Local Search

林伯慎 Bor-shen Lin
bslin@cs.ntust.edu.tw
http:www.cs.ntust.edu.tw/~bslin

Agenda

- Combinatorial Optimization Problems & Constraint Satisfaction Problems (CSPs)
- Local Search Methods
  - Ant colony optimization
  - Particle swarm optimization
  - Simulated annealing
  - Genetic algorithm

Constraint Satisfaction Problems

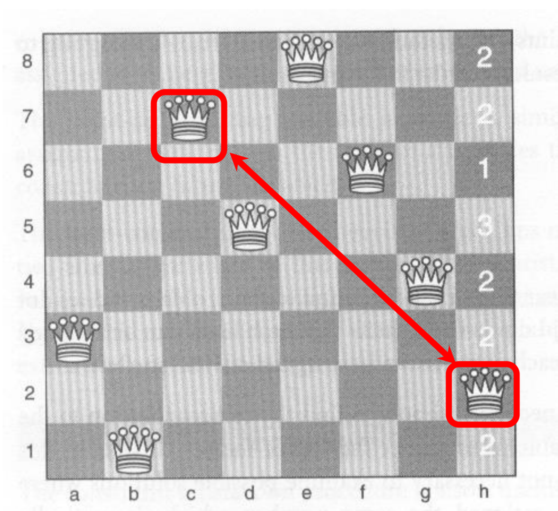
- Combinatorial optimization problems involve assigning values to a number of variables.
- A constraint satisfaction problem (CSP) is a combinatorial optimization problem with a set of constraints.
- CSPs can be modeled as search problems, and search methods and heuristics can be used to solve them.
- Example: 8-queens problem
  - A queen occupies the squares of the same row, column and diagonal, on which no other queens can be put.

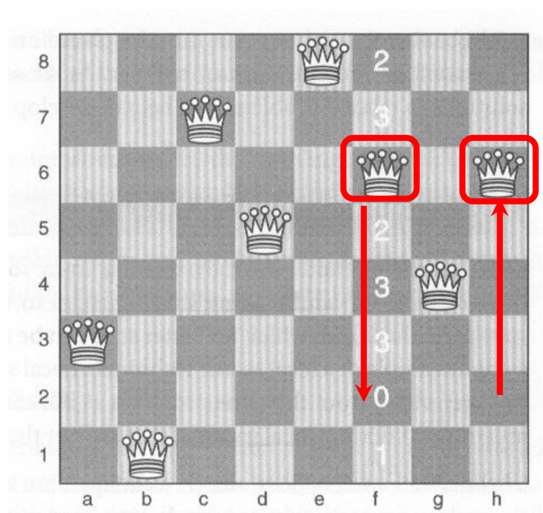
8-Queens Problem

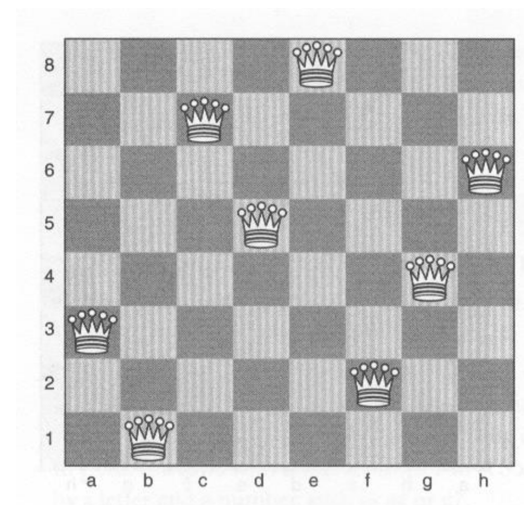
- A 8-ply deep search tree (DFS)
  - Size of search space without constraint: $64 \cdot 63 \cdot \ldots \cdot 57$
  - The branching factor will be smaller if the constraints are applied
- Place one queen for each row (or column)
  - Size of search space: $8 \cdot 7 \cdot \ldots \cdot 1$
  - The choices are even fewer after further applying the constraints
  - Row 5 has 2 choices only
- The search tree is large, but can be significantly reduced by applying the constraints appropriately

HEURISTIC REPAIR

- Generate a possible solution (randomly or with heuristic) and then make changes that reduce the distance of the state from the goal
  - e.g. board state randomly assigned as [3, 1, 7, 5, 8, 6, 4, 2]
- Resolve conflict: min-conflict heuristic for 8-queens







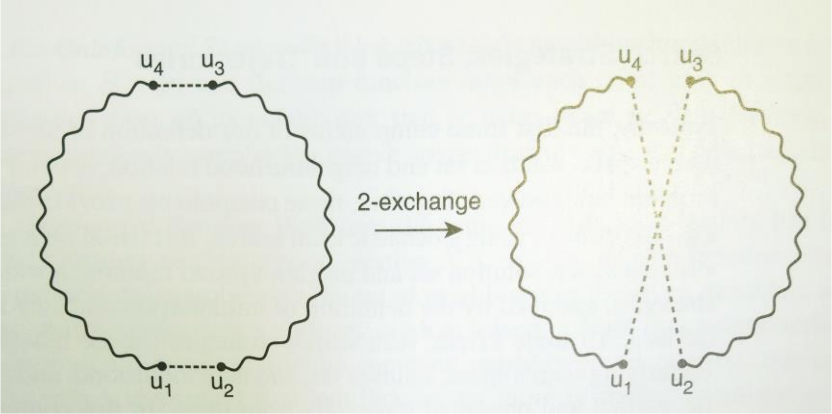
8-Queens Problem (cont'd)

- Most Constrained Variables
  - Considering the problem to be one of assigning a value to eight variables, a through h.
  - Working with the variable that has the least possible number of valid choices
  - Example: a = 1, b = 3, c =5 , and the numbers of choices: d(3), e(3), f(1), g(3), h(3)

    → place a queen in column f (choose variable f) !
- Least Constraining Value
  - Assigning a value to a variable that leaves the greatest number of choices for other variables

Exchanging Heuristics

- Swap two or more variables at each step until a solution is found.
  - A *k-exchange* involves swapping the values of k variables.
- k-exchange for TSP
  - Remove two edges and substitute them for two other edges
  - Higher k leads to higher time complexity
  - k=3 produces good enough results and can be implemented efficiently.



Cited from *stochastic local search Foundations and applications,* H. H. Hoos and T. Stutzle

Combinatorial Optimization Problems

- Finding the best possible set of values for a group of variables
- It is useful to find ways to restrict the number of choices that are available for each queen to avoid the problem of combinatorial explosion
- Example
  - Allocating teachers to classroom
  - Scheduling machines and workers
  - Selecting the best routes for buses
  - Traveling salesman problem

Problems in DFS/BFS/A*

- Large memory required
  - Open list for recording and sorting all frontier nodes
  - Close list for tracking all visited nodes
- Computation cost is high
  - Sorting the open list according to evaluation function
  - Comparing nodes to avoid duplicated state in close list
- Optimality guaranteed for A*
  - May not be achievable in real time for the search tree with large branching factor
- Stochastic local search
  - Fewer memory required
  - Local optimum

Local Search

- Start from a random state, and make small changes until a goal state is achieved
- The methods used by local search techniques are known as meta-heuristics.
  - Ant colony optimization
  - Particle swarm optimization
  - Simulated annealing
  - Genetic algorithm
- Local optimization: attempt to optimize a set of values, but will often find local optimum instead of a global optimum.

Iterated Local Search

- Attempt to overcome the problem of local optima by running the optimization procedure repeatedly, from different initial states.
- If used with *sufficient* iterations, this kind of method will almost always find a global maximum.

Ant Colony Optimization (ACO)

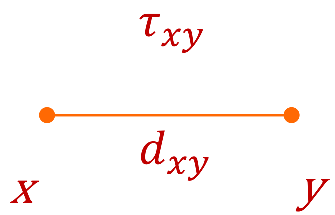
- Behavior of ants
  - Foraging ants leave a trail of pheromones that can lead other ants to find the food they have found
  - The trail of pheromones is renewed regularly
  - If better route is found, the pheromones of old route will gradually fade and new route become most popular choice
- Cope extremely well with *changes in the environment*, such as blockages and new routes
- Adapt to change in real time
  - Traveling salesperson problem
  - Network routing problem

Ant Colony Optimization

- $p_{xy} = \frac{{\tau_{xy}}^{\alpha}{\eta_{xy}}^{\beta}}{\sum_z {\tau_{xz}}^{\alpha}{\eta_{xz}}^{\beta}}$
  - probability for state transition from state x to state y in state space
  - τ_{xy} : amount of pheromone from x to y
  - $\eta_{xy} = \frac{1}{d_{xy}}$, d_{xy} is distance from x to y
- Ant: computational agent
  - moves to candidate states in state space
  - selects its next state according to p_{xy}
- Pheromone update
  - ${\tau_{xy}}' = (1-\rho)\tau_{xy} + \sum_k \Delta\tau_{xy}^{k}$

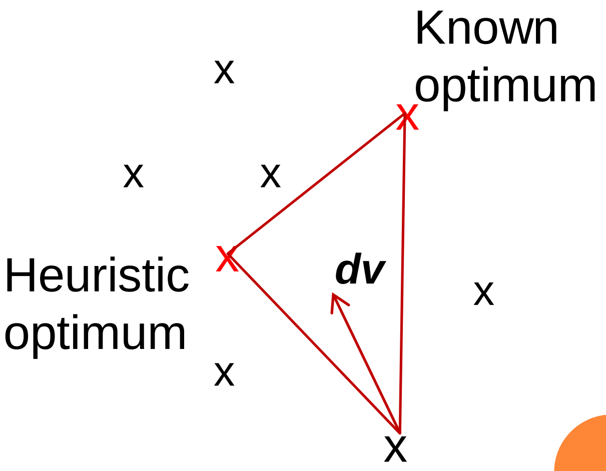
    ρ : evaporation rate of pheromone
  - $\Delta\tau_{xy}^{k} = \frac{\mathrm{Q}}{\mathrm{L}_k}$ if ant k passes through x-y edge

    L_k the tour length of the k-th ant



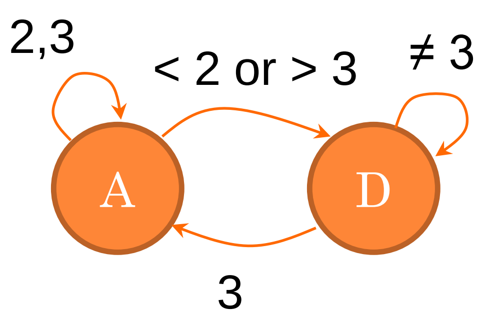
Particle Swarm Optimization (PSO)

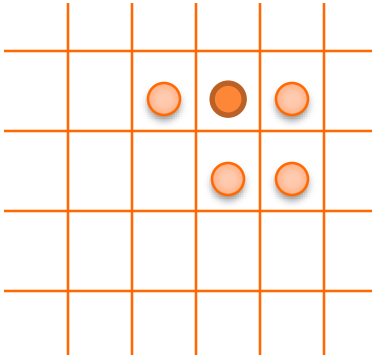
- Observe a school of fishes or a fleet of birds
  - If a fish found the food, there might be food around it
  - Every fish moves towards the food randomly
  - Humans are usually influenced by celebrities



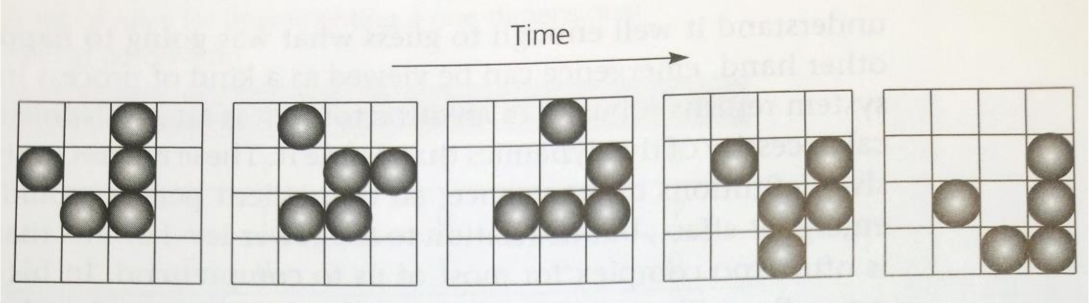
Game of Life

- Cellular automata (by J. H. Conway 1970)
- Rules
  - Every cell is either *Alive* or *Dead* and has 8 neighbors
  - Alive cell
    - change to Dead if too lonely (<2) or too crowded (>3)
  - Dead cell
    - Change to Alive if having exactly three neighbors





SIMULATION BY GAME OF LIFE



- Cited from *swarm intelligence*, J.K and R.C. E

- A simple model of social behavior
  - each particle changes its individual state according to a simple rule
  - A group of particles has a specific pattern
- Similar to social behaviors (like stores)
- Evolution of the social pattern

Axelrod's Cultural Model

- Individual
  - Represented a string of symbols (features)
- Similarity criteria for interaction
  - Probability of interaction as a function of similarity
  - agents who are *similar to each other* are likely to interact

    e.g. 42237 vs. 99217 : with probability 0.4

    42237 vs. 98765 : with probability 0.0 (no interaction)
  - Interaction occurs to change nonmatching feature according to stochastic simulation

    e.g. 42237 → 42217
  - Cognitions, attitudes, and other arrays of psychological phenomena are optimized by interaction among individuals
- What are the simulation results?

27217	74924	31157	53671	22660	37316	07959	57666	33206	92725
66219	08226	26707	45600	48767	39481	62784	89859	27792	35492
37262	66163	89178	60968	91098	19937	62103	07562	03500	13864
87746	66209	94122	72784	03593	16647	19776	87819	22160	48185
16880	09713	76057	30843	92125	41152	74156	98801	64760	00144
86287	66161	23271	46773	53014	44442	25424	98309	32553	16678
90624	65685	68785	32385	90770	24676	68806	25347	16640	30602
98681	11402	57304	68003	16943	01041	44693	63237	76040	61075
52249	30617	91425	92780	82342	30467	19721	84117	96595	55215
79949	70851	29089	89311	19176	67653	95954	64805	51332	74301

- before

22233	22233	22233	22233	22233	22233	22233	22233	22233	22233
22233	22233	22233	22233	22233	22233	22233	22233	22233	22233
22233	65955	65955	22233	22233	22233	22233	33588	33588	33588
22233	22233	22233	22233	22233	22233	22233	33588	33588	33588
22233	22233	22233	22233	22233	22233	22233	22233	22233	22233
22233	22233	22233	22233	22233	22233	22233	22233	22233	22233
22233	22233	22233	22233	22233	22233	22233	13157	22233	22233
22233	22233	22233	22233	22233	22233	22233	13157	22233	22233
22233	22233	22233	22233	22233	22233	22233	22233	22233	22233
22233	22233	22233	22233	22233	22233	22233	22233	22233	22233

- after

- Cited from *swarm intelligence*, J.K and R.C. E

Discussions

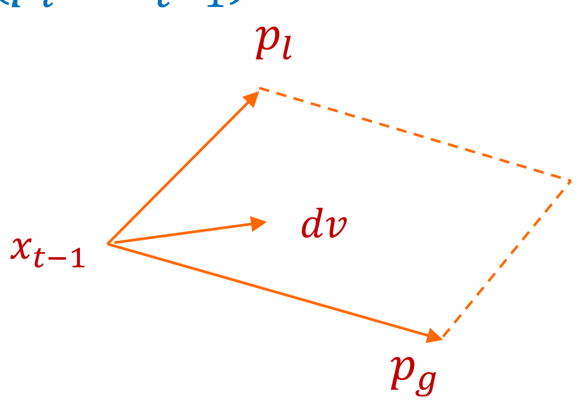
- Form homogeneous groups
  - Polarization
  - Cultural differences become insurmountable
    - Dissimilarity creates boundaries between cultural regions
    - Inter-individual similarities do not facilitate convergence
  - Reason: interaction is influenced by similarity
- Alternative simulation
  - No *similarity criteria* (unconditional interaction)
  - The population converges without polarity

Adaptive Culture Model

- Universal behaviors of individuals
  - Evaluate
    - Evaluate stimuli as positive or negative, attractive or repulsive
    - Distinguish features of the environment (good/bad)
    - Learn to improve the average evaluation
  - Compare
    - Judge ourselves through comparison with others (neighbors)
    - Looks, wealth, humor, intelligence…
  - Imitate
    - Imitate ONLY those neighbors who are superior to themselves
    - Few animals may imitate (humans/birds)
    - True imitation: not only imitating a behavior but realizing its purpose (very rare for animals)
- Adapt to environment challenges

Swarm Intelligence

- A set of particles as candidate solutions
- Each particle is influenced by
  - The very best performance of any member of the entire population (temporarily global optimum p_l)
  - The best performance of a group of k nearest neighbors (local optimum p_l)
- Updating rule for a particle
  - $v_t = v_{t-1} + \varphi_1 \cdot \left(p_g - x_{t-1}\right) + \varphi_2 \cdot (p_l - x_{t-1})$
  - $x_t = x_{t-1} + v_t$
  - φ_1, φ_2 are random numbers



Applications of Swarm Intelligence

- Find optimal solutions
- Optimizing parameters as meta method
  - e.g. PSO + ANN
- Evolution for a group of particles
  - flock of birds, school of fishes, … (bees)
  - Simple rules for each particle

Simulated Annealing

- Annealing: heating material to a very high temperature and then allowing it to cool very slowly
- Some global energy function e($\underline{x}$) of the variables $\underline{x}$ needs to be defined. The aim of simulated annealing is to minimize the energy.
- Simple Monte Carlo simulation vs. Metropolis Monte Carlo simulation

Concept of GPD

- Minimization f(x)
- $\mathrm{df} = \frac{\partial f}{\partial x} \cdot dx$

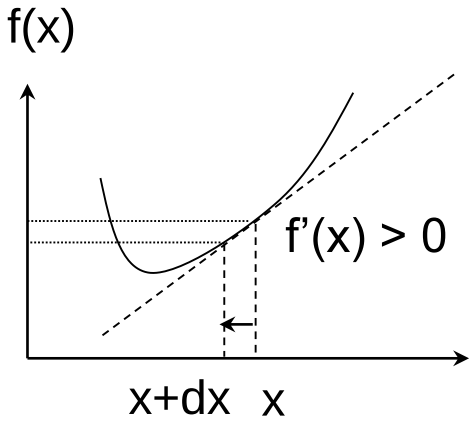
  if $\mathrm{dx} = -\varepsilon \cdot \frac{\partial f}{\partial x}$

  → df < 0

  x updated along the negative of derivative
- Reach local minimum
- Select initial x randomly

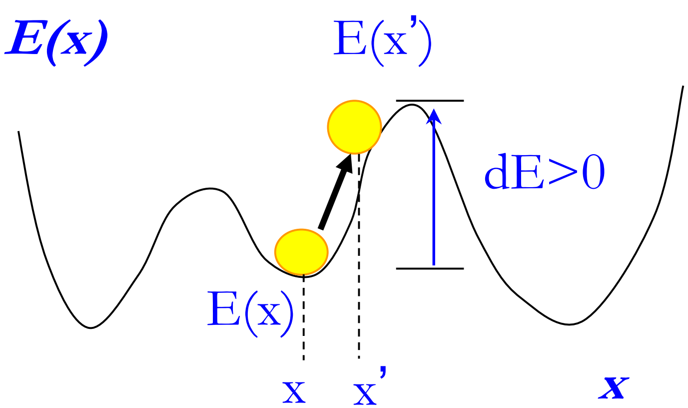
  → compute dx

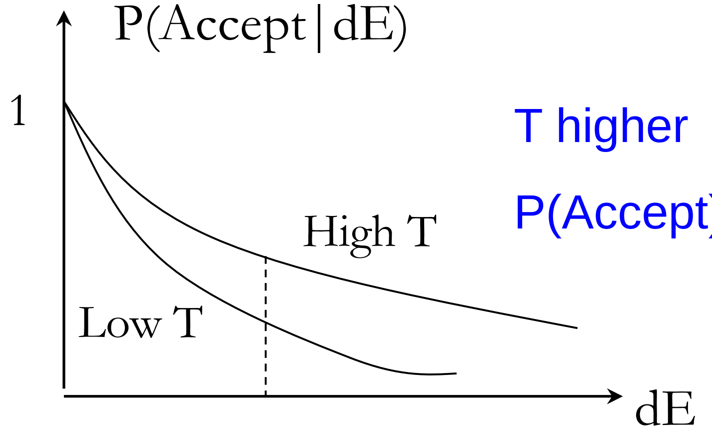
  → x' = x + dx
- Converging → df is close to 0



SIMULATED ANNEALING (CONT'D)

- A random start state ***x*** is selected.
- A small random change is made
  - $x' = x + \delta$.
  - If x' lowers the system energy, it is accepted.
  - If it increases the energy, it may be accepted, depending on a probability called the Boltzmann acceptance criteria P(accept | dE) = $e^{(-dE/T)}$





T higher

P(Accept) higher

Simulated Annealing (cont'd)

- Probability of acceptance: the chance that the ball escape from local minimum
- dE ↑, P(accept) ↓ more difficult to escape
- T↑, P(accept)↑ easier to escape
- T = 0, P(accept) = 0
  - do not escape from local minimum any longer
- Cooling schedule
  - reducing T from initial temperature
  - $T_{new} = T_{old} - dT$
  - $T_{new} = C \cdot T_{old}$ $(0 < C < 1)$

Simulated Annealing (cont'd)

- Has been successfully used for placing VLSI components, scheduling problems and other large combinatorial problems where values need to be assigned to a large number of variables to maximize (or minimize) some function of those variables.
- Example: knapsack problem
  - Solutions representing whether the objects are picked are selected randomly (e.g. (0, 1, 1, 0, 1, 0, 1, 0, 0, 1))
  - $E(\underline{x})$ is the total value of the picked objects (It can also be defined as some other measurement)
  - Use random test to make small changes for the solutions

Discussion on Simulated Annealing

- Compared with gradient descent (GD)
  - $E(x)$ always decreases ($dx = -\varepsilon E'(x) \rightarrow dE < 0$)
  - Arrive at local optimum
  - GD allows ONLY x to have continuous & differentiable variables since it needs to compute $E'(x)$
- Simulated annealing allows the state to jump out of the local optimum when $dE > 0$
- Simulated annealing allows x to have discrete variables

Genetic Algorithms

- A method based on biological evolution.
- Create *chromosomes* which represent possible solutions to a problem.
- A *fitness* value for each chromosome is determined.
- The best chromosomes in each generation are bred with each other to produce a new generation.
- Reproduction is done by applying *crossover* over two or more chromosomes
- *Mutation* is applied so as to make random changes to particular genes.

Introduction

- Local Search
  - Find solutions in a extremely large search space
  - Making small changes to potential solutions to a problem until an optimal solution is identified
- GA is a form of local search
  - Based on evolution
  - Borrow terms from genetics
    - Chromosome, crossover, mutation, fitness

Representation

- Chromosome (染色體)
  - A string of bits
  - Each bit is known as a gene
  - Formulated as a solution to a problem
- Population
  - A set of chromosomes
  - Creatures might have a number of chromosomes

Genetic Algorithm

1. Create a random population of chromosomes (the first generation)
2. If the termination criteria are satisfied, stop.
3. Determine the fitness of each chromosome.
4. Apply crossover and mutation to selected chromosomes from current generation to generate a new population of chromosomes – the next generation
5. Return to step 2.

Algorithm – cont'd

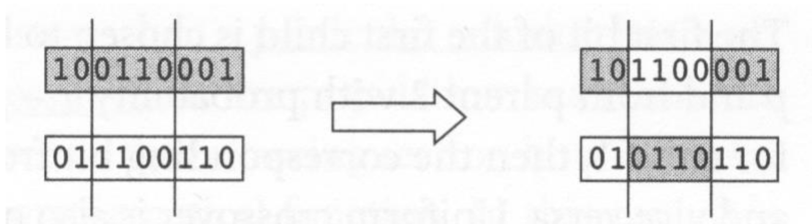
- Size of population
  - Should be determined in advance and remains constant from one generation to the next
- Size of each chromosome
  - Should be the same (variable chromosome size is unusual)
- Fittest chromosomes are selected to mate (with higher probability) and might generate more offspring
- Objective fitness function is used (in general)
  - Subjective fitness function can also be used (e.g. To generate interesting pictures, human judgment is important.)

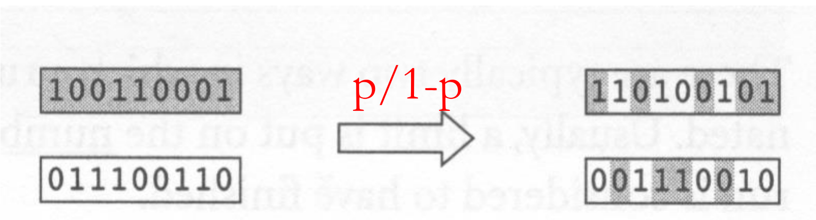
Selection

- The probability that a chromosome is selected to mate is proportional to its fitness value
- $P(c \text{ is selected}) = f(c)/(\Sigma_n f(c_n))$
  - c is selected by random test

Crossover

- Single point crossover
  - 110100 | 110001001 → 110100000111101
    010101 | 000111101 → 010101110001001
- Two-point crossover
- Uniform crossover
- Cloning
  - asexual reproduction
  - single parent





Mutation

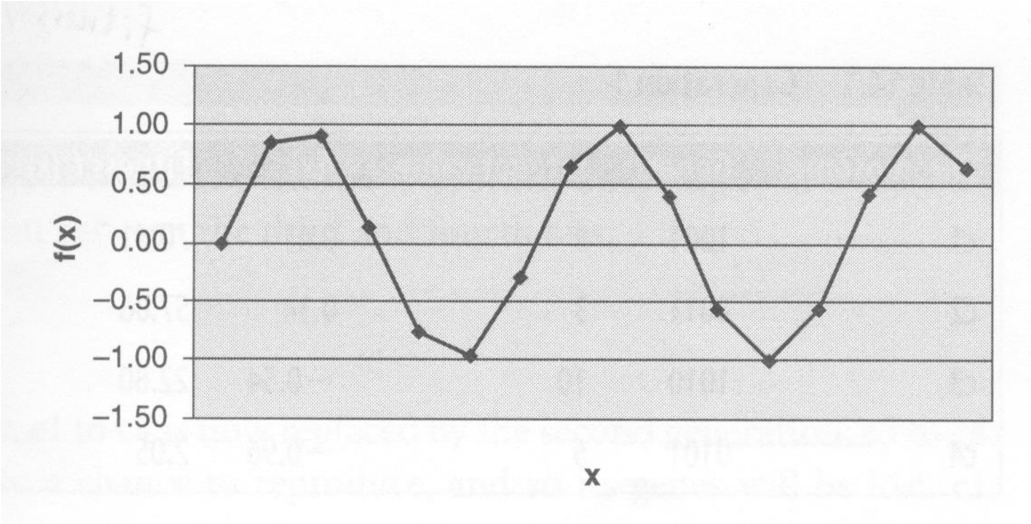
- GA is similar to Hill Climbing…
  - Generating possible solutions to a problem and moving toward a better solution until no better solution can be found
  - Does not perform well with problems with local optima
  - Mutation is introduced to avoid this.
- Mutation is usually applied with low probability

Termination

- GA is terminated if…
  - A limit on the number of generations
  - Particular solution is found
  - The highest fitness level has reached a particular value
- Culling
  - All individuals below a given threshold are discarded
  - Converge faster than the random version

Optimization of a Math Function

- Find maximum for sin(x)
- Fitness
  - 100 for f(x) = 1, 0 for f(x) = -1, 50 for f(x) = 0
  - f(x) = 100(sin(x) + 1)



EXAMPLE

Table 14.1 Generation 1

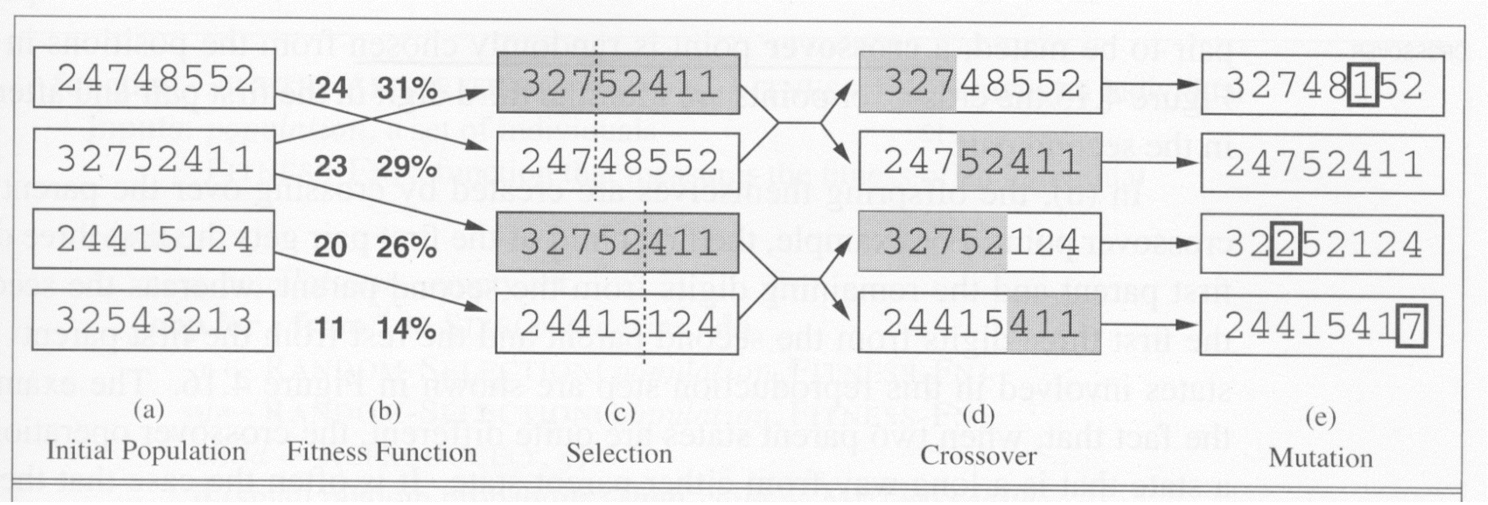
Chromosome	Genes	Integer value	$f(x)$	Fitness $f'(x)$	Fitness ratio
c1	1001	9	0.41	70.61	46.3%
c2	0011	3	0.14	57.06	37.4%
c3	1010	10	−0.54	22.80	14.9%
c4	0101	5	−0.96	2.05	1.34%

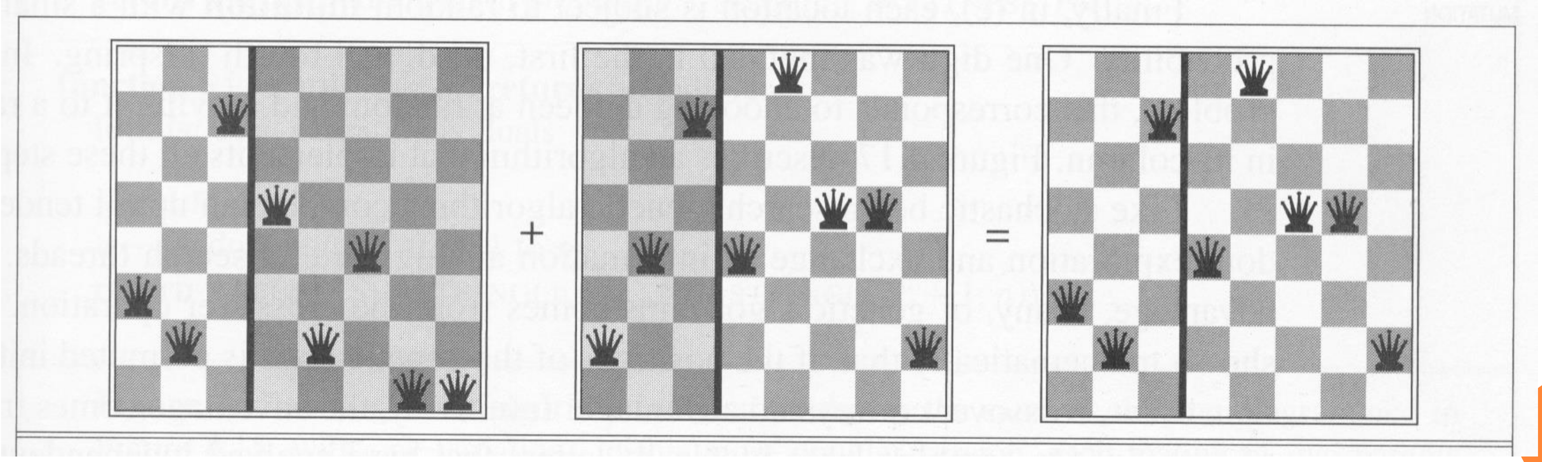
100%

Table 14.2 Generation 2

Chromosome	Genes	Integer value	$f(x)$	Fitness $f'(x)$	Fitness ratio
c5	1011	11	−1	0	0%
c6	0001	1	0.84	92.07	48.1%
c7	1000	8	0.99	99.47	51.9%
c8	1011	11	−1	0	0%

Solving 8-Queens Problem





32752411 24748552 32748552

Sample Program for GA

- Use GA to solve 8-Queens problem
  - In 8 questions, there exist $C^8{}_2$(=28) non-conflicts
- Use non-conflicts as a metric of fitness.
  - Goal: non-conflicts = 28

    e.g. onflicts, non-conflicts =25.
  - Culling: non-conflicts=20 as a threshold
- Applicable to 10 queens

Why GA Works?

- Schema S (pattern of chromosomes)
  - **1011****, 010**1****, *****10*00
- $d_L(S)$: defining length
  - Number of cutting positions between the highest bit and the lowest bit for the pattern
  - 3, 5, 4 respectively
- O(S): order
  - Number of fixed bits for the pattern
  - 4, 4, 4 respectively (for defined bits)
- Shorter, low order schema whose fitness is higher than the average fitness

  → has higher chance to survive!

Schema Theorem

Current generation

C_1 = 01000100101010010001010100101010
C_2 = 10100010100100001001010111010101
C_3 = 01010101011110101010100101010101
C_4 = $\boxed{11010}$101010101001101111010100101
C_5 = $\boxed{11010}$01010101001001010010000 1010
C_6 = 00101001010100101010010101111010
C_7 = 00101010100101010010101001010011
C_8 = 11111010010101010100101001010101
C_9 = 01010101010111101010001010101011
C_{10} = $\boxed{11010}$10010010101001111001010 0001

- S = 11010****************************
- m(S) = 3 (occurring 3 times: C_4, C_5, C_{10})
- f(C_4) = 10, f(C_5) = 22, f(C_{10}) = 40 → f(S) = (10+22+40)/3 = 24 (average)

Schema Theorem – cont'd

- Current generation: $C_1, C_2, \ldots, C_K$
  - average fitness: $a \equiv \Sigma_k f(C_k) / K$
- Probability that C_k is selected
  - $P(C_k) = f(C_k) / \Sigma_k f(C_k)$
- There are K selections to produce the next generations. The expected number that C_k is selected : $m'(C_k)$
  - $m'(C_k) = K * P(C_k) = K * f(C_k)/\Sigma_k f(C_k) = f(C_k) / a$
  - e.g. Assume the probability for head in coin tossing is 0.4. The expected number of heads in 5000 tossings is 5000 * 0.4 = 2000.

Schema Theorem – cont'd

- Schema (pattern) S contains C_n, so the expected number of patterns of schema S is m'(S)
  - $m'(S) = \Sigma_n m'(C_n) = (\Sigma_n f(C_n)) / a$
  - Summed for the chromosomes with pattern S (C_4, C_5, C_{10} in the example)
- Average fitness of Schema S is
  - $f(S) \equiv (\Sigma_n f(C_n)) / m(S)$
- It can be derived that
  - $m'(S) = [f(S)/a] * m(S)$

Schema Theorem – cont'd

- $m_{i+1}(S) = [f_i(S)/a_i] * m_i(S)$
- If in the i-th generation, the average fitness for the chromosomes of Schema S ($f_i(S)$) is higher than the average fitness of all chromosomes(a_i)

  (Schema S is better pattern of chromosomes)

  → $m_{i+1}(S) > m_i(S)$

  → the expected number of chromosomes belonging to Schema S will not decrease through selection

  → Schema S has higher chance to survive

Crossover and Mutation

- Crossover (with rate P_c)
  - $P_{survival}(S) \geqq 1 - P_c * d_L(S)/(L-1)$
  - For single point crossover, Schema S (with $d_L(S)$ cutting positions) tends to be destroyed if the cutting position is between the highest bit and the lowest bit
  - The higher the crossover rate, the more probable that Schema S is destroyed
  - Shorter schema (small $d_L(S)$) easy to survive
- Mutation (with rate P_m)
  - $P_{survival}(S) = (1-P_m)^{O(S)}$
  - Schema S can survive if mutation does not occur to the defined bits

Schema Theorem

$$m(S,i+1) \geq \left(\frac{f(S,i)}{a(i)} \right) \left(1 - p_c \cdot \frac{d_L(S)}{L(S)-1} \right) (1 - p_m)^{O(S)} \cdot m(S,i)$$

Shorter ($d_L(S)$ small), **lower order** (O(S) small) schemata which are **fitter than the average fitness** of the population (f(S,i) > a(i)) will appear with exponentially increasing regularity in subsequent generations.