# STAT 7650: Computational Statistics

## 4. Combinatorial Optimization

Peng Zeng

Department of Mathematics and Statistics
Auburn University

Spring 2026

# Outline



Reference: Givens and Hoeting (2013). Chapter 3.

# Combinatorial Optimization

Consider the optimization problem,

$$\max_{\theta \in \Theta} f(\theta) = f(\theta_1, \ldots, \theta_p),$$

where $\Theta$ consists of $N$ elements for a finite positive integer $N$.

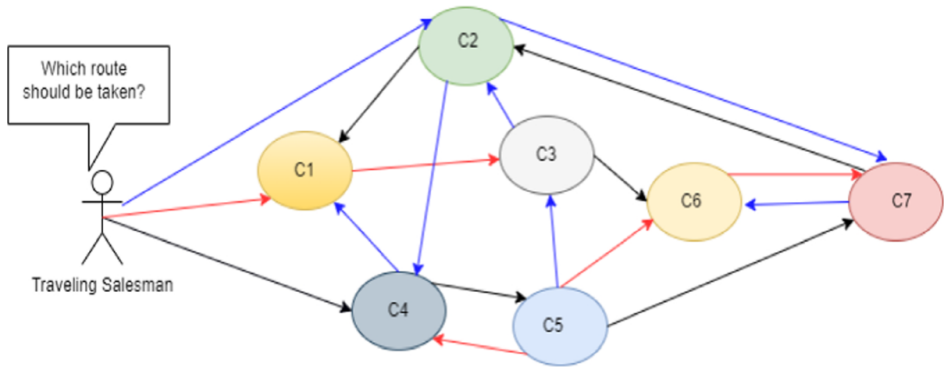
Each $\theta \in \Theta$ is a candidate solution. The set of global maxima is

$$\mathscr{M} = \{\theta \in \Theta : f(\theta) = f_{\max}\},$$

where $f_{\max} = \max f(\theta)$ denotes the global maximum.

# Example: Traveling Salesman Problem

Traveling salesman problem: A salesman visits each of $p$ cities exactly once and return to the starting city. We seek to minimize the total travel distance over all possible routes.




Possible rountes:

2765341

1356742

4572361

- It is a size-$p$ problem, since it is to find a sequence of $p$ cities.
- Let $\pi = (\pi_1, \ldots, \pi_p)$ be a permutation of $1, \ldots, p$,

$$\min f(\pi) = \sum_{i=1}^{p} \text{distance}(\pi_i \to \pi_{i+1}), \quad \text{where } \pi_{p+1} = \pi_1.$$

- There are in total $(p-1)!/2$ possible routes, since the point of origin and direction of travel are arbitrary.

# Complexity of Algorithms

The complexity of a problem is characterized by the number of operations required to solve it in the worst case using the best known algorithm.

- To sort $n$ values from the smallest to the largest
  - selection sort: $(n-1)+\cdots+1 = n(n-1)/2 = O(n^2)$.
  - quick sort: $O(n \log n)$.
- If the number of operations is $O(h(p))$ for a polynomial in $p$, an algorithm is said to be polynomial.
- The complexity of the traveling salesman problem is $O(p!)$.

| | Time to solve problem of order... | |
|---|---|---|
| $p$ | $O(p^2)$ | $O(p!)$ |
| 20 | 1 minute | 1 minute |
| 21 | 1.10 minutes | 21 minutes |
| 25 | 1.57 minutes | 12.1 years |
| 30 | 2.25 minutes | 207 million years |
| 50 | 6.25 minutes | $2.4\times10^{40}$ years |

# Example: Baseball Salaries

Data were collected for 337 baseball players in 1991.

- Response: log of a player's 1992 salary.
- Predictors: 27 baseball performance statistics.

Choose the best subset of predictors to fit a linear regression with the smallest AIC. There are in total $2^{27} \approx 10^9$ models.

| | | |
|---|---|---|
| 1. Batting average | 10. Strikeouts (SOs) | 19. Walks per SO |
| 2. On base pct. (OBP) | 11. Stolen bases (SBs) | 20. OBP / errors |
| 3. Runs scored | 12. Errors | 21. Runs per error |
| 4. Hits | 13. Free agency[a] | 22. Hits per error |
| 5. Doubles | 14. Arbitration[b] | 23. HRs per error |
| 6. Triples | 15. Runs per SO | 24. SOs × errors |
| 7. Home runs (HRs) | 16. Hits per SO | 25. SBs × OBP |
| 8. Runs batted in (RBIs) | 17. HRs per SO | 26. SBs × runs |
| 9. Walks | 18. RBIs per SO | 27. SBs × hits |

# Global vs Local Search

Global search

- guarantee to find the global optimum
- may not complete within a practical time limit

Local search

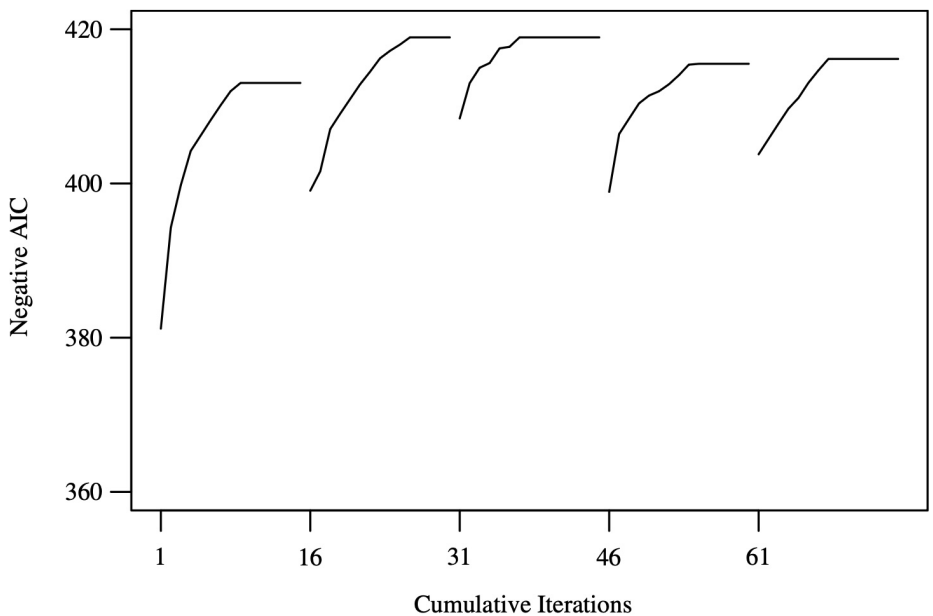
- limit the search to a local neighborhood $\mathcal{N}\left(\theta^{(t)}\right)$ of $\theta^{(t)}$ at any particular iteration.
- iteratively improve a current candidate solution: $\theta^{(t)} \longrightarrow \theta^{(t+1)}$
- may terminate at an uncompetitive local optimum

# Ascent Algorithm

- In a $k$-neighborhood $\mathcal{N}(\theta^{(t)})$, any $\theta$ is obtained from $\theta^{(t)}$ with at most $k$ changes.
- Steepest ascent: choose the best among all candidates in $\mathcal{N}(\theta^{(t)})$
- Random ascent or next ascent: select the first randomly chosen neighbor for which the objective function exceeds its previous value
- It is a greedy algorithm. Sometimes,
  - variable-depth local search: allow one or more steps to be suboptimal (e.g. random)
  - random starts local search: from a large number of randomly chosen starting points

# Example: Baseball Salaries

- 1-neighborhood: either add or remove one predictor
- Start from 5 randomly selected predictors, 14 additional steps
- Each move was made by steepest ascent
- Evaluate the objective function $1890 = 5 \times 14 \times 27$ times



# Results

- The optimal AIC $= -418.95$.
- The stepwise method chooses a model with 12 predictors, yielding AIC $= -418.94$.

| | Predictors selected | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | 1 | 2 | 6 | 7 | 9 | 10 | 12 | 15 | 16 | 18 | 19 | 20 | 21 | 22 | 24 | 25 | 26 | 27 | AIC |
| LS (2,3) | | • | • | | | • | | • | • | | | | | | • | • | • | | −418.95 |
| S-Plus | • | | • | | | • | | • | • | | | | | | • | • | • | | −418.94 |
| LS (5) | | | | | | • | • | • | • | | | • | | • | | • | | • | −416.15 |
| LS (4) | | | | | • | • | | | | | • | • | • | • | | | | | −415.52 |
| LS (1) | | | • | | | | • | | | | | | • | • | • | • | | | −413.04 |
| Efroy. | | | | • | | • | | | | • | | | | | • | | • | | −402.16 |

# Code Template

```
theta = ...                 # initial value

theta_opt = theta           # best solution
fval_opt  = f(theta)        # objective function

for(i in 1:maxiter)
{
    # identify neighborhood, and select one among them
    theta = update_solution(theta)
    fval = f(theta)

    if(fval > fval_opt)
    {   # update theta_opt if a better solution is found
        theta_opt = theta
        fval_opt  = fval
    }
}
```

# Comments

- Represent a solution appropriately
  - traveling salesman: a permutation of $1, \ldots, p$
  - variable selection: a sequence of 1/0 or TRUE/FALSE
- The best solution is not necessarily the last solution
- The algorithm can only search and compare a small portion of candidate values among all possible ones

# Simulated Annealing

Annealing: heating up a solid and then cooling it slowly.

- When a solid is heated, its internal energy increases and its molecules move randomly.
- If the solid is then cooled slowly, the thermal energy generally decreases slowly, but there are also random increases.
- If the cooling is slow enough and deep enough, all the molecules are arranged to have minimal potential energy.

Apply this idea to an optimization problem,

$$\min_{\theta \in \Theta} f(\theta)$$

- $\theta$ corresponds to the state of the material
- $f(\theta)$ corresponds to its energy level
- Randomness: accept an uphill move with probability

# Algorithm - Simulated Annealing

Select an initial point $\theta^{(0)}$ and a temperature $\tau_0$ at time $t = 0$.

1. Select a candidate solution $\theta^*$ within the neighborhood of $\theta^{(t)}$ according to a proposal density $g^{(t)}(\cdot \mid \theta^{(t)})$.
2. Let $p = \min(1, \exp\{[f(\theta^{(t)}) - f(\theta^*)]/\tau_j\})$ and

$$\theta^{(t+1)} = \begin{cases} \theta^*, & \text{with probability } p \\ \theta^{(t)}, & \text{otherwise} \end{cases}$$

3. Repeat steps 1 and 2 a total of $m_j$ times
4. Increment $j$, update $\tau_j = \alpha(\tau_{j-1})$ and $m_j = \beta(m_{j-1})$.
5. Go to step 1.

The stopping rule is often expressed as a minimum temperature. It is also possible to monitor an absolute or relative convergence criterion. After stopping, the best candidate solution found is the minimum.

# Code Template

```
tau = ...; n_iter = ...                  # temperature and num iter
theta = ...; f_theta = f(theta)          # initial value and f-value
theta_opt = theta; f_opt = f_theta       # best solution and f-value

while(stopping_rule) {
    for(i in 1:n_iter) {
        theta_new = randomly_generate(theta) # propose a new one
        f_new = f(theta_new)
        if(log(runif(1)) < (f_theta - f_new) / tau) {
            theta = theta_new; f_theta = f_new;
            if(f_theta < f_opt}
            { theta_opt = theta; f_opt = f_theta }
        }
    }
    tau = update_temperature(tau)    # update temperature
    n_iter = update_n_iter(n_iter)   # update num of iterations
}
```

# Comments

- The function $\alpha(\cdot)$ should slowly decrease the temperature to 0.
- The number of iterations at each temperature ($m_j$) should be large and increasing in $j$. Ideally, the function $\beta$ should scale the $m_j$ exponentially in $p$, but in practice some compromises will be required in order to obtain tolerable computing speed.
- Simulated annealing is a stochastic descent algorithm. Its randomness allows simulated annealing sometimes to escape uncompetitive local minima.

# Neighborhood, Proposal, Objective Function

- Choose neighborhoods that are small and easy to compute.
- Choose neighborhood structure allows all solutions in $\Theta$ to communicate. For $\theta_i$ and $\theta_j$ to communicate, it must be possible to find a finite sequences $\theta_1, \ldots, \theta_k$ such that $\theta_1 \in \mathcal{N}(\theta_i)$, $\theta_2 \in \mathcal{N}(\theta_1), \ldots, \theta_k \in \mathcal{N}(\theta_{k-1})$, and $\theta_j \in \mathcal{N}(\theta_k)$.
- The most common proposal density is discrete uniform.
- To speed up computation, it is common to evaluate the objective function based on the current value $f(\theta^{(t)})$.

Example (traveling salesman problem): generate a neighbor of $\theta$ by removing two non-adjacent links and reconnecting the tour. For example: 123456 to 143256.

$$f(143256) = f(123456) - (D_{1\to 2} + D_{4\to 5}) + (D_{1\to 4} + D_{2\to 5})$$

# Convergence and Stationary Distribution

Assume that

- the temperature is fixed at $\tau$
- for any pair $\theta_i, \theta_j \in \Theta$, proposing $\theta_i$ from $\mathcal{N}(\theta_j)$ has the same probability as proposing $\theta_j$ from $\mathcal{N}(\theta_i)$

The sequence of $\theta^{(t)}$ is generated from a Markov chain with stationary distribution $\pi_t(\theta) \propto \exp\{-f(\theta)/\tau\}$, or

$$\pi_\tau(\theta_i) = \frac{\exp\{-[f(\theta_i) - f_{\min}]/\tau\}}{M + \sum_{j \notin \mathscr{M}} \exp\{-[f(\theta_j) - f_{\min}]/\tau\}}, \qquad \theta_i \in \Theta$$

where $f_{\min} = \min f(\theta)$, $\mathscr{M} = \{\theta : f(\theta) = f_{\min}\}$, and $M$ is the number of $\theta$ in $\mathscr{M}$. The limiting distribution is

$$\lim_{\tau \to 0} \pi_\tau(\theta_i) = 1/M, \text{ for } i \in \mathscr{M}; \qquad = 0, \text{ otherwise.}$$

We would like to run the chain at this fixed temperature long enough.

# Stopping Criterion

We can relate the cooling schedule to a bound on the quality of the final solution.

If one wishes any iterate to have not more than probability $\delta$ of being worse than the global minimum by no more than $\varepsilon$,

$$P(f(\theta^{(t)}) > f_{\min} + \varepsilon) < \delta$$

this can be achieved if one cools until $\tau_j \leq \varepsilon / \log((N-1)/\delta)$, where $N$ is the number of points $\Theta$.

# Choose $\alpha$ and $\beta$

The temperature at stage $j$ is $\tau_j = \alpha(\tau_{j-1})$ and the number of iterations in stage $j$ is $m_j = \beta(m_{j-1})$.

- Set $m_j = 1$ for all $j$ and $\alpha(\tau_{j-1}) = \tau_{j-1}/(1 + a\tau_{j-1})$ for a small value of $a$.
- $\alpha(\tau_{j-1}) = a\tau_{j-1}$ for $a < 1$ (usually $a > 0.9$) and $\beta(m_{j-1}) = bm_{j-1}$ for $b > 1$ or $\beta(m_{j-1}) = b + m_{j-1}$ for $b > 0$.
- Set

$$\alpha(\tau_{j-1}) = \frac{\tau_{j-1}}{1 + \tau_{j-1}\log\{1 + r\}/(3s_{\tau_{j-1}})}$$

where $s^2_{\tau_{j-1}}$ is the mean squared objective function cost at the current temperature minus the square of the mean cost at the current temperature, and $r$ is a small real number.

# More Comments

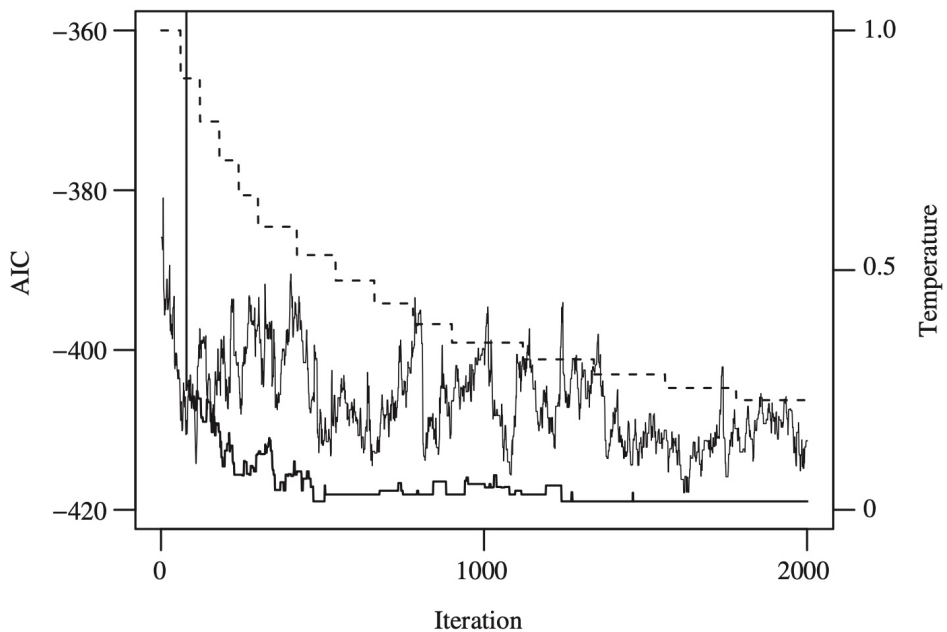
Choose a positive $\tau_0$ so that $\exp\{[f(\theta_i) - f(\theta_j)]/\tau_0\}$ is close to 1 for any pair $\theta_i, \theta_j \in \Theta$. The rationale for this choice is that it provides any point in the parameter space with a reasonable chance of being visited in early iterations of the algorithm.

Choosing $m_j$ to be large can produce a more accurate solution, but can result in long computing times. As a general rule of thumb, larger decreases in temperature require longer runs after the decrease.

Running simulated annealing long at high temperatures is not very useful. Good cooling schedules therefore decrease the temperature rapidly at first.

# Example: Baseball Salaries

- Neighborhood: add or delete one predictor.
- Cooling schedule has 15 stages, with stage length $= 60$ for first 5 stages, 120 for next 5, and 220 for the final 5.
- Set $\alpha(\tau_{j-1}) = 0.9\tau_{j-1}$
- Compare $\tau_0 = 1$ and $\tau_0 = 6$.



# Genetic Algorithms

Genetic algorithms mimic the process of Darwinian natural selection.

$$\max_{\theta \in \Theta} f(\theta)$$

Generate a population of organisms. Breed them to produce offsprings. If fit organisms are more likely to produce offsprings and pass desirable attributes to offsprings, the organisms in the population should evolve to become increasingly fit over time.

| | |
|---|---|
| $\theta$ | biological organisms |
| $\phi$, representation of $\theta$ | genetic code, chromosome |
| $f(\theta)$ | fitness of an organism |
| generate $\theta$ from $\theta_i$ and $\theta_j$ | breeding |

# Chromosome

A chromosome is a sequence of $C$ symbols, each of which consists of a single choice from a pre-determined alphabet.

- One common choice is the binary alphabet, $\{0, 1\}$, in which case a chromosome of length $C = 9$ might look like $100110001$.

More terms borrowed from biology.

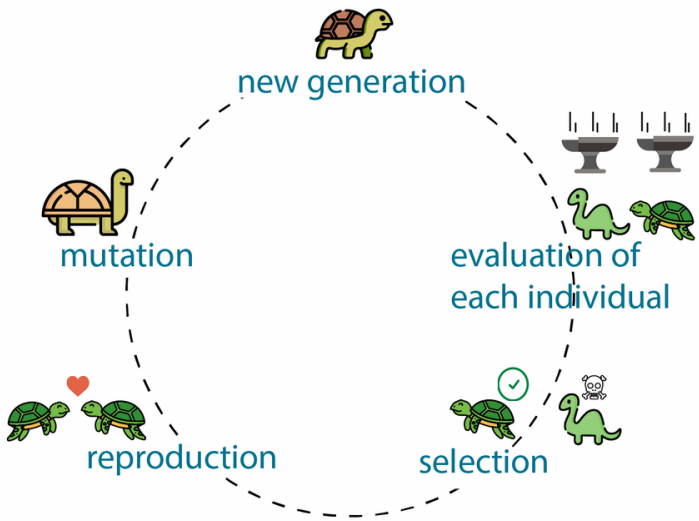
- Gene or allele: each element of the chromosome
- Locus: the position of a gene or allele
- Phenotype: $\theta$, candidate values
- Genotype: $\phi$, representation of $\theta$

# Genetic Algorithms

Let the $t$th generation consist of a collection of $P$ organisms,

| | |
|---|---|
| genotype | $\phi_1^{(t)}, \ldots, \phi_P^{(t)}$ |
| phenotype | $\theta_1^{(t)}, \ldots, \theta_P^{(t)}$ |

- Evaluate fitness $f(\theta_i^{(t)})$
- Select individuals with high fitness to produce offsprings
- Mutation with probability




Repeat the breeding process to produce enough offsprings. The size of the population usually remains fixed across generations.

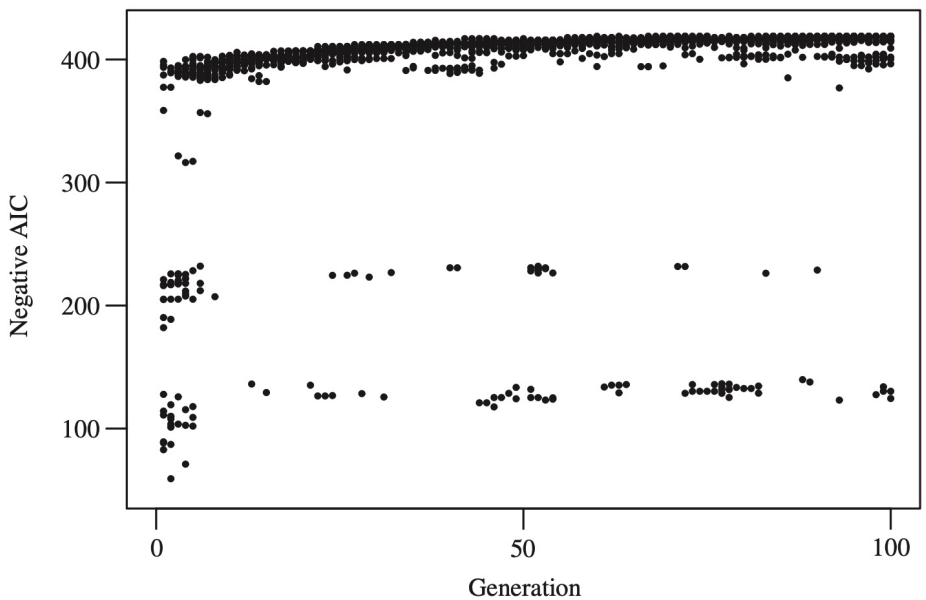
# Example: Variable Selection

Consider a variable selection problem in a linear regression with 9 predictors. Assume that an intercept is included in any model.

- Genotype: a chromosome of length 9, e.g. $\phi_i^{(t)} = 100110001$
- Fitness: the objective function, AIC values
- Two common selection mechanisms:
  - select one parent with probability proportional to fitness and the other parent completely at random
  - select each parent independently with probability proportional to fitness
- Produce an offspring from $\phi_i^{(t)}$ and $\phi_j^{(t)} = 110100110$.
  - crossover: two chromosomes exchange some segments.
  - mutation: change each allele to other values independently with probability $\mu$

# Example: Baseball Salaries

- A chromosome consists of $C = 27$ alleles with values in $\{0, 1\}$
- 100 generation of size $P = 20$
- purely random individuals in the starting generation
- rank-based fitness function
- select one parent with probability proportional to the fitness, and the other one purely at random
- simple crossover, and 1% mutation rate



# Allele Alphabets and Genotypic Representation

The binary alphabet for alleles is very common. For example,

$$f(\theta) = 100 - (\theta - 4)^2, \quad \theta \in [1, 12.999] = [a_1, a_2].$$

We represent a number in $[a_1, a_2]$ as

$$a_1 + \left( \frac{a_2 - a_1}{2^d - 1} \right) \text{decimal}(b)$$

where $b$ is a binary number of $d$ digits and the decimal() function converts from base 2 to base 10. If $c$ decimal places of accuracy are required, then $d$ must be chosen to satisfy

$$(a_2 - a_1)10^c \leq 2^d - 1$$

In the example, 15 binary digits are required for accuracy up to 3 decimal places.

# Comments

Chromosomes that are similar in genotype may have very different phenotypes.

A small mutation may move to a drastically different region of solution space, and a crossover may produce offspring whose phenotypes bear little resemblance to either parent.

| $\phi$ | $\theta$ |
|---|---|
| 01000000000000 | 4.000 |
| 10000000000000 | 7.000 |
| 00000000000000 | 1.000 |
| 00111111111111 | close to 4 |

To resolve such difficulties, a different encoding scheme or modified genetic operators may be required.

# Initialization, Termination, Parameter Values

- The first generation consists of purely random individuals.
- The size of the generation $P$: prefer large values of $P$
  - For binary encoding of chromosomes, $C \leq P \leq 2C$, where $C$ is the chromosome length.
  - For permutation chromosomes, $2C \leq P \leq 20C$.
  - For most real applications, $10 \leq P \leq 200$.
- Mutation rates are typically very low, in the neighborhood of 1%.
- The termination criterion for a genetic algorithm is frequently just a maximum number of iterations chosen to limit computing time.

# Fitness

Let $g(\phi)$ denote the value of a fitness function that describes the fitness of a chromosome.

- Set $g(\phi) = f(\theta)$, the objective function value of its phenotype.
- A rank-based function:

$$g(\phi_i^{(t)}) = \frac{2r_i}{P(P+1)}$$

where $r_i$ is the rank of $f(\theta_i^{(t)})$ among generation $t$.

- selection probability of the median quality candidate is $1/P$
- selection probability of the best candidate is $2/(P+1)$, roughly double that for the median.

Selecting parents on the basis of fitness rank is far more common than using selection probabilities proportional to fitness.

# Selection Mechanisms and Updating Generations

Tournament selection:

- randomly partition the population into $k$ disjoint subsets
- choose the best individual in each group as a parent
- repeat the partition-selection process for more parenets
- pair the parents randomly for breeding

This approach ensures that the best individual will breed $P$ times, the median individual will breed once on average, and the worst individual will not breed at all.

Populations can be partially updated. The generation gap, $G$, is a proportion of the generation to be replaced by generated offspring.

# Permutation Chromosomes

In the traveling salesman problem, a natural chromosome is a permutation of integers $1, \ldots, p$.

We can apply order crossover: A random collection of loci is chosen, and the order in which the alleles in these loci appear in one parent is imposed on the same alleles in the other parent to produce one offspring.

- parents: 752631948 and 912386754
- randomly select loci from the first one: 4th, 6th, 7th $\rightarrow$ 6, 1, 9
- remove 6, 1, 9 in second one **238*754
- insert 6, 1, 9 back in the order as the first one: 612389754
- reverse the role to get 352671948

# Tabu Algorithm

A tabu algorithm is a local search algorithm with variable neighborhoods and additional rules to promote the discovery of a global maximum.

- In a standard ascent algorithm, entrapment in a globally uncompetitive local maximum is likely, because no downhill moves are allowed.
- Tabu search allows downhill moves when no uphill move can be found in the current neighborhood (and possibly in other situations too), thereby potentially escaping entrapment.
- A tabu is a certain move that is temporarily forbidden based on the recent history of the algorithm. For example, in a variable selection problem, do not add one variable back immediately if this variable has recently been deleted.

# Basics

Tabu search is an iterative algorithm.

- Initialization: at time $t = 0$, select a candidate solution $\theta^{(0)}$
- At the $t$th iteration,
  - identify the neighborhood $\mathscr{N}(\theta^{(t)}, H^{(t)})$ that excludes a list of temporarily forbidden moves

$$\mathscr{N}(\theta^{(t)}, H^{(t)}) = \{\theta : \theta \in \mathscr{N}(\theta^{(t)}) \text{ and no attribute of } \theta \text{ is currently tabu}\}$$

    where $H^{(t)}$ is the history of the algorithm through time $t$
  - select a new solution $\theta^{(t+1)}$ in $\mathscr{N}(\theta^{(t)}, H^{(t)})$, which yields steepest ascent or mildest descent if no neighbhor yields an increased $f$.

# Attributes

A single step from $\theta^{(t)}$ to $\theta^{(t+1)}$ can be characterized by attributes, which are used to describe moves or types of moves that will be forbidden, encouraged, or discouraged in future iterations.

| Attribute | Model Selection Example |
|---|---|
| a change in the value of $\theta_i^{(t)}$ | $A_1$: Whether the $i$th predictor is added (or deleted) from the model. |
| a swap in the values of $\theta_i^{(t)}$ and $\theta_j^{(t)}$ | $A_2$: Whether the absent variable is exchanged for the variable present in the model. |
| a change in the value of $f$ resulting from the step, $f(\theta^{(t+1)}) - f(\theta^{(t)})$ | $A_3$: The reduction in AIC achieved by the move. |
| the value $g(\theta^{(t+1)})$ of some other function $g$. | $A_4$: The number of predictors in the new model. |
| a change in the value of $g$ resulting from the step, $g(\theta^{(t+1)}) - g(\theta^{(t)})$. | $A_5$: A change to a different variable selection criterion such as Mallows's $C_p$ |

# Attribute and Recency

Denote the $a$th attribute as $A_a$ and $\bar{A}_a$ as its complement.

- If $A_a$ means swapping the values of $\theta_i^{(t)}$ and $\theta_j^{(t)}$, then $\bar{A}_a$ means not making that swap.

The recency of an attribute is the number of steps that have passed since a move most recently had that attribute.

- $R(A_a, H^{(t)}) = 0$ if $A_a$ is expressed in the move yielding $\theta^{(t)}$
- $R(A_a, H^{(t)}) = 1$ if $A_a$ is expressed in the move yielding $\theta^{(t-1)}$, but not in the move yielding $\theta^{(t)}$
- $R(A_a, H^{(t)}) = 2$ if ...

Each time a move with attribute $A_a$ is taken, $\bar{A}_a$ is put on a tabu list for $\tau$ iterations. When $R(A_a, H^{(t)})$ first equals $\tau$, the tabu expires and $\bar{A}_a$ is removed from the tabu list.

# Tabu Tenure

The tabu list prevents undoing the change for $\tau$ iterations, thereby discouraging cycling.

The tabu tenure $\tau$ is the number of iterations over which an attribute is tabu. This can be a fixed number or it may vary, systematically or randomly, perhaps based on features of the attribute.

- choose $\tau$ between 7 and 20,
- choose $\tau$ between $0.5\sqrt{p}$ and $2\sqrt{p}$, where $p$ is the size of the problem.

# Four Types of Rules

There are four general types of rules added to a tabu algorithm.

- tabu: keep a list of temporarily forbidden moves
- aspiration: override the temporarily forbidden moves
- diversification: promote broader exploration of the solution space
- intensification: increase effort in promising areas

# Aspiration Criteria

An aspiration criterion overrides the tabu list.

- Permit a tabu move if it provides a higher value of the objective function than that has been found in any iteration so far.
- Aspiration criteria can also be used to encourage moves that are not tabu.
  - downweight low-influence moves
  - prefer high-influence moves

  A move or attribute is influential if it is associated with a large change in the value of the objective function.

# Diversification

The frequency of an attribute records the number of moves that manifested that attribute since the search began. Let $C(A_a, H^{(t)})$ represent the count of occurrences of the $a$th attribute thus far.

Increase search diversification by considering $f_{H^{(t)}}$ given by

$$f_{H^{(t)}}(\theta^{(t+1)}) = \begin{cases} f(\theta^{(t+1)}), & \text{if } f(\theta^{(t+1)}) \geq f(\theta^{(t)}) \\ f(\theta^{(t+1)}) - cF(A_a, H^{(t)}), & \text{if } f(\theta^{(t+1)}) < f(\theta^{(t)}) \end{cases}$$

where $F(A_a, H^{(t)}) = C(A_a, H^{(t)})/t$ and the denominator may be replaced by the sum, the maximum, or the average of the counts.

If all non-tabu moves are downhill, then this approach discourages moves that have the high-frequency attribute $A_a$.

# Intensification

Frequencies can also be used to guide intensification that increases search effort in particular areas of solution space.

The time span $v > \tau$ parameterizes the length of a long-term memory to enable search intensification in promising areas of solution space.

- tabulate the frequencies of attributes over the most recent $v$ moves, and keep the corresponding values of objective function
- identify key attributes shared by good candidate solutions
- reward moves that retain such features by defining $\tilde{f}_{H^{(t)}}$

# Comprehensive Tabu Algorithm

1. Determine an augmented objective function $f_{H^{(t)}}$ that depends on $f$ and perhaps on frequency-based penalties or incentives to promote diversification and/or intensification.
2. Identify $\mathscr{N}(\theta^{((t)})$, the neighborhood of $\theta^{(t)}$
3. Rank the neighbors in decreasing order of $f_{H^{(t)}}$
4. Select the highest ranking neighbor.
5. Is this neighbor currently on the tabu list? If not, go to step 8.
6. Does this neighbor pass an aspiration criterion? If so, go to step 8.
7. If all neighbors of $\theta^{(t)}$ have been considered and none have been adopted as $\theta^{(t+1)}$, then stop. Otherwise, select the next most high-ranking neighbor and go to step 5.
8. Adopt this solution as $\theta^{(t+1)}$.
9. Update the tabu list
10. Has a stopping criterion been met? If so, stop. Otherwise, increment $t$ and go to step 1.

# Example: Baseball Salaries

- Only monitor attributes signaling the presence or absence of each predictor
- Tabu list: moves that reverse the inclusion or removal of a predictor, $\tau = 5$
- run the algorithm for $75$ moves from a random start
- Aspiration criterion: permit a tabu move if it yields an objective function value above the current best value