COMP30030: Introduction to Artificial Intelligence

Neil Hurley

School of Computer Science
University College Dublin
neil.hurley@ucd.ie

October 16, 2018

Graph Partitioning I

Definition

Let $G = (V, E)$ be a graph. A partition of the vertex set is a set of vertex sub-sets $\{P_1, \ldots, P_k\}$ such that $P_i \cap P_j = \emptyset$ and $P_1 \cup \cdots \cup P_k = V$. Given k, the balanced graph partitioning problem is to find a vertex partition such that $|P_1| = \cdots = |P_k|$ and the edge-cut is minimised. The edge cut is defined as

$$|\{(v, w) \in E | v \in P_i, w \in P_j \text{ s.t. } i \neq j\}|$$

- Lots of applications of this problem
  - Partitioning a computation across a parallel machine.
  - Module placement in VLSI design
  - FInding communities in social network
  - etc.

Representation of Graph Partitioning Problem I

- Consider the graph bi-partitioning problem (i.e. k=2). We can represent a solution to this problem (i.e. an example partition) by an indicator vector **x** of length $n =$ number of vertices, such that

$$\begin{aligned} x_i &= -1 \quad v_i \in P_0 \\ x_i &= 1 \quad v_i \in P_1 \end{aligned} \tag{1}$$

- The edge-cut objective can be expressed using the Laplacian matrix of the graph.
- The adjacency matrix A of a graph G is defined as

$$a_{ij} = \begin{cases} 1 & (v_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases}$$

- Let D be a diagonal matrix, such that the value on the diagonal is the degree of vertex v_i.

Representation of Graph Partitioning Problem II

- The Laplacian matrix is defined as

$$\mathrm{L} = \mathrm{D} - \mathrm{A}$$

- Now consider the quadratic form

$$\begin{aligned}
\mathbf{x}^T \mathrm{L} \mathbf{x} &= \sum_{i=1}^{n} \sum_{j=1}^{n} l_{ij} x_i x_j \qquad (2) \\
&= \sum_i d_i x_i^2 - \sum_i \sum_j a_{ij} x_i x_j \\
&= \sum d_i - \sum_{x_i = x_j} a_{ij} + \sum_{x_i \neq x_j} a_{ij} \\
&= 2m - 2I + 2E \\
&\quad (m = \text{ no. edges } I = \text{ no. int edges } E = \text{ no. ext edges}) \\
&= 2m - 2(m - E) + 2E \\
&= 4E
\end{aligned}$$

Representation of Graph Partitioning Problem III

- So we can write graph partitioning as the problem to find $\mathbf{x}$

$$\mathbf{x} = \arg\min \mathbf{x}^T \mathrm{L} \mathbf{x} \qquad \text{s.t.} \sum_i x_i = 0 \quad x_i \in \{-1, 1\}$$

This is an integer (specifically, binary) **quadratic programming** problem with <u>linear</u> constraints.

Relaxation

- One useful partial strategy for solving integer optimisation problems is to relax the integer constraints to allow for solutions in the real numbers.
- The minimum of a problem with fewer constraints will necessarily provide a lower bound on the best solution that can be obtained on the fully constrained problem.
- The solution to the relaxed problem can also be used as a starting point for obtaining an integer solution.
- The lower bound can be used in branch and bound techniques to prune the search space.

Relaxing the Graph Partitioning Problem I

- Consider the following relaxation:
  Find $\mathbf{x} \in \mathbb{R}^n$

$$\mathbf{x} = \arg\min \mathbf{x}^T \mathrm{L}\mathbf{x} \qquad \text{s.t.} \|\mathbf{x}\|^2 = n$$

- All solutions to the integer problem must necessarily satisfy the constraints of this problem.
- Also, it has a straight-forward solution using Lagrange multipliers:

$$o(\mathbf{x}) = \mathbf{x}^T \mathrm{L}\mathbf{x} - \lambda(\|\mathbf{x}\|^2 - n)$$

- Computing the gradient of o and setting to zero, we obtain

$$\mathrm{L}\mathbf{x} = \lambda\mathbf{x}$$

- The objective is minmised by solving an eigenvalue problem and selecting the eigenvalue of lowest magnitude.

Relaxing the Graph Partitioning Problem II

- It turns out that $\mathbf{x} = (1, \ldots, 1)^T$ is a solution to this problem with minimum eigenvalue 0, but this corresponds to placing all vertices in a single set.
- The eigenvector corresponding to <u>second smallest eigenvalue</u> the so called <u>Fiedler vector</u> is the required solution which also satisfies the constraint $\sum_i x_i = 0$. (Why? Because all other eigenvectors are orthogonal to the first eigenvector).
- Once we have $\mathbf{x} \in \mathbb{R}^n$ as a real-valued solution, we can quantize it to generate a proposed integer solution – set the lowest $n/2$ values of x_i to -1 and the rest to $+1$.
- Of course, relaxation and quantisation is not guaranteed to produce a globally optimal solution.

Bin Packing Problem

Definition

Given a finite set $O = \{a_1, \ldots, a_n\}$ of positive numbers (corresponding to object sizes) i.e. $a_i > 0$ and a constant B (the bin size), find the minimum number of bins N such that O can be partitioned into N subsets such that the sum of elements in any of the subsets doesn't exceed B i.e. $O = \{S_1 \cup \cdots \cup S_N\}$, $S_i \cap S_j = \emptyset$, $i \neq j$ and

$$\sum_{a_i \in S_k} a_i \leq B \quad 1 \leq k \leq N$$

- Applications of bin packing include machine scheduling - scheduling a set of tasks on a set of machines, in order to minimize the number of machines required to complete the tasks within a specified time.

Representation of Bin Packing Problem I

- We can represent a solution to this problem by an indicator vector $\mathbf{y}$ of length $n =$ maximum number of bins, such that

$$\begin{aligned} y_i &= 0 \quad \text{bin } i \text{ is not used} \\ y_i &= 1 \quad \text{bin } i \text{ is used} \end{aligned} \tag{3}$$

<u>and</u> a matrix X, such that

$$x_{ij} = \begin{cases} 1 & \text{if object } j \text{ is in bin} i \\ 0 & \text{otherwise} \end{cases}$$

Representation of Bin Packing Problem II

- Then the bin packing problem can be expressed as

$$(\mathbf{y}, \mathrm{X}) = \arg\min \sum_{i=1}^{n} y_i$$

subject to

$$\sum_{j=1}^{n} a_j x_{ij} \leq B y_i$$

$$\sum_{i=1}^{n} x_{ij} = 1 \, \forall j \in \{1, \ldots, n\}$$

$$x_{ij} \in \{0, 1\} \quad y_i \in \{0, 1\}$$

- An integer linear programming problem.

Stochastic Processes I

- A stochastic process, $\{X(t), t \in T\}$ is a collection of random variables, indexed by the set T, which is often interpreted as time. We often refer to $X(t)$ as the state of the process at time t.
- The set T may be discrete i.e. containing at most a countable set of values, or continuous i.e. containing uncountably infinite values. The set of possible outcomes of $X(t)$ – usually referred to as the state space rather than the sample space, in this context, may also be discrete or continuous. Hence, we refer to, for example, discrete-time discrete-space stochastic processes, or continuous-time, discrete space stochastic processes and so on.

Stochastic Processes II

- Formally, a stochastic process is a mapping from the sample space S to function of t. Thus, if $e \in S$, the $X(t, e)$ is a function of time (often called the realisation of the stochastic process). For a given value of t, $X(t, e)$ is a random variable. For a given value of e and t, $X(t, e)$ is a fixed number.
- Discrete time stochastic processes are also called random sequences, and may be written as $\{X_n, n = 0, 1, 2, \ldots\}$.
- The time-dependent distribution of $X(t)$ is the distribution of the random variable $X(t)$ at a given instant of time t.
- The stationary distribution of $X(t)$ is the distribution as $t \to \infty$, if it exists.

Stochastic Processes III

- The n^{th} order statistics of a stochastic process $X(t)$ is defined by the joint distribution

$$F_{X_{t_1},\ldots,X_{t_n}}(x_1,\ldots,x_n) = P\{X_{t_1} \leq x_1,\ldots,X_{t_n} \leq x_n\}$$

  for all possible sets $\{t_1,\ldots,t_n\}$. A complete characterisation of $X(t)$ requires knowing the stochastics of all order n.

- A process is said to be stationary if the statistics of all orders are unchanged by a shift in the time axis i.e.

$$F_{X_{t_1+\tau},\ldots,X_{t_n+\tau}}(x_1,\ldots,x_n) = F_{X_{t_1},\ldots,X_{t_n}}(x_1,\ldots,x_n) \ \forall n \ \forall t_1,\ldots,t_n$$

- A stochastic process is called a Markov process if it has the Markov property i.e.

$$\begin{aligned} &P\{X(t_n) \leq x_n | X(t_{n-1}) = x_{n-1},\ldots,X(t_1) = x_1\} \\ &\qquad = P\{X(t_n) \leq x_n | X(t_{n-1}) = x_{n-1}\} \ \forall n, \forall t_1 < \cdots < t_n \end{aligned}$$

Stochastic Processes IV

- That is, the future path of a Markov process, given its current state $X(t_{n-1})$ and the past history before t_{n-1}, depends only on the current state.

Markov Chains I

- A discrete-time, discrete-space Markov process is referred to as a Markov chain. In the following, let $X_n = i$, denote that the process is in state i at time n.
- Thus, we can write the transition probability $p_{i,j}$ that the process will be in state j at time n, given that it was in state i at time $n - 1$

$$p_{i,j}(n) = P\{X_n = j | X_{n-1} = i\}$$

- With the further assumption that the process is homogenous in time i.e. that the transition probabilities are independent of the time n, then the evolution of the Markov chain is determined completely by its initial state and the (one-step) transition matrix $\mathrm{P} = \{p_{i,j}\}$.

Markov Chains II

- The elements of P must satisfy

$$p_{i,j} \geq 0 \quad \text{and} \quad \sum_{j=0}^{\infty} p_{i,j} = 1, \forall i = 0, 1, \ldots$$

Chapman-Kolmogorov Equations I

- The *n*-step transition probability matrix defines the probability that a process in state *i* will be in state *j* after *n* transitions:

$$p_{i,j}^{(n)} = P\{X_{n+k} = j | X_k = i\}, n \geq 0 \ , i, j \geq 0$$

- These *n*-step probabilities can be calculated using the Chapman-Kolmogorov Equations :

$$\begin{aligned} p_{i,j}^{(n+m)} &= P\{X_{n+m} = j | X_0 = i\} \qquad (4) \\ &= \sum_{k=0}^{\infty} P\{X_{n+m} = j, X_n = k | X_0 = i\} \\ &= \sum_{k=0}^{\infty} P\{X_{n+m} = j | X_n = k, X_0 = i\} P\{X_n = k | X_0 = i\} \\ &= \sum_{k=0}^{\infty} p_{k,j}^{(m)} p_{i,k}^{(n)} \end{aligned}$$

Chapman-Kolmogorov Equations II

- This implies that $\mathrm{P}^{(2)} = \mathrm{P}.\mathrm{P} = \mathrm{P}^2$ and similarly $\mathrm{P}^{(n)} = \mathrm{P}^n$
- Let $\pi_{\mathbf{0}}$ be the initial probability distribution i.e. $\pi_{0i} = P\{X_0 = i\}$, then the unconditional probabilities $\pi_{\mathbf{n}}$, can be computed by

$$P\{X_n = j\} = \sum_{i=0}^{\infty} P\{X_n = j | X_0 = i\} P\{X_0 = i\}$$

i.e.

$$\pi_n = \pi_0 \mathrm{P}^n$$

Limiting Probabilities I

- State j is said to be accessible for state i if $P_{i,j}^{(n)} > 0$ for some $n \geq 0$, that is, if, starting in state i it is possible to eventually enter the state j. This follows since if j is not accessible from i,

$$\begin{aligned} P\{\text{ever enter } j| \text{ start in } i\} &= P\{\cup_{n=0}^{\infty} X_n = j | X_0 = i\} \quad (5) \\ &\leq \sum_{n=0}^{\infty} P\{X_n = j | X_0 = i\} \\ &= \sum_{n=0}^{\infty} P_{i,j}^{(n)} \\ &= 0. \end{aligned}$$

Limiting Probabilities II

- Two states that are accessible are said to communicate and we write $i \leftrightarrow j$. It can be checked that this relation statisfies the equivalence properties
  1. $i \leftrightarrow i$
  2. $i \leftrightarrow j \Rightarrow j \leftrightarrow i$
  3. $i \leftrightarrow j$ and $j \leftrightarrow k \Rightarrow i \leftrightarrow k$
- Thus two states that communicate are in the same equivalence class and the set of equivalence classes partitions the state space. The Markov chain is said to be irreducible is there is only one class – i.e. if all states communicate with each other.

Limiting Probabilities III

- For any state i, let f_i denote the probability that, starting in state i, the process will ever reenter state i. State i is said to be recurrent if $f_i = 1$ and transient if $f_i < 1$. Suppose the process starts in the recurrent state i. Hence, with probability 1, the process will eventually reenter i and by definition of a Markov chain, the process starts over again, and so will reenter once again. Hence, if a state i is recurrent, starting in i, the process will reenter i infinitely often.
- On the other hand, if the state i is transient, there is a positive probabiliy, $(1 - f_i)$ that it will never again enter that state. Thus starting in state i the probability that the process will be in state i for exactly n time periods is $f_i^{n-1}(1 - f_i)$ – a geometric distribution. The mean of this distribution is

$$\frac{1}{1 - f_i}\,.$$

Limiting Probabilities IV

- It follows that state i is recurrent if and only if, starting in state i, the expected number of time periods that the process is in state i is infinite. Now let

$$I_n = \begin{cases} 1, & \text{if } X_n = i \\ 0, & \text{if } X_n \neq i \end{cases}$$

  so that the number of times a process is in state i is given by $\sum_{n=0}^{\infty} I_n$.

- Thus

$$\begin{aligned} E\{\sum_{n=0}^{\infty} I_n | X_0 = i\} &= \sum_{n=0}^{\infty} E\{I_n | X_0 = i\} \qquad (6) \\ &= \sum_{n=0}^{\infty} P\{X_n = i | X_0 = i\} \\ &= \sum_{n=0}^{\infty} P_{i,i}^{(n)} \end{aligned}$$

Limiting Probabilities V

- Hence, it can be concluded that state i is recurrent if $\sum_{n=0}^{\infty} P_{i,i}^{(n)} = \infty$ and is transient if $\sum_{n=0}^{\infty} P_{i,i}^{(n)} < \infty$.
- This argument also implies that a transient state will only be visited a finite number of times and hence, in a finite Markov chain, not all states can be transient. It also follows that recurrence is a class property – that is, if $i \leftrightarrow j$ and i is recurrent, then j is recurrent.
- To see this, note that $i \leftrightarrow j$ means there exists integers k and m such that $P_{i,j}^{(k)} > 0$ and $P_{j,i}^{(m)} > 0$. For any integer n, we have

$$P_{j,j}^{(m+n+k)} \geq P_{j,i}^{(m)} P_{i,i}^{(n)} P_{i,j}^{(k)}$$

  since the lhs represents the probability of going from j to j in $m + n + k$ steps, while the rhs represents the probability of doing so, by going to i in m steps, returning to i in a further n steps and going from i to j in k steps.

Limiting Probabilities VI

- Summing over n we get

$$\sum_n P_{j,j}^{(m+n_+k)} \geq P_{j,i}^{(m)} P_{i,j}^{(k)} \sum_n P_{i,i}^{(n)} = \infty$$

  and hence j is recurrent.
- As a consequence, transience is also a class property. All states of a finite irreducible Markov chain must therefore be recurrent.
- State i is said to have period d if $P_{i,i}^{(n)} = 0$ whenever n is not divisable by d and d is the largest integer with this property. A state with period 1 is said to be aperiodic.
- Periodicity is also a class property, that is, if state i has period d and state i and j communicate then state j also has period d.

Limiting Probabilities VII

- A recurrent state i is said to be positive recurrent if, starting in i, the expected time until the process returns to state i is finite. In Markov chains with an infinite number of states, it may be the case that a recurrent state is not positive recurrent. However in a finite-state Markov chain all recurrent states are positive recurrent. Positive recurrent, aperiodic states are called ergodic.

Limiting Probabilities VIII

Theorem

For an irreducible ergodic Markov chain $\lim_{n\to\infty} P_{i,j}^{(n)}$ exists and is independent of i. Furthermore, letting

$$\pi_j = \lim_{n\to\infty} P_{i,j}^{(n)}, j \geq 0 \tag{7}$$

then π_j is the unique non-negative solution of

$$\pi_j = \sum_{i=0}^{\infty} \pi_i P_{i,j}$$

In the irreducible, positive recurrent, periodic case, we still have that π_j are the unique solution of (8), such that $\sum_j \pi_j = 1$, but π_j must be interpreted as the long-run proportion of time that the Markov chain is in state j.

Convergence of SA I

- For a fixed value of the temperature parameter, T, the sequence of states that the SA algorithm generates form a **homogeneous Markov chain**.
  - Provided the neighbourhood structure satisfies conditions to make this chain irreducible and aperiodic, then it converges to a stationary probability distribution.
- To examine what this stationary distribution is, we need to write down the transition matrix of the Markov chain.
  - From any state i, the SA algorithm selects a candidate neighbouring state j.
  - The neighbour may be chosen with equal probability, or more generally, some neighbours may be preferred more than others. Let $g(i,j)$ be the preference for neighbour j from neighbour i. Furthermore, let
$$g(i) \triangleq \sum_{j \in N(i)} g(i,j)$$
    be a normalising constant, so that $g(i,j)/g(i)$ is the probability of choosing neighbour j.

Convergence of SA II

- Once a neighbour is chosen, that neighbour is accepted and the chain moves to that state, when
  - the cost is lower, *or*
  - with probability depending on T if the cost is higher.
- Putting the conditions together, we get the transition value:

$$p_{i,j} \triangleq \Pr[X_n = j | X_{n-1} = i] = \frac{g(i,j)}{g(i)} \min(1, \exp(-(f(j) - f(i))/T))$$

- If the neighbour is not accepted, then the chain remains at state i, so that

$$p_{i,i} = 1 - \sum_{j \in N(i)} p_{i,j}$$

Convergence of SA III

- It may be shown that provided the neighbourhood structure is symmetric, i.e. $i \in N(j) \Rightarrow j \in N(i)$ and $g(i,j) = g(j,i)$, for $i \in N(j)$, then the limiting distribution of the chain is given by

$$\pi_i(T) = \frac{g(i)}{G(T)} \exp\left(-\frac{f(i)}{T}\right) ,$$

  where $G(T)$ is a normalising constant:

$$G(T) \triangleq \sum_i g(i) \exp\left(-\frac{f(i)}{T}\right) .$$

- All that is required is to show that $\pi_i(T)$ is a left eigenvector of the transition matrix, with eigenvalue 1.

Convergence of SA IV

- When this is the case, we must have

$$\begin{aligned}
\pi_j &= \sum_i \pi_i p_{i,j} \\
&= \pi_j p_{j,j} + \sum_{i \in N(j)} \pi_i p_{i,j} \\
(1 - p_{j,j})\pi_j &= \sum_{i \in N(j)} \pi_i p_{i,j} \\
(1 - (1 - \sum_{k \in N(j)} p_{j,k}))\pi_j &= \sum_{i \in N(j)} \pi_i p_{i,j} \\
\sum_{k \in N(j)} \pi_j p_{j,k} &= \sum_{i \in N(j)} \pi_i p_{i,j}
\end{aligned}$$

Convergence of SA V

To check that this holds, we compute

$$
\begin{aligned}
\sum_{k\in N(j)} \pi_j p_{j,k} &= \sum_{k\in N(j), f(j)<f(k)} \pi_j p_{j,k} + \sum_{k\in N(j), f(j)>f(k)} \pi_j p_{j,k} \\
&= \sum_{k\in N(j), f(j)<f(k)} \frac{g(j)}{G(T)} \frac{g(j,k)}{g(j)} \exp\left(-\frac{f(j)}{T}\right) \exp\left(-\frac{f(k)-f(j)}{T}\right) \\
&+ \sum_{k\in N(j), f(j)>f(k)} \frac{g(j)}{G(T)} \frac{g(j,k)}{g(j)} \exp\left(-\frac{f(j)}{T}\right) \\
&= \sum_{k\in N(j), f(j)<f(k)} \frac{g(j,k)}{G(T)} \exp\left(-\frac{f(k)}{T}\right) \\
&+ \sum_{k\in N(j), f(j)>f(k)} \frac{g(j,k)}{G(T)} \exp\left(-\frac{f(j)}{T}\right)
\end{aligned}
$$

Convergence of SA VI

$$
\begin{aligned}
&= \sum_{k\in N(j), f(j)<f(k)} \frac{g(j,k)}{g(k)} .1 . \frac{g(k)}{G(T)} \exp\left(-\frac{f(k)}{T}\right) \\
&+ \sum_{k\in N(j), f(j)>f(k)} \frac{g(j,k)}{g(k)} \exp\left(-\frac{f(j-f(k))}{T}\right) \frac{g(k)}{G(T)} \exp\left(-\frac{f(k)}{T}\right) \\
&= \sum_{k\in N(j), f(j)<f(k)} \pi_k p_{k,j} + \sum_{k\in N(j), f(j)>f(k)} \pi_k p_{k,j} \qquad (8) \\
&= \sum_{k\in N(j)} \pi_k p_{k,j}
\end{aligned}
$$

- So $\pi_i(T)$ is the stationary distribution of the homogeneous Markov chain, with fixed temperature value T.

Convergence of SA VII

- Also,

$$\lim_{T\downarrow 0} \pi_i(T) = \begin{cases} \frac{g(i)}{\sum_{j\in S_{\mathrm{opt}}} g(j)} & \text{if } i \in S_{\mathrm{opt}} \\ 0 & \text{otherwise} \end{cases}$$

Since, (multiplying top and bottom by $\exp(f_{\mathrm{opt}}/T)$),

$$\lim_{T\downarrow 0} g(i) \frac{\exp(-\frac{f(i)}{T})}{\sum_{j\in S} g(j)\exp(-\frac{f(j)}{T})} \quad = \quad \lim_{T\to 0} g(i) \frac{\exp(-\frac{f(i)-f_{\mathrm{opt}}}{T})}{\sum_{j\in S} g(j)\exp(-\frac{f(j)-f_{\mathrm{opt}}}{T})}$$

then, for $i \in S_{\mathrm{opt}}$,

$$\lim_{T\downarrow 0} \pi_i(T) = \lim_{T\to 0} g(i) \frac{\exp(-\frac{0}{T})}{\sum_{j\in S/S_{\mathrm{opt}}} g(j)\exp(-\frac{f(j)-f_{\mathrm{opt}}}{T}) + \sum_{j\in S_{\mathrm{opt}}} g(j)\exp(-\frac{0}{T})}$$

and the result follows since $\lim_{T\downarrow 0}\exp(-a/T) = 0$ for $a > 0$ and $\lim_{T\downarrow 0}\exp(-a/T) = 1$ for $a = 0$.

Convergence of SA VIII

- The general SA process forms an <u>inhomogeneous</u> Markov chain, since the temperature is changed as the iteration proceeds.
- Convergence of the inhomogeneous chain to the optimal set has been shown to hold for annealing schedules of the form

$$T_m = \frac{\gamma}{\log(m + m_0 + 1)}, m = 0, 1, 2, \ldots$$

  where $1 \leq m_0 < \infty$ is an arbitrary parameter and $\gamma \geq rL$ where r is the radius of the graph underlying the Markov chain and L is depends on the cost function and is given by

$$L \triangleq \max_{i \in S} \max_{j \in N(i)} |f(j) - f(i)|.$$

Convergence of Inhomogeneous Chain I

- This convergence is demonstrated by firstly showing that the inhomogeneous chain is weakly ergodic under the above cooling schedule. The existance of the unique left eigenvector $\pi_i(T_m)$ as given above then provides a sufficient condition to conclude (strong) ergodicity.
- Write the cooling schedule as a sequence of temperatures $\{T_n\}$, so that a simulated annealing process can be represented by the triple $(S, f, \{T_n\})$.
- The cooling schedule $\{T_n\}$ is said to be asympototically good if the sequences of solutions that it generate $\{X_n\}$ satisfies

$$\lim_{n\to\infty} \mathrm{Pr}[X_n \in S_{\mathrm{opt}}] = 1$$

where S_{opt} is the set of optimal states.

Convergence of Inhomogeneous Chain II

- It can be shown that if

$$\lim_{n\to\infty} T_n \log n \geq R > 0$$

for R "large enough", then $\{T_n\}$ is asymptotically good.

- All asymptotically good logarithmic annealing schedules have speed of convergence given by

$$\Pr[X_n \notin S_{\mathrm{opt}}] \approx \left(\frac{k}{n}\right)^{\alpha}$$

for n large enough, $k > 0, \alpha > 0$.

- Unfortunately, α can be small and can be made arbitrarily small for perversely selected problems.

Genetic Algorithms I

- Genetic Algorithms (GAs) are an optimization technique for functions defined over finite (discrete) domains.
- GAs are applied to a problem as follows :
  1. The search space of all possible solutions of the problem is mapped onto a set of finite strings over a finite (generally small) alphabet. That is, an encoding is chosen, such that each point in the search space is represented by exactly one string, called a chromosome. The GA works with these representations of solutions, rather than with the solutions themselves.
  2. An initial population of solutions is selected. This first generation is usually selected at random. Unlike standard optimization techniques, a GA performs a parallel search over a set of points in the search space, thus lessening the probability of being stuck in a local optimum.

Genetic Algorithms II

3. A fitness is computed for each of the individuals in the population, reflecting the way each individual is, in comparison to the others, nearer to the optimum. This value expresses the observed quality of the solution each individual represents.
4. The more fit individuals are selected according to a noisy selection, i.e. individuals are selected randomly, but with probability increasing with fitness. The GAs are thus essentially a randomized optimization technique.
5. The selected individuals form the parent set – they are crossed over (in pairs) to produce their progeny. A crossover consists of joining together non-corresponding bits of each parent in order to constitute two new individuals.
6. Another noisy selection is performed, this time biased towards the less fit individuals. These are replaced by the progeny obtained in the previous step. Unlike standard optimization techniques, the GAs proceed by replacing the weak part of a population with new individuals, rather than replacing the current best solution with a new candidate.

Genetic Algorithms III

7. A small part of the resulting population is <u>mutated</u> i.e. small random changes are applied to a few randomly selected individuals. In some GA applications, a small randomly chosen portion of the population is also subject to another genetic operator, the inversion — genes, while retaining their meaning, change their position inside the chromosome.
8. At this point, a new population has been constituted and the optimization process is repeated from 3 for the next generation.

Genetic Algorithms IV

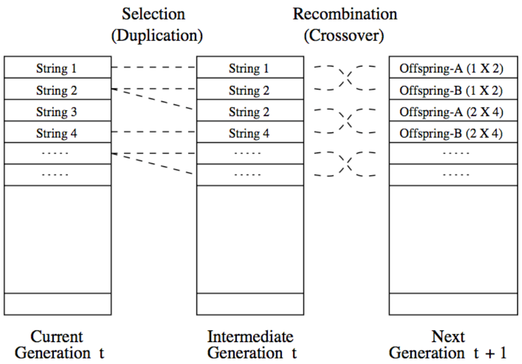

Figure: One generation is broken down into a selection phase and recombination phase. This figure shows strings being assigned into adjacent slots during selection. In fact they can be assigned slots randomly in order to shuffle the intermediate population. Mutation is not shown but can be applied after crossover

Canonical Genetic Algorithm I

- The first step is to generate a population of some size (number of members) P. Each member of this population is a binary string of length L which corresponds to the problem encoding.
- The evaluation function computes the value of the objective to be optimised.
- The fitness is defined with respect to the rest of the population by: $f_i/\bar{f}$ where f_i is the evaluation associated with string i and $\bar{f}$ the average evaluation of all strings in the population.
- Computation of the next generation can be considered as a 2-stage process:
  1. Selection is applied to the current population to create an intermediate population.
  2. Crossover (recombination) and mutation are applied to the intermediate population to create the next population

Selection I

- **Method 1** View population as mapping on to a roulette wheel, where each individual is represented by a space that proportionally corresponds to its fitness. Individuals are chosen using "stochastic sampling with replacement" to fill the intermediate population.
- **Method 2**: For each string i where $f_i/\bar{f} > 1$, the integer portion indicates how many copies of that string are directly placed in the intermediate population. All strings (including those with fitness ≤ 1 then place additional copies in the population with probability proportional to the fractional part of $f_i/\bar{f}$.

Crossover I

- Crossover is applied to randomly paired strings with probability p_c.
- Consider the string 1101001100101101 and another binary string yxyyxyxxyyyxyxxy in which the values 1 and 0 are denoted by x and y. Using a single randomly chosen recombination point **1-point** crossover occurs as follows

$$\begin{array}{ll} 11010 & \backslash/01100101101 \\ yxyyx & \backslash/yxxyyyxyxxy \end{array}$$

- Swapping the fragments between the two parents produces the following offspring

$$11010yxxyyyxyxxy \quad \text{and } yxyyx01100101101$$

Mutation

- For each bit in the population, mutate with some low probability p_m, usually less than 1%
- Sometimes interpreted as randomly generating a new bit – in which case the bit value will only change 50% of the time.
- Otherwise interpreted as ‘flipping’ the bit value.

Representation for Graph Partitioning I

- The graph partitioning problem maps easily to a binary string representation. We can use a string of length $L = n$, the number of vertices in the graph.
- String value 1 implies that the corresponding vertex is in P_1 and string value 0 implies the corresponding vertex is in P_0.
- However, note that the ordering of the sting is completely arbitrary – depends on the labelling of the vertices in the graph.
- The effect of the 1-point crossover operator is highly dependent on the string ordering – its effect is to maintain groups of consecutively numbered vertices in the partition assignment of their parents – but if vertices are numbered randomly, this may make no intuitive sense at all.
- Note also the the 1-point crossover may generate <u>infeasible</u> states i.e. states where # ones $\neq$ # zeros.

Representation for Graph Partitioning II

- It therefore makes sense to move away from the canonical algorithm and choose a crossover that has a better probability of choosing high-quality children.
- One possibility is to maintain in the children the partition assignments of the vertices that both parents agree on and fill out the remainder in some way that at least preserves feasibility.
  - A basic approach is to fill the remaining vertices randomly (while ensuring feasibility).
  - A more sophisticated approach might try to choose the remaining values in a manner that tries to maximise the quality of the resulting child(ren) e.g. assign the vertices in order of maximum gain – where gain is the reduction in edge-cut resulting from the assignment of the vertex to a particular partition. This sort of approach is sometimes called a memetic algorithm.