ucc
Coláiste na hOllscoile Corcaigh, Éire
University College Cork, Ireland

CS4618 Artificial Intelligence I

Today: Introduction to Heuristic Search
Randomised Search Heuristics

Thomas Jansen

October 17th

Plans for Today

Plans for the Next Weeks

so far blind search

knows initial state, transition-model, actions, costs, goal-test

Plans for the Next Weeks

so far blind search

knows initial state, transition-model, actions, costs, goal-test

Examples

Plans for the Next Weeks

so far blind search

knows initial state, transition-model, actions, costs, goal-test

Examples breadth-first search, depth-first search, depth-limited search, iterative deepening depth-first search, uniform-cost search, bidirectional search

Plans for the Next Weeks

so far blind search

knows initial state, transition-model, actions, costs, goal-test

Examples breadth-first search, depth-first search, depth-limited search, iterative deepening depth-first search, uniform-cost search, bidirectional search

and informed search

knows in addition evaluation function f

(often with heuristic h)

Examples

Plans for the Next Weeks

so far blind search

knows initial state, transition-model, actions, costs, goal-test

Examples breadth-first search, depth-first search, depth-limited search, iterative deepening depth-first search, uniform-cost search, bidirectional search

and informed search

knows in addition evaluation function f

(often with heuristic h)

Examples A* search, greedy best-first search, iterative deepening A* search, simplified memory-bounded A* search, recursive best-first search

Plans for the Next Weeks

so far blind search

knows initial state, transition-model, actions, costs, goal-test

Examples breadth-first search, depth-first search, depth-limited search, iterative deepening depth-first search, uniform-cost search, bidirectional search

and informed search

knows in addition evaluation function f

(often with heuristic h)

Examples A* search, greedy best-first search, iterative deepening A* search, simplified memory-bounded A* search, recursive best-first search

Observation not only goal considered

also path to solution

Plans for the Next Weeks

so far blind search

knows initial state, transition-model, actions, costs, goal-test

Examples breadth-first search, depth-first search, depth-limited search, iterative deepening depth-first search, uniform-cost search, bidirectional search

and informed search

knows in addition evaluation function f

(often with heuristic h)

Examples A* search, greedy best-first search, iterative deepening A* search, simplified memory-bounded A* search, recursive best-first search

Observation not only goal considered

also path to solution

Now only 'optimal goal' interesting

Searching for a Solution

Searching for a Solution

Consider problem given by

Searching for a Solution

Consider problem given by

- set of potential solutions S

Searching for a Solution

Consider problem given by

- set of potential solutions $S \approx$ state space

Searching for a Solution

Consider problem given by

- set of potential solutions $S \approx$ state space
- values of potential solutions $f : S \to \mathbb{R}$

Searching for a Solution

Consider problem given by

- set of potential solutions $S \approx$ state space
- values of potential solutions $f : S \to \mathbb{R} \approx$ path cost

Searching for a Solution

Consider problem given by

- set of potential solutions $S \approx$ state space
- values of potential solutions $f : S \to \mathbb{R} \approx$ path cost

What about actions? What about costs of actions?

Searching for a Solution

Consider problem given by

- set of potential solutions $S \approx$ state space
- values of potential solutions $f : S \to \mathbb{R} \approx$ path cost

What about actions? What about costs of actions?

Observation not useful notions in some problems

Searching for a Solution

Consider problem given by

- set of potential solutions $S \approx$ state space
- values of potential solutions $f : S \to \mathbb{R} \approx$ path cost

What about actions? What about costs of actions?

Observation not useful notions in some problems

Example problems

Searching for a Solution

Consider problem given by

- set of potential solutions $S \approx$ state space
- values of potential solutions $f : S \to \mathbb{R} \approx$ path cost

What about actions? What about costs of actions?

Observation not useful notions in some problems

Example problems

- 8-queens problem

Searching for a Solution

Consider problem given by

- set of potential solutions $S \approx$ state space
- values of potential solutions $f : S \to \mathbb{R} \approx$ path cost

What about actions? What about costs of actions?

Observation not useful notions in some problems

Example problems

- 8-queens problem
- integrated circuit design

Searching for a Solution

Consider problem given by

- set of potential solutions $S \approx$ state space
- values of potential solutions $f : S \to \mathbb{R} \approx$ path cost

What about actions? What about costs of actions?

Observation not useful notions in some problems

Example problems

- 8-queens problem
- integrated circuit design
- factory floor layout

Searching for a Solution

Consider problem given by

- set of potential solutions $S \approx$ state space
- values of potential solutions $f : S \to \mathbb{R} \approx$ path cost

What about actions? What about costs of actions?

Observation not useful notions in some problems

Example problems

- 8-queens problem
- integrated circuit design
- factory floor layout
- job-shop scheduling

Searching for a Solution

Consider problem given by

- set of potential solutions $S \approx$ state space
- values of potential solutions $f : S \to \mathbb{R} \approx$ path cost

What about actions? What about costs of actions?

Observation not useful notions in some problems

Example problems

- 8-queens problem
- integrated circuit design
- factory floor layout
- job-shop scheduling
- telecommunication network optimisation

Searching for a Solution

Consider problem given by

- set of potential solutions $S \approx$ state space
- values of potential solutions $f : S \to \mathbb{R} \approx$ path cost

What about actions? What about costs of actions?

Observation not useful notions in some problems

Example problems

- 8-queens problem
- integrated circuit design
- factory floor layout
- job-shop scheduling
- telecommunication network optimisation
- vehicle routing

Searching for a Solution

Consider problem given by

- set of potential solutions $S \approx$ state space
- values of potential solutions $f : S \to \mathbb{R} \approx$ path cost

What about actions? What about costs of actions?

Observation not useful notions in some problems

Example problems

- 8-queens problem
- integrated circuit design
- factory floor layout
- job-shop scheduling
- telecommunication network optimisation
- vehicle routing
- automatic programming

Searching for a Solution

Consider problem given by

- set of potential solutions $S \approx$ state space
- values of potential solutions $f : S \to \mathbb{R} \approx$ path cost

What about actions? What about costs of actions?

Observation not useful notions in some problems

Example problems

- 8-queens problem
- integrated circuit design
- factory floor layout
- job-shop scheduling
- telecommunication network optimisation
- vehicle routing
- automatic programming
- ...

Solving General Search Problems

Consider problem given by

- set of potential solutions S
- values of potential solutions $f : S \to \mathbb{R}$

Solving General Search Problems

Consider problem given by

- set of potential solutions S
- values of potential solutions $f : S \to \mathbb{R}$

Observation search problem of this kind
$\widehat{=}$ general optimisation problem with objective function f

Solving General Search Problems

Consider problem given by

- set of potential solutions S
- values of potential solutions $f\colon S \to \mathbb{R}$

Observation search problem of this kind
$\widehat{=}$ general optimisation problem with objective function f
'Search for optimal solution $s \in S$.'
(optimal $\widehat{=}$ minimal or maximal under f)

Solving General Search Problems

Consider problem given by

- set of potential solutions S
- values of potential solutions $f\colon S \to \mathbb{R}$

Observation search problem of this kind
$\widehat{=}$ general optimisation problem with objective function f
'Search for optimal solution $s \in S$.'
(optimal $\widehat{=}$ minimal or maximal under f)

Assumptions

Solving General Search Problems

Consider problem given by

- set of potential solutions S
- values of potential solutions $f\colon S \to \mathbb{R}$

Observation search problem of this kind
$\widehat{=}$ general optimisation problem with objective function f
'Search for optimal solution $s \in S$.'
(optimal $\widehat{=}$ minimal or maximal under f)

Assumptions

- can construct arbitrary $s \in S$ efficiently

Solving General Search Problems

Consider problem given by

- set of potential solutions S
- values of potential solutions $f\colon S \to \mathbb{R}$

Observation search problem of this kind
$\widehat{=}$ general optimisation problem with objective function f
'Search for optimal solution $s \in S$.'
(optimal $\widehat{=}$ minimal or maximal under f)

Assumptions

- can construct arbitrary $s \in S$ efficiently
- can compute $f(s)$ for arbitrary $s \in S$ reasonably fast

General Randomised Search Heuristics

Consider problem given by

- set of potential solutions S
- values of potential solutions $f : S \to \mathbb{R}$

Assumptions

- can construct arbitrary $s \in S$ efficiently
- can compute $f(s)$ for arbitrary $s \in S$ reasonably fast

General Randomised Search Heuristics

Consider problem given by

- set of potential solutions S
- values of potential solutions $f : S \to \mathbb{R}$

Assumptions

- can construct arbitrary $s \in S$ efficiently
- can compute $f(s)$ for arbitrary $s \in S$ reasonably fast

Remark

- additional assumptions about search space S may be made

General Randomised Search Heuristics

Consider problem given by

- set of potential solutions S
- values of potential solutions $f : S \to \mathbb{R}$

Assumptions

- can construct arbitrary $s \in S$ efficiently
- can compute $f(s)$ for arbitrary $s \in S$ reasonably fast

Remark

- additional assumptions about search space S may be made
  Example neighbourhood N $(N : S \to 2^S)$

General Randomised Search Heuristics

Consider problem given by

- set of potential solutions S
- values of potential solutions $f : S \to \mathbb{R}$

Assumptions

- can construct arbitrary $s \in S$ efficiently
- can compute $f(s)$ for arbitrary $s \in S$ reasonably fast

Remark

- additional assumptions about search space S may be made
  Example neighbourhood N $(N : S \to 2^S)$
- additional assumptions about objective function f not made

General Randomised Search Heuristics

Consider problem given by

- set of potential solutions S
- values of potential solutions $f : S \to \mathbb{R}$

Assumptions

- can construct arbitrary $s \in S$ efficiently
- can compute $f(s)$ for arbitrary $s \in S$ reasonably fast

Remark

- additional assumptions about search space S may be made
  Example neighbourhood N $(N : S \to 2^S)$
- additional assumptions about objective function f not made

In the following

- always S finite

General Randomised Search Heuristics

Consider problem given by

- set of potential solutions S
- values of potential solutions $f : S \to \mathbb{R}$

Assumptions

- can construct arbitrary $s \in S$ efficiently
- can compute $f(s)$ for arbitrary $s \in S$ reasonably fast

Remark

- additional assumptions about search space S may be made
  Example neighbourhood N $(N : S \to 2^S)$
- additional assumptions about objective function f not made

In the following

- always S finite
- sometimes $S = \{0, 1\}^n$

General Randomised Search Heuristics

Consider problem given by

- set of potential solutions S
- values of potential solutions $f : S \to \mathbb{R}$

Assumptions

- can construct arbitrary $s \in S$ efficiently
- can compute $f(s)$ for arbitrary $s \in S$ reasonably fast

Remark

- additional assumptions about search space S may be made
  Example neighbourhood N $(N : S \to 2^S)$
- additional assumptions about objective function f not made

In the following

- always S finite
- sometimes $S = \{0,1\}^n$
- maximisation of f

Local Search

Local Search

Local Search

1. $t := 1$; Choose $x_t \in S$ uniformly at random.

Local Search

Local Search

1. $t := 1$; Choose $x_t \in S$ uniformly at random.
2. Repeat
3. Select $y \in N(x_t)$.

Local Search

Local Search

1. $t := 1$; Choose $x_t \in S$ uniformly at random.
2. Repeat
3. Select $y \in N(x_t)$.
4. $t := t + 1$; If $f(y) > f(x_{t-1})$ then $x_t := y$ else $x_t := x_{t-1}$.

Local Search

Local Search

1. $t := 1$; Choose $x_t \in S$ uniformly at random.
2. Repeat
3. Select $y \in N(x_t)$.
4. $t := t + 1$; If $f(y) > f(x_{t-1})$ then $x_t := y$ else $x_t := x_{t-1}$.
5. Until 'decide to stop'
6. Output x_t.

Local Search

Local Search

1. $t := 1$; Choose $x_t \in S$ uniformly at random.
2. Repeat
3. Select $y \in N(x_t)$.
4. $t := t + 1$; If $f(y) > f(x_{t-1})$ then $x_t := y$ else $x_t := x_{t-1}$.
5. Until 'decide to stop'
6. Output x_t.

Observation '3. Select $y \in N(x_t)$.'
not sufficiently precise

Local Search

Local Search

1. $t := 1$; Choose $x_t \in S$ uniformly at random.
2. Repeat
3. Select $y \in N(x_t)$.
4. $t := t + 1$; If $f(y) > f(x_{t-1})$ then $x_t := y$ else $x_t := x_{t-1}$.
5. Until 'decide to stop'
6. Output x_t.

Observation '3. Select $y \in N(x_t)$.'
not sufficiently precise

Solution precise specification
$\rightsquigarrow$ different variants of local search

Local Search

Local Search

1. $t := 1$; Choose $x_t \in S$ uniformly at random.
2. Repeat
3. Select $y \in N(x_t)$.
4. $t := t + 1$; If $f(y) > f(x_{t-1})$ then $x_t := y$ else $x_t := x_{t-1}$.
5. Until 'decide to stop'
6. Output x_t.

Observation '3. Select $y \in N(x_t)$.'
not sufficiently precise

Solution precise specification
$\rightsquigarrow$ different variants of local search

Observation '5. Until decide to stop' not sufficiently precise
precise definitions later

Local Search Variants

Local Search

1. $t := 1$; Choose $x_t \in S$ uniformly at random.
2. Repeat
3. Select $y \in N(x_t)$.
4. $t := t + 1$; If $f(y) > f(x_{t-1})$ then $x_t := y$ else $x_t := x_{t-1}$.
5. Until 'decide to stop'
6. Output x_t.

Local Search Variants

Local Search

1. $t := 1$; Choose $x_t \in S$ uniformly at random.
2. Repeat
3. Select $y \in N(x_t)$.
4. $t := t + 1$; If $f(y) > f(x_{t-1})$ then $x_t := y$ else $x_t := x_{t-1}$.
5. Until 'decide to stop'
6. Output x_t.

First-Improvement Local Search

Consider all $y \in N(x_t)$ in fixed order
and stop at first y with $f(y) > f(x_t)$.

Local Search Variants

Local Search

1. $t := 1$; Choose $x_t \in S$ uniformly at random.
2. Repeat
3. Select $y \in N(x_t)$.
4. $t := t + 1$; If $f(y) > f(x_{t-1})$ then $x_t := y$ else $x_t := x_{t-1}$.
5. Until 'decide to stop'
6. Output x_t.

First-Improvement Local Search

Consider all $y \in N(x_t)$ in fixed order
and stop at first y with $f(y) > f(x_t)$.

Best-Improvement Local Search

Consider all $y \in N(x_t)$ (in arbitrary order)
and use one y with $f(y) = \max\{f(z) \mid z \in N(x_t)\}$.

Local Search Variants

Local Search

1. $t := 1$; Choose $x_t \in S$ uniformly at random.
2. Repeat
3. Select $y \in N(x_t)$.
4. $t := t + 1$; If $f(y) > f(x_{t-1})$ then $x_t := y$ else $x_t := x_{t-1}$.
5. Until 'decide to stop'
6. Output x_t.

First-Improvement Local Search

Consider all $y \in N(x_t)$ in fixed order
and stop at first y with $f(y) > f(x_t)$.

Best-Improvement Local Search

Consider all $y \in N(x_t)$ (in arbitrary order)
and use one y with $f(y) = \max\{f(z) \mid z \in N(x_t)\}$.

Random Local Search

Consider all $y \in N(x_t)$ in random order
and stop at first y with $f(y) > f(x_t)$.

Deciding to Stop

Remember need to specify ‘decide to stop’

Deciding to Stop

Remember need to specify 'decide to stop'

Possible Stopping Criteria

- Stop if $f(x_t) \geq \max\{f(y) \mid y \in N(x_t)\}$.

Deciding to Stop

Remember need to specify 'decide to stop'

Possible Stopping Criteria

- Stop if $f(x_t) \geq \max\{f(y) \mid y \in N(x_t)\}$.
  $\widehat{=}$ Stop if no further improvement possible. ($\widehat{=}$local optimum)

Deciding to Stop

Remember need to specify 'decide to stop'

Possible Stopping Criteria

- Stop if $f(x_t) \geq \max\{f(y) \mid y \in N(x_t)\}$.
  $\widehat{=}$ Stop if no further improvement possible. ($\widehat{=}$local optimum)
  Caution Example $S := \mathbb{Z}$, $N(z) = \{z-1, z+1\}$, $f(z) = z$.

Deciding to Stop

Remember need to specify 'decide to stop'

Possible Stopping Criteria

- Stop if $f(x_t) \geq \max\{f(y) \mid y \in N(x_t)\}$.
  $\widehat{=}$ Stop if no further improvement possible. ($\widehat{=}$local optimum)
  Caution Example $S := \mathbb{Z}$, $N(z) = \{z-1, z+1\}$, $f(z) = z$.
  Observation never stops

Deciding to Stop

Remember need to specify 'decide to stop'

Possible Stopping Criteria

- Stop if $f(x_t) \geq \max\{f(y) \mid y \in N(x_t)\}$.
  $\widehat{=}$ Stop if no further improvement possible. ($\widehat{=}$local optimum)
  Caution Example $S := \mathbb{Z}$, $N(z) = \{z-1, z+1\}$, $f(z) = z$.
  Observation never stops
- Stop after a pre-specified number of steps.

Deciding to Stop

Remember need to specify 'decide to stop'

Possible Stopping Criteria

- Stop if $f(x_t) \geq \max\{f(y) \mid y \in N(x_t)\}$.
  $\widehat{=}$ Stop if no further improvement possible. ($\widehat{=}$local optimum)
  Caution Example $S := \mathbb{Z}$, $N(z) = \{z-1, z+1\}$, $f(z) = z$.
  Observation never stops
- Stop after a pre-specified number of steps.
  simple to implement, but difficult to set

Deciding to Stop

Remember need to specify 'decide to stop'

Possible Stopping Criteria

- Stop if $f(x_t) \geq \max\{f(y) \mid y \in N(x_t)\}$.
  $\widehat{=}$ Stop if no further improvement possible. ($\widehat{=}$local optimum)
  Caution Example $S := \mathbb{Z}$, $N(z) = \{z-1, z+1\}$, $f(z) = z$.
  Observation never stops
- Stop after a pre-specified number of steps.
  simple to implement, but difficult to set
- Stop after a pre-specified f-value is reached.

Deciding to Stop

Remember need to specify 'decide to stop'

Possible Stopping Criteria

- Stop if $f(x_t) \geq \max\{f(y) \mid y \in N(x_t)\}$.
  $\widehat{=}$ Stop if no further improvement possible. ($\widehat{=}$local optimum)
  Caution Example $S := \mathbb{Z}$, $N(z) = \{z-1, z+1\}$, $f(z) = z$.
  Observation never stops
- Stop after a pre-specified number of steps.
  simple to implement, but difficult to set
- Stop after a pre-specified f-value is reached.
  simple to implement, but difficult to set

Deciding to Stop

Remember need to specify 'decide to stop'

Possible Stopping Criteria

- Stop if $f(x_t) \geq \max\{f(y) \mid y \in N(x_t)\}$.
  $\widehat{=}$ Stop if no further improvement possible. ($\widehat{=}$local optimum)
  Caution Example $S := \mathbb{Z}$, $N(z) = \{z-1, z+1\}$, $f(z) = z$.
  Observation never stops
- Stop after a pre-specified number of steps.
  simple to implement, but difficult to set
- Stop after a pre-specified f-value is reached.
  simple to implement, but difficult to set
- Stop if time is up.

Deciding to Stop

Remember need to specify 'decide to stop'

Possible Stopping Criteria

- Stop if $f(x_t) \geq \max\{f(y) \mid y \in N(x_t)\}$.
  $\widehat{=}$ Stop if no further improvement possible. ($\widehat{=}$local optimum)
  Caution Example $S := \mathbb{Z}$, $N(z) = \{z-1, z+1\}$, $f(z) = z$.
  Observation never stops
- Stop after a pre-specified number of steps.
  simple to implement, but difficult to set
- Stop after a pre-specified f-value is reached.
  simple to implement, but difficult to set
- Stop if time is up.
  simple to implement

Deciding to Stop

Remember need to specify 'decide to stop'

Possible Stopping Criteria

- Stop if $f(x_t) \geq \max\{f(y) \mid y \in N(x_t)\}$.
  $\widehat{=}$ Stop if no further improvement possible. ($\widehat{=}$local optimum)
  Caution Example $S := \mathbb{Z}$, $N(z) = \{z-1, z+1\}$, $f(z) = z$.
  Observation never stops
- Stop after a pre-specified number of steps.
  simple to implement, but difficult to set
- Stop after a pre-specified f-value is reached.
  simple to implement, but difficult to set
- Stop if time is up.
  simple to implement
- ...

Metropolis Algorithm

Metropolis Algorithm

Parameter $T \in \mathbb{R}^+$ (temperature)

Metropolis Algorithm

Parameter $T \in \mathbb{R}^+$ (temperature)

Metropolis Algorithm

1. $t := 1$; Choose $x_t \in S$ uniformly at random.

Metropolis Algorithm

Parameter $T \in \mathbb{R}^+$ (temperature)

Metropolis Algorithm

1. $t := 1$; Choose $x_t \in S$ uniformly at random.
2. Repeat
3. Select $y \in N(x_t)$ uniformly at random.

Metropolis Algorithm

Parameter $T \in \mathbb{R}^+$ (temperature)

Metropolis Algorithm

1. $t := 1$; Choose $x_t \in S$ uniformly at random.
2. Repeat
3. Select $y \in N(x_t)$ uniformly at random.
4. $t := t + 1$
   With probability $\min\{1, e^{(f(y)-f(x_{t-1}))/T}\}$ set $x_t := y$
   else $x_t := x_{t-1}$.

Metropolis Algorithm

Parameter $T \in \mathbb{R}^+$ (temperature)

Metropolis Algorithm

1. $t := 1$; Choose $x_t \in S$ uniformly at random.
2. Repeat
3. Select $y \in N(x_t)$ uniformly at random.
4. $t := t + 1$
   With probability $\min\{1, e^{(f(y)-f(x_{t-1}))/T}\}$ set $x_t := y$
   else $x_t := x_{t-1}$.
5. Until 'decide to stop'
6. Output x_t.

Metropolis Algorithm

Parameter $T \in \mathbb{R}^+$ (temperature)

Metropolis Algorithm

1. $t := 1$; Choose $x_t \in S$ uniformly at random.
2. Repeat
3. Select $y \in N(x_t)$ uniformly at random.
4. $t := t + 1$
   With probability $\min\{1, e^{(f(y)-f(x_{t-1}))/T}\}$ set $x_t := y$
   else $x_t := x_{t-1}$.
5. Until 'decide to stop'
6. Output x_t.

Remark decision about stopping as for local search

Metropolis Algorithm

Parameter $T \in \mathbb{R}^+$ (temperature)

Metropolis Algorithm

1. $t := 1$; Choose $x_t \in S$ uniformly at random.
2. Repeat
3. Select $y \in N(x_t)$ uniformly at random.
4. $t := t + 1$
   With probability $\min\{1, e^{(f(y)-f(x_{t-1}))/T}\}$ set $x_t := y$
   else $x_t := x_{t-1}$.
5. Until 'decide to stop'
6. Output x_t.

Remark decision about stopping as for local search
'Stop, if no improvement within pre-specified number of steps.'
also useful

Metropolis Algorithm

Parameter $T \in \mathbb{R}^+$ (temperature)

Metropolis Algorithm

1. $t := 1$; Choose $x_t \in S$ uniformly at random.
2. Repeat
3. Select $y \in N(x_t)$ uniformly at random.
4. $t := t + 1$
   With probability $\min\{1, e^{(f(y)-f(x_{t-1}))/T}\}$ set $x_t := y$
   else $x_t := x_{t-1}$.
5. Until 'decide to stop'
6. Output x_t.

Remark decision about stopping as for local search
'Stop, if no improvement within pre-specified number of steps.'
also useful
simple to implement, but difficult to set

About T in the Metropolis Algorithm

Parameter $T \in \mathbb{R}^+$ (temperature)

Metropolis Algorithm

1. $t := 1$; Choose $x_t \in S$ uniformly at random.
2. Repeat
3. Select $y \in N(x_t)$ uniformly at random.
4. $t := t + 1$
   With probability $\min\{1, e^{(f(y)-f(x_{t-1}))/T}\}$ set $x_t := y$
   else $x_t := x_{t-1}$.
5. Until 'decide to stop'
6. Output x_t.

About T in the Metropolis Algorithm

Parameter $T \in \mathbb{R}^+$ (temperature)

Metropolis Algorithm

1. $t := 1$; Choose $x_t \in S$ uniformly at random.
2. Repeat
3. Select $y \in N(x_t)$ uniformly at random.
4. $t := t + 1$
   With probability $\min\{1, e^{(f(y)-f(x_{t-1}))/T}\}$ set $x_t := y$
   else $x_t := x_{t-1}$.
5. Until 'decide to stop'
6. Output x_t.

Observation

- Metropolis can escape from local optima

About T in the Metropolis Algorithm

Parameter $T \in \mathbb{R}^+$ (temperature)

Metropolis Algorithm

1. $t := 1$; Choose $x_t \in S$ uniformly at random.
2. Repeat
3. Select $y \in N(x_t)$ uniformly at random.
4. $t := t + 1$
   With probability $\min\{1, e^{(f(y)-f(x_{t-1}))/T}\}$ set $x_t := y$
   else $x_t := x_{t-1}$.
5. Until 'decide to stop'
6. Output x_t.

Observation

- Metropolis can escape from local optima
- large $T \rightsquigarrow$ easy escape $\widehat{=}$ willing to take large risks

About T in the Metropolis Algorithm

Parameter $T \in \mathbb{R}^+$ (temperature)

Metropolis Algorithm

1. $t := 1$; Choose $x_t \in S$ uniformly at random.
2. Repeat
3. Select $y \in N(x_t)$ uniformly at random.
4. $t := t + 1$
   With probability $\min\{1, e^{(f(y)-f(x_{t-1}))/T}\}$ set $x_t := y$
   else $x_t := x_{t-1}$.
5. Until 'decide to stop'
6. Output x_t.

Observation

- Metropolis can escape from local optima
- large $T \rightsquigarrow$ easy escape $\widehat{=}$ willing to take large risks
- small $T \rightsquigarrow$ difficult escape $\widehat{=}$ unwilling to take large risks

Simulated Annealing

Simulated Annealing

Parameter $T\colon \mathbb{N} \to \mathbb{R}^+$ (cooling schedule)

Simulated Annealing

Parameter $T\colon \mathbb{N} \to \mathbb{R}^+$ (cooling schedule)

Simulated Annealing

1. $t := 1$; Choose $x_t \in S$ uniformly at random.

Simulated Annealing

Parameter $T\colon \mathbb{N} \to \mathbb{R}^+$ (cooling schedule)

Simulated Annealing

1. $t := 1$; Choose $x_t \in S$ uniformly at random.
2. Repeat
3. Select $y \in N(x_t)$ uniformly at random.

Simulated Annealing

Parameter $T\colon \mathbb{N} \to \mathbb{R}^+$ (cooling schedule)

Simulated Annealing

1. $t := 1$; Choose $x_t \in S$ uniformly at random.
2. Repeat
3. Select $y \in N(x_t)$ uniformly at random.
4. $t := t + 1$
   With probability $\min\{1, e^{(f(y)-f(x_{t-1}))/T(t)}\}$ set $x_t := y$
   else $x_t := x_{t-1}$.

Simulated Annealing

Parameter $T\colon \mathbb{N} \to \mathbb{R}^+$ (cooling schedule)

Simulated Annealing

1. $t := 1$; Choose $x_t \in S$ uniformly at random.
2. Repeat
3. Select $y \in N(x_t)$ uniformly at random.
4. $t := t + 1$
   With probability $\min\{1, e^{(f(y)-f(x_{t-1}))/T(t)}\}$ set $x_t := y$
   else $x_t := x_{t-1}$.
5. Until 'decide to stop'
6. Output x_t.

Simulated Annealing

Parameter $T\colon \mathbb{N} \to \mathbb{R}^+$ (cooling schedule)

Simulated Annealing

1. $t := 1$; Choose $x_t \in S$ uniformly at random.
2. Repeat
3. Select $y \in N(x_t)$ uniformly at random.
4. $t := t + 1$
   With probability $\min\{1, e^{(f(y)-f(x_{t-1}))/T(t)}\}$ set $x_t := y$
   else $x_t := x_{t-1}$.
5. Until 'decide to stop'
6. Output x_t.

Remark Metropolis $\widehat{=}$ simulated annealing with fixed temperature
most often cooling schedule non-increasing

Randomised Search Heuristics

Randomised Search Heuristics

Scenario Find some $s \in S$ that optimises $f : S \to \mathbb{R}$.

Randomised Search Heuristics

Scenario Find some $s \in S$ that optimises $f : S \to \mathbb{R}$.

Always necessary assumptions

- $\forall s \in S$: s is easy to construct

Randomised Search Heuristics

Scenario Find some $s \in S$ that optimises $f : S \to \mathbb{R}$.

Always necessary assumptions

- $\forall s \in S$: s is easy to construct
- $\forall s \in S$: $f(s)$ is reasonably easy to compute

Randomised Search Heuristics

Scenario Find some $s \in S$ that optimises $f : S \to \mathbb{R}$.

Always necessary assumptions

- $\forall s \in S$: s is easy to construct
- $\forall s \in S$: $f(s)$ is reasonably easy to compute

Here we make the following additional assumptions

- S finite (often $S = \{0,1\}^n$)

Randomised Search Heuristics

Scenario Find some $s \in S$ that optimises $f : S \to \mathbb{R}$.

Always necessary assumptions

- $\forall s \in S$: s is easy to construct
- $\forall s \in S$: $f(s)$ is reasonably easy to compute

Here we make the following additional assumptions

- S finite (often $S = \{0,1\}^n$)
- 'optimise' means maximise

Randomised Search Heuristics

Scenario Find some $s \in S$ that optimises $f : S \to \mathbb{R}$.

Always necessary assumptions

- $\forall s \in S$: s is easy to construct
- $\forall s \in S$: $f(s)$ is reasonably easy to compute

Here we make the following additional assumptions

- S finite (often $S = \{0,1\}^n$)
- 'optimise' means maximise

Common additional assumption

- neighbourhood $N : S \to 2^S$ known

Some Randomised Search Heuristics

Some Randomised Search Heuristics

- local search

Some Randomised Search Heuristics

- local search
  - first-improvement local search
  - best-improvement local search
  - random local search

Some Randomised Search Heuristics

- local search
  - first-improvement local search
  - best-improvement local search
  - random local search
- Metropolis algorithm
  $\widehat{=}$ random local search but accepting new search point with probability $\min\{1, e^{(f(y)-f(x_{t-1}))/T}\}$, parameter $T \in \mathbb{R}^+$

Some Randomised Search Heuristics

- local search
  - first-improvement local search
  - best-improvement local search
  - random local search
- Metropolis algorithm
  $\widehat{=}$ random local search but accepting new search point with probability $\min\{1, e^{(f(y)-f(x_{t-1}))/T}\}$, parameter $T \in \mathbb{R}^+$
- simulated annealing
  $\widehat{=}$ Metropolis algorithm but with annealing schedule $T\colon \mathbb{N} \to \mathbb{R}^+$ instead of fixed temperature

Evolutionary Algorithms

Evolutionary Algorithms

Parameters

- population size $\mu \in \mathbb{N}$
- offspring population size $\lambda \in \mathbb{N}$
- crossover probability $p_c \in [0, 1]$

Evolutionary Algorithms

Parameters

- population size $\mu \in \mathbb{N}$
- offspring population size $\lambda \in \mathbb{N}$
- crossover probability $p_c \in [0, 1]$

Evolutionary Algorithm

1. Choose $x_1, x_2, \ldots, x_\mu \in S$ uniformly at random.

Evolutionary Algorithms

Parameters

- population size $\mu \in \mathbb{N}$
- offspring population size $\lambda \in \mathbb{N}$
- crossover probability $p_c \in [0, 1]$

Evolutionary Algorithm

1. Choose $x_1, x_2, \ldots, x_\mu \in S$ uniformly at random.
2. Repeat
3. For $i \in \{1, 2, \ldots, \lambda\}$ do

Evolutionary Algorithms

Parameters

- population size $\mu \in \mathbb{N}$
- offspring population size $\lambda \in \mathbb{N}$
- crossover probability $p_c \in [0, 1]$

Evolutionary Algorithm

1. Choose $x_1, x_2, \ldots, x_\mu \in S$ uniformly at random.
2. Repeat
3. For $i \in \{1, 2, \ldots, \lambda\}$ do
4. With probability p_c select $z_1, z_2 \in \{x_1, x_2, \ldots, x_\mu\}$.
   $z :=$ crossover(z_1, z_2)

Evolutionary Algorithms

Parameters

- population size $\mu \in \mathbb{N}$
- offspring population size $\lambda \in \mathbb{N}$
- crossover probability $p_c \in [0, 1]$

Evolutionary Algorithm

1. Choose $x_1, x_2, \ldots, x_\mu \in S$ uniformly at random.
2. Repeat
3. For $i \in \{1, 2, \ldots, \lambda\}$ do
4. With probability p_c select $z_1, z_2 \in \{x_1, x_2, \ldots, x_\mu\}$.
   $z :=$ crossover(z_1, z_2)
   Else select $z \in \{x_1, x_2, \ldots, x_\mu\}$.

Evolutionary Algorithms

Parameters

- population size $\mu \in \mathbb{N}$
- offspring population size $\lambda \in \mathbb{N}$
- crossover probability $p_c \in [0, 1]$

Evolutionary Algorithm

1. Choose $x_1, x_2, \ldots, x_\mu \in S$ uniformly at random.
2. Repeat
3. For $i \in \{1, 2, \ldots, \lambda\}$ do
4. With probability p_c select $z_1, z_2 \in \{x_1, x_2, \ldots, x_\mu\}$.
   $z :=$ crossover(z_1, z_2)
   Else select $z \in \{x_1, x_2, \ldots, x_\mu\}$.
5. $y_i :=$ mutation(z)

Evolutionary Algorithms

Parameters

- population size $\mu \in \mathbb{N}$
- offspring population size $\lambda \in \mathbb{N}$
- crossover probability $p_c \in [0, 1]$

Evolutionary Algorithm

1. Choose $x_1, x_2, \ldots, x_\mu \in S$ uniformly at random.
2. Repeat
3. For $i \in \{1, 2, \ldots, \lambda\}$ do
4. With probability p_c select $z_1, z_2 \in \{x_1, x_2, \ldots, x_\mu\}$.
   $z :=$ crossover(z_1, z_2)
   Else select $z \in \{x_1, x_2, \ldots, x_\mu\}$.
5. $y_i :=$ mutation(z)
6. Select new $x_1, x_2, \ldots, x_\mu$ out of old $x_1, x_2, \ldots, x_\mu$ and $y_1, y_2, \ldots, y_\lambda$.

Evolutionary Algorithms

Parameters

- population size $\mu \in \mathbb{N}$
- offspring population size $\lambda \in \mathbb{N}$
- crossover probability $p_c \in [0, 1]$

Evolutionary Algorithm

1. Choose $x_1, x_2, \ldots, x_\mu \in S$ uniformly at random.
2. Repeat
3. For $i \in \{1, 2, \ldots, \lambda\}$ do
4. With probability p_c select $z_1, z_2 \in \{x_1, x_2, \ldots, x_\mu\}$.
   $z :=$ crossover(z_1, z_2)
   Else select $z \in \{x_1, x_2, \ldots, x_\mu\}$.
5. $y_i :=$ mutation(z)
6. Select new $x_1, x_2, \ldots, x_\mu$ out of old $x_1, x_2, \ldots, x_\mu$ and $y_1, y_2, \ldots, y_\lambda$.
7. Until 'decide to stop'
8. Output x_i with $f(x_i) = \max\{f(x_j) \mid 1 \leq j \leq \mu\}$.

Modules of Evolutionary Algorithms

Observation need to specifiy

Modules of Evolutionary Algorithms

Observation need to specifiy

- Selection for Reproduction

Modules of Evolutionary Algorithms

Observation need to specifiy

- Selection for Reproduction
  - uniform $\mathrm{Prob}\left(\text{select z}\right) = 1/\mu$

Modules of Evolutionary Algorithms

Observation need to specifiy

- Selection for Reproduction
  - uniform $\mathrm{Prob}\left(\text{select z}\right) = 1/\mu$
  - fitness-proportional $\mathrm{Prob}\left(\text{select z}\right) = f(z)/\sum_{i=1}^{\mu} f(x_i)$

Modules of Evolutionary Algorithms

Observation need to specifiy

- Selection for Reproduction
  - uniform $\mathrm{Prob}\,(\text{select z}) = 1/\mu$
  - fitness-proportional $\mathrm{Prob}\,(\text{select z}) = f(z)/\sum_{i=1}^{\mu} f(x_i)$
  - tournament Select k uniform and of those a best.

Modules of Evolutionary Algorithms

Observation need to specifiy

- Selection for Reproduction
  - uniform $\mathrm{Prob}\,(\text{select z}) = 1/\mu$
  - fitness-proportional $\mathrm{Prob}\,(\text{select z}) = f(z)/\sum_{i=1}^{\mu} f(x_i)$
  - tournament Select k uniform and of those a best.
- Selection for Survival

Modules of Evolutionary Algorithms

Observation need to specifiy

- Selection for Reproduction
  - uniform $\mathrm{Prob}\left(\text{select z}\right) = 1/\mu$
  - fitness-proportional $\mathrm{Prob}\left(\text{select z}\right) = f(z)/\sum_{i=1}^{\mu} f(x_i)$
  - tournament Select k uniform and of those a best.
- Selection for Survival
  - strictly fitness-based among all (plus-selection)

Modules of Evolutionary Algorithms

Observation need to specifiy

- Selection for Reproduction
  - uniform $\mathrm{Prob}\,(\text{select z}) = 1/\mu$
  - fitness-proportional $\mathrm{Prob}\,(\text{select z}) = f(z)/\sum_{i=1}^{\mu} f(x_i)$
  - tournament Select k uniform and of those a best.
- Selection for Survival
  - strictly fitness-based among all (plus-selection)
  - strictly fitness-based among offspring (comma-selection)

Modules of Evolutionary Algorithms

Observation need to specifiy

- Selection for Reproduction
  - uniform $\mathrm{Prob}\,(\text{select z}) = 1/\mu$
  - fitness-proportional $\mathrm{Prob}\,(\text{select z}) = f(z)/\sum_{i=1}^{\mu} f(x_i)$
  - tournament Select k uniform and of those a best.
- Selection for Survival
  - strictly fitness-based among all (plus-selection)
  - strictly fitness-based among offspring (comma-selection)
  - uniform

Modules of Evolutionary Algorithms

Observation need to specifiy

- Selection for Reproduction
  - uniform $\mathrm{Prob}\,(\text{select z}) = 1/\mu$
  - fitness-proportional $\mathrm{Prob}\,(\text{select z}) = f(z)/\sum_{i=1}^{\mu} f(x_i)$
  - tournament Select k uniform and of those a best.
- Selection for Survival
  - strictly fitness-based among all (plus-selection)
  - strictly fitness-based among offspring (comma-selection)
  - uniform
- Variation here only for $S = \{0,1\}^n$

Modules of Evolutionary Algorithms

Observation need to specifiy

- Selection for Reproduction
  - uniform $\mathrm{Prob}\,(\text{select z}) = 1/\mu$
  - fitness-proportional $\mathrm{Prob}\,(\text{select z}) = f(z)/\sum_{i=1}^{\mu} f(x_i)$
  - tournament Select k uniform and of those a best.
- Selection for Survival
  - strictly fitness-based among all (plus-selection)
  - strictly fitness-based among offspring (comma-selection)
  - uniform
- Variation here only for $S = \{0,1\}^n$
  - Crossover(z_1, z_2)
    - uniform crossover Independently for all $1 \leq i \leq n$ select $y[i] \in \{z_1[i], z_2[i]\}$ uniformly at random.

Modules of Evolutionary Algorithms

Observation need to specifiy

- Selection for Reproduction
  - uniform $\mathrm{Prob}\,(\text{select z}) = 1/\mu$
  - fitness-proportional $\mathrm{Prob}\,(\text{select z}) = f(z)/\sum_{i=1}^{\mu} f(x_i)$
  - tournament Select k uniform and of those a best.
- Selection for Survival
  - strictly fitness-based among all (plus-selection)
  - strictly fitness-based among offspring (comma-selection)
  - uniform
- Variation here only for $S = \{0,1\}^n$
  - Crossover(z_1, z_2)
    - uniform crossover Independently for all $1 \leq i \leq n$ select $y[i] \in \{z_1[i], z_2[i]\}$ uniformly at random.
    - k-point crossover Choose $p_1 < p_2 < \ldots p_k \in \{0,1,\ldots,n\}$ uniformly at random.
      $y = z_1[1]z_1[2]\ldots z_1[p_1]z_2[p_1+1]z_2[p_1+2]\cdots z_2[p_2]z_1[p_2+1]\cdots$

Modules of Evolutionary Algorithms

Observation need to specifiy

- Selection for Reproduction
  - uniform $\mathrm{Prob}\,(\text{select z}) = 1/\mu$
  - fitness-proportional $\mathrm{Prob}\,(\text{select z}) = f(z)/\sum_{i=1}^{\mu} f(x_i)$
  - tournament Select k uniform and of those a best.
- Selection for Survival
  - strictly fitness-based among all (plus-selection)
  - strictly fitness-based among offspring (comma-selection)
  - uniform
- Variation here only for $S = \{0,1\}^n$
  - Crossover(z_1, z_2)
    - uniform crossover Independently for all $1 \leq i \leq n$ select $y[i] \in \{z_1[i], z_2[i]\}$ uniformly at random.
    - k-point crossover Choose $p_1 < p_2 < \ldots p_k \in \{0, 1, \ldots, n\}$ uniformly at random.
      $y = z_1[1]z_1[2]\ldots z_1[p_1]z_2[p_1+1]z_2[p_1+2]\cdots z_2[p_2]z_1[p_2+1]\cdots$
      number of crossover points $k \in O(1)$, most often $k \in \{1, 2\}$

Modules of Evolutionary Algorithms (continued)

Observation need to specifiy

- Selection for Reproduction
- Selection for Survival
- Variation here only for $S = \{0,1\}^n$
  - Crossover(z_1, z_2)

Modules of Evolutionary Algorithms (continued)

Observation need to specifiy

- Selection for Reproduction
- Selection for Survival
- Variation here only for $S = \{0,1\}^n$
  - Crossover(z_1, z_2)
  - Mutation(z)

Modules of Evolutionary Algorithms (continued)

Observation need to specifiy

- Selection for Reproduction
- Selection for Survival
- Variation here only for $S = \{0,1\}^n$
  - Crossover(z_1, z_2)
  - Mutation(z)
    - standard bit mutation Independently for all $1 \leq i \leq n$ with probability p_m set $y[i] := 1 - z[i]$ else $y[i] := z[i]$.

Modules of Evolutionary Algorithms (continued)

Observation need to specifiy

- Selection for Reproduction
- Selection for Survival
- Variation here only for $S = \{0,1\}^n$
  - Crossover(z_1, z_2)
  - Mutation(z)
    - standard bit mutation Independently for all $1 \leq i \leq n$ with probability p_m set $y[i] := 1 - z[i]$ else $y[i] := z[i]$.
      mutation probability $p_m \in (0, 1/2]$, most often $p_m = 1/n$

Modules of Evolutionary Algorithms (continued)

Observation need to specifiy

- Selection for Reproduction
- Selection for Survival
- Variation here only for $S = \{0,1\}^n$
  - Crossover(z_1, z_2)
  - Mutation(z)
    - standard bit mutation Independently for all $1 \leq i \leq n$ with probability p_m set $y[i] := 1 - z[i]$ else $y[i] := z[i]$. mutation probability $p_m \in (0, 1/2]$, most often $p_m = 1/n$
    - b-bit mutation Select $p_1 \neq p_2 \neq \cdots \neq p_b \in \{1, 2, \ldots, n\}$ uniformly at random. Set $y[i] := 1 - z[i]$ for $i \in \{p_1, p_2, \ldots, p_b\}$ and $y[i] := z[i]$ otherwise.

Modules of Evolutionary Algorithms (continued)

Observation need to specifiy

- Selection for Reproduction
- Selection for Survival
- Variation here only for $S = \{0,1\}^n$
  - Crossover(z_1, z_2)
  - Mutation(z)
    - standard bit mutation Independently for all $1 \leq i \leq n$ with probability p_m set $y[i] := 1 - z[i]$ else $y[i] := z[i]$.
      mutation probability $p_m \in (0, 1/2]$, most often $p_m = 1/n$
    - b-bit mutation Select $p_1 \neq p_2 \neq \cdots \neq p_b \in \{1, 2, \ldots, n\}$ uniformly at random. Set $y[i] := 1 - z[i]$ for $i \in \{p_1, p_2, \ldots, p_b\}$ and $y[i] := z[i]$ otherwise.
      number of flipping $b \in O(1)$, most often $b \in \{1, 2\}$

Summary & Take Home Message

Things to remember

Summary & Take Home Message

Things to remember

- search problems, optimisation problems

Summary & Take Home Message

Things to remember

- search problems, optimisation problems
- local search (first improvement, best improvement, random)

Summary & Take Home Message

Things to remember

- search problems, optimisation problems
- local search (first improvement, best improvement, random)
- stopping criteria

Summary & Take Home Message

Things to remember

- search problems, optimisation problems
- local search (first improvement, best improvement, random)
- stopping criteria
- Metropolis algorithm, simulated annealing

Summary & Take Home Message

Things to remember

- search problems, optimisation problems
- local search (first improvement, best improvement, random)
- stopping criteria
- Metropolis algorithm, simulated annealing
- evolutionary algorithms

Summary & Take Home Message

Things to remember

- search problems, optimisation problems
- local search (first improvement, best improvement, random)
- stopping criteria
- Metropolis algorithm, simulated annealing
- evolutionary algorithms
  - modules: selection (uniform, fitness-proportional, tournament, plus, comma)

Summary & Take Home Message

Things to remember

- search problems, optimisation problems
- local search (first improvement, best improvement, random)
- stopping criteria
- Metropolis algorithm, simulated annealing
- evolutionary algorithms
  - modules: selection (uniform, fitness-proportional, tournament, plus, comma)
  - modules: variation (uniform crossover, k-point crossover, standard bit mutations)

Summary & Take Home Message

Things to remember

- search problems, optimisation problems
- local search (first improvement, best improvement, random)
- stopping criteria
- Metropolis algorithm, simulated annealing
- evolutionary algorithms
  - modules: selection (uniform, fitness-proportional, tournament, plus, comma)
  - modules: variation (uniform crossover, k-point crossover, standard bit mutations)

Take Home Message

Summary & Take Home Message

Things to remember

- search problems, optimisation problems
- local search (first improvement, best improvement, random)
- stopping criteria
- Metropolis algorithm, simulated annealing
- evolutionary algorithms
  - modules: selection (uniform, fitness-proportional, tournament, plus, comma)
  - modules: variation (uniform crossover, k-point crossover, standard bit mutations)

Take Home Message

- Optimisation is searching for an optimal solution.

Summary & Take Home Message

Things to remember

- search problems, optimisation problems
- local search (first improvement, best improvement, random)
- stopping criteria
- Metropolis algorithm, simulated annealing
- evolutionary algorithms
  - modules: selection (uniform, fitness-proportional, tournament, plus, comma)
  - modules: variation (uniform crossover, k-point crossover, standard bit mutations)

Take Home Message

- Optimisation is searching for an optimal solution.
- RSH are easy to implement and easy to apply.

Summary & Take Home Message

Things to remember

- search problems, optimisation problems
- local search (first improvement, best improvement, random)
- stopping criteria
- Metropolis algorithm, simulated annealing
- evolutionary algorithms
  - modules: selection (uniform, fitness-proportional, tournament, plus, comma)
  - modules: variation (uniform crossover, k-point crossover, standard bit mutations)

Take Home Message

- Optimisation is searching for an optimal solution.
- RSH are easy to implement and easy to apply.
- Many different randomised search heuristics exist.