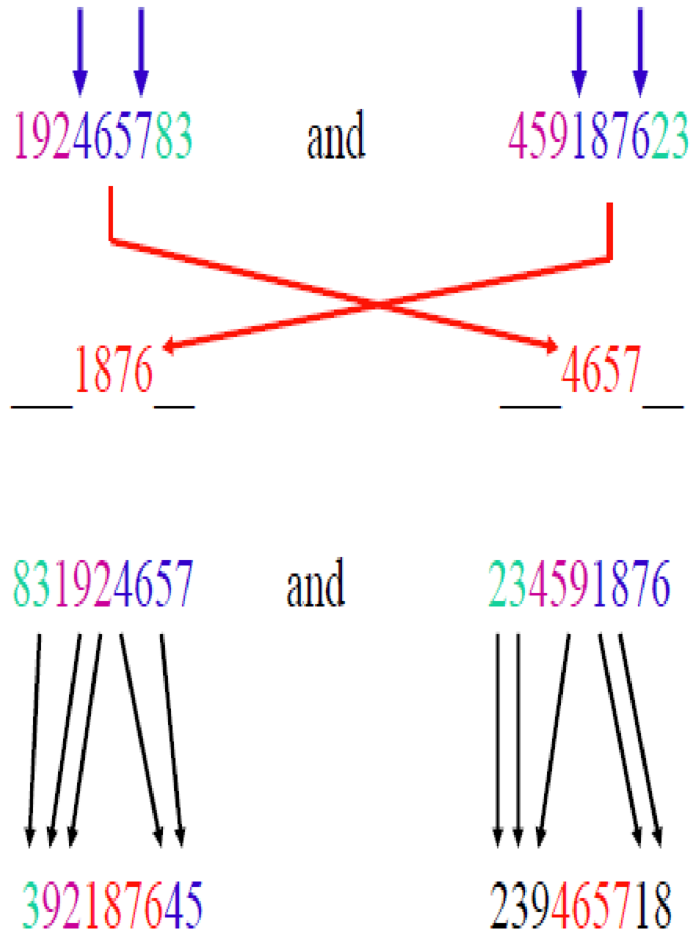
But genetic operators working on these bit-strings would create many problems: positions beyond the tour, more than one city with the same position, etc.

Example (cont…)

- So, for our genetic code we use unique digits (and if necessary characters) to represent the cities, or to represent positions in the tour. Such as 192465783 to indicate city A is first, city B is 9th, etc.

2. **Genetic operators**:
   - *Mutation* could produce invalid sequences, so it should be modified to **make a random exchange** of elements, rather than randomly replacing elements.
   - *Inversion* works fine.
   - *Cross-over* also creates invalid sequences, and needs to be modified.

Example (cont…)



- Here is an example of how crossover could be modified:
- From each of a pair of reproducing individuals, select 2 random locations in the strings (same 2 for each individual) – **2 point crossover**.
- Cross over the selected portions of each individual
- Reconstruct the originals to start after the selected portion.
- Fill in the blanks left to right using any that are not already in the string from the cross-over.

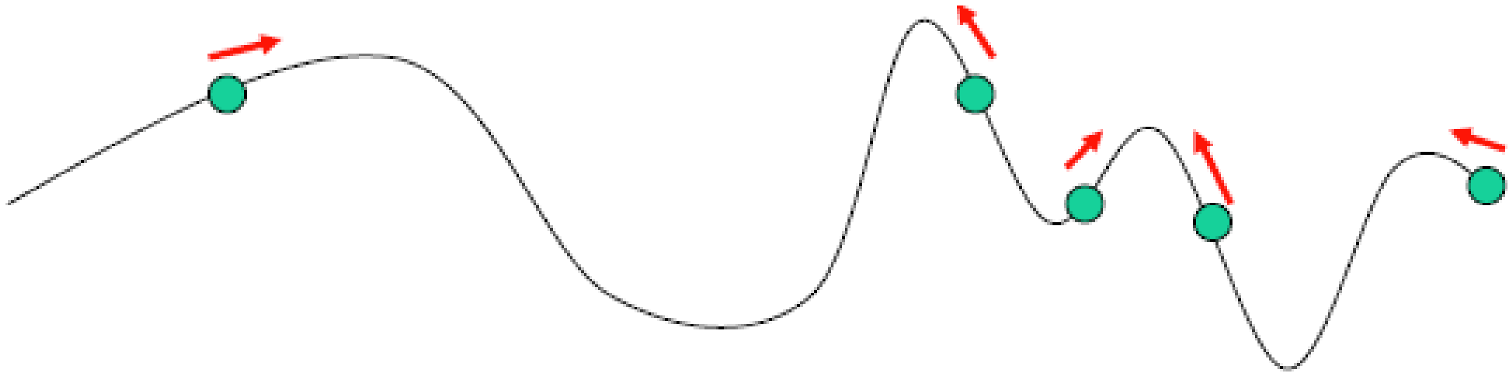
Discussion

3. **Fitness:** in this case is obviously 1/(length of tour)

- In the examples, we have seen that all three aspects
  1. Representation
  2. Genetic operators
  3. Fitness
- May have to be "tuned" to the particular problem in order to use a Genetic Algorithm ensuring that
  - Only viable offspring are generated
  - Meaningful aspects of the solutions are retained

Discussion (cont…)

- Does the Genetic Algorithm work?
  - Yes. Practical experience indicates that it is far better than blind search.
- We can view GA as a parallel search in which different individuals move to locally optimal locations through the use of operators that maintain solution structure and propagate it through generations.



Discussion (cont…)

- But operators such as mutation allow that sometimes solutions move away from locally optimal solutions, introducing a randomness that is similar to the energy increases in simulated annealing or a Boltzmann machine.

Schema in GA

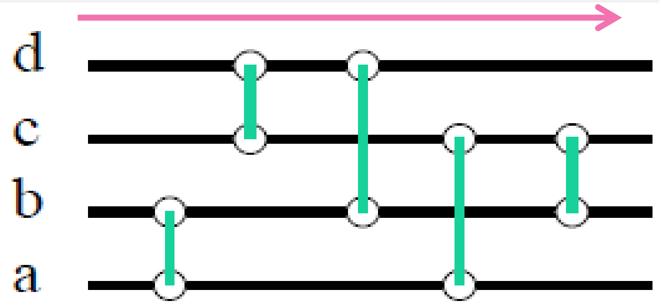
- Typical arguments about the optimality of GA employ the idea of a schema (or the plural schemata).
- A schema is a pattern that may be found in a solution string.
- Example: #1##0# represents all solutions that have a 1 and a 0 in the designated position, regardless of what else is present. Such as: 0**1**00**0**0 and 1**1**10**0**1 , etc.
- A string (solution) with better than average fitness propagates all its schemata in parallel.
- Schemata which are found in fit-strings propagate exponentially into the population.
- Exponential propagation leads to optimality.

Fitness Measures and Parasites

- Often a fitness measure can be absolutely determined - as in optimizing a function value, or in the graph bi-partitioning problem.
- In other cases, absolute determination of fitness would require an unreasonable number of computations.
- Example: If each solution is a backpropagation network, we would have to train it with a variety of starting weight values, trained and tested with the entire set of samples, and still we would not know for sure how well it could solve the task.

GA to Develop a Sorting Network

- A *Sorting Network* is a specification of exchanges to consider in order to sort a specific number of inputs (n).
- Example: Here is a depiction of a sorting network for n=4 (4 numbers to sort). This involves 5 possible exchanges



	a	b	c	d
	1	3	2	4
ab	3	1	2	4
cd	3	1	4	2
bd	3	2	4	1
ac	4	2	3	1
bc	4	3	2	1

GA for Sorting Network (cont…)

- We can use a GA to find the best possible sorting network for a given value of n. The genetic code is just the sequence of exchange considerations.

  <u>ab</u> <u>cd</u> <u>bd</u> <u>ac</u> <u>bc</u> works very well

  <u>ad</u> <u>ac</u> <u>bc</u> <u>dc</u> <u>ac</u> does not work well

  - 4321→2341→2431→2413 →1423

a	B	C	D
1	3	2	4

- We could try for a solution with 5 exchanges, and if we find one that works, we would go on to try 4 exchanges, etc., until we reach a point at which we can't get a working solution.

GA for Sorting Network (cont…)

- The established minimum number of exchanges required in a sorting algorithm is $nlog_2n$, but only for an arbitrary value of *n*.
- For specific values of *n*, sorting networks exist that use less than $nlog_2n$ exchanges, For *n*=16, sorting networks exist with only 60 exchanges where $16log_216$ = 64.
- A sorting network that works well for *n* samples in {0,1}, will also work for *n* integer samples.

GA for Sorting Network (cont…)

- In developing the GA, it is unreasonable (for large *n*) to test out each individual solution with all 2^n possible orderings that must be sorted (until the very end to see if we really have a solution).
- We can take a smaller test set, drawn from the entire set of 2^n, but this may lead to networks that are *perfect on the test set*, *but still fails on some examples* that are not in the test set.

GA for Sorting Network (cont…)

- So, we treat the test sets as initial populations.
- Each test set in the population has a fitness which is the number of errors that it causes in the sorting solutions.
- Based on its own fitness, the population of tests sets evolve using the GA.
  - As better test sets emerge in the population, the sorting networks are being pushed to become even better.
  - This type of co-evolving system is called a host-parasite co-evolution system or more simply a parasite system.