Local Search Algorithms

Todd Ebert

Outline

1. Introduction
2. Markov-Chains
3. Simulated Annealing
4. Genetic Algorithms
5. WalkSAT Algorithm

State Spaces

The Set of Possible Assignments

State Spaces

The Set of Possible Assignments

- **Review.** Given constraint model $M = (V, D, C)$, $\mathcal{A}(V)$ denotes the set of all possible assignments a over V.

State Spaces

The Set of Possible Assignments

- **Review.** Given constraint model $M = (V, D, C)$, $\mathcal{A}(V)$ denotes the set of all possible assignments a over V.
- **Cartesian Product Representation:** $\mathcal{A}(V) = D_1 \times D_2 \times \cdots \times D_n$.

State Spaces

The Set of Possible Assignments

- **Review.** Given constraint model $M = (V, D, C)$, $\mathcal{A}(V)$ denotes the set of all possible assignments a over V.
- **Cartesian Product Representation:** $\mathcal{A}(V) = D_1 \times D_2 \times \cdots \times D_n$.
- $\mathcal{A}(V)$ is called the **state space** or **solution space**, since elements of $\mathcal{A}(V)$, i.e. assignments, are candidates for providing a model solution.

State Space Example: Finding a Hamilton Path

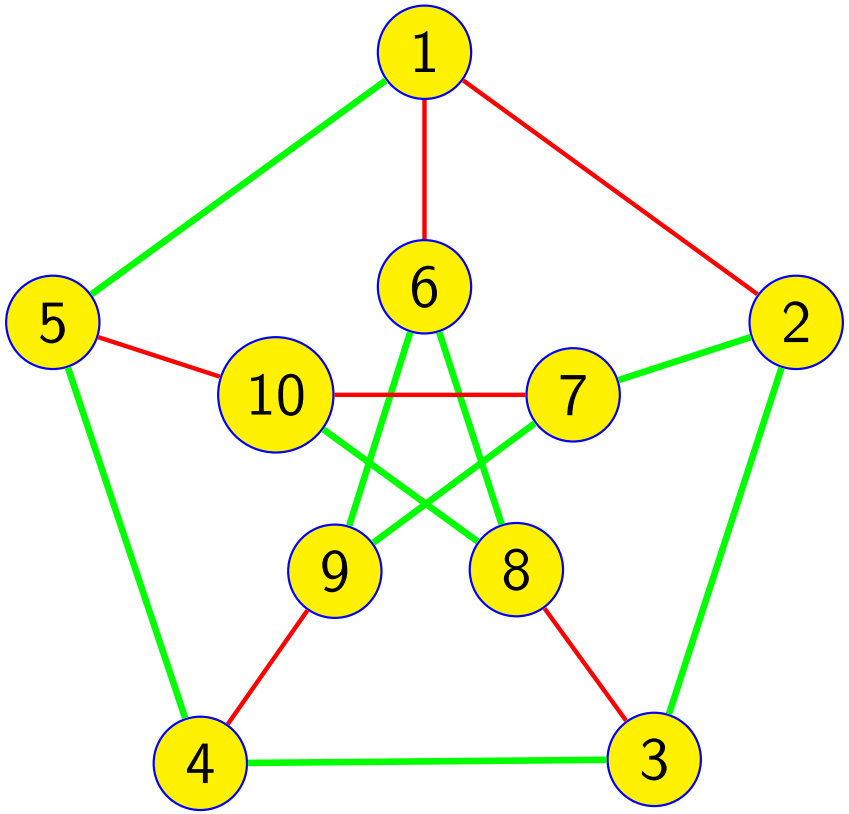


Figure: Hamilton Path (green edges) for the Peterson Graph

State Space Example: Finding a Hamilton Path

Problem Model for Graph $G = (\{1, \ldots, n\}, E)$

State Space Example: Finding a Hamilton Path

Problem Model for Graph $G = (\{1, \ldots, n\}, E)$

- **Variables.** Boolean x_{ij}, for each $(i, j) \in E$.

State Space Example: Finding a Hamilton Path

Problem Model for Graph $G = (\{1, \ldots, n\}, E)$

- **Variables.** Boolean x_{ij}, for each $(i, j) \in E$.
- **Tree Constraint.**
  Tree$(\{1, \ldots, n\}, \{(i, j) | x_{ij}$ is assigned true$\})$. In other words, the graph whose vertices are $\{1, \ldots, n\}$, and whose edges are those edges $(i, j) \in E$ for which x_{ij} is assigned true, must have a tree structure.

State Space Example: Finding a Hamilton Path

Problem Model for Graph $G = (\{1, \ldots, n\}, E)$

- **Variables.** Boolean x_{ij}, for each $(i, j) \in E$.
- **Tree Constraint.**
  $\text{Tree}(\{1, \ldots, n\}, \{(i, j) | x_{ij} \text{ is assigned true}\})$. In other words, the graph whose vertices are $\{1, \ldots, n\}$, and whose edges are those edges $(i, j) \in E$ for which x_{ij} is assigned true, must have a tree structure.
- **Degree Bound Constraints.** For each $i = 1, \ldots, n$,
  $$\sum_{\{j | (i,j) \in E\}} x_{ij} \leq 2.$$

State Space Example: Finding a Hamilton Path

Problem Model for Graph $G = (\{1, \ldots, n\}, E)$

- **Variables.** Boolean x_{ij}, for each $(i, j) \in E$.
- **Tree Constraint.**
  $\text{Tree}(\{1, \ldots, n\}, \{(i, j) | x_{ij} \text{ is assigned true}\})$. In other words, the graph whose vertices are $\{1, \ldots, n\}$, and whose edges are those edges $(i, j) \in E$ for which x_{ij} is assigned true, must have a tree structure.
- **Degree Bound Constraints.** For each $i = 1, \ldots, n$,
  $$\sum_{\{j | (i,j) \in E\}} x_{ij} \leq 2.$$
- **State Space.** $\{0, 1\}^m$, where $m = |E|$; i.e. the set of binary strings of length m.

State-Space Graphs

Graph Terminology for State Space S

State-Space Graphs

Graph Terminology for State Space S

- **Neighborhood function:** $\nu : S \rightarrow \text{subset}(S)$ maps each state to a subset of **neighboring states**, called the **neighborhood** of S.

State-Space Graphs

Graph Terminology for State Space S

- **Neighborhood function:** $\nu : S \rightarrow \text{subset}(S)$ maps each state to a subset of **neighboring states**, called the **neighborhood** of S.
- **State-space graph** $G(S)$**:** A graph whose vertex set is S and whose edge set is $\{(s,t) | s \in S \text{ and } t \in \nu(s)\}$

State-Space Graphs

Graph Terminology for State Space S

- **Neighborhood function:** $\nu : S \rightarrow \text{subset}(S)$ maps each state to a subset of **neighboring states**, called the **neighborhood** of S.
- **State-space graph** $G(S)$**:** A graph whose vertex set is S and whose edge set is $\{(s,t) | s \in S \text{ and } t \in \nu(s)\}$
- **Local path:** a sequence of states $s_0, s_1, \ldots, s_k \in S$, where $s_i \in \nu(s_{i-1})$, for all $i = 1, \ldots, k$. In other words, a directed path of $G(S)$.

State-Space Graphs

Graph Terminology for State Space S

- **Neighborhood function:** $\nu : S \rightarrow \text{subset}(S)$ maps each state to a subset of **neighboring states**, called the **neighborhood** of S.
- **State-space graph** $G(S)$**:** A graph whose vertex set is S and whose edge set is $\{(s,t) | s \in S \text{ and } t \in \nu(s)\}$
- **Local path:** a sequence of states $s_0, s_1, \ldots, s_k \in S$, where $s_i \in \nu(s_{i-1})$, for all $i = 1, \ldots, k$. In other words, a directed path of $G(S)$.
- **Solution state:** a state $s \in S$ (when viewed as a variable assignment) that satisfies all model constraints.

State-Space Graphs

Graph Terminology for State Space S

- **Neighborhood function:** $\nu : S \rightarrow \text{subset}(S)$ maps each state to a subset of **neighboring states**, called the **neighborhood** of S.
- **State-space graph** $G(S)$**:** A graph whose vertex set is S and whose edge set is $\{(s,t) | s \in S \text{ and } t \in \nu(s)\}$
- **Local path:** a sequence of states $s_0, s_1, \ldots, s_k \in S$, where $s_i \in \nu(s_{i-1})$, for all $i = 1, \ldots, k$. In other words, a directed path of $G(S)$.
- **Solution state:** a state $s \in S$ (when viewed as a variable assignment) that satisfies all model constraints.
- **Local search:** searching for solution states within S by traversing $G(S)$ along its edges.

State-Space Graph Example

Problem Definition

State-Space Graph Example

Problem Definition

- **Variables:** Boolean x, y, z

State-Space Graph Example

Problem Definition

- **Variables:** Boolean x, y, z
- **Constraint:** $x \wedge y \wedge z$

State-Space Graph Example

Problem Definition

- **Variables:** Boolean x, y, z
- **Constraint:** $x \wedge y \wedge z$
- **State Space:** $\{0, 1\}^3$

State-Space Graph Example

Problem Definition

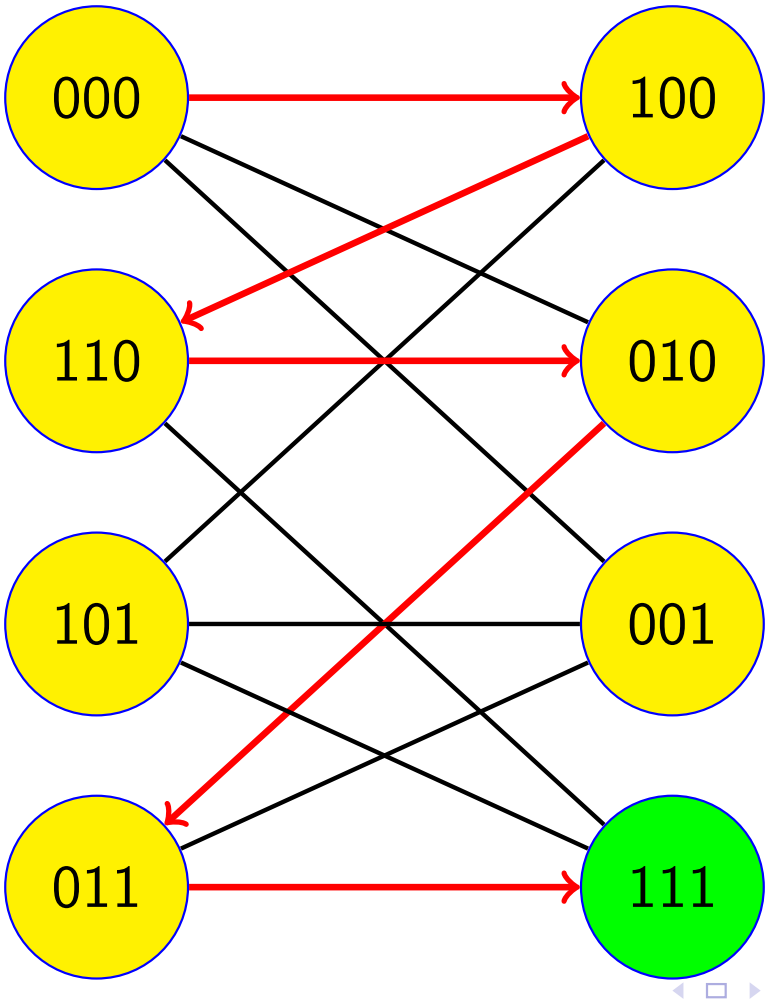
- **Variables:** Boolean x, y, z
- **Constraint:** $x \wedge y \wedge z$
- **State Space:** $\{0, 1\}^3$
- **Neighborhood function:** $\nu(s)$ is the set of strings that differ in one bit place with s.

State-Space Graph Example

Problem Definition

- **Variables:** Boolean x, y, z
- **Constraint:** $x \wedge y \wedge z$
- **State Space:** $\{0, 1\}^3$
- **Neighborhood function:** $\nu(s)$ is the set of strings that differ in one bit place with s.
- **Solution state:** $s = (x = 1, y = 1, z = 1)$

State-Space Graph Example



Transition Models

A **transition model** for a state-space graph is a means for determining the next state of a local-search path given the current state.

Transition Models

A **transition model** for a state-space graph is a means for determining the next state of a local-search path given the current state.

Markov-Chain Transition Models

Transition Models

A **transition model** for a state-space graph is a means for determining the next state of a local-search path given the current state.

Markov-Chain Transition Models

- **Markov-Chain Transition Model**. The next state only depends on knowledge of the current state s according to some fixed conditional probability distribution $p(|s)$ over $\nu(s)$.

Transition Models

A **transition model** for a state-space graph is a means for determining the next state of a local-search path given the current state.

Markov-Chain Transition Models

- **Markov-Chain Transition Model**. The next state only depends on knowledge of the current state s according to some fixed conditional probability distribution $p(|s)$ over $\nu(s)$.
- $p(y|x)$**:** probability of transitioning to state $y \in \nu(x)$ on condition current state is x.

Transition Models

A **transition model** for a state-space graph is a means for determining the next state of a local-search path given the current state.

Markov-Chain Transition Models

- **Markov-Chain Transition Model**. The next state only depends on knowledge of the current state s according to some fixed conditional probability distribution $p(|s)$ over $\nu(s)$.
- $p(y|x)$**:** probability of transitioning to state $y \in \nu(x)$ on condition current state is x.
- $p(x \to y)$**:** weight assigned to state $y \in \nu(x)$ to reflect the likelihood of transitioning from x to y. $p(x \to)$ not necessarily a probability distribution.

Transition Models

A **transition model** for a state-space graph is a means for determining the next state of a local-search path given the current state.

Markov-Chain Transition Models

- **Markov-Chain Transition Model**. The next state only depends on knowledge of the current state s according to some fixed conditional probability distribution $p(|s)$ over $\nu(s)$.
- $p(y|x)$**:** probability of transitioning to state $y \in \nu(x)$ on condition current state is x.
- $p(x \to y)$**:** weight assigned to state $y \in \nu(x)$ to reflect the likelihood of transitioning from x to y. $p(x \to)$ not necessarily a probability distribution.
- π_0**:** a probability distribution over S where $\pi_0(s)$ is the probability that s will be chosen as the initial state of a local-search path.

Transition Models

A **transition model** for a state-space graph is a means for determining the next state of a local-search path given the current state.

Markov-Chain Transition Models

- **Markov-Chain Transition Model**. The next state only depends on knowledge of the current state s according to some fixed conditional probability distribution $p(|s)$ over $\nu(s)$.
- $p(y|x)$**:** probability of transitioning to state $y \in \nu(x)$ on condition current state is x.
- $p(x \to y)$**:** weight assigned to state $y \in \nu(x)$ to reflect the likelihood of transitioning from x to y. $p(x \to)$ not necessarily a probability distribution.
- π_0**:** a probability distribution over S where $\pi_0(s)$ is the probability that s will be chosen as the initial state of a local-search path.
- **Time homogeneous (or stationary) Markov chain**. The transition probabilities stay constant over time.

Transition Models

A **transition model** for a state-space graph is a means for determining the next state of a local-search path given the current state.

Markov-Chain Transition Models

- **Markov-Chain Transition Model**. The next state only depends on knowledge of the current state s according to some fixed conditional probability distribution $p(|s)$ over $\nu(s)$.
- $p(y|x)$**:** probability of transitioning to state $y \in \nu(x)$ on condition current state is x.
- $p(x \to y)$**:** weight assigned to state $y \in \nu(x)$ to reflect the likelihood of transitioning from x to y. $p(x \to)$ not necessarily a probability distribution.
- π_0**:** a probability distribution over S where $\pi_0(s)$ is the probability that s will be chosen as the initial state of a local-search path.
- **Time homogeneous (or stationary) Markov chain**. The transition probabilities stay constant over time.
- **Path probability.** Given $L = x_0, x_1, \ldots, x_n$, $p(L) = \pi_0(x_0)p(x_1|x_0)\cdots p(x_n|x_{n-1})$.

Transition Models

A **transition model** for a state-space graph is a means for determining the next state of a local-search path given the current state.

Markov-Chain Transition Models

- **Markov-Chain Transition Model**. The next state only depends on knowledge of the current state s according to some fixed conditional probability distribution $p(|s)$ over $\nu(s)$.
- $p(y|x)$**:** probability of transitioning to state $y \in \nu(x)$ on condition current state is x.
- $p(x \to y)$**:** weight assigned to state $y \in \nu(x)$ to reflect the likelihood of transitioning from x to y. $p(x \to)$ not necessarily a probability distribution.
- π_0**:** a probability distribution over S where $\pi_0(s)$ is the probability that s will be chosen as the initial state of a local-search path.
- **Time homogeneous (or stationary) Markov chain**. The transition probabilities stay constant over time.
- **Path probability.** Given $L = x_0, x_1, \ldots, x_n$, $p(L) = \pi_0(x_0)p(x_1|x_0)\cdots p(x_n|x_{n-1})$.
- **Stochastic Search.** The general term used to describe search techniques where state changes are based on random factors.

Transition Model Examples

Random Walk

For a **random walk search**, $\nu(x)$ is given the uniform distribution. For example, if $|\nu(x)| = m$, then $p(y|x) = 1/m$, for all $y \in \nu(x)$.

Transition Model Examples

Random Walk

For a **random walk search**, $\nu(x)$ is given the uniform distribution. For example, if $|\nu(x)| = m$, then $p(y|x) = 1/m$, for all $y \in \nu(x)$.

Hill Climbing

Let $H : S \to \mathcal{R}$ a function such that $H(s) \geq 0$ measures how close s is to a solution state (the lower the measure, the closer to a solution state). Then **hill climbing** local search assigns $p(y|x) = 1$, where $y = \underset{z \in \nu(x)}{\operatorname{argmin}}(h(z))$. In other words, then next state is chosen as the one measured closest to a solution state.

Outline

1 Introduction

2 Markov-Chains

3 Simulated Annealing

4 Genetic Algorithms

5 WalkSAT Algorithm

Viewing Stochastic Local Search with Markov-Chains

Markov-Chain State Transition Model

Given a finite set of states $\{1, \ldots, n\}$, a **Markov-chain state-transition model** is an $n \times n$ matrix P, where entry P_{ij} is the probability of transitioning from state i to state j.

Viewing Stochastic Local Search with Markov-Chains

Markov-Chain State Transition Model

Given a finite set of states $\{1, \ldots, n\}$, a **Markov-chain state-transition model** is an $n \times n$ matrix P, where entry P_{ij} is the probability of transitioning from state i to state j.

Markov-chain Example

Viewing Stochastic Local Search with Markov-Chains

Markov-Chain State Transition Model

Given a finite set of states $\{1, \ldots, n\}$, a **Markov-chain state-transition model** is an $n \times n$ matrix P, where entry P_{ij} is the probability of transitioning from state i to state j.

Markov-chain Example

- **States:** $\{1 = \text{no rain}, 2 = \text{rain}\}$.

Viewing Stochastic Local Search with Markov-Chains

Markov-Chain State Transition Model

Given a finite set of states $\{1, \ldots, n\}$, a **Markov-chain state-transition model** is an $n \times n$ matrix P, where entry P_{ij} is the probability of transitioning from state i to state j.

Markov-chain Example

- **States:** $\{1 = \text{no rain}, 2 = \text{rain}\}$.
- **State Transition:** moving from one day to the next.

Viewing Stochastic Local Search with Markov-Chains

Markov-Chain State Transition Model

Given a finite set of states $\{1, \ldots, n\}$, a **Markov-chain state-transition model** is an $n \times n$ matrix P, where entry P_{ij} is the probability of transitioning from state i to state j.

Markov-chain Example

- **States:** $\{1 = \text{no rain}, 2 = \text{rain}\}$.
- **State Transition:** moving from one day to the next.
- **State-Transition matrix.**

$$P = \begin{pmatrix} 0.8 & 0.2 \\ 0.5 & 0.5 \end{pmatrix}$$

Markov-Chains Can Use Both Past and Present

Markov-chain Example

Markov-Chains Can Use Both Past and Present

Markov-chain Example

- **States:**
  $\{(\text{no rain, no rain}), (\text{no rain, rain}), (\text{rain, no rain}), (\text{rain,rain})\}$.

Markov-Chains Can Use Both Past and Present

Markov-chain Example

- **States:**
  $\{(\text{no rain, no rain}), (\text{no rain, rain}), (\text{rain, no rain}), (\text{rain,rain})\}$.
- **State Interpretation.** For example, (no rain, rain) means "no rain yesterday, but rain today".

Markov-Chains Can Use Both Past and Present

Markov-chain Example

- **States:**
  {(no rain, no rain), (no rain, rain), (rain, no rain), (rain,rain)}.
- **State Interpretation.** For example, (no rain, rain) means "no rain yesterday, but rain today".

State-Transition Matrix P

	(nr, nr)	(nr, r)	(r, nr)	(r,r)
(nr,nr)	0.85	0.15	0	0
(nr, r)	0	0	0.6	0.4
(r, nr)	0.65	0.35	0	0
(r, r)	0	0	0.7	0.3

Predicting Further into the Future

t-Step Transition Matrix P^t

The t-**step transition matrix** P^t is defined so that P^t_{ij} represents the probability of being in state j t steps after being in state i.

Predicting Further into the Future

t-Step Transition Matrix P^t

The t-**step transition matrix** P^t is defined so that P^t_{ij} represents the probability of being in state j t steps after being in state i.

Proposition 1

$P^t = P \cdot P^{(t-1)}$. In other words, the t-step transition matrix is obtained by multiplying the one-step matrix with the $(t-1)$-step matrix.

Proof of Proposition 1: $t = 2$ Basis Step

Let S_i, $i = 0, 1, 2, \ldots$, be the current state at time i. Then for $t = 2$,

$$P_{ij}^2 = p(S_2 = j | S_0 = i) =$$

$$\sum_{k=1}^{n} p(S_2 = j | S_1 = k, S_0 = i) p(S_1 = k | S_0 = i) =$$

$$\sum_{k=1}^{n} p(S_2 = j | S_1 = k) p(S_1 = k | S_0 = i) = \sum_{k=1}^{n} P_{ik} P_{kj},$$

which is obtained by taking the inner product of row i of P with column j of P. Thus, $P^2 = P \cdot P$.

Proof of Proposition 1: Inductive Step

Now assume the result holds for some $t \geq 2$. We show that it is also true for $t+1$.

$$P_{ij}^{(t+1)} = p(S_{(t+1)} = j | S_0 = i) =$$

$$\sum_{k=1}^{n} p(S_{(t+1)} = j | S_t = k, S_0 = i) p(S_t = k | S_0 = i) =$$

$$\sum_{k=1}^{n} p(S_{(t+1)} = j | S_t = k) p(S_t = k | S_0 = i) = \sum_{k=1}^{n} P_{ik} P_{kj}^{t},$$

which is obtained by taking the inner product of row i of P with column j of P^t. Thus, $P^{(t+1)} = P \cdot P^t$, and the proposition is proved by induction on t.

t-Step Transition Weather Example

$$P^2 = \begin{pmatrix} 0.8 & 0.2 \\ 0.5 & 0.5 \end{pmatrix} \begin{pmatrix} 0.8 & 0.2 \\ 0.5 & 0.5 \end{pmatrix} = \begin{pmatrix} 0.74 & 0.26 \\ 0.65 & 0.35 \end{pmatrix}$$

t-Step Transition Weather Example

$$P^2 = \begin{pmatrix} 0.8 & 0.2 \\ 0.5 & 0.5 \end{pmatrix} \begin{pmatrix} 0.8 & 0.2 \\ 0.5 & 0.5 \end{pmatrix} = \begin{pmatrix} 0.74 & 0.26 \\ 0.65 & 0.35 \end{pmatrix}$$

$$P^4 = \begin{pmatrix} 0.74 & 0.26 \\ 0.65 & 0.35 \end{pmatrix} \begin{pmatrix} 0.74 & 0.26 \\ 0.65 & 0.35 \end{pmatrix} = \begin{pmatrix} 0.7166 & 0.2834 \\ 0.7085 & 0.2915 \end{pmatrix}$$

t-Step Transition Weather Example

$$P^2 = \begin{pmatrix} 0.8 & 0.2 \\ 0.5 & 0.5 \end{pmatrix} \begin{pmatrix} 0.8 & 0.2 \\ 0.5 & 0.5 \end{pmatrix} = \begin{pmatrix} 0.74 & 0.26 \\ 0.65 & 0.35 \end{pmatrix}$$

$$P^4 = \begin{pmatrix} 0.74 & 0.26 \\ 0.65 & 0.35 \end{pmatrix} \begin{pmatrix} 0.74 & 0.26 \\ 0.65 & 0.35 \end{pmatrix} = \begin{pmatrix} 0.7166 & 0.2834 \\ 0.7085 & 0.2915 \end{pmatrix}$$

Interpretation of P^4

t-Step Transition Weather Example

$$P^2 = \begin{pmatrix} 0.8 & 0.2 \\ 0.5 & 0.5 \end{pmatrix} \begin{pmatrix} 0.8 & 0.2 \\ 0.5 & 0.5 \end{pmatrix} = \begin{pmatrix} 0.74 & 0.26 \\ 0.65 & 0.35 \end{pmatrix}$$

$$P^4 = \begin{pmatrix} 0.74 & 0.26 \\ 0.65 & 0.35 \end{pmatrix} \begin{pmatrix} 0.74 & 0.26 \\ 0.65 & 0.35 \end{pmatrix} = \begin{pmatrix} 0.7166 & 0.2834 \\ 0.7085 & 0.2915 \end{pmatrix}$$

Interpretation of P^4

- If it is not raining today, then there is a 71.66% chance of no rain in 4 days.

t-Step Transition Weather Example

$$P^2 = \begin{pmatrix} 0.8 & 0.2 \\ 0.5 & 0.5 \end{pmatrix} \begin{pmatrix} 0.8 & 0.2 \\ 0.5 & 0.5 \end{pmatrix} = \begin{pmatrix} 0.74 & 0.26 \\ 0.65 & 0.35 \end{pmatrix}$$

$$P^4 = \begin{pmatrix} 0.74 & 0.26 \\ 0.65 & 0.35 \end{pmatrix} \begin{pmatrix} 0.74 & 0.26 \\ 0.65 & 0.35 \end{pmatrix} = \begin{pmatrix} 0.7166 & 0.2834 \\ 0.7085 & 0.2915 \end{pmatrix}$$

Interpretation of P^4

- If it is not raining today, then there is a 71.66% chance of no rain in 4 days.
- If it is raining today, then there is a 29.15% chance of rain in 4 days.

Ergodic Markov-Chains

Some Convenient State Graph Properties for $G(S)$

Ergodic Markov-Chains

Some Convenient State Graph Properties for $G(S)$

- **Finite States.** S is said to be a **finite** state space iff $|S| < \infty$.

Ergodic Markov-Chains

Some Convenient State Graph Properties for $G(S)$

- **Finite States.** S is said to be a **finite** state space iff $|S| < \infty$.
- **Undirected.** $G(S)$ is said to be **undirected** iff, for all states $s, t \in S$, $t \in \nu(s)$ and $p(t|s) > 0$ iff $s \in \nu(t)$ and $p(s|t) > 0$.

Ergodic Markov-Chains

Some Convenient State Graph Properties for $G(S)$

- **Finite States.** S is said to be a **finite** state space iff $|S| < \infty$.
- **Undirected.** $G(S)$ is said to be **undirected** iff, for all states $s, t \in S$, $t \in \nu(s)$ and $p(t|s) > 0$ iff $s \in \nu(t)$ and $p(s|t) > 0$.
- **Self Transitioning.** $G(S)$ is said to be self-transitioning iff, for all $s \in S$, $s \in \nu(s)$ and $p(s|s) > 0$.

Ergodic Markov-Chains

Some Convenient State Graph Properties for $G(S)$

- **Finite States.** S is said to be a **finite** state space iff $|S| < \infty$.
- **Undirected.** $G(S)$ is said to be **undirected** iff, for all states $s, t \in S$, $t \in \nu(s)$ and $p(t|s) > 0$ iff $s \in \nu(t)$ and $p(s|t) > 0$.
- **Self Transitioning.** $G(S)$ is said to be self-transitioning iff, for all $s \in S$, $s \in \nu(s)$ and $p(s|s) > 0$.
- **Irreducible.** $G(S)$ is **connected** iff there is a (positive-probability) path from any one state to any other state.

Ergodic Markov-Chains

Ergodicity

A Markov-chain with the first three properties is said to be **ergodic**, since, when transitioning from any state $s \in S$, the expected time that will elapse before the system returns to s is finite, and a return to s can occur after t steps, for any $t \geq 1$

Fundamental Theorem of Markov Chains

Stationary Distributions

Given a Markov-chain matrix P, a probability distribution π over S is called **stationary** (or an **equilibrium** distribution) provided that $\pi = \pi \cdot P$, where π is viewed as a $1 \times n$ vector/matrix. Thus, the probability of being in state i, $i = 1, \ldots, n$, after t steps is independent of t and is given by $\pi(i)$.

Fundamental Theorem of Markov Chains

Fundamental Theorem of Irreducible, Ergodic Markov Chains

Let P be the transition matrix for a finite, irreducible, ergodic Markov chain. Then associated with P is a unique stationary distribution π for which

$$\pi_i = \lim_{t->\infty} P_{ji}^t = 1/s_i,$$

for all $j = 1, 2, \ldots n$, and where s_i is the expected number of steps it takes for a random walk beginning at state i to return to state i. Moreover, π satisfies the equation $\pi = \pi \cdot P$. Conversely, if an irreducible, ergodic Markov chain's transition matrix satisfies such an equation, for some distribution π, then π is the chain's stationary distribution.

Stationary Distribution for the Weather Example

Let $\pi = (x, y)$. Then

$$(x, y) = (x, y) \begin{pmatrix} 0.8 & 0.2 \\ 0.5 & 0.5 \end{pmatrix} \Rightarrow x = 0.8x + 0.5y \Rightarrow x = 5y/2$$

by equating the first components of both sides. Since $x + y = 1$, This yields

$$5y/2 + y = 7y/2 = 1 \Rightarrow y = 2/7 \text{ and } x = 5/7.$$

Stationary Distribution for the Weather Example

Let $\pi = (x, y)$. Then

$$(x, y) = (x, y) \begin{pmatrix} 0.8 & 0.2 \\ 0.5 & 0.5 \end{pmatrix} \Rightarrow x = 0.8x + 0.5y \Rightarrow x = 5y/2$$

by equating the first components of both sides. Since $x + y = 1$, This yields

$$5y/2 + y = 7y/2 = 1 \Rightarrow y = 2/7 \text{ and } x = 5/7.$$

Stationary Distribution Interpretation

Regardless of today's weather, the probability that it will be not be raining in exactly one year from today (or some other day in the distant future) is approximately 5/7.

Markov Chain Master Equation

Master Equation Corollary for P with Stationary Distribution π

For every $i \in S$,

$$\pi(i)\sum_{j\in S} P_{ij} = \sum_{j\in S} \pi(j) P_{ji}$$

Markov Chain Master Equation

Master Equation Corollary for P with Stationary Distribution π

For every $i \in S$,

$$\pi(i)\sum_{j\in S} P_{ij} = \sum_{j\in S} \pi(j) P_{ji}$$

Proof of Master Equation

Simply note that the left side is the i th component of π, while the right side is the i th component of $\pi \cdot P$.

Markov Chain Master Equation

Interpretation: Conservation of State

For each $i \in S$, the Master Equation states that the probability of the event that the system transitions into state i equals the probability of the event that the system transitions out of state i.

Markov Chain Detailed-Balance Equation

Detailed-Balance Equation Corollary for Markov Chain P

If for every $i, j \in S$,

$$\pi(i)P_{ij} = \pi(j)P_{ji},$$

then π is the stationary distribution for P.

Markov Chain Detailed-Balance Equation

Detailed-Balance Equation Corollary for Markov Chain P

If for every $i, j \in S$,

$$\pi(i)P_{ij} = \pi(j)P_{ji},$$

then π is the stationary distribution for P.

Proof of Detailed-Balance Corollary

By fixing i and summing both sides over j, one obtains the Master Equation. Hence, the Master-Equation Corollary implies that π is the stationary distribution.

Markov Chain Detailed-Balance Equation

Interpretataion: Conservation of Inter-State Transitions

For each $i, j \in S$, probability that system is in state i and then transitions to state j equals the probability that system is in state j and transitions to state i.

Hastings-Metropolis Algorithm

Eating the π that You Desire

Given Markov Chain P, desired state distribution π, and $i, j \in S$, define $\alpha(i,j)$ and $\alpha(j,i)$ so that

$$\pi(i)\alpha(i,j)P_{ij} = \pi(j)\alpha(j,i)P_{ji},$$

where, e.g. $\alpha(i,j) = 1$ in the case that $\pi(i)P_{ij} \leq \pi(j)P_{ji}$.

Hastings-Metropolis Algorithm

Hastings-Metropolis Algorithm

Hastings-Metropolis Algorithm

Hastings-Metropolis Algorithm

- When in state i, apply transition matrix P to generate candidate next-state j.

Hastings-Metropolis Algorithm

Hastings-Metropolis Algorithm

- When in state i, apply transition matrix P to generate candidate next-state j.
- Generate a random real number U from interval $[0, 1]$.

Hastings-Metropolis Algorithm

Hastings-Metropolis Algorithm

- When in state i, apply transition matrix P to generate candidate next-state j.
- Generate a random real number U from interval $[0, 1]$.
- If $U \leq \alpha(i, j)$, transition to next state j. Otherwise, remain in (next) state i.

Hastings-Metropolis Example

$$P = \begin{pmatrix} 0.4 & 0.2 & 0.4 \\ 0.25 & 0.5 & 0.25 \\ 0.8 & 0.2 & 0 \end{pmatrix}$$

Desired Distribution: $\pi = (1/3, 1/3, 1/3)$.

$$(1/3)\alpha(1,2)(0.2) = (1/3)\alpha(2,1)(0.25) \Rightarrow \alpha(1,2) = 1 \ , \ \alpha(2,1) = 4/5$$

$$(1/3)\alpha(1,3)(0.4) = (1/3)\alpha(3,1)(0.8) \Rightarrow \alpha(1,3) = 1 \ , \ \alpha(3,1) = 1/2$$

$$(1/3)\alpha(2,3)(0.25) = (1/3)\alpha(3,2)(0.2) \Rightarrow \alpha(2,3) = 4/5 \ , \ \alpha(3,2) = 1$$

Hastings-Metropolis Example

Final α Matrix

$$P = \begin{pmatrix} 0.4 & 0.2 & 0.4 \\ 0.25 & 0.5 & 0.25 \\ 0.8 & 0.2 & 0 \end{pmatrix} \quad \alpha = \begin{pmatrix} 1 & 1 & 1 \\ 4/5 & 1 & 4/5 \\ 1/2 & 1 & 1 \end{pmatrix}$$

Hastings-Metropolis Example

Final α Matrix

$$P = \begin{pmatrix} 0.4 & 0.2 & 0.4 \\ 0.25 & 0.5 & 0.25 \\ 0.8 & 0.2 & 0 \end{pmatrix} \quad \alpha = \begin{pmatrix} 1 & 1 & 1 \\ 4/5 & 1 & 4/5 \\ 1/2 & 1 & 1 \end{pmatrix}$$

Example Uses of P and α

Hastings-Metropolis Example

Final α Matrix

$$P = \begin{pmatrix} 0.4 & 0.2 & 0.4 \\ 0.25 & 0.5 & 0.25 \\ 0.8 & 0.2 & 0 \end{pmatrix} \quad \alpha = \begin{pmatrix} 1 & 1 & 1 \\ 4/5 & 1 & 4/5 \\ 1/2 & 1 & 1 \end{pmatrix}$$

Example Uses of P and α

- Current state: $i = 3$. Next-state candidtate: $j = 1$. Generate random real: $U = 0.631$. $U > \alpha(3, 1) \Rightarrow$ next state remains $i = 3$.

Hastings-Metropolis Example

Final α Matrix

$$P = \begin{pmatrix} 0.4 & 0.2 & 0.4 \\ 0.25 & 0.5 & 0.25 \\ 0.8 & 0.2 & 0 \end{pmatrix} \quad \alpha = \begin{pmatrix} 1 & 1 & 1 \\ 4/5 & 1 & 4/5 \\ 1/2 & 1 & 1 \end{pmatrix}$$

Example Uses of P and α

- Current state: $i = 3$. Next-state candidtate: $j = 1$. Generate random real: $U = 0.631$. $U > \alpha(3,1) \Rightarrow$ next state remains $i = 3$.
- Current state: $i = 2$. Next-state candidtate: $j = 3$. Generate random real: $U = 0.417$. $U \leq \alpha(2,3) \Rightarrow$ next state is 3.

Outline

1. Introduction
2. Markov-Chains
3. Simulated Annealing
4. Genetic Algorithms
5. WalkSAT Algorithm

Choosing the Stationary Distribution

The General Problem

Given state space S, some states represent solution states. Therefore, these states should be assigned higher stationary probabilities.

Choosing the Stationary Distribution

The General Problem

Given state space S, some states represent solution states. Therefore, these states should be assigned higher stationary probabilities.

Two Step Approach

Choosing the Stationary Distribution

The General Problem

Given state space S, some states represent solution states. Therefore, these states should be assigned higher stationary probabilities.

Two Step Approach

1. Define a **proximity** function $H : S \to \mathcal{R}$, for which $H(s) \geq 0$ measures how close s is to being a solution, with $H(s) = 0$ iff s is a solution.

Choosing the Stationary Distribution

The General Problem

Given state space S, some states represent solution states. Therefore, these states should be assigned higher stationary probabilities.

Two Step Approach

1. Define a **proximity** function $H : S \to \mathcal{R}$, for which $H(s) \geq 0$ measures how close s is to being a solution, with $H(s) = 0$ iff s is a solution.
2. Define the stationary probability $\pi(s)$ in terms of $H(s)$, where $\pi(s)$ decreases as $H(s)$ increases..

Simulated Annealing Stationary Distribution

A Distribution Inspired from Physics

$$\pi(s) = \frac{\exp(\frac{-H(s)}{T})}{\sum_{t \in S} H(t)},$$

where $T > 0$ is a **temperature parameter** .

Simulated Annealing Stationary Distribution

A Distribution Inspired from Physics

$$\pi(s) = \frac{\exp(\frac{-H(s)}{T})}{\sum_{t \in S} H(t)},$$

where $T > 0$ is a **temperature parameter** .

The Annealing Process

Simulated Annealing Stationary Distribution

A Distribution Inspired from Physics

$$\pi(s) = \frac{\exp(\frac{-H(s)}{T})}{\sum_{t \in S} H(t)},$$

where $T > 0$ is a **temperature parameter** .

The Annealing Process

- **Annealing** is a heating method for creating metals with desirable physical properties.

Simulated Annealing Stationary Distribution

A Distribution Inspired from Physics

$$\pi(s) = \frac{\exp(\frac{-H(s)}{T})}{\sum_{t \in S} H(t)},$$

where $T > 0$ is a **temperature parameter** .

The Annealing Process

- **Annealing** is a heating method for creating metals with desirable physical properties.
- Metal is heated to a temperature below its melting point, but high enough so that the crystalline lattice structures within the metal break apart.

Simulated Annealing Stationary Distribution

A Distribution Inspired from Physics

$$\pi(s) = \frac{\exp(\frac{-H(s)}{T})}{\sum_{t \in S} H(t)},$$

where $T > 0$ is a **temperature parameter** .

The Annealing Process

- **Annealing** is a heating method for creating metals with desirable physical properties.
- Metal is heated to a temperature below its melting point, but high enough so that the crystalline lattice structures within the metal break apart.
- The Crystalline structures re-form and grow larger the more slowly the metal is cooled.

Simulated Annealing Stationary Distribution

A Distribution Inspired from Physics

$$\pi(s) = \frac{\exp(\frac{-H(s)}{T})}{\sum_{t \in S} H(t)},$$

where $T > 0$ is a **temperature parameter** .

The Annealing Process

- **Annealing** is a heating method for creating metals with desirable physical properties.
- Metal is heated to a temperature below its melting point, but high enough so that the crystalline lattice structures within the metal break apart.
- The Crystalline structures re-form and grow larger the more slowly the metal is cooled.
- These structures correspond with a low-energy state.

Crystalline Structures in Metals

Polonium Crystals

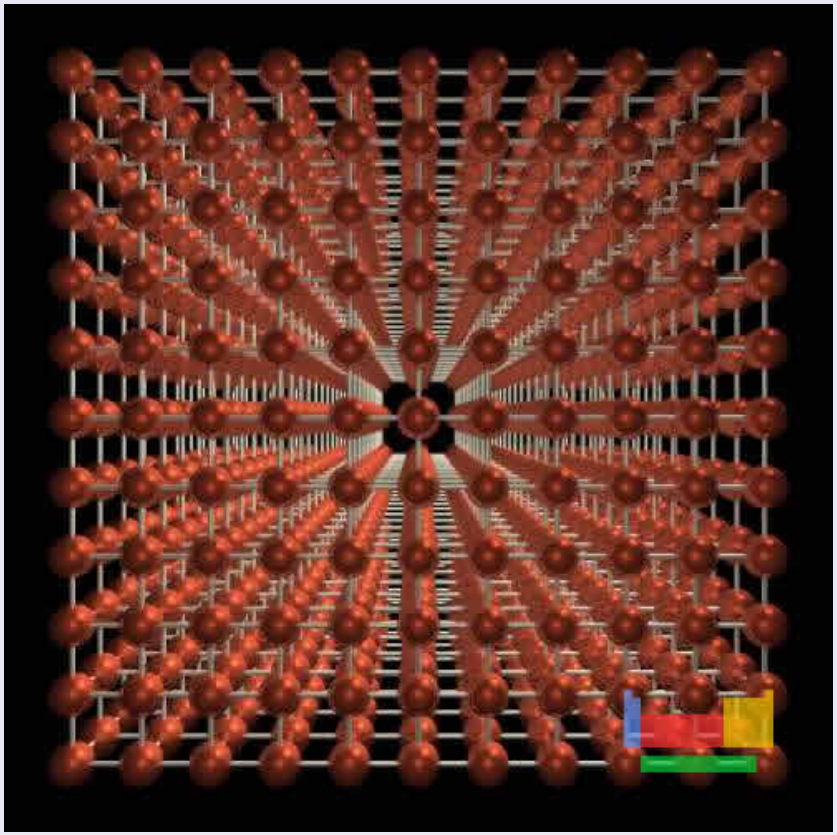


Figure: Polonium Metal Crystals

Simulated Annealing Stationary Distribution

Notes on Using the Distribution

Simulated Annealing Stationary Distribution

Notes on Using the Distribution

- **Symmetry of P.** P is assumed **symmetric**; i.e. $P_{ij} = P_{ji}$.

Simulated Annealing Stationary Distribution

Notes on Using the Distribution

- **Symmetry of P.** P is assumed **symmetric**; i.e. $P_{ij} = P_{ji}$.
- Temperature parameter T is successively lowered according to a **cooling schedule**.

Simulated Annealing Stationary Distribution

Notes on Using the Distribution

- **Symmetry of P.** P is assumed **symmetric**; i.e. $P_{ij} = P_{ji}$.
- Temperature parameter T is successively lowered according to a **cooling schedule**.
- Higher temperatures produce more uniform-looking distributions. True since, if $H(s) < H(t)$, then $\pi(s)/\pi(t) = \exp((H(t) - H(s))/T) \to 1$ as $T \to \infty$.

Simulated Annealing Stationary Distribution

Notes on Using the Distribution

- **Symmetry of P.** P is assumed **symmetric**; i.e. $P_{ij} = P_{ji}$.
- Temperature parameter T is successively lowered according to a **cooling schedule**.
- Higher temperatures produce more uniform-looking distributions. True since, if $H(s) < H(t)$, then $\pi(s)/\pi(t) = \exp((H(t) - H(s))/T) \to 1$ as $T \to \infty$.
- Lower temperatures produce distributions more concentrated about low $H(s)$ states. True since, if $H(s) < H(t)$, then $\pi(s)/\pi(t) = \exp((H(t) - H(s))/T) \to \infty$ as $T \to 0$.

Simulated Annealing Algorithm

Generate an Initial State s_0
Initialize $T_0 = \infty$

While a solution state has not been found at step $k \geq 0$
    Use P to generate a next state j from current state i.
    If j is a solution state, then return j.
    If $H(j) \leq H(i)$, then transition to next state j.
    Otherwise
        Generate random $U \in [0, 1]$
        If $U \leq \exp((H(i) - H(j))/T_k)$,
            then transition to next state j
        Otherwise remain in state i
    $T_{k+1} = \text{cooling\_function}(k + 1)$

Cooling Techniques

Geman and Geman's Theorem

A necessary and sufficience condition for a having a probability of one of ending in a global optimum is that the temperature decreases more slowly than

$$T = \frac{a}{b + \log(t)},$$

with a and b being problem-dependent constants, and t is the number of steps.

Cooling Techniques

Popular Cooling Schedules

Cooling Techniques

Popular Cooling Schedules

- **Linear Cooling:** $T = a - bt$, where a is the initial temperature, and b is usually chosen within the range of $[0.01, 0.2]$.

Cooling Techniques

Popular Cooling Schedules

- **Linear Cooling:** $T = a - bt$, where a is the initial temperature, and b is usually chosen within the range of $[0.01, 0.2]$.
- **Exponential Cooling:** $T = a \cdot b^t$, where a is the initial temperature, and b is usually chosen within the range of $[0.8, 0.99]$.

Simulated Annealing and Finding Model Solutions

Proximity Function H

Given $P = (V, D, C)$ and assignment (i.e. state) $a \in \mathcal{A}(V)$

$$H(a) = \sum_{c \in C} h_c(a),$$

where $h_c(a)$ measures how close constraint c is to being satisfied by a.

Example: Hamilton Path Problem for Graph $G = (V, E)$

Defining the Proximity Functions

Let $G_a = (V, E_a)$ denote the graph whose edge set E_a consists of those edges in $e \in E$ for which $a(x_e) = 1$.

Example: Hamilton Path Problem for Graph $G = (V, E)$

Defining the Proximity Functions

Let $G_a = (V, E_a)$ denote the graph whose edge set E_a consists of those edges in $e \in E$ for which $a(x_e) = 1$.

- **Tree Constraint** c. $h_c(a) = n(C - 1) + ||E_a| - n + 1|$, where C is the number of connected components of G_a, and $n = |V|$.

Example: Hamilton Path Problem for Graph $G = (V, E)$

Defining the Proximity Functions

Let $G_a = (V, E_a)$ denote the graph whose edge set E_a consists of those edges in $e \in E$ for which $a(x_e) = 1$.

- **Tree Constraint** c**.** $h_c(a) = n(C - 1) + ||E_a| - n + 1|$, where C is the number of connected components of G_a, and $n = |V|$.
- **Degree Constraint** c **for each vertex** v**.** Let $\deg_a(v)$ denote the degree of v in G_a. Then $h_c(a)$ equals 0 if $\deg_a(v) \leq 2$. Otherwise $h_c(a) = \deg(v) - 2$.

Example: Hamilton Path Problem for Graph $G = (V, E)$

Assignment a**:** E_a consists of all green edges.
Tree Constraint c**:** $h_c(a) = 10(2) + (9 - 7) = 22$.
Degree Constraint c**:** violated by vertex 6. $h_c(a) = (3 - 2) = 1$, for vertex 6, and $h_c(a) = 0$ for all other vertices.

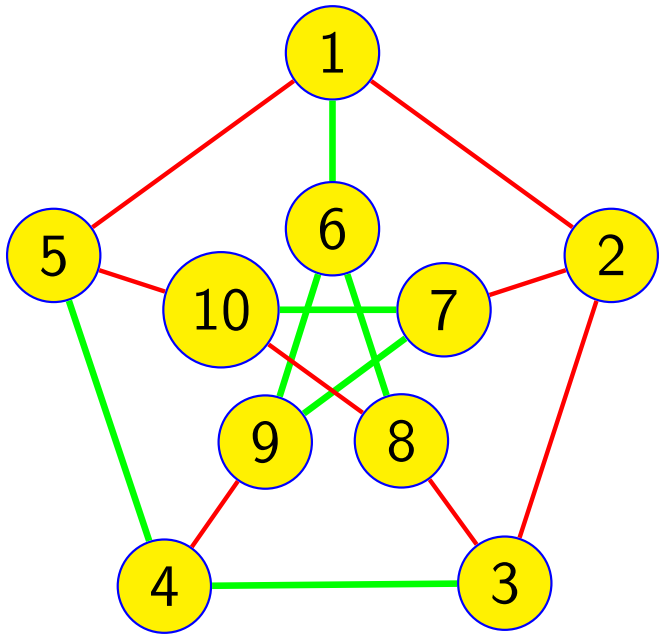


Figure: $H(a) = 22 + 1 = 23$

Enhancements to Simulated Annealing

Escaping Local Minima and Speeding up Convergence

Enhancements to Simulated Annealing

Escaping Local Minima and Speeding up Convergence

- **Tabu States.** Tabu states are recently-visited states that should are avoided in the near future so as to promote more variation in the search path. For example, a tabu number of five would prevent the return to a state that had beeen visited in the past five steps.

Enhancements to Simulated Annealing

Escaping Local Minima and Speeding up Convergence

- **Tabu States.** Tabu states are recently-visited states that should are avoided in the near future so as to promote more variation in the search path. For example, a tabu number of five would prevent the return to a state that had beeen visited in the past five steps.
- **Simulated Tempering.** Rather than allowing the temperature to continually decrease, simulated tempering treats the temperature as a state space that can be navigated via a Markov-chain model. This allows for gradual temperature fluctuations, which can assist in the escaping of local minima.

Enhancements to Simulated Annealing

Escaping Local Minima and Speeding up Convergence

- **Tabu States.** Tabu states are recently-visited states that should are avoided in the near future so as to promote more variation in the search path. For example, a tabu number of five would prevent the return to a state that had beeen visited in the past five steps.
- **Simulated Tempering.** Rather than allowing the temperature to continually decrease, simulated tempering treats the temperature as a state space that can be navigated via a Markov-chain model. This allows for gradual temperature fluctuations, which can assist in the escaping of local minima.
- **Swapping.** Instead of a single state-graph path, r such paths are generated in parallel, where the path temperatures are uniformly distributed. State transitions of paths alternate with swapping the states of two paths. This allows for states to be subjected to different temperatures which can help escape local minima.

Enhancements to Simulated Annealing

Escaping Local Minima and Speeding up Convergence

- **Tabu States.** Tabu states are recently-visited states that should are avoided in the near future so as to promote more variation in the search path. For example, a tabu number of five would prevent the return to a state that had beeen visited in the past five steps.
- **Simulated Tempering.** Rather than allowing the temperature to continually decrease, simulated tempering treats the temperature as a state space that can be navigated via a Markov-chain model. This allows for gradual temperature fluctuations, which can assist in the escaping of local minima.
- **Swapping.** Instead of a single state-graph path, r such paths are generated in parallel, where the path temperatures are uniformly distributed. State transitions of paths alternate with swapping the states of two paths. This allows for states to be subjected to different temperatures which can help escape local minima.
- **Hybrid Tree Search.** The root of the search tree $r \in S$ is chosen randomly. Given visited state x, its children are states $y \in \nu(x)$ for which $H(y) < H(x)$. If no such y exists, then x is a leaf. Otherwise, with probabiliy p_x randomly choose a child to visit next, and with probability $1 - p_x$ backtrack to the parent of x, where p_x increases as $H(x)$ decreases.

Assessment of Simulated Annealing

Advantages

Assessment of Simulated Annealing

Advantages

- Based on a well-developed mathematical theory.

Assessment of Simulated Annealing

Advantages

- Based on a well-developed mathematical theory.
- Local state transitions allow one to zoom in on increasingly improved states.

Assessment of Simulated Annealing

Advantages

- Based on a well-developed mathematical theory.
- Local state transitions allow one to zoom in on increasingly improved states.

Disadvantages

Assessment of Simulated Annealing

Advantages

- Based on a well-developed mathematical theory.
- Local state transitions allow one to zoom in on increasingly improved states.

Disadvantages

- Local state transitions means that one cannot immediately "jump" to more promising regions of the state space.

Assessment of Simulated Annealing

Advantages

- Based on a well-developed mathematical theory.
- Local state transitions allow one to zoom in on increasingly improved states.

Disadvantages

- Local state transitions means that one cannot immediately "jump" to more promising regions of the state space.
- There is a propensity under low temperatures to become trapped in sub-optimal regions due to the lack of neighobors who offer improvement to the objective functions.

Finding the Tallest Peak in the Grand Canyon

Figure: Challenging Search Problem: Find the Tallest Peak!

Outline

1 Introduction

2 Markov-Chains

3 Simulated Annealing

4 Genetic Algorithms

5 WalkSAT Algorithm

The Genetic Algorithm

Overview

The **Genetic Algorithm** is a general optimization algorithm that maintains an entire population of current states initially distributed throughout the state space. State changes occur by combining existing states to form new states via the use genetic operators.

The Genetic Algorithm

Overview

The **Genetic Algorithm** is a general optimization algorithm that maintains an entire population of current states initially distributed throughout the state space. State changes occur by combining existing states to form new states via the use genetic operators.

Philosophy of the Genetic Algorithm

States that are closer to being solutions (according to proximity function H) encode partial information about a solution to the problem. By combining the information from two such states, one can perhaps produce new states that are in closer proximity to a solution.

The Genetic Algorithm

Terminology

The Genetic Algorithm

Terminology

- The constraint-model variables V is the set of **genes**. The variable domain values are called **alleles**, and represent the set of possible ways to realize a particular gene.

The Genetic Algorithm

Terminology

- The constraint-model variables V is the set of **genes**. The variable domain values are called **alleles**, and represent the set of possible ways to realize a particular gene.
- The state space S is now called the **genotype space**, and represents the set of possible combinations of alleles. A member of the genotype space is called an **individual**.

The Genetic Algorithm

Terminology

- The constraint-model variables V is the set of **genes**. The variable domain values are called **alleles**, and represent the set of possible ways to realize a particular gene.
- The state space S is now called the **genotype space**, and represents the set of possible combinations of alleles. A member of the genotype space is called an **individual**.
- The proximity function H is now replaced by a **fitness function** $f : S \rightarrow \mathcal{R}$, where $f(s)$ indicates the degree to which individual s is fit to being a solution.

The Genetic Algorithm

Terminology

- The constraint-model variables V is the set of **genes**. The variable domain values are called **alleles**, and represent the set of possible ways to realize a particular gene.
- The state space S is now called the **genotype space**, and represents the set of possible combinations of alleles. A member of the genotype space is called an **individual**.
- The proximity function H is now replaced by a **fitness function** $f : S \to \mathcal{R}$, where $f(s)$ indicates the degree to which individual s is fit to being a solution.
- At time $t = 0, 1, \ldots$ the genetic algorithm maintains a **population** P_t of M individuals, where M is typically in the tens or hundreds of individuals.

Description of the Genetic Algorithm

Initializing P_0

P_0 is obtained by randomly selecting M individuals using some (usually uniform) distribution π_0.

Description of the Genetic Algorithm

Initializing P_0

P_0 is obtained by randomly selecting M individuals using some (usually uniform) distribution π_0.

Creating the Next Generation

Description of the Genetic Algorithm

Initializing P_0

P_0 is obtained by randomly selecting M individuals using some (usually uniform) distribution π_0.

Creating the Next Generation

- Given $P_t = \{s_1, \ldots, s_M\}$, choose $M/2$ pairs of individuals, called **parents**, from P_t, using the distribution $p_1, \ldots, p_M$, where p_i is proportional to fitness $f(s_i)$.

Description of the Genetic Algorithm

Initializing P_0

P_0 is obtained by randomly selecting M individuals using some (usually uniform) distribution π_0.

Creating the Next Generation

- Given $P_t = \{s_1, \ldots, s_M\}$, choose $M/2$ pairs of individuals, called **parents**, from P_t, using the distribution $p_1, \ldots, p_M$, where p_i is proportional to fitness $f(s_i)$.
- With probability p_c a set of parents will produce two **offspring** individuals to place in P_{t+1}. With probability $(1 - p_c)$, the parents will not have offspring, and themselves be admitted to P_{t+1}.

Description of the Genetic Algorithm

Initializing P_0

P_0 is obtained by randomly selecting M individuals using some (usually uniform) distribution π_0.

Creating the Next Generation

- Given $P_t = \{s_1, \ldots, s_M\}$, choose $M/2$ pairs of individuals, called **parents**, from P_t, using the distribution $p_1, \ldots, p_M$, where p_i is proportional to fitness $f(s_i)$.
- With probability p_c a set of parents will produce two **offspring** individuals to place in P_{t+1}. With probability $(1 - p_c)$, the parents will not have offspring, and themselves be admitted to P_{t+1}.
- To complete the next generation, each P_{t+1} individual is **mutated**; meaning that zero or more of its alleles are randomly changed.

Description of the Genetic Algorithm

Initializing P_0

P_0 is obtained by randomly selecting M individuals using some (usually uniform) distribution π_0.

Creating the Next Generation

- Given $P_t = \{s_1, \ldots, s_M\}$, choose $M/2$ pairs of individuals, called **parents**, from P_t, using the distribution $p_1, \ldots, p_M$, where p_i is proportional to fitness $f(s_i)$.
- With probability p_c a set of parents will produce two **offspring** individuals to place in P_{t+1}. With probability $(1 - p_c)$, the parents will not have offspring, and themselves be admitted to P_{t+1}.
- To complete the next generation, each P_{t+1} individual is **mutated**; meaning that zero or more of its alleles are randomly changed.
- The algorithm is terminated if either a model solution has been found, or the rate of increase of average population fitness has fallen below a pre-defined threhsold.

Offspring Production: Genetic Crossover

Crossover Operator

The **crossover operator** with **crossover point** k, $1 \leq k < n$, acts on two parents p_1 and p_2 to create two offspring o_1 and o_2, where o_1 (respectively, o_2) is comprised of the first k alleles of p_1 (respectively, p_2), followed by the last $n - k$ alleles of p_2, (repsectively, p_1).

Offspring Production: Genetic Crossover

Crossover Operator

The **crossover operator** with **crossover point** k, $1 \leq k < n$, acts on two parents p_1 and p_2 to create two offspring o_1 and o_2, where o_1 (respectively, o_2) is comprised of the first k alleles of p_1 (respectively, p_2), followed by the last $n - k$ alleles of p_2, (repsectively, p_1).

Crossover Example

Parent 1	0	1	0	1	1
Parent 2	1	1	0	0	0
Offspring 1	0	1	0	0	0
Offpring 2	1	1	0	1	1

Figure: Crossover with $n = 5$, $k = 3$, and Boolean Alleles

Offspring Production: Gene Mutation

Mutation Operator

Offspring Production: Gene Mutation

Mutation Operator

- A **mutation operator** acts on a single individual by changing zero or more of its alleles via some random process.

Offspring Production: Gene Mutation

Mutation Operator

- A **mutation operator** acts on a single individual by changing zero or more of its alleles via some random process.
- Example operator: change each allele independently and with probability p_m, where $p_m = 1/n$.

Offspring Production: Gene Mutation

Mutation Operator

- A **mutation operator** acts on a single individual by changing zero or more of its alleles via some random process.
- Example operator: change each allele independently and with probability p_m, where $p_m = 1/n$.

Mutation Example

Individual	0	1	0	1	1
Mutated Individual	1	1	0	0	1

Figure: Mutated individual with genes 1 and 4 being mutated (in red)

Genetic Algorithm Example

Genotype Space. Binary strings of length $n = 5$.

P_1 **member index**	s_i	p_i
1	00100	1/15
2	00101	2/15
3	11000	2/15
4	01110	1/5
5	10110	1/5
6	01111	4/15

Average fitness $= 15/6 =$ 2.5

fitness function. $f(s) =$ number of one bits of s.

Parent selection and crossover: parent pairs $(4, 5)$, $(2, 6)$, and $(3, 2)$. Assuming $p_c = 1$, the three crossover points are 1,4, and 2.

Mutation. $p_m = 1/5$; offspring 1,2,3, and 6 had mutations at respective locations 3,5,1, and 4.

Genetic Algorithm Example: Crossovers

Offspring of s_4 and s_5

s_4	0	1	1	1	0
s_5	1	0	1	1	0
$\hat{s}_1$	0	0	1	1	0
$\hat{s}_2$	1	1	1	1	0

Figure: Crossover s_4 and s_5 with crossover point $k = 1$.

Genetic Algorithm Example: Crossovers

Offspring of s_2 and s_6

s_2	0	0	1	0	1
s_6	0	1	1	1	1
$\hat{s}_3$	0	0	1	0	1
$\hat{s}_4$	0	1	1	1	1

Figure: Crossover s_2 and s_6 with crossover point $k = 4$.

Genetic Algorithm Example: Crossovers

Offspring of s_3 and s_2

s_3	1	1	0	0	0
s_2	0	0	1	0	1
$\hat{s}_5$	1	1	1	0	1
$\hat{s}_6$	0	0	0	0	0

Figure: Crossover s_3 and s_2 with crossover point $k = 2$.

Genetic Algorithm Example: Mutations

Mutations of $\hat{s}_1$, $\hat{s}_2$, $\hat{s}_3$, and $\hat{s}_6$ at locations 3,5,1,4

Mutated $\hat{s}_1$	0	0	0	1	0
Mutated $\hat{s}_2$	1	1	1	1	1
Mutated $\hat{s}_3$	1	0	1	0	1
Mutated $\hat{s}_6$	0	0	0	1	0

Figure: Mutations of Population P_1

Genetic Algorithm Example

Population P_1

P_0 **member index**	s_i	p_i
1	00010	1/18
2	11111	5/18
3	10101	1/6
4	01111	2/9
5	11101	2/9
6	00010	1/18

Average fitness $= 18/6 = 3$

Schemata for Boolean Models

Schema Definitions

Schemata for Boolean Models

Schema Definitions

- A **schema** for Boolean variables/genes $x_1, \ldots, x_n$, is a string $u \in \{0, 1, *\}^n$. For example, $01*1$, $****$, and $*01*$ are schemata.

Schemata for Boolean Models

Schema Definitions

- A **schema** for Boolean variables/genes $x_1, \ldots, x_n$, is a string $u \in \{0, 1, *\}^n$. For example, $01*1$, $****$, and $*01*$ are schemata.
- The **order** $o(u)$ of a schema is the number of bit characters that it possesses.

Schemata for Boolean Models

Schema Definitions

- A **schema** for Boolean variables/genes $x_1, \ldots, x_n$, is a string $u \in \{0, 1, *\}^n$. For example, $01*1$, $****$, and $*01*$ are schemata.
- The **order** $o(u)$ of a schema is the number of bit characters that it possesses.
- Schema **length** $\delta(u)$: distance between first and last bit characters.

Schemata for Boolean Models

Schema Definitions

- A **schema** for Boolean variables/genes $x_1, \ldots, x_n$, is a string $u \in \{0, 1, *\}^n$. For example, $01*1$, $****$, and $*01*$ are schemata.
- The **order** $o(u)$ of a schema is the number of bit characters that it possesses.
- Schema **length** $\delta(u)$: distance between first and last bit characters.
- A bit string $v \in \{0, 1\}^n$ **contains** u (written $u \subseteq v$) provided u and v agree at each place where u has a bit value.

Schemata for Boolean Models

Schema Definitions

- A **schema** for Boolean variables/genes $x_1, \ldots, x_n$, is a string $u \in \{0, 1, *\}^n$. For example, $01*1$, $****$, and $*01*$ are schemata.
- The **order** $o(u)$ of a schema is the number of bit characters that it possesses.
- Schema **length** $\delta(u)$: distance between first and last bit characters.
- A bit string $v \in \{0, 1\}^n$ **contains** u (written $u \subseteq v$) provided u and v agree at each place where u has a bit value.
- Given fitness function f defined over $\{0, 1\}^n$, the **average fitness** of schema u, is

$$f(u) = \frac{1}{m(u)} \sum_{u \subseteq v} f(v),$$

where $m(u)$ is the number of bit strings that contain u.

Schema Example

Schema $u = 01 * 1*$

Schema Example

Schema $u = 01 * 1*$

- $o(u) = 3$.

Schema Example

Schema $u = 01 * 1*$

- $o(u) = 3.$
- $\delta(u) = 3.$

Schema Example

Schema $u = 01 * 1*$

- $o(u) = 3$.
- $\delta(u) = 3$.
- $v_1 = 01110$ contains u, but $v_2 = 00110$ does not.

Schema Example

Schema $u = 01 * 1*$

- $o(u) = 3$.
- $\delta(u) = 3$.
- $v_1 = 01110$ contains u, but $v_2 = 00110$ does not.
- Given fitness function $f(v) = \overset{n}{\underset{i=1}{v_i}}$, average fitness of u is $(2 + 3 + 3 + 4)/4 = 3$.

Fundamental Theorem of The Genetic Algorithm

Fundamental Theorem of the Genetic Algorithm

Above-average-fitness, low-order, and short schemata will be contained by an exponentially increasing number of population members. More specifically,letting $m(u,t)$ denote the number of population members that contain u at time t, and $\overline{f}$ denote the average fitness of the entire population, then

$$m(u,t+1) \geq \frac{f(u)}{\overline{f}}[1 - p_c \frac{\delta(u)}{(n-1)} - o(u)p_m]m(u,t).$$

Proof of the Fundamental Theorem

Claim. The expected number of members who are selected to reproduce at time t and that contain schema u is equal to $\frac{f(u)}{\overline{f}} m(u,t)$.

Proof of Claim. When randomly selecting a single member to reproduce, the probability of selecting a member that contains u is given by

$$\frac{\sum_{s \in m(u,t)} f(s)}{\sum_{s \in P_t} f(s)}.$$

Thus, the expected number of members selected to reproduce is

$$M \frac{\sum_{s \in m(u,t)} f(s)}{\sum_{s \in P_t} f(s)} = M \frac{m(u,t) \sum_{s \in m(u,t)} f(s)}{m(u,t) \sum_{s \in P_t} f(s)} = \frac{f(u)}{\overline{f}} m(u,t),$$

which proves the claim.

Proof of the Fundamental Theorem Continued

Now for each member s that contains u and was chosen to reproduce at time t, we must calculate the probability that s passes u to one of its offspring. For this to *not* happen, it is necessary that crossover is performed, and the crossover point divides u. The likelihood of both these events occuring is $p_c \frac{\delta(u)}{n-1}$. Hence, with probability $1 - p_c \frac{\delta(u)}{n-1}$, s will pass u to one of its offspring.

Finally, after s passes u to an offspring, u must survive mutation within that offspring. The probability of surviving mutation is $(1 - p_m)^{o(u)} \geq 1 - o(u)p_m$.

Therefore, the expected number of population members who will possess u at time $t + 1$ is at least

$$\frac{f(u)}{\bar{f}}[1 - p_c \frac{\delta(u)}{n-1}][1 - o(u)p_m]m(u, t),$$

and the result follows via multiplication and dropping the lowest order term.

Outline

1. Introduction
2. Markov-Chains
3. Simulated Annealing
4. Genetic Algorithms
5. WalkSAT Algorithm

Applying Local Search to CNF-SAT

CNF-SAT Problem Definition

Applying Local Search to CNF-SAT

CNF-SAT Problem Definition

- **Variables.** Boolean variables $V = \{x_1, \ldots, x_n\}$.

Applying Local Search to CNF-SAT

CNF-SAT Problem Definition

- **Variables.** Boolean variables $V = \{x_1, \ldots, x_n\}$.
- **Constraints.** Each constraint c has the form $(l_1 \vee l_2 \vee \cdots \vee l_k)$, where each l_i, $1 \leq i \leq k$, is a **literal**, i.e. either a variable or its negation.

Applying Local Search to CNF-SAT

CNF-SAT Problem Definition

- **Variables.** Boolean variables $V = \{x_1, \ldots, x_n\}$.
- **Constraints.** Each constraint c has the form $(l_1 \vee l_2 \vee \cdots \vee l_k)$, where each l_i, $1 \leq i \leq k$, is a **literal**, i.e. either a variable or its negation.

WalkSAT Local-Search Algorithm

Applying Local Search to CNF-SAT

CNF-SAT Problem Definition

- **Variables.** Boolean variables $V = \{x_1, \ldots, x_n\}$.
- **Constraints.** Each constraint c has the form $(l_1 \vee l_2 \vee \cdots \vee l_k)$, where each l_i, $1 \leq i \leq k$, is a **literal**, i.e. either a variable or its negation.

WalkSAT Local-Search Algorithm

- Introduced by B. Selman, H. Katz, and B. Cohen in 1994.

Applying Local Search to CNF-SAT

CNF-SAT Problem Definition

- **Variables.** Boolean variables $V = \{x_1, \ldots, x_n\}$.
- **Constraints.** Each constraint c has the form $(l_1 \vee l_2 \vee \cdots \vee l_k)$, where each l_i, $1 \leq i \leq k$, is a **literal**, i.e. either a variable or its negation.

WalkSAT Local-Search Algorithm

- Introduced by B. Selman, H. Katz, and B. Cohen in 1994.
- **Local Neighborhoods.** For each assignment $a \in \mathcal{A}(V)$, $\hat{a} \in \nu(a)$ iff $\hat{a}(x) = a(x)$ for all but one variable $x \in V$.

Applying Local Search to CNF-SAT

CNF-SAT Problem Definition

- **Variables.** Boolean variables $V = \{x_1, \ldots, x_n\}$.
- **Constraints.** Each constraint c has the form $(l_1 \vee l_2 \vee \cdots \vee l_k)$, where each l_i, $1 \leq i \leq k$, is a **literal**, i.e. either a variable or its negation.

WalkSAT Local-Search Algorithm

- Introduced by B. Selman, H. Katz, and B. Cohen in 1994.
- **Local Neighborhoods.** For each assignment $a \in \mathcal{A}(V)$, $\hat{a} \in \nu(a)$ iff $\hat{a}(x) = a(x)$ for all but one variable $x \in V$.
- **Transition Rule.** Randomly select $c \in C$ violated by a. With probability p, change $a(x)$ for randomly chosen $x \in \mathrm{var}(c)$. With probability $(1 - p)$, greedily change $x \in c$, where x maximizes the number of constraints that become satisfied due to changing x, minus those constraints that become violated by changing x.